Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présenté et soutenu le *26/11/2024* **par :**

RACHID EL MONTASSIR

Approche hybride basée sur la physique et l'IA pour l'advection des champs de probabilités. Application à la prévision immédiate de la couverture nuageuse.

JURY

FABRICE GAMBOA	IMT / Univ. Toulouse III	Président du jury
FRANÇOIS FLEURET	Meta / Univ. Genève	Rapporteur
GUILLAUME GASTINEAU	LOCEAN	Rapporteur
GUILLAUME BALARAC	LEGI / Grenoble INP	Examinateur
SIDONIE LEFEBVRE	ONERA	Examinatrice
OLIVIER PANNEKOUCKE	CNRM / CERFACS	Directeur de thèse
CORENTIN LAPEYRE	NVIDIA	Invité

École doctorale et spécialité :
SDU2E : Océan, Atmosphère, Climat

Unité de Recherche :
UMR 5318 - CECI - Climat, Environnement, Couplages et Incertitudes / CERFACS

Directeurs de Thèse :
Olivier Pannekoucke et *Corentin Lapeyre*

Rapporteurs :
François Fleuret et *Guillaume Gastineau*

Abstract

During the last decades, as the global warming has accelerated, so has the frequency of extreme weather events, significantly affecting societies and the economies. These events highlight the growing need for accurate weather forecasting. Traditional numerical weather prediction models, while effective, remain computationally expensive and struggle to predict small-scale phenomena such as thunderstorms. Meanwhile, deep learning models have shown promise in weather forecasting but often lack physical consistency and generalisation capabilities.

This thesis addresses the limitations of traditional deep learning methods in producing realistic and physically consistent results that can generalise to unseen data. In this thesis, we explore hybrid methods that seek to reconcile the accuracy of first-principle methods with the data-leveraging power of learning techniques, with an application to cloud cover nowcasting. The cloud cover data used are satellite images with cloud type classification, and the goal is to predict the cloud cover position over the next two hours while preserving the classification of the cloud types.

The proposed approach, named HyPhAICC, enforces physical behaviour based on probability fields advection. In the first model, denoted HyPhAICC-1, multi-level advection dynamics are used to guide the learning of a U-Net model. This is achieved by solving the advection equation for multiple probability fields, each corresponding to a different cloud type, while simultaneously learning the unknown velocity field.

Our experiments show that the hybrid formulation outperforms not only the EUMETSAT Extrapolated Imagery model (EXIM) but also the U-Net model in terms of standard metrics such as F1 score, Critical Success Index (CSI), and accuracy. We also demonstrate that the HyPhAICC-1 model preserves more details and produces more realistic results compared to the U-Net model. To quantitatively measure this aspect, we use a modified version of the Hausdorff distance which is, to the best of our knowledge, the first time this metric is used for this purpose in the literature. This first version shows also a significant faster convergence. It also performed significantly better compared to the U-Net when trained on smaller datasets, highlighting the computational efficiency of the proposed approach. Another model, denoted HyPhAICC-2, adds a source term to the advection equation. While this impaired the visual rendering, it displayed the best performance in terms of accuracy.

These results suggest that the proposed hybrid Physics-AI architecture provides a promising solution to overcome the limitations of traditional AI methods. This could motivate further research to combine physical knowledge with deep learning models for more accurate and efficient weather forecasting.

Résumé

Au cours des dernières décennies, le réchauffement climatique s'est accéléré, tout comme la fréquence des événements météorologiques extrêmes, affectant considérablement les sociétés et les économies. Ces événements soulignent le besoin croissant de prévisions météorologiques précises. Les modèles traditionnels de prévision numérique du temps, bien qu'efficaces, restent coûteux en termes de calcul et peinent à prédire les phénomènes à petite échelle tels que les orages. Parallèlement, les modèles d'apprentissage profond se sont révélés prometteurs dans les prévisions météorologiques, mais manquent souvent de cohérence physique et de capacités de généralisation.

Cette thèse aborde les limites des méthodes traditionnelles d'apprentissage profond dans la production de résultats réalistes et physiquement cohérents qui peuvent se généraliser à des données non vues. Dans cette thèse, nous explorons des méthodes hybrides qui cherchent à concilier la précision des méthodes de premier principe avec la puissance d'exploitation des données des techniques d'apprentissage, avec une application à la prévision immédiate de la couverture nuageuse. Les données de couverture nuageuse utilisées sont des images satellites avec classification des types de nuages, et l'objectif est de prédire la position de la couverture nuageuse au cours des deux prochaines heures tout en préservant la classification des types de nuages.

L'approche proposée, nommée HyPhAICC, impose un comportement physique basé sur l'advection des champs de probabilités. Dans le premier modèle, dénommé HyPhAICC-1, des dynamiques d'advection multi-niveaux sont utilisées pour guider l'apprentissage d'un modèle U-Net. Cela est réalisé en résolvant l'équation d'advection pour plusieurs champs de probabilité, chacun correspondant à un type de nuage différent, tout en apprenant simultanément le champ de vitesse inconnu.

Nos expériences montrent que la formulation hybride surpasse non seulement le modèle d'imagerie extrapolée d'EUMETSAT (EXIM), mais également le modèle U-Net en termes de métriques standard telles que le score F1, l'indice de succès critique (CSI) et l'accuracy. Nous démontrons également que le modèle HyPhAICC-1 préserve plus de détails et produit des résultats plus réalistes par rapport au modèle U-Net. Pour mesurer quantitativement cet aspect, nous utilisons une version modifiée de la distance de Hausdorf qui est, à notre connaissance, la première fois que cette métrique est utilisée à cette fin dans la littérature. Cette première version montre aussi une convergence remarquablement rapide. Elle a également affiché de meilleures performances par rapport au U-Net lorsqu'elle a été entraînée sur des ensembles de données plus petits, soulignant l'efficacité computationnelle de l'approche proposée.

Un autre modèle, dénommé HyPhAICC-2, ajoute un terme source à l'équation d'advection. Bien que cela ait dégradé le rendu visuel, il a affiché les meilleures performances en termes d'accuracy. Ces résultats suggèrent que l'architecture hybride physique-IA proposée

constitue une solution prometteuse pour surmonter les limitations des méthodes d'IA traditionnelles. Cela pourrait motiver des recherches supplémentaires pour combiner les connaissances physiques avec les modèles d'apprentissage profond afin d'améliorer la précision et l'efficacité des prévisions météorologiques.

Acknowledgements

First, I would like to thank the members of the jury for taking the time to review this work, for attending my defense, and for their helpful feedback.

I am very grateful to my thesis advisors, Olivier Pannekouce and Corentin Lapeyre, for their support, guidance, and advice throughout these three years. Their knowledge and availability were essential for the success of this project.

I would also like to thank my parents and my wife for their constant support and encouragement, which have motivated me during this journey.

A big thank you to my colleagues and friends for their valuable discussions and assistance. In particular, my office mate Victor Coulon, and all the members of the Algo-COOP and GlobC teams. I also want to thank Laurent Terray and Luciano Drozda for their support and advice during the past few months.

I would like to thank the technical support and administrative teams, especially Isabelle D'Ast and Chantal Nasri for their help and efficiency throughout these years.

Finally, I am grateful to CERFACS for giving me the opportunity to complete this thesis in such a supportive and stimulating environment.

Contents

List of Figures

List of Tables

List of abbreviations

The next list describes several abbreviations that will be later used within the body of the document

AdaGrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
CFL	Courant-Friedrichs-Lewy
CNN	Convolutional Neural Network
CSI	Critical Success Index
DL	Deep Learning
ECMWF	European Centre for Medium-Range Weather Forecasts (international organisation)
EDP	Équations aux Dérivées Partielles
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
FCN	Fully Convolutional Networks
GAN	Generative Adversarial Network
GFS	Global Forecast System (NCEP)
GRU	Gated Recurrent Unit
HyPhAICC	Hybrid Physics-AI architecture for Cloud Cover nowcasting

IFS	Integrated Forecasting System (ECMWF)
IoU	Intersection over Union
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
MSG	Meteosat Second Generation
NAG	Nesterov Accelerated Gradient
NCEP	National Centers for Environmental Prediction (US)
NLP	Natural Language Processing
NWP	Numerical Weather Prediction
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Networks
PNT	Prévision Numérique du Temps
ResNet	Residual Network
rHD	Restricted Hausdorff Distance
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
SDE	Stochastic Differential Equation
SGD	Stochastic Gradient Descent
ViT	Vision Transformer

Chapter 1 Introduction

We would like to start this thesis with Fig. 1.1. This picture was taken in September 2024 in the south-east of Morocco, more precisely in Merzouga.

Figure 1.1: Large ponds formed in the middle of a desert in Merzouga, Morocco after heavy rains (September 2024). Source: Al Jazeera.

This picture shows a beautiful contrast between the desert and the water. However, it also shows the devastating reality of the floods that hit the region. The world is facing an alarming increase of extreme weather events. Thus, accurate, timely and reliable weather forecasting is essential more than ever to reduce the impact of these events on human lives and activities.

Weather forecasting has been essential for centuries, linked to farming, navigation, and safety. Ancient civilisations, like the Babylonians, predicted weather using cloud formations and animal behaviour [Ossendrijver, 2021, Hazen, 1900]. The introduction of meteorological instruments in the 17th century, including the barometer, thermometer, and hygrometer, enabled systematic atmospheric observations, though early forecasts remained qualitative.

In the 19th century, advances in thermodynamics and fluid dynamics deepened the

understanding of weather forces. In 1904, physicist Vilhelm Bjerknes proposed that weather could be mathematically predicted by solving atmospheric equations, marking a significant shift in meteorology.

Building on this concept, numerical weather prediction (NWP) models began to emerge. These models rely on mathematical equations to simulate the atmosphere's behaviour using data like temperature, pressure, and wind conditions. The first real revolution came in the 1920s, when Lewis Fry Richardson developed an NWP model that laid the foundation for the sophisticated forecasting systems we rely on today. Even if Richardson's manual calculations were slow and impractical for real-time forecasting, his work laid the foundation for modern numerical methods. The advent of digital computers in the mid-20th century revolutionised weather prediction by enabling the rapid computation of complex equations. The 1950s saw the first operational NWP models Charney et al. [1950], which have since become the cornerstone of weather forecasting.

However, even with their success, NWP models have their limitations. They often struggle to capture small-scale weather events, such as thunderstorms, tornadoes, and localised heavy rainfall events [Schultz et al., 2021, Matte et al., 2022, Joe et al., 2022]. These phenomena often evolve too quickly and on a finer scale for traditional NWP models to accurately predict.

Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) have opened new possibilities for weather prediction. Deep Learning (DL), a subset of ML, has transformed fields like computer vision, natural language processing, and now, weather forecasting. In weather prediction, DL models have shown promise in processing large datasets, including satellite imagery and radar data, leading to improvements in forecast accuracy and efficiency.

One of the first notable applications of DL in weather prediction was the "Convolutional LSTM (Long Short-Term Memory)" model introduced by Shi et al. [2015], designed for precipitation nowcasting by predicting radar echo maps. This pioneering work has inspired further research to apply DL to various weather prediction tasks. However, DL models for weather forecasting face significant limitations. One of these limitations is the lack of physical consistency in the predictions, as illustrated in Fig. 1.2 for cloud cover nowcasting task, where a significant loss of small cloud structures is observed in the prediction.

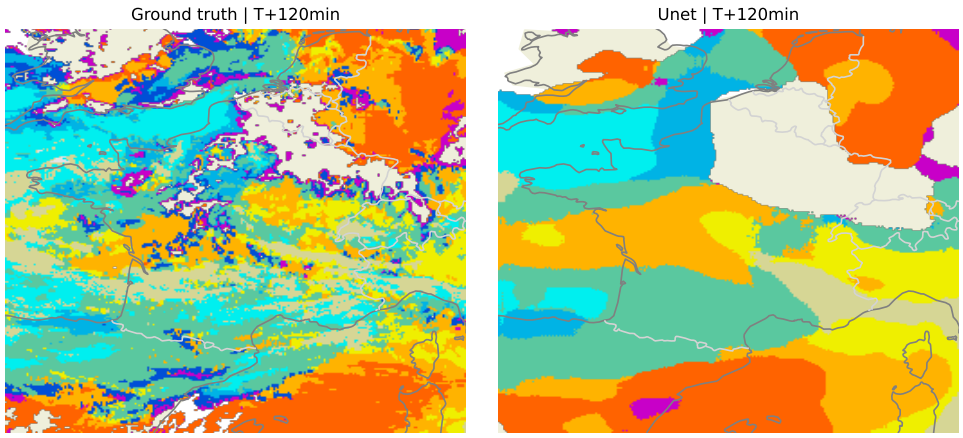


Figure 1.2: Example of cloud cover prediction using a U-Net model at 120 minutes ahead. Left: observed cloud cover. Right: predicted cloud cover, each colour represents a different cloud type.

Indeed, traditional DL models are typically trained to minimise a loss function, often without regard to the physical principles that govern atmospheric dynamics. This can lead to predictions that are accurate in terms of data-driven metrics but physically implausible. Another limitation is the challenge of generalisation. DL models tend to excel when making predictions on data similar to what they were trained on but can struggle with unseen scenarios. This limitation is more crucial in meteorology, where weather patterns can vary widely depending on the geographic location or season. This means that a DL-based model that performs well in one region or season may not necessarily do so in another, which poses a significant challenge for global or long-term forecasting. In addition, DL often requires large amounts of labelled data, which can be difficult to obtain for rare or extreme weather events.

To address these limitations, there is a growing interest in integrating physical laws into ML models, particularly in domains where such laws play a critical role in governing the behaviour of the system. This approach, often referred to as physics-informed ML or hybrid modelling, combines the pattern recognition capabilities of DL with the rigour and consistency of physical models.

In weather forecasting, hybrid models that combine DL with physical dynamics offer the potential to produce more accurate and physically consistent forecasts. By guiding the learning process with physical principles, these models can better capture the underlying mechanisms of atmospheric phenomena, leading to improved performance, particularly in scenarios where traditional DL models struggle.

In this study, we consider a particular weather forecasting task: cloud cover nowcasting. Nowcasting refers to the process of generating short-term weather forecasts - typically on the timescale of minutes to a few hours - using recently acquired high-resolution observational data. This data, which can come from sources such as radar,

satellite imagery, and ground-based sensors, is used to generate real-time estimates of weather conditions. Nowcasting is particularly valuable for predicting rapidly developing weather phenomena, such as thunderstorms, heavy rain, and severe winds, which can pose significant risks to public safety and property Reyniers [2016], Wilson et al. [1998].

Although traditional NWP models are effective fsor longer-term forecasts, they often struggle with the spatial and temporal resolution required for nowcasting. Here, nowcasting techniques excel by leveraging near-real-time data. By processing recent observations, nowcasting models can provide highly localised and timely forecasts, making them essential for applications such as aviation, emergency management, and outdoor event planning.

Cloud cover nowcasting is less treated in the literature compared to other weather forecasting tasks such as precipitation nowcasting. However, it is a crucial component of weather forecasting, as cloud formation and movement are closely related to the development of precipitation, thunderstorms, and other hazardous weather events. Accurate short-term predictions of cloud cover are particularly important for sectors such as aviation, agriculture, and renewable energy, where even slight inaccuracies can lead to significant operational challenges.

Traditionally, cloud cover forecasting has relied on physics-based methods, such as tracking cloud motion vectors, optical flow, or NWP-based Data Assimilation. These methods model the evolution of clouds on the basis of the laws of physics, such as fluid dynamics and thermodynamics, providing predictions that are consistent with our understanding of atmospheric processes.

Recently, there has been a growing interest in data-driven approaches for cloud cover nowcasting. These methods use historical data to learn patterns and make predictions, offering the potential for improved accuracy and efficiency. However, as discussed earlier, purely data-driven approaches can struggle with physical consistency and generalisation, particularly in the complex and variable environment of the atmosphere. Above all,

why train a model to learn processes that are already known and can be modelled ?

Given the limitations of both traditional physics-based methods and purely data-driven models, a hybrid approach that combines the strengths of both is necessary. This research proposes a hybrid architecture for cloud cover nowcasting, integrating physical dynamics into neural networks to enhance accuracy, efficiency, and physical consistency. However,

how can we effectively integrate physical knowledge into AI models?

The cloud cover data used in this study consists of satellite images with cloud type classifications, leading us to adopt a probabilistic approach for predicting cloud type probabilities. Yet,

how can we maintain this probabilistic characteristic within the hybrid model?

Moreover,

what if we want to introduce a source term into the physical formulation?

The central hypothesis of this research is that incorporating physical knowledge into AI models will enhance cloud cover nowcasting performance. The key objectives include developing a hybrid model, evaluating its performance against traditional models, and analysing the computational efficiency (training time and data requirements) of the proposed approach, assessing the potential for generalisation to unseen situations, and exploring the potential trade-offs between the different approaches. This study contributes to the emerging field of hybrid modelling, with broader implications for improving the accuracy, reliability and efficiency in weather forecasting and other areas where physical laws should be considered.

The remainder of this thesis is structured as follows:

- **Chapter** 2 provides an overview of the foundations and basic concepts of deep learning and some classical architectures.
- **Chapter** 3 is an introduction to numerical methods for solving partial differential equations, focusing on the finite differences.
- **Chapter** 4 is an overview of the hybrid modelling approaches that have been proposed in the literature, focusing on the method used in this study for solving PDEs within neural networks.
- **Chapter** 5 presents the methods used in the literature for weather forecasting, first focusing on NWP models and then on DL models.
- **Chapter** 6 presents the proposed hybrid model for cloud cover nowcasting, detailing the architecture, training process, and evaluation procedure.
- **Chapter** 7 presents other versions of the hybrid model extending the physical formulation to include source terms.
- **Chapter** 8 is the conclusion of this thesis, summarising the main results and discussing potential future research directions.

Figure 1.3 illustrates the structure of this thesis.

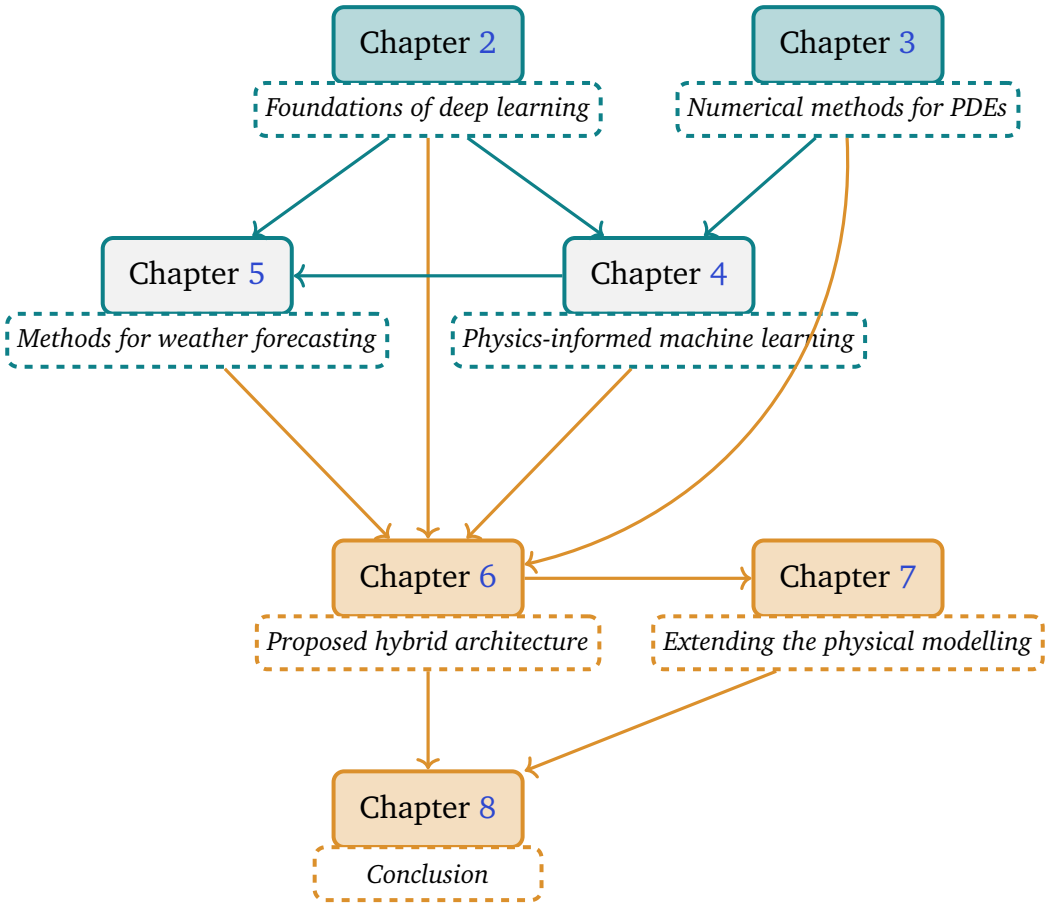


Figure 1.3: Structure and interactions between thesis chapters. The orange colour represents the parts containing the contributions of this thesis.

Introduction (French version)

Nous souhaitons commencer cette thèse par la figure 1.4. Cette image a été prise en septembre 2024 dans le sud-est du Maroc, plus précisément à Merzouga.

Figure 1.4: De grands étangs formés au milieu du désert à Merzouga, au Maroc, après de fortes pluies (septembre 2024). Source : Al Jazeera.

Cette image montre un beau contraste entre le désert et l'eau. Cependant, elle révèle également la réalité dévastatrice des inondations qui ont frappé la région. Le monde est confronté à une augmentation alarmante des événements météorologiques extrêmes. Ainsi, des prévisions météorologiques précises, rapides et fiables sont plus essentielles que jamais pour réduire l'impact de ces événements sur la vie humaine et les activités.

Les prévisions météorologiques sont essentielles depuis des siècles, elles sont liées à l'agriculture, à la navigation et à la sécurité. Les anciennes civilisations, comme les Babyloniens, prédisaient le temps en utilisant les formations nuageuses et le comportement animal [Ossendrijver, 2021, Hazen, 1900]. L'introduction d'instruments météorologiques au 17^{e} siècle, y compris le baromètre, le thermomètre et l'hygromètre, a permis d'avoir des observations atmosphériques systématiques. Bien que les premières prévisions demeurent qualitatives.

Au 19^{e} siècle, les avancées en thermodynamique et en dynamique des fluides ont approfondi notre compréhension des forces météorologiques. En 1904, le physicien Vilhelm Bjerknes a suggéré que la météo pourrait être prédite mathématiquement en résolvant des équations atmosphériques, marquant un tournant significatif en météorologie.

En s'appuyant sur ce concept, des modèles de prévision numérique du temps (PNT) ont commencé à émerger. Ces modèles reposent sur des équations mathématiques pour simuler le comportement de l'atmosphère en utilisant des données telles que la température, la pression et les conditions de vent. La première véritable révolution est survenue dans les années 1920, lorsque Lewis Fry Richardson a développé un modèle PNT qui a jeté les bases des systèmes de prévisions sophistiqués sur lesquels nous comptons aujourd'hui. Bien que les calculs manuels de Richardson aient été lents et peu pratiques pour les prévisions en temps réel, son travail a ouvert la voie aux méthodes numériques modernes. L'avènement des ordinateurs numériques au milieu du 20^{e} siècle a révolutionné la prévision météorologique en permettant le calcul rapide d'équations complexes. Les années 1950 ont vu les premiers modèles PNT opérationnels Charney et al. [1950], qui sont depuis devenus la pierre angulaire des prévisions météorologiques.

Cependant, même avec leur succès, les modèles PNT ont leurs limites. Ils peinent souvent à capturer des événements météorologiques à fine échelle, tels que les orages, les tornades et les événements de fortes pluies localisées [Schultz et al., 2021, Matte et al., 2022, Joe et al., 2022]. Ces phénomènes évoluent souvent trop rapidement et à une échelle plus fine pour que les modèles PNT traditionnels puissent les prédire avec précision.

Les récentes avancées en intelligence artificielle (IA) et en apprentissage automatique (ML) ont ouvert de nouvelles possibilités pour la prévision météorologique. L'apprentissage profond (DL), un sous-ensemble du ML, a transformé des domaines comme la vision par ordinateur, le traitement du langage naturel, et maintenant, la prévision météorologique. Dans la prévision météorologique, les modèles DL ont montré leur promesse en traitant de grands ensembles de données, y compris des images satellites et des données radar, conduisant à des améliorations de la précision et de l'efficacité des prévisions.

L'une des premières applications notables du DL dans la prévision météorologique était le modèle "Convolutional LSTM (Long Short-Term Memory)" introduit par Shi et al. [2015], conçu pour la prévision immédiate des précipitations en prédisant des cartes d'écho radar. Ce travail pionnier a inspiré d'autres recherches visant à appliquer le DL à diverses tâches de prévision météorologique. Cependant, les modèles DL pour la prévision météorologique font face à des limitations significatives. L'une de ces limitations est le manque de cohérence physique dans les prédictions, comme illustré dans Fig. 1.5 sur une tâche de prévision immédiate de la couverture nuageuse, sur laquelle une perte significative des petites structures nuageuses est observée dans la prédiction.

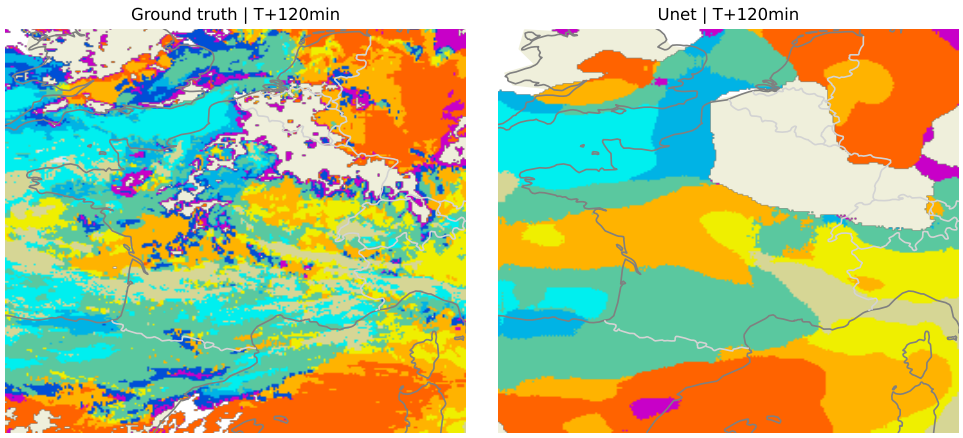


Figure 1.5: Exemple de prédiction de la couverture nuageuse à l'aide d'un modèle U-Net à 120 minutes. Gauche : couverture nuageuse observée. Droite : couverture nuageuse prédite, chaque couleur représente un type de nuage différent.

En effet, les modèles DL traditionnels sont généralement entraînés pour minimiser une fonction de perte, souvent sans tenir compte des principes physiques qui régissent la dynamique atmosphérique. Cela peut conduire à des prédictions qui sont précises en termes de métriques basées sur les données, mais physiquement peu plausibles. Une autre limitation est le défi de la généralisation. Les modèles DL ont tendance à exceller lorsqu'il s'agit de faire des prédictions sur des données similaires à celles sur lesquelles ils ont été entraînés, mais peuvent avoir des difficultés avec des situations non vues, ou avec des conditions extrêmes qui ne sont pas représentées dans les données d'entraînement.

Cette limitation est particulièrement prononcée en météorologie, où les modèles météorologiques peuvent varier considérablement en fonction de l'emplacement géographique ou de la saison. Cela signifie qu'un modèle d'apprentissage profond qui fonctionne bien dans une région ou une saison donnée peut ne pas nécessairement fonctionner de la même manière ailleurs, ce qui représente un défi important pour les prévisions globales ou à long terme. De plus, le DL nécessite souvent de grandes quantités de données étiquetées, ce qui peut être difficile à obtenir pour des événements météorologiques rares ou extrêmes.

Pour remédier à ces limitations, un intérêt croissant se manifeste pour l'intégration des lois physiques dans les modèles d'apprentissage automatique (ML), en particulier dans des domaines dans lesquels ces lois jouent un rôle critique dans le comportement du système. Cette approche, souvent appelée *physics-informed ML* ou modélisation hybride, combine les capacités de reconnaissance des patterns des réseaux de neurones avec la rigueur et la cohérence des modèles physiques.

Dans les prévisions météorologiques, les modèles hybrides qui combinent le DL avec les dynamiques physiques offrent le potentiel de produire des prévisions plus

précises et physiquement cohérentes. En guidant le processus d'apprentissage par des principes physiques, ces modèles peuvent mieux capturer les mécanismes sous-jacents des phénomènes atmosphériques, menant à une performance améliorée, notamment dans des scénarios où les modèles DL traditionnels rencontrent des difficultés.

Dans cette étude, nous considérons une tâche particulière de prévision météorologique : la prévision immédiate de la couverture nuageuse. La prévision immédiate fait référence au processus de génération de prévisions météorologiques à court terme — typiquement à l'échelle de quelques minutes à quelques heures — en utilisant des données d'observation, généralement de haute résolution, les plus récentes. Ces données, qui peuvent provenir de sources telles que le radar, les images satellites et les capteurs au sol, sont utilisées pour générer des estimations en temps réel des conditions météorologiques. La prévision immédiate est particulièrement utile pour prédire des phénomènes météorologiques en développement rapide, tels que les orages, les fortes pluies et les vents violents, qui peuvent poser des risques significatifs pour la sécurité publique et les biens.

Bien que les modèles PNT traditionnels soient efficaces pour les prévisions à long terme, ils ont souvent du mal avec la résolution spatiale et temporelle requise pour la prévision immédiate. Ici, les techniques de prévision immédiate excellent en tirant parti des données quasi temps réel. En traitant des observations récentes, les modèles de prévision immédiate peuvent fournir des prévisions très localisées et opportunes, ce qui les rend essentiels pour des applications telles que l'aviation, la gestion des urgences et la planification d'événements en plein air Reyniers [2016], Wilson et al. [1998].

La prévision immédiate de la couverture nuageuse est moins traité dans la littérature par rapport à d'autres tâches de prévision météorologique, telles que la prévision immédiate des précipitations. Cependant, il reste un élément crucial de la prévision météorologique, car la formation et le mouvement des nuages sont étroitement liés au développement des précipitations, des orages et d'autres événements météorologiques dangereux. Des prévisions précises à court terme de la couverture nuageuse sont particulièrement importantes pour des secteurs tels que l'aviation, l'agriculture et les énergies renouvelables, où même de légères inexactitudes peuvent entraîner d'importants défis opérationnels.

Traditionnellement, la prévision de la couverture nuageuse s'est appuyée sur des méthodes basées sur la physique, telles que le suivi des vecteurs de mouvement des nuages, le flux optique ou les modèles PNT [Dupuy et al., 2020]. Ces méthodes modélisent l'évolution des nuages sur la base des lois de la physique, telles que la dynamique des fluides et la thermodynamique, fournissant des prévisions qui sont cohérentes avec notre compréhension des processus atmosphériques.

Récemment, un intérêt croissant s'est manifesté pour les approches basées sur les données pour la prévision immédiate de la couverture nuageuse [Berthomier et al., 2020]. Ces méthodes utilisent des données historiques pour apprendre des modèles et

faire des prévisions, offrant un potentiel d'amélioration de la précision et de l'efficacité. Cependant, comme discuté précédemment, les approches purement basées sur les données peuvent rencontrer des difficultés en matière de cohérence physique et de généralisation, en particulier dans l'environnement complexe et variable de l'atmosphère. Avant tout,

pourquoi entraîner un modèle pour apprendre des processus qui sont déjà connus et peuvent être modélisés ?

Compte tenu des limitations des méthodes traditionnelles basées sur la physique et des modèles purement basés sur les données, une approche hybride qui combine les forces des deux est nécessaire. Cette recherche propose une architecture hybride pour la prévision immédiate de la couverture nuageuse, intégrant les dynamiques physiques dans des réseaux neuronaux pour améliorer la précision, l'efficacité et la cohérence physique. Cependant,

comment pouvons-nous intégrer efficacement les connaissances physiques dans les modèles d'IA ?

Les données de couverture nuageuse utilisées dans cette étude se composent d'images satellites avec des classifications de types de nuages, ce qui nous amène à adopter une approche probabiliste pour prédire les probabilités des types de nuages. Pourtant,

comment pouvons-nous maintenir ce caractère probabiliste au sein du modèle hybride ?

De plus,

que se passe-t-il si nous voulons introduire un terme source dans la formulation physique ?

L'hypothèse centrale de cette recherche est que l'incorporation de connaissances physiques dans les modèles d'IA améliorera les résultats de la prévision immédiate de la couverture nuageuse. Les objectifs clés comprennent le développement d'un modèle hybride, l'évaluation de ses performances par rapport aux modèles traditionnels, l'analyse de l'efficacité computationnelle (temps d'entraînement et exigences en matière de données) de l'approche proposée, l'évaluation du potentiel de généralisation à des situations non vues, et l'exploration des compromis potentiels entre les différentes approches. Cette étude contribue au domaine émergent de la modélisation hybride, avec des implications plus larges pour améliorer la précision, la fiabilité et l'efficacité des prévisions météorologiques et d'autres domaines où les lois physiques doivent être prises en compte.

Le reste de cette thèse est structuré comme suit :

- **Chapitre** 2 fournit un aperçu des fondements et des concepts de base de l'apprentissage profond et de certaines architectures classiques.

- **Chapitre** 3 est une introduction aux méthodes numériques pour résoudre les équations dérivées partielles (EDP), en se concentrant sur les différences finies.
- **Chapitre** 4 donne un aperçu sur les approches de modélisation hybride qui ont été proposées dans la littérature, en se concentrant sur la méthode utilisée dans cette étude pour résoudre les EDP au sein des réseaux neuronaux.
- **Chapitre** 5 présente les méthodes utilisées dans la littérature pour la prévision météorologique, en se concentrant d'abord sur les modèles PNT, puis sur les modèles DL.
- **Chapitre** 6 présente le modèle hybride proposé pour la prévision immédiate de la couverture nuageuse, détaillant l'architecture, le processus d'entraînement et la procédure d'évaluation.
- **Chapitre** 7 présente d'autres versions du modèle hybride étendant la formulation physique pour inclure des termes sources.
- **Chapitre** 8 est la conclusion de cette thèse, résumant les principaux résultats et discutant des orientations possibles pour des future travaux.

Figure 1.6 illustre la structure de cette thèse.

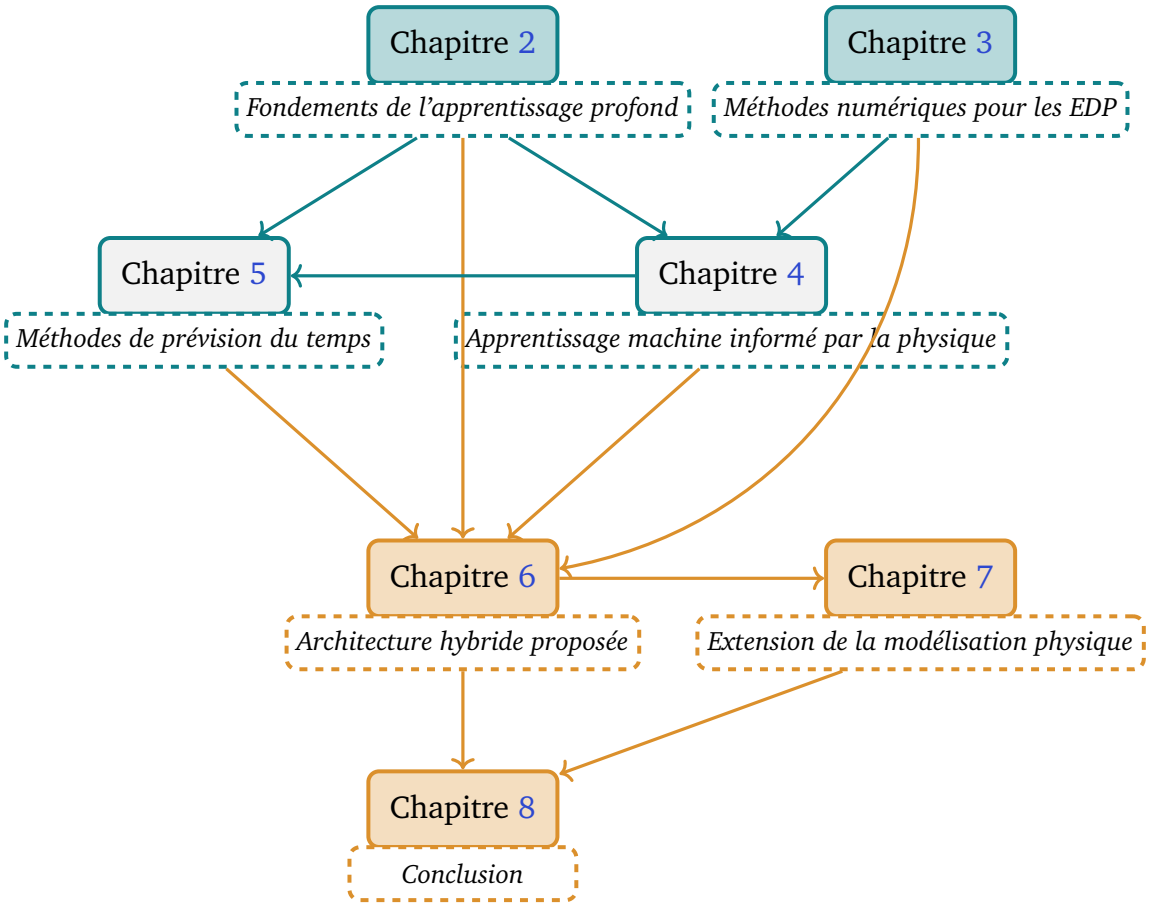


Figure 1.6: Structure et interactions entre les chapitres de la thèse. La couleur orange représente les parties contenant des contributions de cette thèse.

Chapter 2 Fundamentals of deep learning

This work relies on deep learning and related concepts. To provide a solid foundation, this chapter offers an introduction to the core principles of DL, along with the key concepts that will be referenced throughout the subsequent chapters.

The chapter is organised as follows: Section 1 provides a historical overview of DL and highlights key developments that have led to its widespread adoption in recent years. Sections 2.2 to 2.9 introduce the fundamental concepts of DL, including perceptrons, multi-layer perceptrons and gradient descent optimisation methods. Followed by some classical architectures in Section 2.10.

2.1 A brief history of deep learning (DL)

The concept of artificial neural networks (ANNs) was first introduced in the 1940s by McCulloch and Pitts [1943], who proposed a simplified mathematical abstraction of a biological neuron (Figure 2.1). The model consisted of a network of interconnected

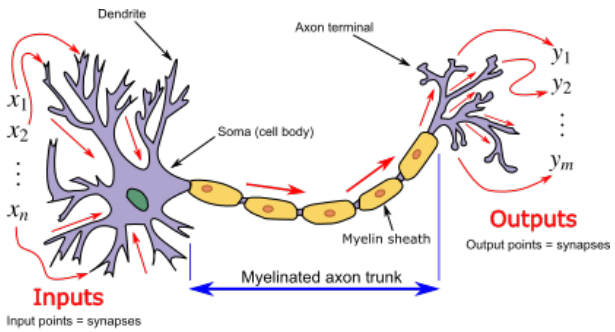


Figure 2.1: Schematic of a biological neuron. The dendrites receive signals from other neurons, which are transmitted to the cell body. The axon transmits signals to other neurons. Synapses are the connections between neurons. (Image source: Prof. Loc Vu-Quoc, University of Florida).

neurons, each of which could be in one of two states: *on* or *off*. The state of each neuron was determined by the weighted sum of the states of its input neurons, with the weights representing the strength of the connections between neurons. This model, known as the McCulloch-Pitts neuron, provided the foundation for subsequent developments in neural network theory. In 1957, Rosenblatt introduced the perceptron, an early neural network model designed for binary classification tasks. The perceptron is able to learn weights from input features to make decisions, but its limitations in solving complex problems and handling non-linearities became apparent. In 1969, Minsky and Papert demonstrated that single-layer perceptrons had limitations in solving problems that were not linearly separable, contributing to a period of decreased interest in AI research during the 1970s and 1980s known as the "AI winter".

In the 1980s, the development of the backpropagation algorithm by Rumelhart et al. [1986] enabled the training of multi-layer neural networks, overcoming the limitations of single-layer perceptrons. This breakthrough paved the way for the development of more sophisticated neural network architectures, including recurrent neural networks (RNN) and radial basis function networks. However, the computational complexity of training these models limited their widespread adoption. In the late 1980s and early 1990s, convolution-based neural networks (CNNs) were introduced, offering a more efficient approach to training neural networks.

The 21st century witnessed a resurgence of interest in neural networks, particularly with the advent of DL. Advances in hardware, such as graphics processing units (GPUs), and the availability of large datasets enabled the training of deep neural networks [LeCun et al., 2015]. Breakthroughs in computer vision, natural language processing, and other domains have propelled the widespread adoption of DL across various industries.

In the 21st century, artificial neural networks (ANNs) have experienced significant advancements, fueled by increased computational power, the availability of large datasets, and breakthroughs in training algorithms. In 2012, the ImageNet Large Scale Visual Recognition Challenge demonstrated the effectiveness of DL in image classification. AlexNet, a deep CNN architecture, outperformed traditional methods and paved the way for the widespread adoption of convolutional neural networks in computer vision tasks [Krizhevsky et al., 2012].

RNNs gained prominence for their ability to model sequential data, making them well-suited for natural language processing and time-series analysis. Long Short-Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Units (GRUs) were introduced to address challenges related to vanishing gradients [Pascanu et al., 2013] and long-term dependencies in RNNs.

Researchers started leveraging pre-trained models on large datasets to improve the performance of neural networks on specific tasks with limited data. Transfer learning, where a model trained on one task is adapted to another related task, became a common

practice, leading to the development of versatile and effective models [Pan and Yang, 2010].

Introduced by Ian Goodfellow and his colleagues in 2014, Generative Adversarial Networks (GANs) revolutionised the field of generative modelling. GANs consist of two neural networks, a generator and a discriminator, trained adversarially to generate realistic data, such as images or text [Goodfellow et al., 2014].

The Transformer architecture, introduced by Vaswani et al. [2017], revolutionised natural language processing tasks. The attention mechanisms within Transformers allowed models to focus on different parts of the input sequence, which led to improved performance in tasks such as machine translation and language understanding [Brown et al., 2020]. In 2018, Bidirectional Encoder Representations from Transformers (BERT) demonstrated remarkable success in natural language understanding tasks. Pre-trained language models, such as GPT-3 (Generative Pre-trained Transformer 3) by OpenAI, showcase the power of large-scale pre-training on diverse language tasks [Devlin et al., 2019].

Today, deep neural networks are fundamental components of many cutting-edge technologies, including image recognition, natural language processing, and autonomous systems, reaching human-level performance in some tasks [Silver et al., 2017]. The field of DL continues to evolve rapidly, with ongoing research in areas such as reinforcement learning, unsupervised learning, and explainable AI. The next sections provide an introduction to some fundamental concepts of DL.

2.2 Artificial neural networks

The perceptron, introduced by Rosenblatt [1958], is a simple neural network model that can be used for binary classification tasks. The perceptron consists of a single layer of neurons, each of which is connected to the input features by a set of weights. The output of the perceptron is determined by the weighted sum of the inputs, which is passed through an activation function to produce a binary output as shown in Figure 2.2. It

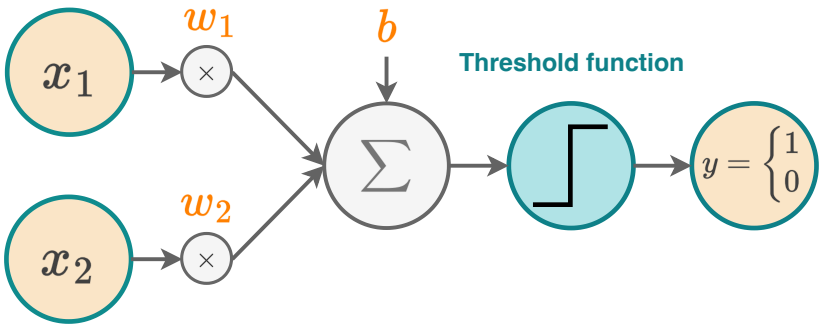


Figure 2.2: A single-layer perceptron with two input features and a single output.

can be represented mathematically as follows:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where x_i is the i^{th} input feature, w_i is the weight associated with the i^{th} input, b is the bias term, and y is the output of the perceptron. The bias term is a constant value that is added to the weighted sum of the inputs.
The weighted sum $\sum_{i=1}^{n} w_i x_i$ can be written as $W.x$, where $W \in \mathbb{R}^{p \times n}$ and $x \in \mathbb{R}^{n \times 1}$ are, respectively, the weight matrix and input vector, with p being the number of neurons in the layer ($p = 1$ in this case) and n being the number of input features ($n = 2$ in this case).

The activation function is a non-linear function that help the model to represent non-linear complex relationships between the inputs and outputs. The most common activation function is the *sigmoid* function, which is defined as follows:

$$\sigma(x) \triangleq \frac{1}{1 + e^{-x}} \tag{2.2}$$

The *sigmoid* function is bounded between 0 and 1, making it suitable for binary classification tasks. Other activation functions, such as the hyperbolic tangent function, can also be used for this purpose.

2.3 Multi-layer perceptron (MLP)

The MLP is a feedforward neural network consisting of multiple layers of neurons, each of which is connected to the neurons in the previous layer. The first layer is the input layer, which receives the input features. The last layer is the output layer, which produces the output of the network. The layers in between are known as hidden layers. The neurons in each layer are connected to the neurons in the previous layer by a set of parameters. The output of each neuron is determined by the weighted sum of the inputs, which is passed through an activation function. The output of the previous layer is the input to the next layer, until the output layer. A two-layer MLP example is shown in Figure 2.3 and can be represented mathematically as follows:

$$y = \sigma(\boldsymbol{W}^{(2)}(\boldsymbol{W}^{(1)}x + b^{(1)}) + b^{(2)}) \tag{2.3}$$

where x is the input vector, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ are the weight matrices, $b^{(1)}$ and $b^{(2)}$ are the bias vectors, and σ is an activation function.

A generalised representation form of the MLP can be written as a composition of

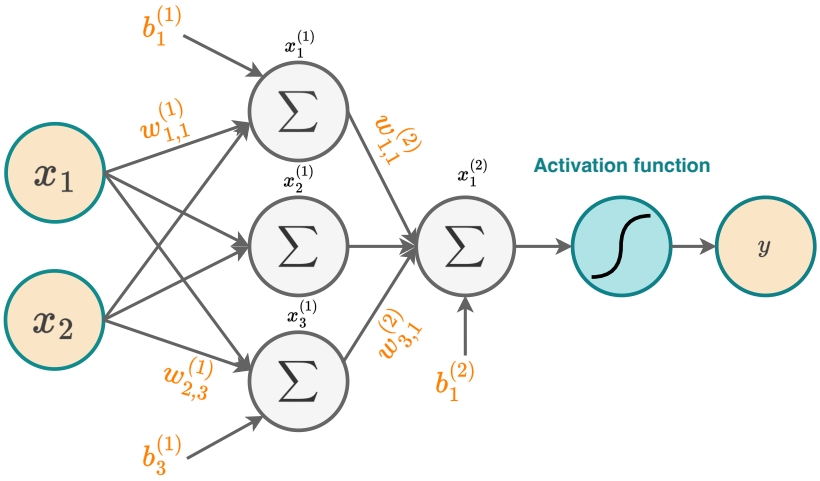


Figure 2.3: A two-layer MLP with two input features, three hidden neurons, and a single output.

linear transformations and activation functions as follows,

$$y = \sigma^{(L)} \circ \mathcal{A}^{(L)} \circ \sigma^{(L-1)} \circ \mathcal{A}^{(L-1)} \circ \cdots \circ \sigma^{(1)} \circ \mathcal{A}^{(1)}(x) \tag{2.4}$$

where $\sigma^{(l)}$ is the activation function of the l^{th} layer, $\mathcal{A}^{(l)}(x) = \boldsymbol{W}^{(l)}x + b^{(l)}$ is the linear transformation of the l^{th} layer, and L is the number of layers in the network. Note that the identity function can be used as an activation function.

The possibility of training a neural network with multiple layers was at the origin of the term "DL", generally attributed to Dechter [1986]. The following sections introduce the training process of neural networks.

2.4 Loss functions

The training process involves adjusting the weights and biases of the neural network to minimise a loss function. This loss function measures the difference between the predicted output and the actual output and can be expressed as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\mathcal{F}\left(\boldsymbol{x}_i; \boldsymbol{\theta}\right), \hat{\boldsymbol{y}}_i\right) \tag{2.5}$$

where $\boldsymbol{\theta}$ is the set of parameters of the neural network (weights and biases), $\mathcal{F}$ is the neural network function, $\boldsymbol{x}_i$ is the i^{th} input and $\hat{\boldsymbol{y}}_i$ is the i^{th} output. Note that throughout this manuscript, $\hat{y}$ denotes the estimator of the target variable y. Here, N represents the number of training samples considered at each iteration, also known as the batch size, and $\mathcal{L}$ is the function that measures the model error. Among the most common loss functions are the mean-squared error (MSE) and the mean absolute error (MAE).

The MSE is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right) - \hat{\boldsymbol{y}}_i\right)^2 \tag{2.6}$$

and the MAE is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left|\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right) - \hat{\boldsymbol{y}}_i\right| \tag{2.7}$$

The process of adjusting the weights and biases of the neural network to minimise the loss is called optimisation. The optimisation process is performed using gradient descent algorithms, which are discussed in the following section.

2.5 Gradient descent optimisation methods

In order to minimise the loss function of a DL model, the goal is to find the optimal values for the model parameters. This can be achieved by computing the loss function gradient with respect to the model parameters and adjusting the parameters in the opposite direction of the gradient. This process is known as gradient descent.

2.5.1 Gradient descent

The gradient descent algorithm is an optimisation method that iteratively updates the variables of a function by moving in the direction of the negative gradient of the function. The algorithm is based on the observation that the gradient at a point indicates the direction of the steepest ascent of the function. Thus, by moving in the opposite direction, one can minimize the function. In the following, we present a mathematical formulation of the gradient descent algorithm.

Let $\mathbb{E}$ be a Hilbert space with the inner product denoted by $\langle\cdot,\cdot\rangle$ and the associated norm denoted by $\|\cdot\|$. Consider a derivable function $f : \mathbb{E} \to \mathbb{R}$. The gradient of f at $x \in \mathbb{E}$ is denoted by $\nabla f(x)$. For a small $\delta x \in \mathbb{E}$, the first-order Taylor expansion of f around x is given by:

$$f(x + \delta x) = f(x) + \langle \nabla f(x), \delta x\rangle + o(\|\delta x\|),$$

where $o(\|\delta x\|)$ represents a function of δx such that $\lim_{\|\delta x\|\to 0} \frac{o(\|\delta x\|)}{\|\delta x\|} = 0$.

To minimize f, we seek δx that decreases $f(x)$. Specifically, we want to minimize the linear approximation:

$$\langle \nabla f(x), \delta x\rangle,$$

subject to $\|\delta x\| \leq 1$. This leads to the optimisation problem:

$$\min_{\|d\|\leq 1} \langle \nabla f(x), d \rangle, \tag{2.8}$$

assuming that $\nabla f(x) \neq 0$. If $\nabla f(x) = 0$, the function is already at a local minimum.

The optimisation problem in Eq. (2.8) have as a solution $d = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$[1], which points in the opposite direction of the gradient. Thus, the opposite direction of the gradient is a direction of descent. The maximum decrease possible on the unit sphere is given by:

$$\begin{aligned} f(x+d) - f(x) &= \left\langle \nabla f(x), -\frac{\nabla f(x)}{\|\nabla f(x)\|} \right\rangle + o(\left\| \frac{\nabla f(x)}{\|\nabla f(x)\|} \right\|) \\ &= -\|\nabla f(x)\| + o(1). \end{aligned}$$

At each iteration t, the gradient descent algorithm updates the variable x as follows:

$$x^{(t+1)} = x^{(t)} - \frac{\nabla f(x^{(t)})}{\|\nabla f(x^{(t)})\|},$$

Generally, we consider small research regions around $x^{(t)}$ of radius $\alpha > 0$, the same reasoning as above leads to the update rule:

$$x^{(t+1)} = x^{(t)} - \alpha \nabla f\left(x^{(t)}\right).$$

This α is also known as the step size, or learning rate in the context of DL.

This forms the foundation of the gradient descent algorithm, as introduced by Cauchy [1847]. In the following sections, we will explore common variants of gradient descent used in DL.

2.5.2 Stochastic gradient descent (SGD)

In the context of neural networks, the loss function is a function of the model parameters. These parameters are adjusted iteratively during the training using optimisation algorithms. Using the gradient descent algorithm, the model parameters are updated as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{2.9}$$

where $\boldsymbol{\theta}^{(t)}$ is the set of model parameters at iteration t, α is the learning rate, and $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)})$ is the gradient of the loss function with respect to the model parameters and

[1] By the Cauchy-Schwarz inequality, the minimum value of this problem is $\frac{-\|\nabla f(x)\|}{\|\nabla f(x)\|}$, which is achieved by d

computed as follows:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \frac{1}{N}\sum_{i=1}^{N}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)};\boldsymbol{x}_i,\boldsymbol{y}_i) \tag{2.10}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are the input and output of the i^{th} training sample, respectively. The learning rate is a hyperparameter that controls the step size of the gradient descent algorithm. A high learning rate can cause the algorithm to diverge or fail to converge, while a low learning rate can result in slow convergence. The learning rate is generally chosen based on the problem at hand and the size of the dataset. The gradient descent algorithm is repeated until the loss function converges to a minimum.

In practice, the gradient descent algorithm is computationally expensive when the dataset is large, as it requires computing the gradient of the loss function for all training samples at each iteration. To address this issue, the stochastic gradient descent (SGD) algorithm was introduced. It is based on the idea of considering the gradient of the loss over a small random subset $\mathcal{B}$of size $m \ll N$, called *stochastic gradient*, as an estimator of the full gradient, as follows:

$$\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)}) = \frac{1}{m}\sum_{i\in\mathcal{B}}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)};\boldsymbol{x}_i,\boldsymbol{y}_i). \tag{2.11}$$

The stochastic gradient is considered an unbiased estimator of the full gradient, i.e.,

$$\mathbb{E}\left[\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)})\right] = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)}).$$

The SGD algorithm is expressed as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{2.12}$$

This simple gradient descent has some known limitations, such as the sensitivity to the learning rate. Indeed, a small learning rate can lead to a slow but precise convergence in the best scenario, but it can also lead to the algorithm getting stuck in a local minimum. On the other hand, a large learning rate may accelerate the convergence, but it can stop the algorithm from converging to the minimum, as it may oscillate around it.

2.5.3 SGD with momentum

To address the limitations of the SGD algorithm, discussed in Section 2.5.2, Polyak [1964] introduced the momentum method, which adds a fraction of the previous update

to the current update. The momentum method can be implemented as follows:

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \mu \boldsymbol{v}^{(t)} + (1-\mu)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \alpha \boldsymbol{v}^{(t+1)} \end{aligned} \tag{2.13}$$

where α is the learning rate, $\boldsymbol{v}^{(t)}$ is the velocity vector, and μ is the momentum parameter. This is the implementation considered in Liu et al. [2020] and in PyTorch. Another implementation which is considered in Sutskever et al. [2013] and in TensorFlow is the following:

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \mu \boldsymbol{v}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} + \boldsymbol{v}^{(t+1)} \end{aligned} \tag{2.14}$$

The term 'momentum' in μ stems from its role in helping the algorithm gather momentum in the descent direction, facilitating escape from local minima and saddle points. μ characterises the resistance to change in the direction of the gradient and is usually set to a value close to 1. The momentum method is also known to reduce the oscillations in the convergence path.

2.5.4 Nesterov accelerated gradient

A very well-known variant of the momentum method (see Section 2.5.3) is the Nesterov accelerated gradient (NAG) [Nesterov, 1983], which computes the gradient at the point $\boldsymbol{\theta}^{(t)} + \mu \boldsymbol{v}^{(t)}$ instead of $\boldsymbol{\theta}^{(t)}$. This method can be expressed as follows:

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \mu \boldsymbol{v}^{(t)} + (1-\mu)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)} + \mu \boldsymbol{v}^{(t)}) \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \alpha \boldsymbol{v}^{(t+1)}, \end{aligned} \tag{2.15}$$

This method is known to converge faster than the momentum method because it anticipates the gradient at the next point. This anticipation allows the algorithm to adjust the velocity vector more accurately, leading to a more precise convergence.

This idea of computing the gradient at an anticipated point is also used in the higher order time integration methods, such as the second order Runge–Kutta method (see Eq. (3.14)).

2.5.5 AdaGrad

AdaGrad [Duchi et al., 2011] is another optimisation algorithm that adapts the learning rate for each parameter based on historical gradients. The idea behind AdaGrad is to decrease the learning rate for parameters that have accumulated large gradients and

vice versa. This method is expressed as follows:

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \boldsymbol{v}^{(t)} + \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)})\right)^2 \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \frac{\alpha}{\sqrt{\boldsymbol{v}^{(t+1)} + \epsilon}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}), \end{aligned} \tag{2.16}$$

where ϵ is a small constant added to the denominator to prevent division by zero and $\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)})\right)^2$ and $\sqrt{\boldsymbol{v}^{(t+1)} + \epsilon}$ are element-wise operations. AdaGrad balances the learning rate for each parameter based on historical gradients, which can be useful in some cases. However, AdaGrad has been shown to have limitations in practice, such as a decaying learning rate that can become too small before reaching the minimum point. This limitation led to the development of other optimisation algorithms, such as RMSprop and Adam.

2.5.6 RMSprop

Unlike AdaGrad, discussed in Section 2.5.5 which uses the sum of the squared gradients to adjust the learning rate, another method called Root Mean Square Propagation (RMSprop) uses a moving average of the squared gradients to adjust the learning rate for each parameter, with a decaying factor controlling the influence of the historical gradients. The RMSprop algorithm is expressed as follows:

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \mu \boldsymbol{v}^{(t)} + (1 - \mu) \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)})\right)^2 \\ \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \frac{\alpha}{\sqrt{\boldsymbol{v}^{(t+1)} + \epsilon}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \end{aligned} \tag{2.17}$$

This unpublished method, developed in the team of Geoffrey Hinton, showed better performance than AdaGrad in practice, as it prevents the learning rate from decaying too quickly. However, RMSprop still has limitations, such as the need to manually adjust the learning rate and the decaying factor.

2.5.7 Adam

Adam [Kingma and Ba, 2015] (Adaptive Moment Estimation) is an optimisation algorithm that combines the advantages of RMSprop (see Section 2.5.6) and the momentum methods (see Section 2.5.3, Section 2.5.4). Adam uses the moving average of the gradients and the squared gradients to adjust the learning rate for each parameter. It is

expressed as follows:

$$\begin{aligned}
\boldsymbol{m}^{(t+1)} &= \mu_1 \boldsymbol{m}^{(t)} + (1-\mu_1)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)}) \\
\boldsymbol{v}^{(t+1)} &= \mu_2 \boldsymbol{v}^{(t)} + (1-\mu_2)\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(t)})\right)^2 \\
\hat{\boldsymbol{m}}^{(t+1)} &= \frac{\boldsymbol{m}^{(t+1)}}{1-\mu_1^{t+1}} \\
\hat{\boldsymbol{v}}^{(t+1)} &= \frac{\boldsymbol{v}^{(t+1)}}{1-\mu_2^{t+1}} \\
\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \frac{\alpha}{\sqrt{\hat{\boldsymbol{v}}^{(t+1)}+\epsilon}}\hat{\boldsymbol{m}}^{(t+1)}
\end{aligned} \tag{2.18}$$

where $\boldsymbol{m}^{(t)}$ and $\boldsymbol{v}^{(t)}$ are the first and second moments of the gradients, respectively, μ_1 and μ_2 are the exponential decay rates for the first and second moments. The Adam algorithm has been shown to be effective in practice and is widely used in training deep neural networks.

The choice of the optimisation algorithm depends on the problem at hand and the characteristics of the dataset. In practice, it is common to use Adam as a default optimisation algorithm due to its robustness and ease of use. However, it is recommended to experiment with different optimisation algorithms and learning rates to find the best combination for a given problem.

2.6 Back-propagation

The back-propagation algorithm, introduced by Rumelhart et al. [1986], is a method for computing the gradient of the loss function. The algorithm consists of two phases: the forward and the backward pass. In the forward pass, the input is propagated through the network layers to compute the output as follows:

$$\begin{aligned}
\boldsymbol{z}^{(1)} &= \sigma^{(1)}(\mathcal{A}^{(1)}(\boldsymbol{x})) \\
&\vdots \\
\boldsymbol{z}^{(L-1)} &= \sigma^{(L-2)}(\mathcal{A}^{(L-2)}(\boldsymbol{z}^{(L-2)})) \\
\hat{\boldsymbol{y}} &= \sigma^{(L-1)}(\mathcal{A}^{(L-1)}(\boldsymbol{z}^{(L-1)}))
\end{aligned} \tag{2.19}$$

where $\boldsymbol{x}$ is the input vector, $\hat{\boldsymbol{y}}$ is the output vector, $\boldsymbol{z}^{(l)}$ is the output of the l^{th} layer, $\mathcal{A}^{(l)}$ is the linear transformation of the l^{th} layer, and $\sigma^{(l)}$ is the activation function of the l^{th} layer.

In order for the model to learn, the gradient of the loss function with respect to the model parameters should be computed, as we have seen in Section 2.5.

Let's consider the single-layer MLP example in Figure 2.2 with two input features,

one output, and a *sigmoid* activation function instead of the threshold function. The MLP can be represented as follows:

$$\begin{aligned} \boldsymbol{z}^{(1)} &= \mathcal{A}^{(1)}(\boldsymbol{x}) = \boldsymbol{W}^{(1)}\boldsymbol{x} + b^{(1)} \\ \hat{\boldsymbol{y}} &= \sigma^{(1)}(\boldsymbol{z}^{(1)}) = \sigma^{(1)}(\boldsymbol{W}^{(1)}\boldsymbol{x} + b^{(1)}) \end{aligned} \tag{2.20}$$

where $\boldsymbol{x} = [x_1, x_2]^T$ is the input vector, $\hat{\boldsymbol{y}} = [y_1]$ is the output vector, $\boldsymbol{W}^{(1)} = [w_1, w_2]$ is the weight vector, $b^{(1)}$ is the bias term, and $\sigma^{(1)}$ is the *sigmoid* activation function. The loss function is the MSE, which can be written as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right) - \boldsymbol{y}_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right)^2 \tag{2.21}$$

where $\boldsymbol{\theta} = [\boldsymbol{W}^{(1)}, b^{(1)}]$ is the set of model parameters, N is the number of training samples, and $\boldsymbol{y}_i$ is the target value of the i^{th} sample. In the case of multidimensional output, the $\left(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right)^2$ term is replaced by $\|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|^2$.

The gradient of the loss function with respect to the model parameters can be computed as follows:

$$\begin{aligned} \nabla_{\boldsymbol{W}^{(1)}}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{W}^{(1)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\hat{\boldsymbol{y}}}\frac{\partial\hat{\boldsymbol{y}}}{\partial\boldsymbol{z}^{(1)}}\frac{\partial\boldsymbol{z}^{(1)}}{\partial\boldsymbol{W}^{(1)}} \\ \nabla_{b^{(1)}}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial b^{(1)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\hat{\boldsymbol{y}}}\frac{\partial\hat{\boldsymbol{y}}}{\partial\boldsymbol{z}^{(1)}}\frac{\partial\boldsymbol{z}^{(1)}}{\partial b^{(1)}} \end{aligned} \tag{2.22}$$

this method is known as the chain rule [Leibniz, 1920], where $\partial\mathcal{L}(\boldsymbol{\theta})/\partial\hat{\boldsymbol{y}}$ is the gradient of the loss function with respect to the output, and it's given by the following formula:

$$\frac{\partial\mathcal{L}_{MSE}(\boldsymbol{\theta})}{\partial\hat{\boldsymbol{y}}} = \frac{\partial}{\partial\hat{\boldsymbol{y}}}\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right)^2 = 2\left(\hat{\boldsymbol{y}} - \boldsymbol{y}\right) \tag{2.23}$$

For the MAE loss function, the gradient is given by the following formula:

$$\frac{\partial\mathcal{L}_{MAE}(\boldsymbol{\theta})}{\partial\hat{\boldsymbol{y}}} = \frac{\partial}{\partial\hat{\boldsymbol{y}}}\left|\hat{\boldsymbol{y}} - \boldsymbol{y}\right| = \text{sign}(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{2.24}$$

$\partial\hat{\boldsymbol{y}}/\partial\boldsymbol{z}^{(1)}$ is the gradient of the output with respect to the output of the first layer and it's given by the following formula[2]:

$$\frac{\partial\hat{\boldsymbol{y}}}{\partial\boldsymbol{z}^{(1)}} = \frac{\partial\sigma^{(1)}(\boldsymbol{z})}{\partial\boldsymbol{z}} = \hat{\boldsymbol{y}} \odot (1 - \hat{\boldsymbol{y}}). \tag{2.25}$$

where $\odot$ is the element-wise product, also known as the Hadamard product. For the

[2]The *sigmoid* derivative is given by $\sigma'(x) = \left(\frac{1}{1+e^{-x}}\right)' = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)\cdot(1-\sigma(x))$.

ReLU activation function, the gradient is given by the following formula:

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}^{(1)}} = \frac{\partial \text{ReLU}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \begin{cases} 1 & \text{if } \boldsymbol{z} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

Next, $\partial \boldsymbol{z}^{(1)}/\partial \boldsymbol{W}^{(1)}$ and $\partial \boldsymbol{z}^{(1)}/\partial b^{(1)}$ are the gradients of the output of the first layer with respect to the model parameters. The formulas for these intermediate gradients are theoretically known and computed as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{W}^{(1)}} &= \left.\frac{\partial \left(\boldsymbol{W}\boldsymbol{x} + b\right)}{\partial \boldsymbol{W}}\right|_{\boldsymbol{W}=\boldsymbol{W}^{(1)}} = \boldsymbol{x} \\ \frac{\partial \boldsymbol{z}^{(1)}}{\partial b^{(1)}} &= \left.\frac{\partial \left(\boldsymbol{W}\boldsymbol{x} + b\right)}{\partial b}\right|_{b=b^{(1)}} = 1 \end{aligned} \tag{2.27}$$

Now, we have all the intermediate gradients to compute the gradient of the loss function with respect to the model parameters. The gradients are computed as follows:

$$\begin{aligned} \nabla_{\boldsymbol{W}^{(1)}} \mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} 2\left(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right) \odot \hat{\boldsymbol{y}}_i \odot \left(1 - \hat{\boldsymbol{y}}_i\right) \odot \boldsymbol{x}_i \\ \nabla_{b^{(1)}} \mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} 2\left(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\right) \odot \hat{\boldsymbol{y}}_i \odot \left(1 - \hat{\boldsymbol{y}}_i\right) \odot 1, \end{aligned} \tag{2.28}$$

where $\hat{\boldsymbol{y}} = \sigma^{(1)}(\boldsymbol{z}_i^{(1)}) = \sigma^{(1)}(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + b^{(1)})$.

This process of computing the gradient of a function using the chain rule and the exact mathematical formulas for the intermediate gradients is known as automatic differentiation. In order to avoid redundant computations of the same intermediate gradients, the gradients can be computed iteratively, starting from the output layer and moving backward to the input layer. This process is known as reverse-mode automatic differentiation, reverse accumulation, or more commonly in the context of neural networks, *backpropagation*. Efficient backpropagation implementations are provided by key libraries such as TensorFlow [Abadi et al., 2016], PyTorch [Paszke et al., 2019], and JAX [Frostig et al., 2018]. These libraries represent the chain of operations using a graph structure, called a computation graph. Each node in the graph represents an operation, and the edges represent the variables involved in the operation. Figure 2.4 shows an example of a computation graph. TensorFlow uses a static computation graph representing the operations of the neural network, while PyTorch uses a dynamic one. The static computation graph is constructed before the training process, storing the operations and the variables involved in the computation. This allows for more optimisations and better performance. The dynamic computation graph is built on the fly as the operations are executed at each iteration. This allows for more flexibility and ease

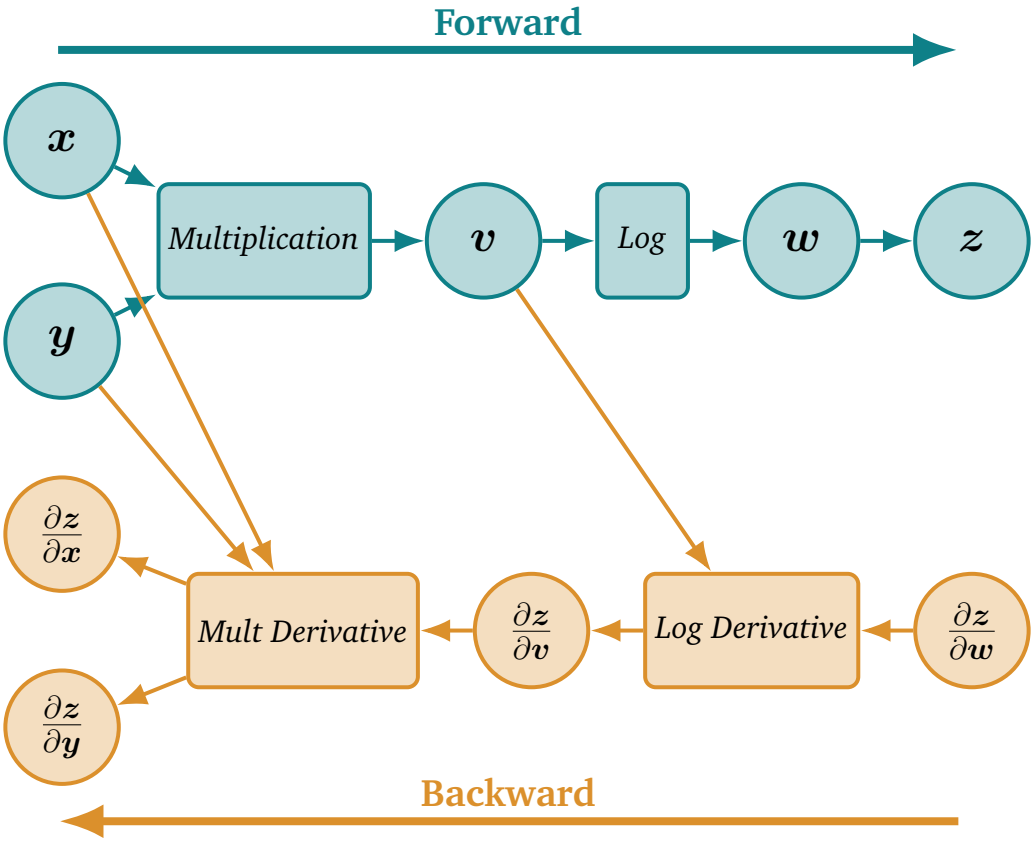


Figure 2.4: The computation graph of $f(x, y) = \log(x * y)$.

of use, which means that the size of the inputs can be changed at every iteration, and control flow operations such as loops with variable lengths and conditional statements can be used.

2.7 Universal approximation theorem

The universal approximation theorem states that a feedforward neural network with a single hidden layer (i.e. the layer between the input and output layers) containing a finite number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$ [Hornik et al., 1989]. This theorem provides a theoretical basis for the use of neural networks as universal function approximators. However, it does not provide any insight into the number of neurons required to approximate a given function.

Later studies have proposed bounds on the number of neurons required to approximate a function with a given error [e.g. Pinkus, 1999, Yarotsky, 2017, Kidger and Lyons, 2020]. These bounds are based on the width (the number of neurons) of the hidden layer. The following theorems provide bounds on the width of the hidden layer required to approximate a function with a given error.

We assume $K \subset \mathbb{R}^d$ is a compact set.

Theorem [Pinkus, 1999]. Let $f : K \to \mathbb{R}$ with $d = 1$, be a continuous function. Then given an $\epsilon > 0$ and a non-polynomial continuous activation σ, there exists an MLP, $\mathcal{F}_{\epsilon,H,L}$, with a single hidden layer $(L = 1)$ of width H, such that

$$\max_{\boldsymbol{x}\in K} |\mathcal{F}_{\epsilon,H,L,\sigma}(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})| \leq \epsilon.$$

Theorem [Kidger and Lyons, 2020]. Let $f : K \to \mathbb{R}^D$ be a continuous vector-valued function, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be any nonaffine continuous function which is continuously derivable at at least one point, with nonzero derivative at that point. Then given an $\epsilon > 0$, there exists an MLP, $\mathcal{F}_{\epsilon,L,H,\sigma}$, with L hidden layers, each having width $H \geq d + D + 2$, such that

$$\max_{\boldsymbol{x}\in K} \|\mathcal{F}_{\epsilon,L,H,\sigma}(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{f}(\boldsymbol{x})\| \leq \epsilon.$$

To observe the first theorem [Pinkus, 1999] in practice, let's consider the following functions:

$$f_n(x) = \sin^{2n}(x) + \cos(x) \tag{2.29}$$

where n is a positive integer. Theses functions are C^∞ for all n and their curves have a cat-like shape.

In the following experiment, we consider $n = 14$. Figure 2.5 shows the approximation of the function f_{14} with a single hidden layer MLP with different numbers of neurons, referred to as H, and a ReLU activation function. We observe that the approximation improves as the number of neurons in the hidden layer increases until the approximation fully captures the function with $H = 32$. However, this doesn't mean that the approximation will be accurate outside the training range. The approximation is only guaranteed to be accurate within the training range, which is expected.

To observe the second theorem [Kidger and Lyons, 2020] in practice, we consider more complex functions, e.g. images. Here, three images are considered, and the goal is to approximate each image with an analytical function of the form

$$\begin{aligned} f :&[0,1] \times [0,1] \to [0,1]^3 \\ &(x, y) \mapsto (R(x, y), G(x, y), B(x, y)) \end{aligned} \tag{2.30}$$

where $R(x, y)$, $G(x, y)$, and $B(x, y)$ are the red, green, and blue values of the pixel at a position (x, y) in the image, respectively.

We implemented 16 hidden layers MLP to approximate the function f with different numbers of neurons in the hidden layers, referred to as H, and $H \geq 2 + 3 + 2 = 7$ to satisfy the theorem. The ReLU activation function is used after each hidden layer. The approximation functions learned by the MLP for each image and different numbers of neurons in the hidden layers are shown in Figure 2.6.

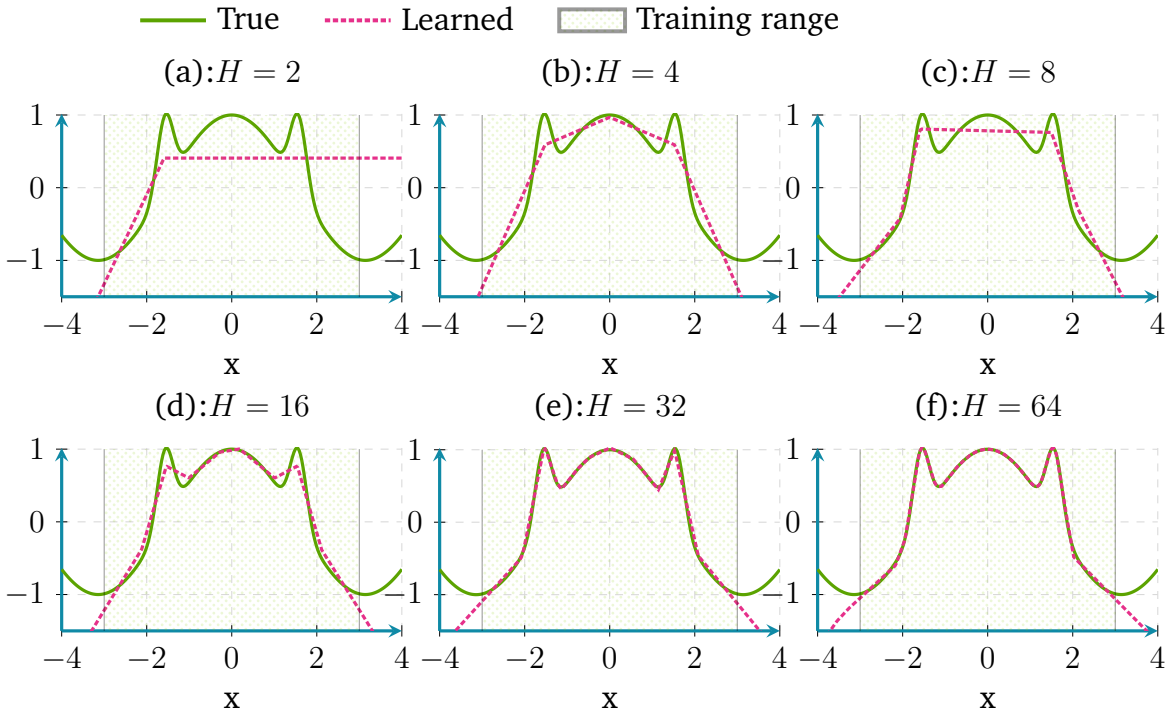


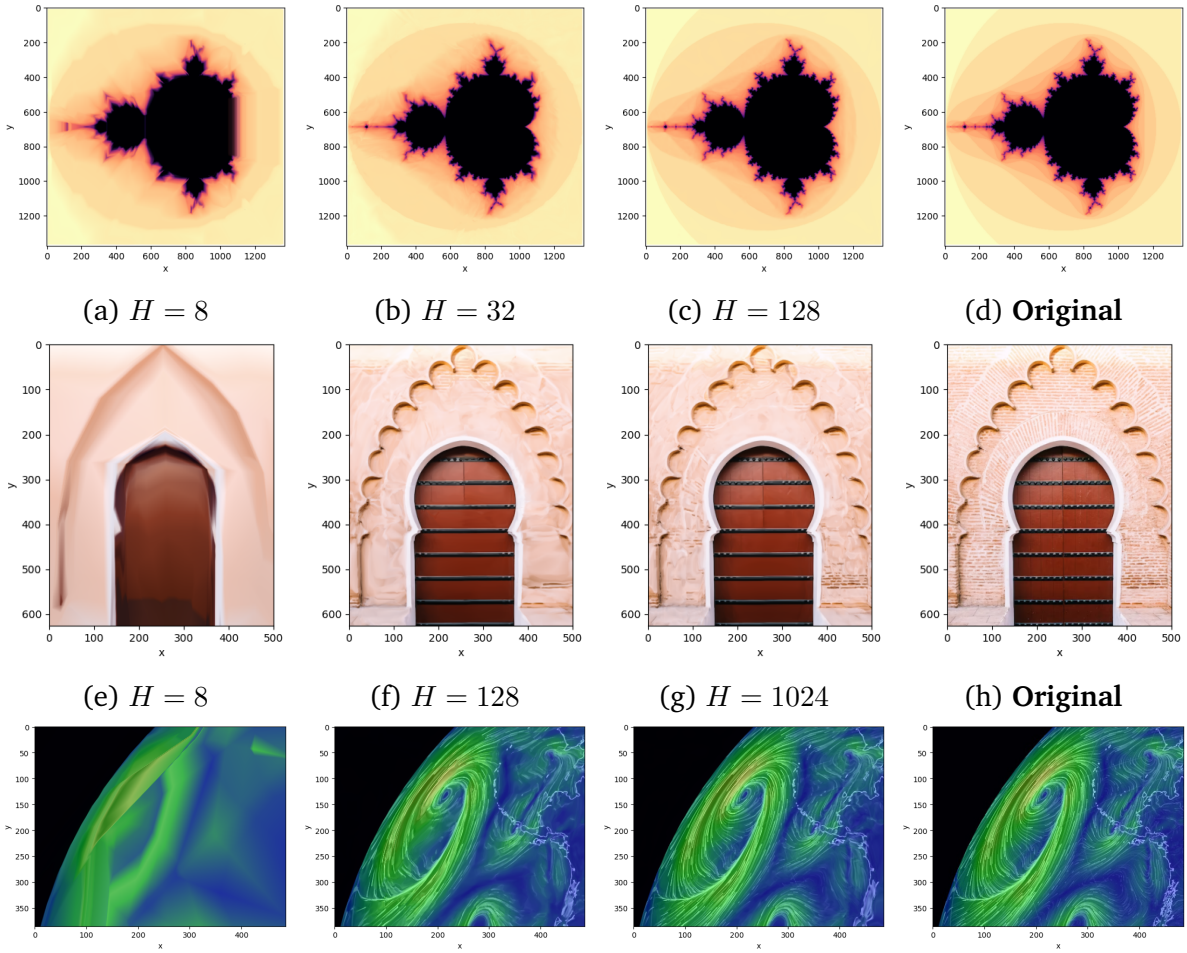
Figure 2.5: Approximation of the function f_{14} with a single hidden layer MLP. For training, we used a learning rate starting from 0.001, a batch size of 256, and 20 epochs. The neural networks were trained using data points sampled from the interval $[-3, 3]$ (training range).

As expected, the approximation improves with the width of the network. We observe that with $H \geq d + D + 2$, we are able to have a relevant approximation of the two first images, which is consistent with the theorem, but the third one, which is more complex, shows that it's problem dependent, which is still consistent with the theorem. The condition about the width of the hidden layer is a necessary condition, but not a sufficient one. While challenging, it would be valuable to establish an upper bound on the number of neurons and layers required to achieve a given approximation error.

Interestingly, once the network has learned the image, we have access to an analytical function approximating the image. This function can be used, for example, to generate a higher resolution image as shown in Figure 2.7, as we are no longer limited by the initial discretisation of the image.

2.8 Regularisation

Neural networks can be easily complex and present a high capacity to learn complex patterns from the data. However, this high capacity can lead to overfitting, where the model learns the noise in the training data and fails to generalise to new data, especially when the training data is limited. Figure 2.8 shows an illustrative overfitting situation.

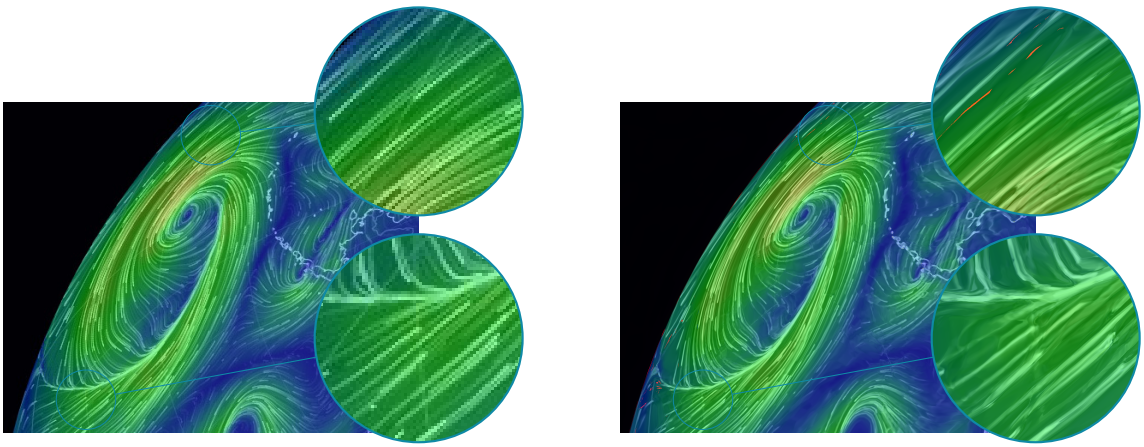


(a) $H = 8$ (b) $H = 32$ (c) $H = 128$ (d) **Original**

(e) $H = 8$ (f) $H = 128$ (g) $H = 1024$ (h) **Original**

(i) $H = 8$ (j) $H = 128$ (k) $H = 1024$ (l) **Original**

Figure 2.6: Approximation of three images with three hidden layers MLP. For training, we used a learning rate starting from 0.001, a batch size of 256 and 10 epochs.



(a) Original image (486×386) (b) Generated high-res image (4860×3860)

Figure 2.7: Generating a high resolution image using an MLP of 16 hidden layers of 1024 neurons each.

One of the reasons of overfitting is the model parameters taking large values, this can

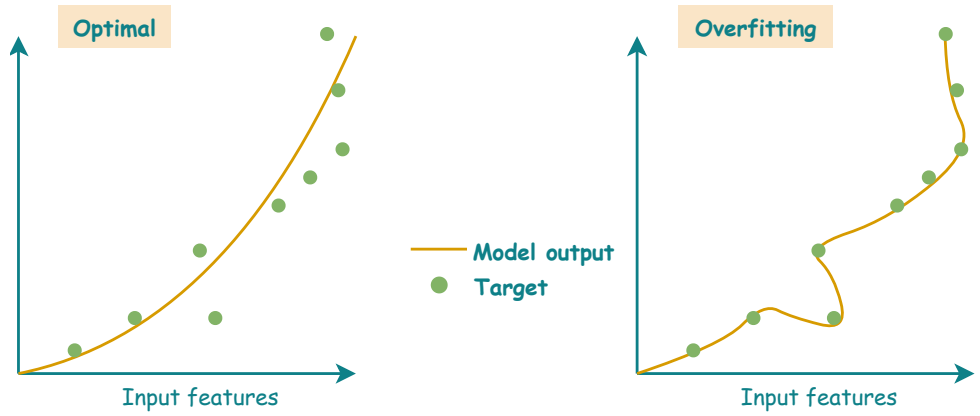


Figure 2.8: Overfitting illustration.

be mitigated by adding a regularisation term to the loss function. The most common terms are the L1 and L2 regularisation, which are defined as follows:

$$\mathcal{L}_{\text{L1}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \tag{2.31}$$

$$\mathcal{L}_{\text{L2}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \tag{2.32}$$

where λ is the regularisation parameter, $\|\boldsymbol{\theta}\|_1$ is the L1 norm of the model parameters, and $\|\boldsymbol{\theta}\|_2$ is the L2 norm of the model parameters. The L1 norm is defined as $\|\boldsymbol{\theta}\|_1 = \sum_{i=1}^{n} |\theta_i|$, and the L2 norm is defined as $\|\boldsymbol{\theta}\|_2 = \sqrt{\sum_{i=1}^{n} \theta_i^2}$. The L1 regularisation is also known as Lasso, and the L2 regularisation is also known as Ridge. The regularisation parameter λ controls the strength of the regularisation. A large value of λ leads to a stronger regularisation, which can help prevent overfitting.

Beyond the regularisation aim, constraints can be added to the loss function for other reasons such as enforcing physical consistency, as shown in Raissi et al. [2017] and will be discussed in Chapter 4.

2.9 Classification and regression

Neural networks can be used for both classification and regression tasks. In the case of regression tasks, the output of the neural network is a continuous value, generally representing a quantity of interest and can be in bounded or unbounded intervals. In the case of classification tasks, the output of the neural network is a categorical value, representing a class or a label. The optimisation algorithms used to train neural networks are designed for continuous optimisation, Hence, the output of the neural network is a continuous value. Hence, the model is trained to predict the probability of each class rather than the class itself. The predicted class is then determined by the class with the

highest probability. To ensure that the output of the neural network is a valid probability distribution, the *softmax* function is used to transform the output of the neural network into a probability distribution over the classes. The softmax function is defined as follows:

$$\text{softmax}(\boldsymbol{z})_i \triangleq \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.33}$$

where $\boldsymbol{z}$ is the output of the neural network, z_i is the i^{th} element of $\boldsymbol{z}$, and K is the number of classes. The softmax function ensures that the output of the neural network is positive and is a valid probability distribution by normalising the output to sum to 1. The predicted class is then determined by the class with the highest probability. The cross-entropy loss is the commonly used loss function for classification tasks. For a binary classification task, the cross-entropy loss is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}_i \log\left(\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)\right) + \left(1-\boldsymbol{y}_i\right)\log\left(1-\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)\right)\right) \tag{2.34}$$

where $\boldsymbol{y}_i$ is the target value of the i^{th} sample (0 or 1), $\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)$ is the output of the neural network after applying the softmax, and N is the number of training samples. For a multi-class classification task, the cross-entropy loss is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\boldsymbol{y}_{i,k}\log\left(\mathcal{F}\left(\boldsymbol{x}_i;\boldsymbol{\theta}\right)_k\right) \tag{2.35}$$

where $\boldsymbol{y}_i$ is the one-hot encoded vector of the i^{th} output. The one-hot encoding is a representation of categorical variables as binary vectors, where each vector has a length equal to the number of categories. The vector has a value of 1 in the position corresponding to the category and 0 in all other positions. For example, a categorical variable with three categories can be represented as follows:

Category	One-hot encoding	Index
A	$[1,0,0]$	0
B	$[0,1,0]$	1
C	$[0,0,1]$	2

. (2.36)

The cross-entropy loss is commonly used for classification tasks because it encourages the model to predict the correct class with high confidence.

2.10 Classical architectures

In this section, we present some classical neural network architectures that are widely used in practice.

2.10.1 Convolutional neural networks

MLPs can be used for diverse tasks, including image classification, time series prediction, and natural language processing. However, they face challenges when dealing with these types of tasks. For example, images are high-dimensional data, and the number of parameters in an MLP grows quadratically with the number of input features, making it difficult to train MLPs on large images. Additionally, MLPs apply pixel-wise operations, which do not capture the spatial structure of the input data, making a slightly shifted image completely different from the original image from the MLP's perspective.

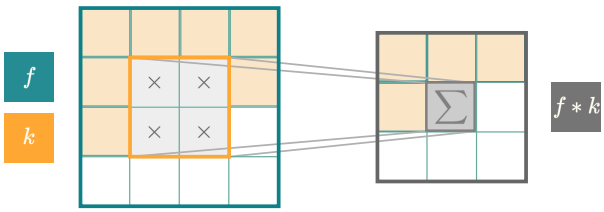


Figure 2.9: Convolution operation using a 2×2 kernel and a 4×4 input feature map.

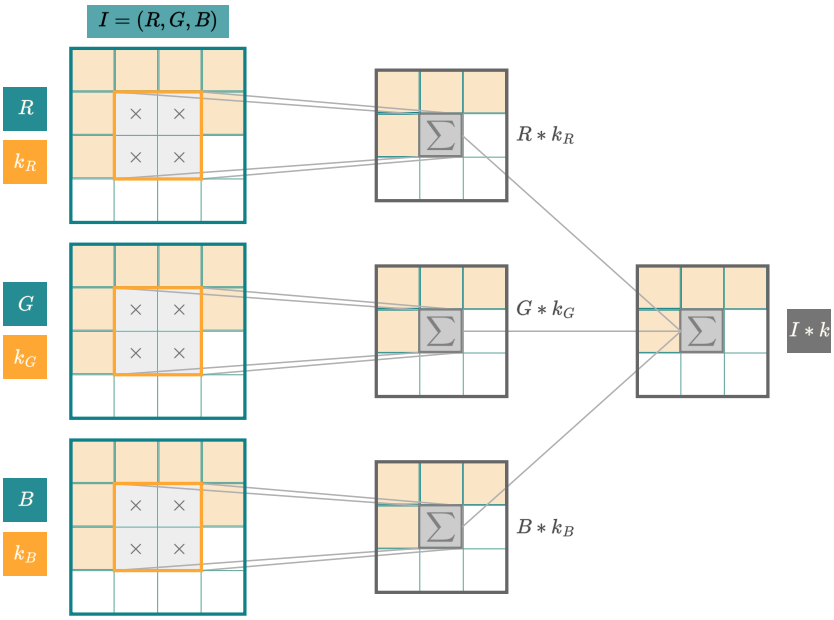


Figure 2.10: How multichannel convolution is applied using a 2×2 kernel on a 4×4 input with 3 channels (e.g. RGB).

Convolutional neural networks are designed to capture the spatial structure of the input data, making them well suited for tasks involving images, videos, and other mul-

tidimensional data. CNNs are based on the concept of convolution, which is a mathematical operation that is defined as follows:

$$(f * k)(t) \triangleq \int_{-\infty}^{\infty} f(\tau)k(t-\tau)\,d\tau \tag{2.37}$$

where f and k are functions, and $*$ is the convolution operator.

The convolution operation is applied to the input data using a set of filters, also known as kernels. Filters are learned during the training process and capture different features of the input data.

In a CNN, the input data is passed through a set of convolutional layers, activation functions, and also fully connected layers (especially in the last layers). Another operation commonly used in CNNs is the *pooling* operation. The pooling operation is used to reduce the dimensionality of the feature maps and to capture the most important features of the input data. The most common pooling operation is the *max-pooling*, which takes the maximum value of a set of values in the feature map.

Figure 2.9 shows the convolution operation using a 2×2 kernel and a 4×4 input feature map.

The convolution operation can be extended to multiple channels, e.g. image with RGB channels. In this case, the kernels have the same number of channels as the input feature map. The convolution operation is applied to each channel of the input feature map, and the results are summed to produce the output feature map (see Figure 2.10). Notice that the output feature map has a size of 3×3, which is smaller than the input feature map. This size reduction is due to the kernel size, as the kernel cannot be applied to the edges of the input. Generally, the output size is given by the formula:

$$\text{Output size} = \text{Input size} - \text{Kernel size} + 1. \tag{2.38}$$

Some tasks require maintaining the input size or that the intermediate feature maps have a specific size. In these cases, padding can be added to the border of the input feature map to ensure that the output feature map has the desired size. The padding is generally added as zeros around the input feature map along each direction. For odd kernel sizes, the padding is generally added symmetrically on both sides and given by the formula:

$$\text{Padding size}[i] = \frac{\text{Kernel size}[i] - 1}{2}. \tag{2.39}$$

where i is the axis (for example, $i = 1$ for the vertical axis and $i = 2$ for the horizontal axis).

For even kernel sizes, it is common to add different padding sizes to the top and bottom/left and right of the input feature map to ensure that the output feature map

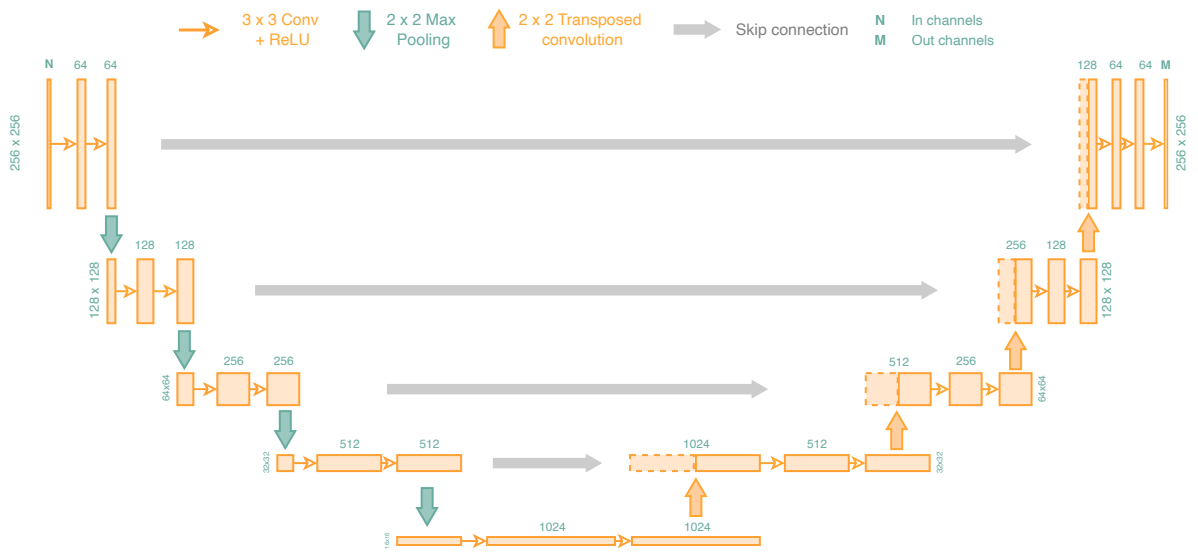


Figure 2.11: The U-Net architecture consists of an encoder part and a decoder part. The encoder part captures the features of the input data, and the decoder part reconstructs the input data from the features captured by the encoder part.

matches the input size. In this case, the padding size is given by the formula:

$$\text{Padding size}[i] = \begin{cases} \frac{\text{Kernel size}[i]}{2} & \text{is added in one side} \\ \frac{\text{Kernel size}[i]}{2} - 1 & \text{is added in the other side} \end{cases} \tag{2.40}$$

where i is the axis (e.g. $i = 1$ for the vertical axis and $i = 2$ for the horizontal axis).

One of the most popular CNN architectures is the U-Net architecture, which is widely used for computer vision tasks. The U-Net architecture introduced by Ronneberger et al. [2015] is based on the concept of an autoencoder, which is a neural network that learns to reconstruct data from a compressed representation, called latent space, which is the bottleneck in Fig. 2.11.

The U-Net architecture has been shown to be effective in capturing the spatial structure of the input data and has been widely used in image segmentation tasks, but also in other tasks such as weather nowcasting [see Ayzel et al., 2020, Berthomier et al., 2020].

2.10.2 Recurrent neural networks (RNN)

RNN are designed to capture the temporal structure of the input data, making them well-suited for tasks involving time series data, natural language processing, and other sequential data. RNNs are based on the concept of recurrence, which is a mathematical operation that is defined as follows:

$$h_t = f(h_{t-1}, x_t) \tag{2.41}$$

where h_t is the output at time t, h_{t-1} is the output at time $t-1$, x_t is the input at time t, and f is the recurrence function. The recurrence function is generally a non-linear function that merges the input and the previous output to produce the current output. The output at time t is then used as the input at time $t+1$, and the process is repeated for the entire sequence (see Figure 2.12). The last output or all the outputs concatenated can be used to make predictions using a fully connected layer. RNNs face challenges

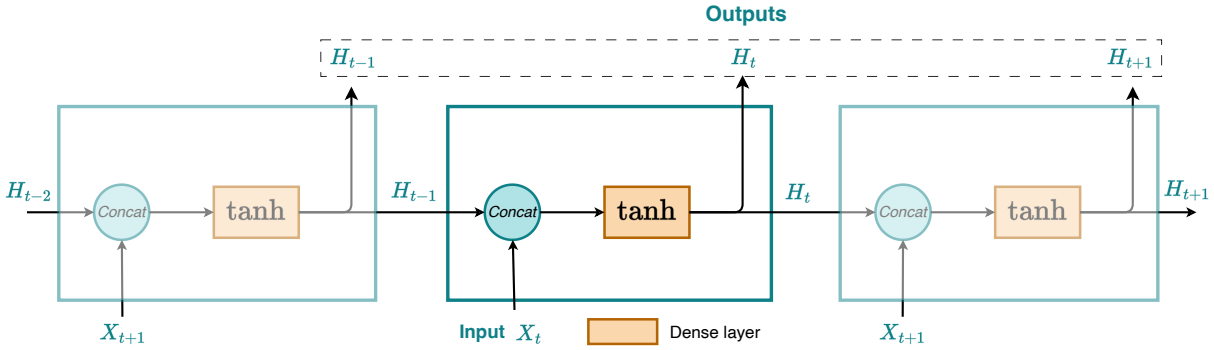


Figure 2.12: An example of a Recurrent Neural Network.

when dealing with long sequences, as the gradients can vanish or explode during the training process. This problem is known as the vanishing gradient problem. Indeed, as explained in Section 2.6, the gradients are computed iteratively starting from the output layer and moving backward to the input layer. The gradients are multiplied at each step, and if these gradients are small (< 1), it results in a very small value of the gradient of the loss function with respect to the model parameters, which can lead to slow convergence or no convergence at all. On the other hand, if the gradients are large (> 1), it results in a very large value of the gradient of the loss function, which can lead to divergence. This problem is commonly referred to as the exploding gradient.

To address the vanishing gradient problem, several RNN variants have been proposed, including LSTM networks [Hochreiter and Schmidhuber, 1997] and GRU networks [Cho et al., 2014]. These networks introduce additional gates that control the flow of information in the network, allowing the network to capture long-range dependencies in the input data and avoid the vanishing gradient problem (see Figure 2.13 for an illustration of an LSTM cell).

RNNs were not the only architectures to suffer from the vanishing gradient problem, CNNs also faced this problem. To address this problem, the ResNet architecture was proposed, as explained in Section 2.10.3.

2.10.3 Residual networks

Some convolutional networks also have suffered from the vanishing gradient problem, especially when the network is deep, i.e. when it has many layers. To address this

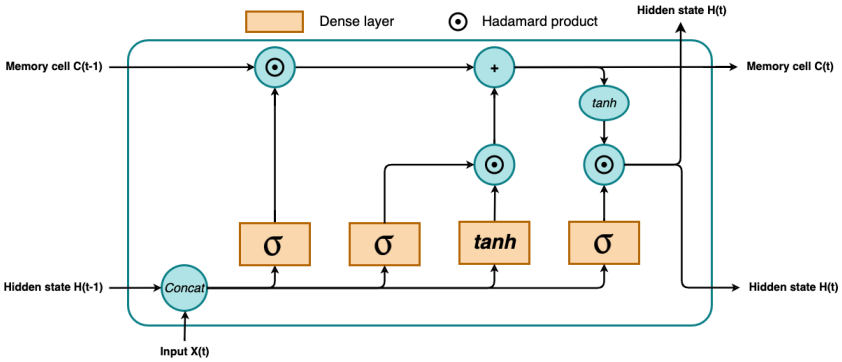


Figure 2.13: Illustration of an LSTM cell: an additional input has been added to the cell compared to a standard RNN cell. This input is used to store the information from the previous time steps. Inside the cell, some of the information is kept, some is forgotten, and some new information is added.

problem, He et al. [2016] proposed the Residual Network (ResNet) architecture. The ResNet architecture introduces a particular building block called the *residual block*. This block uses skip connections, which allow a part of the gradients to flow directly from the output to the input of the network, bypassing the intermediate layers. Figure 2.14 shows the general form of a residual block. This architecture has been successful in

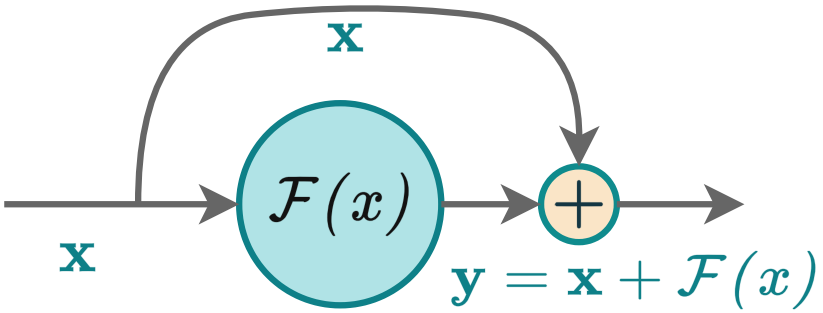


Figure 2.14: Illustration of a residual block.

training very deep networks and has been widely used in image classification tasks.

2.10.4 Transformers

Transformer network is a DL architecture introduced by [Vaswani et al., 2017]. These models have since revolutionised natural language processing (NLP) and other fields due to their ability to handle sequential data without relying on traditional recurrent or convolutional neural network architectures.

The core innovation of transformers is the self-attention mechanism (Fig. 2.15), which allows the model to weigh the importance of different words in a sentence when making predictions. This mechanism captures dependencies regardless of their distance in the sequence, unlike traditional models that are limited by fixed context windows.

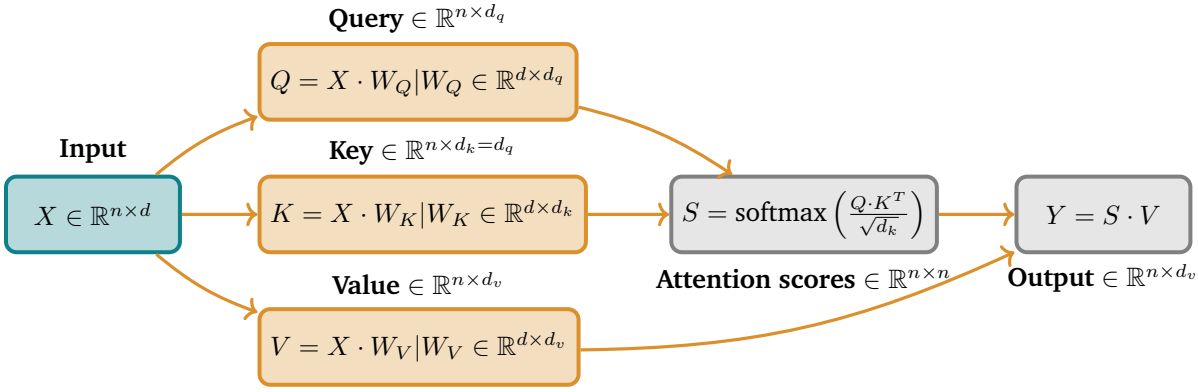


Figure 2.15: The self-attention mechanism works as follows: Let's consider an input vector $X \in \mathbb{R}^{n \times d}$, with n being the number of tokens in the sequence and d the dimension of each token, also known as the embedding dimension. Three vectors are generated using linear transformations of the input: $Q = X \cdot W_Q$, $K = X \cdot W_K$, and $V = X \cdot W_V$. $W_Q \in \mathbb{R}^{d \times d_q}$, and $W_V \in \mathbb{R}^{d \times d_v}$ are learned weights. Attention scores are computed by taking the dot product of $Q \in \mathbb{R}^{n \times d_q}$ and $K^T \in \mathbb{R}^{d_q \times n}$, followed by scaling and applying the softmax function as follows: $S = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)$. The output is then computed as a weighted sum of the values $V \in \mathbb{R}^{n \times d_v}$ using the attention scores: $Y = S \cdot V$.

As it can be noticed on Fig. 2.15, the size of the weight matrices W_Q, W_K, and W_V is independent of the input size n, which allows the model to capture dependencies regardless of the sequence length. This is one of the key advantages of the self-attention mechanism over recurrent-based architectures. Since transformers do not inherently understand the order of sequences (unlike RNNs), positional encodings are added to the input embeddings to provide information about the position of a word in a sentence. To capture different aspects of the relationships between words, transformers use multi-head attention (Fig. 2.16, which involves multiple self-attention operations in parallel. Each head can focus on different parts of the sequence. Other layers, such as layer normalisation and residual connections, are used to stabilise and speed up training. Transformers have been used in a wide range of applications, especially in natural language processing, showing scalability properties that allow them to perform well on large datasets compared to LSTM and GRU models. Transformers were not used for image processing as images contains more data points than sentences. However, recently a study by Dosovitskiy et al. [2021] introduced the Vision Transformer (ViT) model, which applies the transformer architecture to image data. The ViT model divides the image into patches, which are then flattened and passed through the transformer encoder. This ViT model shows promising results and can leverage the scalability of transformers to process large datasets for computer vision tasks.

2.11 Conclusion

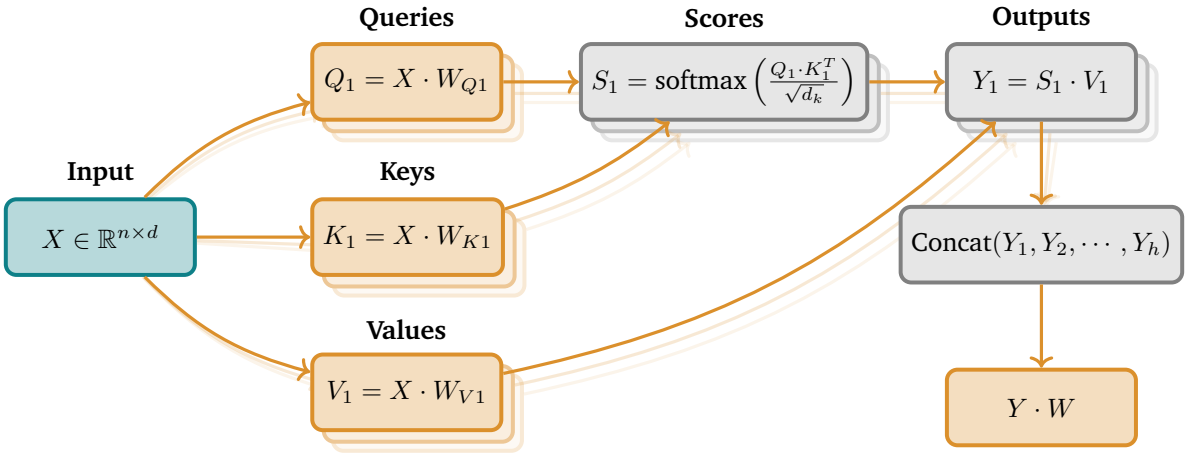


Figure 2.16: Multi-Head Attention mechanism. The input vector $X \in \mathbb{R}^{n \times d}$ is transformed into multiple sets of Query (Q_i), Key (K_i), and Value (V_i) vectors for each attention head $i \in \{1, 2, \cdots, h\}$. The attention scores S_i are computed for each head, and the outputs Y_i are obtained by applying the attention scores to the values. The outputs of all heads are concatenated and projected using a linear transformation.

In this chapter, we have introduced the basic concepts of deep learning, including neural networks, activation functions, loss functions, and optimisation algorithms. We have also presented some classical neural network architectures, including convolutional neural networks, recurrent neural networks, and residual networks. We have also introduced the transformer architecture, which has revolutionised natural language processing and started to be used in other fields such as computer vision and recently in climate science[Nguyen et al., 2023]. In the next chapter, we will present some basics of numerical methods for solving partial differential equations.

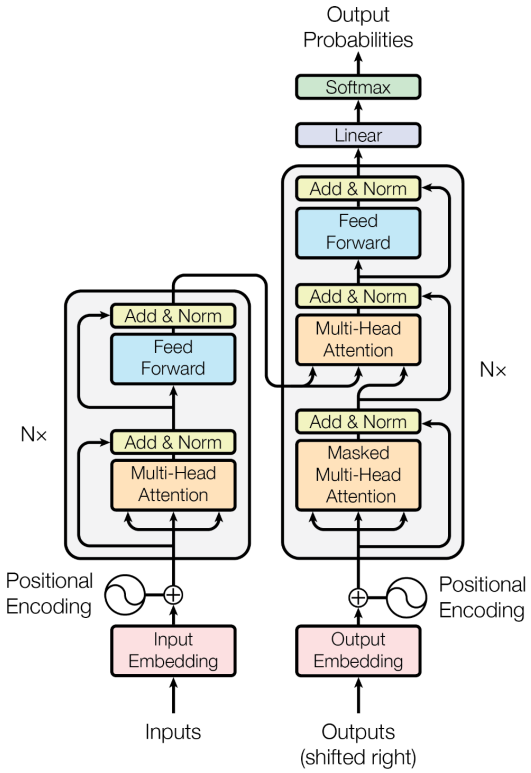


Figure 2.17: An overview of the transformer architecture, it is an encoder-decoder structure with multi-head attention and feed-forward layers. The encoder processes the input sequence into a context-rich representation, which the decoder then uses to generate the output sequence. Figure from [Vaswani et al., 2017].

Chapter 3 Numerical resolution of partial differential equations

In this chapter, we introduce the numerical methods used to solve partial differential equations (PDEs). In particular, we discuss the spatial discretisation of a PDE using finite differences and the time integration of the resulting ordinary differential equation (ODE), and we highlight the numerical errors that can arise from these methods and discuss potential challenges and limitations.

3.1 Introduction

PDEs are fundamental mathematical tools used to model a wide range of physical phenomena, including heat transfer, fluid dynamics, and electromagnetic fields. These equations involve multiple independent variables and their partial derivatives, making them challenging to solve analytically. Consequently, numerical methods, such as finite difference methods, play a crucial role in approximating solutions to PDEs.

Numerical weather prediction involves addressing PDEs representing physical phenomena, e.g. the advection equation or the diffusion equation. These equations can be expressed as follows:

$$\partial_t f + v\partial_x f = 0, \tag{3.1}$$

$$\partial_t f - \kappa\partial_x^2 f = 0, \tag{3.2}$$

where f is the field of interest, v is the velocity field, and κ is the diffusion coefficient. Or more generally, the Navier–Stokes equations, which describe the motion of fluid

substances, can be expressed as follows:

$$\begin{aligned} \partial_t u + (u \cdot \nabla)u &= -\nabla p + \nu \nabla^2 u, \\ \nabla \cdot u &= 0, \end{aligned} \tag{3.3}$$

where p is the pressure field, and ν is the diffusion coefficient.

These equations can be expressed at a more abstract level as follows:

$$\partial_t f = \mathscr{F}\left(f, \partial_x f, \partial_x^2 f, \ldots\right), \tag{3.4}$$

where $\mathscr{F}$ is a function that describes the evolution of a univariate or multivariate field f over time. Computers cannot directly solve symbolic PDEs, and a common approach involves a two-stage process to transform the PDE into a mathematical formulation more suitable for computational handling. This process begins by discretising the partial derivatives with respect to spatial coordinates, resulting in an ODE. Subsequently, a temporal integration describes the evolution of the system over time.

3.2 Spatial discretisation using finite differences

Spatial discretisation can be performed using several methods, e.g. finite volumes, finite elements, or spectral methods. However, the simplest one is the finite-difference method. This is among the most widely used techniques for numerically solving PDEs.

Finite-difference discretisation involves dividing the spatial and temporal domains of the PDE into a grid or mesh of discrete points. The value of the solution at each grid point is then approximated using finite-difference approximations of the partial derivatives. For example, on a 1D periodic domain $[0, L]$ of coordinate x, discretised in N grid points $(x_i)_{[0,n-1]}$ ($x_n = x_0$), the spatial derivatives of a function f at a point x_i can be approximated using central differences, forward differences, backward differences or more complex schemes, such as first-order upwind schemes [Hirsch, 1990] or the WENO scheme [Liu et al., 1994]. These approximations are generally obtained from the Taylor expansion of the function around the grid point x_i which is the basis of the finite difference method. The third-order Taylor expansion of a function f around a point x_i is given by

$$\begin{aligned} f(t, x + \delta x) &= f(t, x) + \delta x \partial_x f(t, x) + \frac{\delta x^2}{2} \partial_x^2 f(t, x) + \frac{\delta x^3}{6} \partial_x^3 f(t, x) + o(\delta x^4), \\ f(t, x - \delta x) &= f(t, x) - \delta x \partial_x f(t, x) + \frac{\delta x^2}{2} \partial_x^2 f(t, x) - \frac{\delta x^3}{6} \partial_x^3 f(t, x) + o(\delta x^4). \end{aligned} \tag{3.5}$$

Using central differences, the first-order derivative can be approximated as follows:

$$\partial_x f(t, x_i) \approx \frac{f(t, x_{i+1}) - f(t, x_{i-1})}{2\delta x}, \tag{3.6}$$

where $\delta x = x_{i+1} - x_i$ represents the grid resolution. Forward and backward differences are given, respectively, by

$$\partial_x f(t, x_i) \approx \frac{f(t, x_{i+1}) - f(t, x_i)}{\delta x}, \tag{3.7}$$

and

$$\partial_x f(t, x_i) \approx \frac{f(t, x_i) - f(t, x_{i-1})}{\delta x}. \tag{3.8}$$

This discretisation scheme can be used to solve the advection equation (Eq. (3.1)) or diffusion equation (Eq. (3.2)), among others.

In this case of the advection equation (Eq. (3.1)), the first-order upwind scheme is used to approximate the derivative of a function at a point x_i using the value of the function at the point x_{i-1} or x_{i+1} depending on the sign of the velocity field. Its formula is given by

$$\partial_x f(t, x_i) \approx \begin{cases} \frac{f(t,x_i) - f(t,x_{i-1})}{\delta x} & \text{if } v_i > 0, \\ \frac{f(t,x_{i+1}) - f(t,x_i)}{\delta x} & \text{if } v_i \leq 0, \end{cases} \tag{3.9}$$

3.3 Time integration

Following spatial discretisation, Eq. (3.4) can be written as an ODE as follows:

$$\frac{d\mathbf{f}}{dt} = \hat{F}(\mathbf{f}), \tag{3.10}$$

where $\mathbf{f}(t)$ is the discretised form of f over the spatial domain, e.g. the vector of grid-point values of f at time t, i.e. $\mathbf{f}(t) = (f(t, x_i))_i$ in the 1D domain mentioned above.

For time integration, various methods can also be employed, e.g. Euler's method and Runge–Kutta methods [Runge, 1895, Kutta, 1901]. These methods differ in their accuracy, stability, and computational cost. As for the spatial discretisation, these methods are based on the Taylor expansion of the solution around the time t which is given by

$$f(t + \delta t, x) = f(t, x) + \delta t \partial_t f(t, x) + \frac{\delta t^2}{2} \partial_t^2 f(t, x) + \mathcal{O}(\delta t^3). \tag{3.11}$$

An explicit Euler time integration of Eq. (3.10) reads

$$\mathbf{f}_{q+1} = \mathbf{f}_q + \delta t \hat{F}(\mathbf{f}_q), \tag{3.12}$$

where $\mathbf{f}_q = \mathbf{f}(t_q)$, with $t_q = q\delta t$ the discretised time of time step δt. Euler method is the simplest time integration scheme, but it is known to be less accurate than higher-order methods. However, it is computationally less expensive and can be used as a starting point for more complex methods.

Runge–Kutta methods are a family of implicit and explicit iterative methods that are used to solve ODE. They are based on the idea of solving the ODE by taking multiple intermediate steps between two time steps. Here, we focus on the explicit Runge–Kutta methods, which can be expressed as follows:

$$\begin{aligned} \mathbf{f}_{q+1} &= \mathbf{f}_q + \delta t \sum_{i=1}^{s} b_i k_i, \\ k_i &= \hat{F}(\mathbf{f}_q + \delta t \sum_{j=1}^{i-1} a_{ij} k_j), \\ k_1 &= \hat{F}(\mathbf{f}_q), \end{aligned} \tag{3.13}$$

where $\mathbf{f}_q$ is the solution at time t_q, k_i are the intermediate time derivatives, a_{ij} and b_i are the coefficients of the method, and s represents the order of the method. However, few particular cases are commonly used, such as the second-order Runge–Kutta method (RK2) and the fourth-order Runge–Kutta method (RK4). The second order Runge–Kutta method (RK2) is given by

$$\begin{aligned} \mathbf{f}_{q+1/2} &= \mathbf{f}_q + \frac{\delta t}{2} \hat{F}(\mathbf{f}_q), \\ \mathbf{f}_{q+1} &= \mathbf{f}_q + \delta t \hat{F}(\mathbf{f}_{q+1/2}), \end{aligned} \tag{3.14}$$

where an intermediate time integration is done at the midpoint of the time step, $\mathbf{f}_{q+1/2}$, before computing the complete time integration step starting from this intermediate estimation of the solution. This method is slightly more accurate than the Euler method and requires two times more function evaluations per time step.

The most popular Runge–Kutta method is the fourth-order method (RK4), which is given by

$$\begin{aligned} \mathbf{k}_1 &= \hat{F}(\mathbf{f}_q), \\ \mathbf{k}_2 &= \hat{F}(\mathbf{f}_q + \frac{\delta t}{2}\mathbf{k}_1), \\ \mathbf{k}_3 &= \hat{F}(\mathbf{f}_q + \frac{\delta t}{2}\mathbf{k}_2), \\ \mathbf{k}_4 &= \hat{F}(\mathbf{f}_q + \delta t \mathbf{k}_3), \\ \mathbf{f}_{q+1} &= \mathbf{f}_q + \frac{\delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \end{aligned} \tag{3.15}$$

In this method, four intermediate time derivatives are estimated at different points within the time step, and the time integration is computed using a weighted average of these intermediate time derivatives. This method is more accurate than the RK2

method, but it requires two times more function evaluations per time step. However, this double cost could be partially compensated using a higher time step, which is possible with higher order time schemes such as the RK4.

3.4 Example: advection equation

For the sake of illustration, we consider the advection over the above-mentioned 1D periodic domain, given by the following equation:

$$\partial_t f + v\partial_x f = 0, \tag{3.16}$$

where v is a velocity field whose values on the grid are denotes as $(v_i)_{i\in[0,n-1]}$. Applying central difference and an Euler scheme discretisation to the advection equation, we obtain the following equation:

$$\frac{f_{q+1,i} - f_{q,i}}{\delta t} + v_i \frac{f_{q,i+1} - f_{q,i-1}}{2\delta x} = 0, \tag{3.17}$$

which can be rearranged to yield the following sequential resolution:

$$f_{q+1,i} = f_{q,i} - \frac{\delta t}{2\delta x} v_i (f_{q,i+1} - f_{q,i-1}). \tag{3.18}$$

Using the first-order upwind scheme, we obtain the following equation:

$$\frac{f_{q+1,i} - f_{q,i}}{\delta t} + \begin{cases} \frac{v_i}{\delta x}(f_{q,i} - f_{q,i-1}) & \text{if } v_i > 0, \\ \frac{v_i}{\delta x}(f_{q,i+1} - f_{q,i}) & \text{if } v_i \leq 0, \end{cases} = 0, \tag{3.19}$$

which can be rearranged to yield the following sequential resolution:

$$f_{q+1,i} = f_{q,i} - \frac{\delta t}{\delta x} \begin{cases} v_i(f_{q,i} - f_{q,i-1}) & \text{if } v_i > 0, \\ v_i(f_{q,i+1} - f_{q,i}) & \text{if } v_i \leq 0. \end{cases} \tag{3.20}$$

This example illustrates the integration of the advection equation over time using an explicit Euler method. However, depending on the characteristics and requirements of the problem, other time integration schemes may be more suitable.

3.5 Numerical errors

Numerical methods for solving PDEs, such as the central, forward, and backward difference schemes, may introduce various numerical effects and errors that can impact

the accuracy and stability of the solutions especially in the presence of discontinuities or sharp gradients. These errors include truncation errors, round-off errors, diffusion, and dispersion errorsBoyd [1989], Hirsch [2007]. Truncation errors arise from the approximation of the derivatives, while round-off errors result from the finite precision of the numerical representation of real numbers.

3.5.1 Central finite differences

For example, in the advection equation, the central difference scheme introduces numerical diffusion and dispersion effects; these effects can be observed in the equivalent equation of the advection equation using this numerical scheme. The equivalent equation is obtained by replacing the spatial derivatives in the original PDE by their Taylor formula approximations, and represents the actual equation that is solved by the numerical scheme. In the following lines, we present the *equivalent equations* (also called the *modified equations*) [Hirt, 1968] of the central differences and the first-order upwind scheme for the advection equation.

Applying the Taylor formulas (see Eq. (3.5)) to the advection equation Eq. (3.17), we get the following expansion:

$$\partial_t f + \frac{\delta t}{2}\partial_t^2 f + o(\delta t) = -v\left(\partial_x f - \frac{\delta x^2}{6}\partial_x^3 f + o(\delta x^2)\right) \tag{3.21}$$

Whereas we require just a first-order expansion in time, we can replace the second-order time derivative by another term coming from a Taylor first order expansion of the Eq. (3.17):

$$\partial_t(\partial_t f) + o(\delta t) = -\partial_t(v\partial_x f) + o(\delta x) \tag{3.22}$$

Then,

$$\partial_t^2 f = -\partial_t v\partial_x f - v\partial_{xt}^2 f + o(\delta t, \delta x)$$

Using the same approach, as in Eq. (3.22), the derivative $\partial_{xt}^2 f$ can be computed as follows:

$$\partial_x(\partial_t f) = -\partial_x v\partial_x f - v\partial_x^2 f + o(\delta t, \delta x)$$

We replace the derivative $\partial_{xt}^2 f$ in the last formula :

$$\partial_t^2 f = -\partial_t v\partial_x f - v\left(-\partial_x v\partial_x f - v\partial_x^2 f\right) + o(\delta t, \delta x) \tag{3.23}$$

Finally, we replace the second-order derivative in Eq. (3.21) by the expression in Eq. (3.23):

$$\begin{aligned}&\partial_t f + \frac{\delta t}{2}\left(-\partial_t v \partial_x f - v\left(-\partial_x v \partial_x f - v\partial_x^2 f\right)\right)\\&= -v\left(\partial_x f - \frac{\delta x^2}{6}\partial_x^3 f\right) + \mathcal{O}\left(\delta t^2, \delta x^2\right).\end{aligned}$$

Hence,

$$\partial_t f + \tilde{v}\partial_x f = -\kappa\partial_x^2 f + \frac{\delta x^2}{6} v \partial_x^3 f + \mathcal{O}(\delta x^2), \tag{3.24}$$

where $\tilde{v} = v - \frac{\delta t}{2}\partial_t v + \frac{\delta t}{2} v \partial_x v$ is the effective velocity field, $\kappa = \frac{\delta t}{2} v^2$ is the diffusion coefficient, the $-\kappa\partial_x^2 f$ term introduces a negative diffusion effect, and the $\frac{\delta x^2}{6} v \partial_x^3 f$ term introduces a dispersion effect. The negative diffusion term keeps adding energy to the system, making it unstable and exploding, while the dispersion term introduces high-frequency oscillations.

It is important to note that the negative diffusion term comes from the second-order time derivative present in the time scheme expansion (check Eq. (3.23)). On the other hand, the dispersion term comes from the spatial discretisation scheme. In fact, when using higher-order time integration schemes, the negative diffusion term may disappear. Indeed, the approximation error of an k^{th} order time integration scheme presents only terms of order δt^p, and $(p+1)^{\text{th}}$ order time derivatives, with $p \geq k$. Thus, higher order time integration schemes will not introduce any type of diffusion.

Overall, the central differences method has a number of advantages. It is relatively simple to implement and can be used to approximate derivatives with a few function evaluations. It is also well-suited for use with functions that have smooth, well-behaved derivatives. However, it can be less accurate than some other methods for approximating derivatives, particularly for functions that have discontinuities or sharp changes in slope. In these cases, other methods, such as the first-order upwind scheme, may be more appropriate.

3.5.2 First-order upwind scheme

The first-order upwind scheme, shown in Eq. (3.9), is a method to discretise PDEs that takes into account the direction in which waves propagate. It is called an *upwind* scheme because it treats the flow of the solution as moving upwind relative to the spatial discretisation, meaning that the discretisation stencil, i.e. the group of grid points used to compute a solution at a particular point, is biased towards points in the direction opposite the flow. This helps to prevent numerical oscillations and instabilities that can occur when using finite difference methods to solve PDEs with advection terms. In the following lines, we present the equivalent equation of the first-order upwind scheme for the advection equation.

Considering the case of $v \geq 0$ of Eq. (3.19), using the Taylor formulas, we get the following.

$$\partial_t f + \frac{\delta t}{2}\partial_t^2 f + o\left(\delta t^2\right) = -v\left(\partial_x f - \frac{\delta x}{2}\partial_x^2 f + o\left(\delta x^2\right)\right) \tag{3.25}$$

As in the case of the central differences, we replace the second-order time derivative in the Eq. (3.25) by the expression in Eq. (3.23). :

$$\begin{aligned}&\partial_t f + \frac{\delta t}{2}\left(-\partial_t v\partial_x f - v\left(-\partial_x v\partial_x f - v\partial_x^2 f\right)\right)\\ &= -v\left(\partial_x f - \frac{\delta x}{2}\partial_x^2 f\right) + o\left(\delta t^2, \delta x^2\right)\end{aligned}$$

Hence,

$$\partial_t f + \tilde{v}\partial_x f = v_{num}\partial_x^2 f + o\left(\delta t^2, \delta x^2\right), \tag{3.26}$$

where $\tilde{v} = v - \frac{\delta t}{2}\partial_t v + \frac{\delta t}{2}v\partial_x v$, and $v_{num} = \frac{v}{2}\left(\delta x - v\delta t\right)$ the introduced numerical viscosity/diffusion.

The equivalent equation for the second case of Eq. (3.19) (case $v \leq 0$) is written as:

$$\partial_t f + \tilde{v}\partial_x f = v_{num}\partial_x^2 f + o\left(\delta t^2, \delta x^2\right), \tag{3.27}$$

where $v_{num} = \frac{v}{2}\left(-\delta x - v\delta t\right)$

From Eq. (3.26) and Eq. (3.27) we can write the equivalent equation as follows:

$$\partial_t f + \tilde{v}\partial_x f = v_{num}\partial_x^2 f + o(\delta t^2, \delta x^2), \tag{3.28}$$

where $\tilde{v} = v - \frac{\delta t}{2}\partial_t v + \frac{\delta t}{2}v\partial_x v$, and $v_{num} = \frac{v}{2}\left(sign(v)\delta x - v\delta t\right)$. The term $v_{num}\partial_x^2 f$ represents the introduced numerical diffusion. For the solution to this equation to be damped over time, the diffusion coefficient v_{num} must be positive, unlike the central difference scheme, where the diffusion coefficient is negative. This added positive diffusion/viscosity stabilises the solution and prevents it from oscillating. However, it also introduces a damping effect that can smooth out the solution and reduce its accuracy. This is a trade-off that must be considered when choosing a numerical method for solving PDEs.

For v_{num} to be positive the following condition must be satisfied

$$C = \frac{v \cdot \delta t}{\delta x} \leq C_{\text{max}} = 1, \tag{3.29}$$

where C is called the Courant number, and C_{max} is the maximum Courant number, in this case, it is equal to 1. This condition is called the Courant-Friedrichs-Lewy (CFL) condition [Courant et al., 1928], and ensures that the time step is small enough to capture the fastest propagation in the system.

There are other numerical methods that can be used to solve the advection equation, such as the Lax-Wendroff scheme. These methods have different properties and trade-offs, and the choice of method depends on the specific requirements of the problem to be solved.

There are other stability analysis methods that can be used to determine the stability of a numerical method, such as the von Neumann stability analysis, which can be used to determine the stability of a numerical method based on the properties of the discrete Fourier transform of the solution. However, this method can be applied only to linear PDEs.

Chapter 4 Physics-informed machine learning

This study introduces a hybrid architecture to address the difficulties encountered by purely data-driven models, in particular in the context of cloud cover nowcasting. Before delving into our approach, this chapter reviews the existing literature on physics-guided learning, which integrates physical principles into ML models. Section 4.2 examines how physical constraints can be incorporated into loss functions, while Section 4.3 covers physics-guided initialisation techniques. Section 4.4 discusses residual modelling for error correction, and Section 4.5 explores the development of hybrid physics-ML models. Finally, Section 4.6 describes a method for embedding the resolution of PDEs within neural networks.

4.1 Introduction

Machine learning models hold great promise for addressing scientific challenges associated with processes that cannot be fully simulated, either due to lack of resources or complexity of the physical process. However, their application in scientific domains faced challenges, including constraints related to large data needs, difficulty generating physically coherent outcomes, limited generalisability, and issues related to explainability [Karpatne et al., 2017]. To overcome these challenges, incorporating physics into ML models is of paramount importance. It uses the inherent structure and principles of physical laws to improve the interpretability of the model, handle limited labelled data, ensure consistency with known scientific principles during optimisation, and ultimately improve the overall performance and applicability of the models, making them more likely to be generalisable to *out-of-sample* scenarios. As discussed by Willard et al. [2022], the physics-ML hybridisation available techniques leverage different aspects of

ML models, e.g. the cost function, the design of the architecture or the weights' initialisation.

4.2 Physical constraints in the loss

A common method to ensure the consistency of ML models with physical laws is to embed physical constraints within the model's loss function [Karpatne et al., 2017]. This involves incorporating a physics-based term weighted by a hyperparameter, alongside the supervised error term in the loss function as follows:

$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \lambda \mathcal{L}_{\text{physics}}, \tag{4.1}$$

the hyperparameter λ controls the trade-off between the two terms, ensuring that the model learns to balance the physics-based constraints with the data-driven constraints. Fig. 4.1 illustrates this concept. This addition enhances prediction accuracy and accommodates unlabelled data. It has proven to be effective in addressing a range of problems, including uncertainty quantification, parameterisation, and inverse problems [Daw et al., 2022, Jia et al., 2019, Raissi et al., 2019b]. Raissi et al. [2019a] introduced the well-known Physics-Informed Neural Networks (PINNs) that leverage this concept to solve PDEs in a data-driven way and to learn the underlying physics from the data.

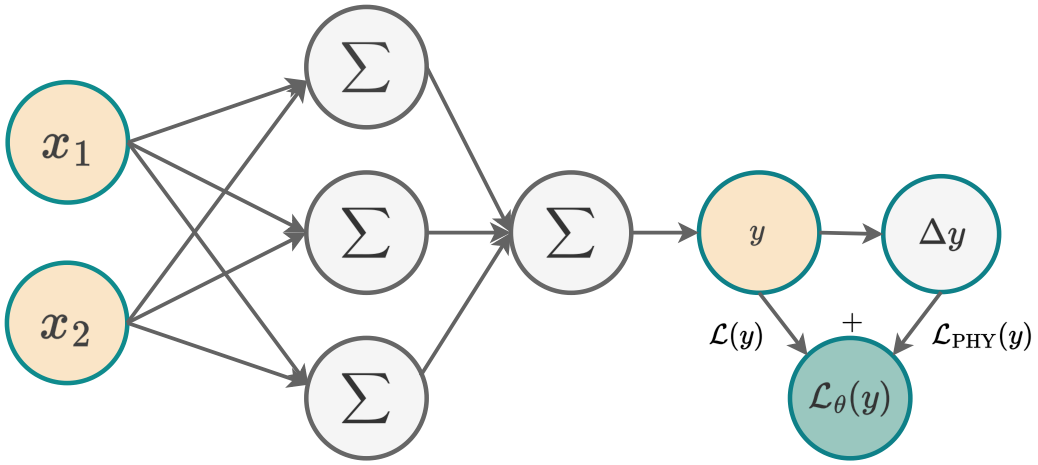


Figure 4.1: Illustration of a loss constrained feedforward neural network. The loss function is a combination of a supervised error term and a physics-based term.

This method has also been applied with generative models. Indeed, a very well-known challenge in GANs is to generate samples that are physically consistent. Thus, incorporating physical constraints into the loss function can help generate physically plausible samples even using a small dataset [Cang et al., 2017, Wu et al., 2020], to accelerate convergence [Yang et al., 2020] or improve resolution [Bode et al., 2019].

4.3 Physics-guided initialisation

Given that many ML models necessitate initialisation of parameters before training, researchers explore ways to inform the initial state of a model with physical insights. For instance, in neural networks, weights are commonly initialized through random distributions prior to training. However, inadequate initialisation may lead models to get trapped in local minima, a particularly prevalent issue in deep neural networks. However, leveraging physical knowledge to guide weight initialisation can accelerate or enhance model training. An effective approach involves employing transfer learning, an ML technique in which a model is initially trained on a related task and subsequently fine-tuned with limited data to address the desired task [Torrey and Shavlik, 2010]. This pre-trained model serves as an informed starting point, ideally positioning it closer to the desired parameters for the target task compared to random initialisation, thus facilitating faster convergence, improved performance and avoiding local minima [Jia et al., 2021]. The term *transfer learning* is generally used when the pre-training is done on large datasets.

Another approach to guide the model's training is to use simulated data from physics-based models for pre-training, before fine-tuning on real-world data. This method has found applications in diverse fields. For instance, in temperature modelling, Jia et al. [e.g. 2019] where the model is pre-trained on generated data from a physics-based model before being fine-tuned using a smaller real-world dataset. We can also mention the work of Shah et al. [2018] on autonomous vehicle training, where driving algorithms are pre-trained in a simulated environment before being fine-tuned in the real world. This method is also applied in biophysics [Sultan et al., 2018]. However, this method requires the assumption that the underlying physics of the simulated data aligns with the real-world data.

4.4 Residual modelling

To address imperfections in physics-based models, a common strategy is residual modelling. Here, an ML model learns to predict the errors (residuals) made by the physics-based model [Forssell and Lindskog, 1997]. This approach takes advantage of learned biases to correct predictions. For example, San and Maulik [2018a,b] propose a data-driven machine learning framework, employing a feedforward network, to predict closure terms within differential equations stabilising temporal mode evolution, which often suffers from amplitude errors due to truncation of dissipative modes. Another example is the work of Wan et al. [2018] who used a neural network to complete an imperfect reduced-order extreme event model.

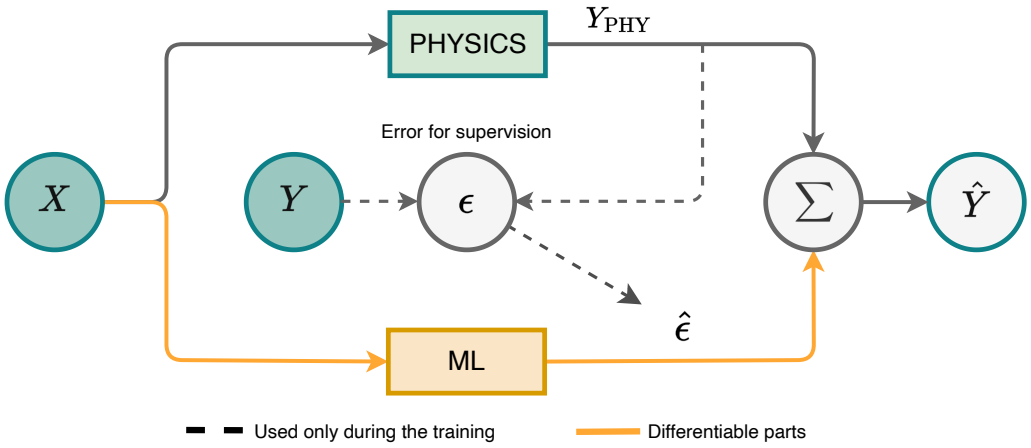


Figure 4.2: Illustration of residual modelling. The physics-based model is used to predict the output and the ML model is used to predict the residuals. Adapted from Forssell and Lindskog [1997].

This method has some limitations. For instance, it does not have the ability to enforce physics-based constraints, as it primarily deals with errors rather than physical states. In addition, the predicted errors may be overfitted to the training data, leading to poor generalisation.

4.5 Hybrid physics-ML models

An advanced variation of residual modelling involves the integration of physics-based models and ML models. This hybridisation can take different forms depending on the problem at hand.

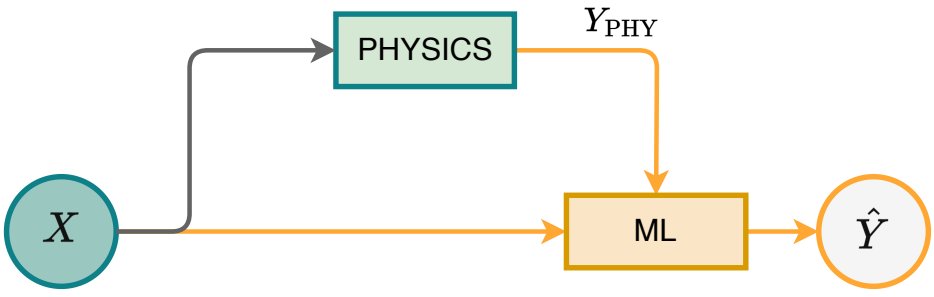


Figure 4.3: Illustration of a hybrid physics-ML model. The output of the physics-based model is used as an input to the ML model.

In scenarios where the dynamics of Physics are fully defined, a straightforward method involves using the output of a physics-based model as an input to an ML model (see Fig. 4.3). This approach has demonstrated enhanced predictions in tasks such as lake temperature modelling [Daw et al., 2022].

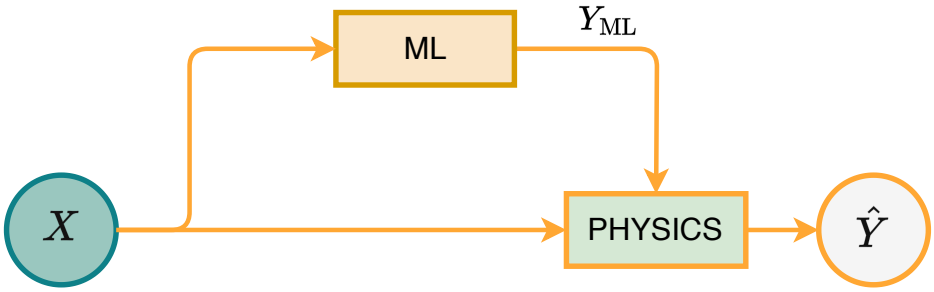


Figure 4.4: Illustration of a hybrid physics-ML model. The output of the ML model is used as an input to the physics-based model.

In other scenarios, hybridisation is done by replacing a component of the physics-based model with an ML model., e.g. Vlachas et al. [2018] used a neural network to estimate the trend term of a PDE, given the past states of the system, before integrating it to predict the future states.

In cases where a physical model contains unknown elements requiring coupling with an ML model for joint resolution, a viable strategy involves introducing a neural network to estimate the missing information; in this situation the machine learning model provides inputs to the physics-based model as illustrated in Fig. 4.4. An illustrative example is a study on sea surface temperature prediction, where de Bézenac et al. [2019] used a neural network to estimate the motion field. However, this method uses an analytical solution of the advection equation to estimate the motion field, which is not always possible, making it less generalisable to other modelling scenarios. In the following section, we discuss another way to integrate physics into neural networks.

4.6 Implementing and solving PDEs using neural layers

Let's consider an artificial neural network f_θ with parameters θ that takes input x and produces output $\hat{y}$. We consider a physics-based model ϕ representing the underlying physical processes, or equations, that govern the behaviour of a system. To incorporate physics into the neural network, one possible approach involves feeding the output of the physics-based model as an input to the neural network as follows:

$$\hat{y} = f_\theta\left(x, \phi\left(x_{\text{Phy}}\right)\right), \tag{4.2}$$

where x_{Phy} are the inputs of the physics-based model ϕ. The residual modelling approach, discussed in Section 4.4, is a particular case of this method. This approach is illustrated in Fig. 4.3. This method could be effective when the physics-based model is

self-contained and its components are explicitly known. However, it becomes impractical in scenarios where the physics-based model presents unknown variables that need to be estimated. This is the case in the application considered in this work, where the cloud motion field is unknown. In such instances, a more suitable approach is to pursue a joint resolution. Here, the physical model takes the output of the neural network and computes the predictions as shown in Fig. 4.4. This results in a composition of f_θ and ϕ as follows:

$$\hat{y} = \phi \circ f_\theta \left(x, x_{\text{Phy}}\right). \tag{4.3}$$

In this approach, ϕ implicitly applies a hard constraint on these outputs, this could contribute to accelerate the convergence of the neural network during the training process.

Unlike the first method (Eq. (4.2)), where the physics-based model ϕ is passive and not involved in the training procedure, the second method raises some challenges regarding the trainability of the architecture.

4.6.1 Automatic differentiability

As mentioned in Section 2.4, neural networks learn to minimise a loss function $\mathcal{L}_\theta$ by adjusting its set of parameters θ using data. The loss function measures the error between the predicted outcomes $\hat{y}$ and the ground truth y.

During this training process, the backpropagation algorithm is used to optimise model parameters by computing the gradient of the loss function with respect to the network's parameters. These gradients indicate how much each weight contributed to the error.

In fact, neural networks are part of a broader class of algorithms called *differentiable programming*. Differentiable programming can be defined as

> a programming paradigm in which complex computer programs (including those with control flows and data structures) can be differentiated end-to-end automatically, enabling gradient-based optimization of parameters in the program. – Blondel and Roulet [2024].

In order to perform backpropagation, we assume that the gradient of the loss function with respect to the model parameters could be calculated using the chain rule. This assumption is called (automatic) *differentiability*. This is a crucial property that allows the use of gradient-based optimisation algorithms to update the model's parameters during training. In fact, neural networks are based on differentiable activation functions and operations, allowing gradients to be propagated backward through network layers. Examples of differentiable operations commonly used in neural networks include linear transformations (such as matrix multiplications and additions), element-wise operations (such as sigmoid, tanh, and ReLU activations), and pooling operations (such as

average pooling).

However, there are certain operations that are non-differentiable in the context of automatic differentiation. For example, operations involving creating a new tensor in the middle of the computation graph without using the tensor from the last operation leads to non-differentiability, as the chain rule breaks down. The *argmax* operation is a typical example of a non-differentiable operation. This operation returns the index of the maximum value in a tensor, and it is not differentiable because the gradient of the output with respect to the input is not well-defined.

In this proposed hybrid approach, PDEs are solved to produce model predictions. If the PDE solver includes non-differentiable steps, the chain rule breaks down, making it impossible to compute gradients within the standard deep learning frameworks. In what follows, we describe our strategy for modelling and solving PDEs using basic differentiable operations commonly employed in neural networks.

4.6.2 Approximating derivatives and time integration in neural networks

In this study, we propose to model and solve PDEs of the form Eq. (3.4) within a neural network. This is achieved by describing the equivalent of spatial and temporal discretisation in the frame of neural network layers; thus, it can be implemented in a DL framework such as TensorFlow [Abadi et al., 2016] or PyTorch [Paszke et al., 2019].

4.6.3 Finite-difference methods and convolutional layers

To implement a finite-difference discretisation, different approaches can be used. For instance, the spatial discretisation could be implemented as a matrix-vector product, using a matrix where each row corresponds to a grid point and each column to a finite-difference coefficient. The general form of the spatial discretisation can be written as

$$\partial_x^\alpha f \approx \mathbf{M} f, \tag{4.4}$$

where $\mathbf{M}$ is a diagonal-constant matrix (also called a Toeplitz matrix) containing the finite-difference coefficients for the α^{th} derivative. For example, the central difference scheme in Eq. (3.6) can be implemented as a matrix-vector product as follows:

$$\partial_x f \approx \frac{1}{2\delta x} \begin{pmatrix} 0 & 1 & 0 & \dots & -1 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 1 & \dots & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix}, \tag{4.5}$$

where N is the number of grid points. This method is simple and could be implemented in a deep learning framework. However, it is not the most efficient way to implement finite-difference methods in neural networks. Indeed, the matrix-vector product is computationally expensive and requires a large amount of memory, e.g. for a 1D domain the complexity is $\mathcal{O}(N^2)$.

An alternative approach is to use convolutional layers [Després, 2022]. Using convolutions (presented in the Section 2.10.1) is simpler and more efficient than matrix-vector products. For a 1D domain, the complexity is reduced to $\mathcal{O}(N)$. For example, the 1D convolution associated with Eq. (3.6) can be mathematically written as:

$$\left(K^1 * f\right)[i] = \sum_{m=0}^{M-1} K^1[i]\, f[m+i]\,, \tag{4.6}$$

where K^1 is the kernel or filter used for the convolution and expressed as

$$K^1 = \left[\begin{array}{ccc} \frac{-1}{2\delta x} & 0 & \frac{1}{2\delta x} \end{array}\right],$$

and f represents the input data. The variable M corresponds to the size of the kernel. It is the number of finite-difference coefficients, also called the stencil size. In this case, a three-point stencil is considered ($M = 3$). Finally, $*$ is the convolution operator.

This leads to an interesting interaction with Deep Learning frameworks. Indeed, CNNs rely on the operation

$$\text{ConvLayer}(f)[i] = \sigma\left(\sum_{m=0}^{M-1} K[m]\, f[m+i] + b\right),$$

where σ is called *activation function* and b is a parameter representing the *bias*. Observing that using $\sigma = \textit{identity}$ and b $= 0$ leads to the same operation as in Eq. (4.6), deep learning frameworks can be used to approximate derivatives, which enables derivative-based operations in neural networks, as shown in Fig. 4.5. The same principle applies to higher derivative orders. For any positive integer α, we can write the approximation of the α^{th} derivative of f as

$$\partial^\alpha f \approx K^\alpha * f, \tag{4.7}$$

where K^α are the finite difference coefficients for the α^{th} derivative.

Finally, using convolutions is a straightforward method to model the spatial term of a PDE as follows:

$$\hat{F}(f) = \mathcal{N}(f). \tag{4.8}$$

This results in a neural network that can be used for time integration.

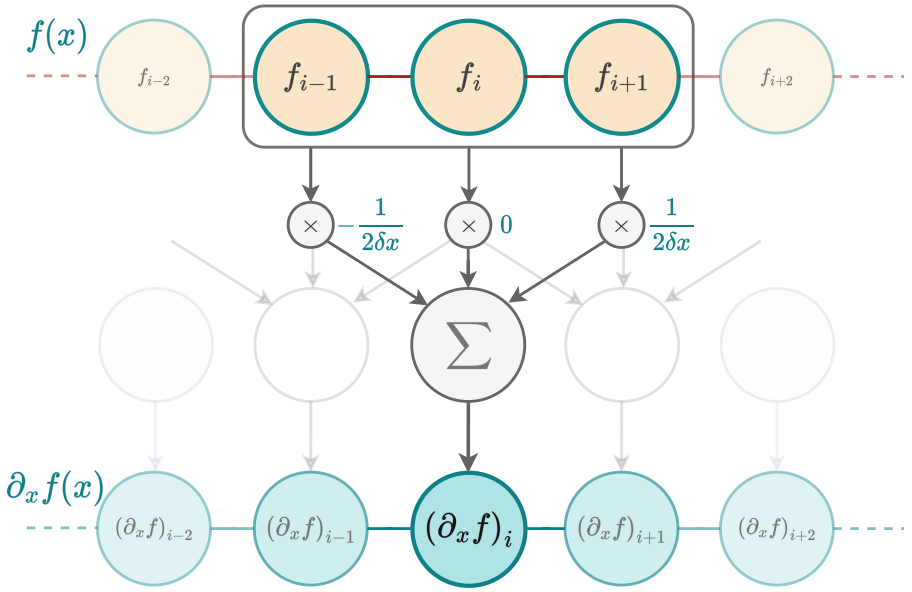


Figure 4.5: In order to calculate the numerical derivative of f, a kernel K^1 is used to slide across an input vector, which is a discretised version of f with N elements, element-wise multiplying values within its window and summing the results to approximate the derivative at each position. The result is a new vector of size $N-2$ containing the numerical derivative of f (using zero-pair or duplicate values in the input vector can be applied at the bounds to produce an output vector of size N). This is equivalent to a convolution between K^1 and f, and can be reproduced using a 1D convolutional layer with K^1 as a kernel.

4.6.4 Temporal schemes and residual networks

The time integration expressed in the Eq. (3.12) can be written by using the neural network implementation $\mathcal{N}$ of the trend as

$$f_{n+1} = f_n + \delta t \mathcal{N}(f_n). \tag{4.9}$$

Interestingly, this formulation is very similar to that of the residual block (see Fig. 2.14). Indeed, the residual block operation is formulated as follows:

$$y = x + \mathcal{F}(x), \tag{4.10}$$

where x is the input to the block, y is the output, and $\mathcal{F}$ is called a *residual function*, made up of multiple neural layers. These layers represent the difference between the input and output. This function aims to capture the additional information or adjustments needed to transform the input into the desired output. This similarity between residual blocks and time schemes, also observed in Ruthotto and Haber [2020], Chen et al. [2018], Fablet et al. [2018], suggests that the time integration step can be done inside a neural network, all we need is the residual function, which can be modelled

using convolutional layers as shown previously. Pannekoucke and Fablet [2020] proposed a general framework called PDE-NetGen[1], to model a PDE in a neural network form using this method.

Residual blocks were originally designed to address vanishing gradient issues in image classification tasks. Intriguingly, these blocks were shown to function similarly to time schemes, where they introduce small changes over incremental time steps. This challenges the traditional black-box perception of neural networks, although full interpretability remains a distant goal.

We have shown that spatial derivatives of PDEs can be approximated within a neural network in a differentiable way. This allows us to compute gradients and backpropagate them during the training process. This fundamental knowledge serves as a foundation for our investigation of novel hybrid Physics-AI architectures. With these established principles, we present in Chapter 6 the proposed hybrid architecture, which is used for cloud cover nowcasting. Before that, in the next chapter, we present the methods used in weather forecasting.

[1]https://github.com/opannekoucke/pdenetgen

Chapter 5 Methods for weather forecasting

This current chapter is a review of the literature on weather forecasting, focusing on the evolution of weather forecasting methods, the limitations of numerical weather prediction models, the use of deep learning for weather and climate forecasting, and the challenges and limitations faced by deep learning models in weather forecasting. We also discuss the potential of hybrid models that combine physical knowledge with deep learning methods to improve the accuracy and physical consistency of weather predictions.

5.1 Numerical weather prediction (NWP)

For several decades, NWP has been the primary method for weather forecasting. NWP models are based on the laws of physics and use mathematical equations to simulate the atmosphere and predict the weather. These models are used by meteorological agencies around the world to produce weather forecasts for different time scales, from short-term forecasts (up to 3 days) to medium-range forecasts (up to 10 days) and long-range forecasts (up to 30 days).

The equations used in NWP models includes the Navier-Stokes and mass continuity equations (accounting for Earth's rotation), the first law of thermodynamics, and the ideal gas law. These equations describe changes in atmospheric wind, pressure, density, and temperature.

These equations are highly complex and cannot be solved analytically for practical weather prediction. Instead, they must be solved numerically. This approach means that the model only directly resolves motions and processes that occur on scales larger than the grid size, known as the resolved scales. However, many important atmospheric phenomena, such as small-scale turbulence, cloud microphysics, friction, and processes

like condensation, evaporation, and radiative heating occur at scales smaller than the grid size and are therefore not directly resolved by the model. These are referred to as unresolved scales.

To account for the effects of these unresolved processes on the larger, resolved scales, NWP models use parametrisations. Parametrisations involves representing the effects of small-scale processes with simplified mathematical formulas or empirical relationships that relate them to the larger-scale variables that the model does resolve. For example, instead of simulating each droplet within a cloud, the model uses parametrisations to estimate the overall impact of cloud processes on temperature, humidity, and other variables.

This distinction between resolved and unresolved scales is crucial. This is because it allows NWP models to efficiently simulate the atmosphere's large-scale dynamics without needing to directly compute every tiny detail, which would be computationally unfeasible. Parametrisations, therefore, play a key role in making weather and climate predictions practical, though it also introduces some level of approximation and potential uncertainty into the model's predictions.

The accuracy of NWP models is heavily dependent on initial conditions, which are determined by assimilating observational data from satellites, weather stations, radar, and other sources. Data assimilation techniques, such as variational methods and ensemble Kalman filters, are used to optimally blend observations with model predictions to generate the initial state of the atmosphere. High-resolution models can capture small-scale weather phenomena (e.g., thunderstorms), but they require significant computational resources. Models vary in their resolution and complexity, with global models covering the entire Earth and regional models focusing on specific areas with finer grids.

Simplifications have been applied to reduce the complexity of these equations, starting with Richardson's early attempts, although with limited success. The first use of an electronic computer for weather prediction occurred in Princeton in 1950, with the first real-time forecasts made in Stockholm in 1954. It was not until the 1970s, with the advent of supercomputers, that it became feasible to solve the full set of equations proposed by Abbe and Bjerknes.

Various numerical methods emerged to enhance stability, accuracy, and computational speed, incorporating key components like spatial discretisation, time-stepping methods, boundary treatments, and initialisation approaches. These developments laid the foundation for NWP. Today, a diverse hierarchy of models exists, ranging from global climate projections to local weather and air-quality prediction, each with varying complexity to address specific forecasting needs.

Key numerical weather prediction (NWP) models include the Integrated Forecasting System (IFS) from the European Centre for Medium-Range Weather Forecasts (ECMWF), the Global Forecast System (GFS) developed by the National Centers for Environmen-

tal Prediction (NCEP), the Unified Model (UM) from the UK Met Office, and ARPEGE, developed by the Centre National de Recherches Météorologiques (CNRM) in France in collaboration with ECMWF. Additionally, AROME is a regional model designed to explicitly address a part of the convection process [Seity et al., 2011]. These models are continually improved with advancements in data assimilation, computational techniques, and atmospheric science, making them central tools in modern weather and climate forecasting.

5.2 Limitations of numerical weather prediction

Although NWP models are globally used for weather forecasting, they have several limitations. One of these limitations is that the accuracy of NWP models is highly dependent on the initial conditions of the atmosphere. Due to the chaotic nature of the atmosphere, small errors - at the atmosphere scale - in initial data can grow rapidly, leading to large differences in the outcome of the system over time and inaccurate forecasts. This fundamental property of chaotic systems is known as the "butterfly effect"; referring to the idea that

> a butterfly flapping its wings in Brazil can set off a tornado in Texas. Lorenz.

This sensitivity to initial conditions imposes a limit on the predictability of the atmosphere, beyond which forecasts become unreliable. Ensemble forecasting techniques have been developed to account for this uncertainty by running multiple simulations with slightly different initial conditions to generate probabilistic forecasts. However, these simulations increase the computational cost of the forecast.

Another limitation of NWP models is that the representation of physical processes in the atmosphere is based on parametrisations that can introduce significant errors. The development of accurate parametrisation schemes is a major challenge, as these processes are highly variable and often not well understood.

Although high-resolution models can capture fine-scale weather phenomena, such as thunderstorms and local wind patterns, they require significant computational resources. This trade-off between model resolution and computational feasibility limits the ability to run very high-resolution models over large domains. Operational centers must balance model resolution, domain size, forecast lead time, update frequency, and the number of ensemble members, often compromising on one or more aspects due to computational constraints.

Other limitations of NWP models include systematic bias due to imperfect model physics, handling of extreme events, and integration of new types of data. Meteorological agencies are continuously working to improve NWP models and are beginning to

slowly integrate machine learning and deep learning methods to improve the efficiency of weather forecasts and correct some of the model biases.

5.3 Deep learning for weather and climate forecasting

Deep learning has been increasingly used in earth science for a wide range of tasks, including weather and climate prediction[e.g. Espeholt et al., 2022, Ravuri et al., 2021, Trebing et al., 2021, Ayzel et al., 2020, Berthomier et al., 2020, Shi et al., 2015]. The use of deep learning in weather and climate science has been motivated by the need to improve the accuracy of weather and climate models. Weather and climate models are complex and computationally expensive. Deep learning methods are effective for learning complex patterns from large amounts of data and for making accurate predictions within a reasonable amount of time.

Deep learning methods have been used for time series prediction tasks in weather and climate science. Tasks like temperature, pressure, wind speed at a specific site are examples of time series prediction tasks. For example, Kuligowski and Barros [1998] uses a feedforward neural network for localised precipitation prediction, El Mghouchi et al. [2019] also uses a feedforward neural network for filtering the most relevant input features for a solar radiation prediction task, and Han et al. [2021] used a Recurrent Neural Network to generate synthetic weather data.

Other tasks in weather and climate science involve the prediction of spatial data, such as precipitation, cloud cover, and temperature maps. These types of data are more challenging to process and analyse as the methods used for time series data are not directly applicable and computationally expensive. Convolutional neural networks are well-suited for processing satellite data, as they are designed to capture the spatial structure of the input data. Convolutional neural networks have been used for tasks such as cloud detection, precipitation nowcasting, and the prediction of extreme weather events. For example, Kim et al. [2023] used a CNN for cloud cover estimation and Berthomier et al. [2020] used a U-Net architecture for cloud cover nowcasting. A U-Net has also been used for precipitation nowcasting as highlighted by Ayzel et al. [2020], a modified version was used for a similar task in Trebing et al. [2021] and in another study, Baño-Medina et al. [2021] proposed using CNN to downscale climate change projections.

Generative adversary networks have also been used to generate realistic skilful precipitation forecasts [Ravuri et al., 2021] or to generate realistic climate states, as in Besombes et al. [2021].

During the last three years, end-to-end learning models capable of forecasting mul-

tiple variables at a global scale have emerged with promising results. Although meteorological agencies possess vast datasets and substantial computational resources, there has been significant scepticism and hesitation regarding the performance of these models compared to traditional NWP models. This hesitation is partly due to their existing hardware being optimised for running physics-based models, which is not directly suitable for training large deep learning models, requiring GPUs. In addition, the lack of expertise in deep learning within meteorological agencies and the high computational cost of training these models have presented significant challenges. Consequently, the development of these advanced models has predominantly been driven by major private technology companies, such as Google, Microsoft, and Huawei, which have the capacity to train large-scale deep learning models on extensive datasets.

Several studies have introduced these global forecasting models, demonstrating impressive results that challenge the long-standing dominance of traditional NWP models. Notable examples include the transformer-based Pangu-Weather [Bi et al., 2022], the graph neural network-based GraphCast [Lam et al., 2022], the transformer-based Climax [Nguyen et al., 2023], Feng-Wu [Chen et al., 2023] and AURORA [Bodnar et al., 2024]. AURORA and Climax, are large models that havebeen trained on large corpus of data in order to learn the intrinsic patterns of the atmosphere, then fine-tuned on smaller datasets to make more accurate for specific tasks, these types of models are known as foundation models.

These models have shown that deep learning approaches can outperform traditional NWP models, such as the ECMWF IFS, across a wide range of climate variables.

These advancements were possible due to a particular dataset known as ERA5, the fifth generation of the ECMWF reanalysis dataset, which provides high-quality global climate data at a spatial resolution of 0.25 degrees and a temporal resolution of one hour from 1940 onwards. A large number of variables are available in ERA5 and at different altitudes in the atmosphere, weighting around 5 petabytes in total, and it is continuously updated.

These advancements are prompting a strategic shift within meteorological agencies, which are increasingly prioritising AI development in their forecasting efforts. Even the ECMWF, renowned for its high-performing IFS, considered to be the most accurate physics-based global NWP forecasting system in the world, is now investing in a data-driven forecasting model known as AIFS (Artificial Intelligence/Integrated Forecasting System), which leverages Graph Neural Networks [Lang et al., 2024].

The success of these deep learning models marks a significant shift in the landscape of weather and climate prediction. They have demonstrated competitive accuracy and higher computational efficiency, making them a compelling alternative for forecasting tasks, even if high-resolution phenomena are not captured due to the coarse resolution of training data (ERA5). However, despite their promising capabilities, these models are

still in the early stages of development. Substantial work remains to be done to further enhance their performance, address their current limitations, and fully integrate them into operational forecasting systems.

5.4 Challenges and limitations of deep learning in weather forecasting

Despite the success of deep learning methods in weather and climate science, at a point where they are replacing traditional numerical weather prediction models, they face several challenges and limitations. One of the main challenges is the need for large amounts of data to train deep learning models. Weather and climate data are often limited, especially for extreme weather events, which are rare events with heavy consequences. Another challenge is the need for large computational resources to train deep learning models, especially for large-scale weather and climate prediction tasks. Deep learning models are complex and require large amounts of memory and computational power to train. This can be a barrier to the use of deep learning methods in weather and climate science, especially for researchers and institutions with limited computational resources. In the following table 5.1, we show the resources needed to train some of the recent deep learning models:

Model	Training time	Hardware
Pangu-Weather [Bi et al., 2022]	15 days	192 Nvidia V100
GraphCast [Lam et al., 2022]	21 days	32 TPU v4
Feng-Wu [Chen et al., 2023]	17 days	32 Nvidia A100

Table 5.1: Computational resources needed to train recent deep learning models for weather prediction.

This implies that each training run can incur costs of up to $100,000 when using cloud services. Once the model is trained, a single GPU can be enough for inference. However, the training process is not a one-shot attempt; it involves extensive tuning and testing. Moreover, even after initial training, the model may require regular retraining to stay aligned with new data, especially in the context of climate change.

One of the most common challenges when deep learning models are used for weather prediction is the inconsistency with physical laws and real world data, which lead to unrealistic predictions. For example, deep learning models do not always respect fundamental principles of physics, such as the conservation of mass and energyBeucler et al. [2019]. Another example is producing progressively smoother and blurrier predictions as the forecast lead time increases [Ayzel et al., 2020, Tran and Song, 2019, Ravuri

et al., 2021, Sønderby et al., 2020]. This is due to lack of physical constraints in the model, the increase in uncertainty as the forecast lead time increases, and using loss functions such as the MSE that have as an optimal solution the average of the training data, this explains also the absence of intensive values in the predictions. This is a major limitation of deep learning models in weather and climate science, as the predictions of these models are used to make decisions that have a significant impact on society and the environment and should be able to predict the extreme values of the variables.

A study by Selz and Craig [2023] found that deep learning models, such as Pangu-Weather [Bi et al., 2022], struggle to capture the butterfly effect. The research revealed that, compared to physics-based models, deep learning models, Pangu-Weather in particular, exhibit limited error growth when subjected to small perturbations in the initial conditions. However, when perturbations of magnitudes comparable to current initial condition uncertainties were introduced, the error growth in deep learning models was consistent with that of physics-based models. This indicates that while deep learning models may not capture the butterfly effect as effectively as physics-based models, they still behave normally within the range of current uncertainties. Thus, this should not be viewed as a limitation of deep learning for weather forecasting but rather a constraint on their potential future use as digital twins for the Earth system.

Another issue with studies presenting deep learning models for weather prediction is the benchmarking. Classical machine learning metrics such as the MSE or accuracy are used to evaluate the performance of the models, but these metrics are not sufficient to assess the performance of a weather prediction model. Although these metrics provide a measure of the model's accuracy, they do not capture the model's ability to predict, for instance, intensive values or extreme events. Indeed, the available objective metrics are not generally suitable for a comprehensive evaluation of weather prediction models, especially the sharpness and visual quality of the predictions. Several studies already pointed this metric issue, in particular, Ravuri et al. [2021] called in meteorologists to assess the performance of their proposed generative approach in comparaison to other models.

For a weather forecasting model to pass from the research phase to the operational phase, it passes through a series of tests and evaluations, a post processing, and including output interpretation. Deep learning models interpretability is a point that has been raised by several studies, as it is important to understand how these models make predictions and to be able to explain their outputs. However, deep learning models are frequently perceived as black boxes, posing challenges in discerning the mechanisms underlying their predictions. This is particularly important in weather and climate science, where the predictions of deep learning models can have far-reaching consequences, hence, the interpretability of deep learning models is an active area of research in this field Yang et al. [2024].

5.5 Hybrid models

Hybrid Physics-AI models are a promising approach to either improve the performance of traditional numerical weather prediction models or to address the limitations of deep learning models in weather forecasting. Several studies have proposed hybrid models for different tasks, as discussed in the Chapter 4. However, the use of hybrid models in weather forecasting is still in its early stages and limited to specific tasks, mostly related to lake and sea temperature prediction. For example, de Bézenac et al. [2019] proposed a hybrid model using the analytical solution of the advection equation with a learned motion field to predict sea surface temperature, Beucler et al. [2019] proposed to apply a conservation equation to a neural network to simulate the outgoing lowgwave radiation. Jia et al. [2019, 2021] proposed an recurrent neural network with energy conservation incorporated to predict lake temperature and Daw et al. [2019] used an LSTM-based model while ensuring that the monotonicity of water density is preserved to predict the lake temperature profiles.

These studies show that integrating physical knowledge can improve the physical consistency of the predictions and the generalisation of the model. However, these are still limited applications and the methods used are very task-specific and not generalisable to other tasks.

In this context, we propose a hybrid architecture allowing to combine the physical knowledge of the atmosphere with the data-driven capabilities of deep learning models to improve the accuracy and the physical consistency of cloud cover predictions, while being flexible and generalisable to different tasks.

Chapter 6 Proposed hybrid architecture

In the following sections, we present the proposed hybrid architecture for cloud cover nowcasting, which combines physical models based on probability fields advection with trainable neural networks to learn the unknown quantities. We discuss the challenges of advecting probability fields and the discretisation schemes used in this study. We then present the first hybrid model, HYPHAICC-1, using the advection dynamics to estimate the velocity field from the cloud cover data.

6.1 The principle of the proposed hybrid architecture

The proposed hybrid architecture is a dual-component system (see Fig. 6.2). The first component is composed of one or more classical deep learning models. These models process the most recent observations, yielding predictions for the physical unknowns of interest. The second block takes as inputs the physical variables, whether known or predicted by the neural networks, along with initial conditions. This second component time integrates one or multiple PDEs to generate the subsequent state of the system. The 4^{th}-order Runge–Kutta (RK4) method is used for time integration. These PDEs encode essential physical knowledge. As discussed in the Section 4.6, the spatial derivatives are approximated using convolutional layers.

The parameters of the first component are trainable; they are optimised during training to estimate the unknown variables. However, we froze the parameters of the second block, as it represents already-known operations. This ensures that the second block maintains its fixed structure, representing the known physical knowledge encoded in the equations, while the trainable block focusses on learning and predicting the unknown aspects of the system. This architecture combines the physical knowledge encoded in the equations with the pattern-extraction capabilities of neural networks.

In this work, a particular application of the proposed hybrid architecture is considered, which is cloud cover nowcasting. It is a challenging application that requires the prediction of cloud cover over a short period of time, typically from a few minutes to a few hours. Cloud cover is an important meteorological variable that affects various aspects of weather and climate, such as temperature, precipitation, and radiation. Accurate nowcasting of cloud cover is essential for various applications, such as weather forecasting, climate modelling, and renewable energy production. In this chapter, we present the proposed hybrid model for cloud cover nowcasting, which combines physical models based on probability fields advection with trainable neural networks to learn the unknown quantities.

6.2 Cloud cover data

Cloud cover data is typically represented as a 2D field, where each grid point corresponds to a specific location on the Earth's surface. In our application, the data are satellite images captured by the Meteosat Second Generation (MSG) satellite at 0 degrees longitude. The spatial resolution of the data over France is $\approx$ 4.5 km and the time step is 15 minutes, and each image is of size 256×256. These images have been processed by EUMETSAT [García-Pereda et al., 2019], to provide the presence or not and the type of the cloud cover at each grid point[1] (see Fig. 6.1). This cloud type data is typically classified into sixteen categories, three of them are not cloud related :

- Non-processed: no data or corrupted data
- Snow over land
- Ice over sea

These three categories are merged into a single category, denoted as "No cloud" along with two other categories:

- Cloud-free land
- Cloud-free sea

The other categories are:

- Very low clouds
- Low clouds
- Mid-level clouds

[1] https://www.nwcsaf.org/ct_description

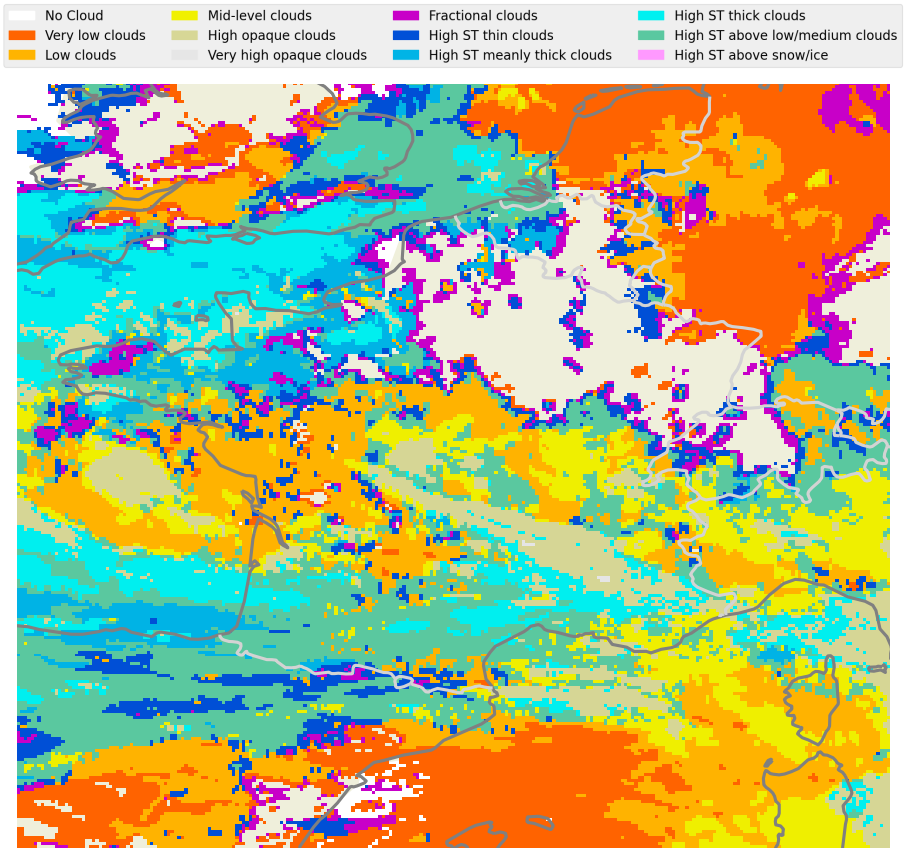


Figure 6.1: Sample of cloud cover data

- High clouds
- Very high clouds
- Fractional clouds
- High semitransparent thin clouds
- High semitransparent meanly thick clouds
- High semitransparent thick clouds
- High semitransparent above low or medium clouds
- High semitransparent above snow/ice

6.3 Advection of cloud cover: HyPhAICC-1

The dynamics of cloud cover can be modeled using a simplified advection equation, where the cloud cover is transported by the wind. One possible approach is to advect the 2D cloud cover map using a velocity field, expressed as:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f = 0,$$

where f represents the cloud cover field, $\mathbf{v}$ is the velocity field, and ∇ denotes the gradient operator.

Although this method is straightforward, it is not well-suited for cloud cover data. The numerical solution of the advection equation tends to introduce diffusion or dispersion, which alters the cloud cover labels, particularly at the boundaries of different cloud types. For instance, in the case of numerical diffusion, the cloud cover labels at these boundaries can be smoothed or degraded to lower values, leading to a loss of information and a degradation in the accuracy of cloud cover predictions.

The most natural way is to model the cloud cover as a probability fields, where each map or field represents the probability of the presence of a specific type of cloud cover at each grid point. This allows to model the uncertainty and variability of the cloud cover, and to account for the different types of clouds that can be present in the atmosphere. In this case, the advection equation becomes a set of advection equations for each cloud type, expressed as:

$$\frac{\partial P_i}{\partial t} + \mathbf{v} \cdot \nabla P_i = 0 \quad \text{for } i = 1, \dots, C, \tag{6.1}$$

where P_i is the cloud cover probability field for the i^{th} type of cloud cover, and $\mathbf{v}$ is the velocity field.

The velocity field can be obtained from weather models or observed data. However, in practice, the velocity field is not always available or accurate. In this case, a neural network can be used to learn the velocity field from the cloud cover data. This is the approach we propose in this work. This modelling is the base of the first hybrid model, denoted HYPHAICC-1, see Fig. 6.2.

In this hybrid model we use a U-Net Xception-style model [Tamvakis et al., 2022] inspired by the Xception architecture [Chollet, 2017]. It takes the last four observations stacked on the channel axis and estimates the velocity field. This model will be guided during training by the advection equation to learn the cloud motion patterns.

This modelling considers the same velocity field for all cloud types, which is a simplification of the real dynamics where we could have different velocities at different altitudes. However, using different velocity fields will yield difficulties to preserve the probability conservation property. The following section delve deeper into these properties, and the challenges that arise when advecting probability fields.

6.4 Which discretisation scheme to use?

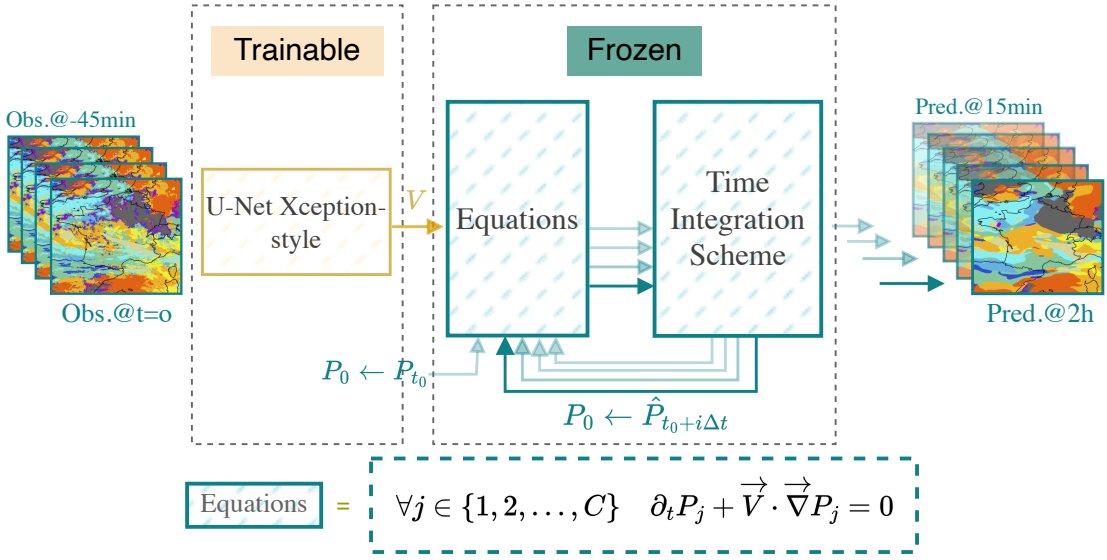


Figure 6.2: HYPHAICC-1: The proposed hybrid model consists of a U-Net Xception-style model to estimate the velocity field from the last observations, the estimated velocity field is smoothed using a Gaussian filter. The equation is numerically integrated using the 4^{th}-order Runge–Kutta method over multiple time steps. The initial condition (f_0) is updated, after each time step, to the current state, allowing the computation of the next state.

Advecting Probability fields causes less issues than advecting labels, however it is still a challenging task. In this section, we are considering a three-class problem where we have a discrete random variable X with values in the set $1, 2, 3$, and we denote by $X(t, x)$ the value of X at time t and space x, with $t \in [0, T]$ and $x \in [0, L]$. We are interested in studying the evolution of the state probabilities of X with respect to t and x. For this purpose, we define a vector $\mathcal{P}$ as

$$\mathcal{P} = \begin{bmatrix} P_X^1 \\ P_X^2 \\ P_X^3 \end{bmatrix},$$

here, $P_X^c(t, x)$ represents the probability of the c^{th} class;

$$P_X^c(t, x) = P(X(t, x) = c)$$

.

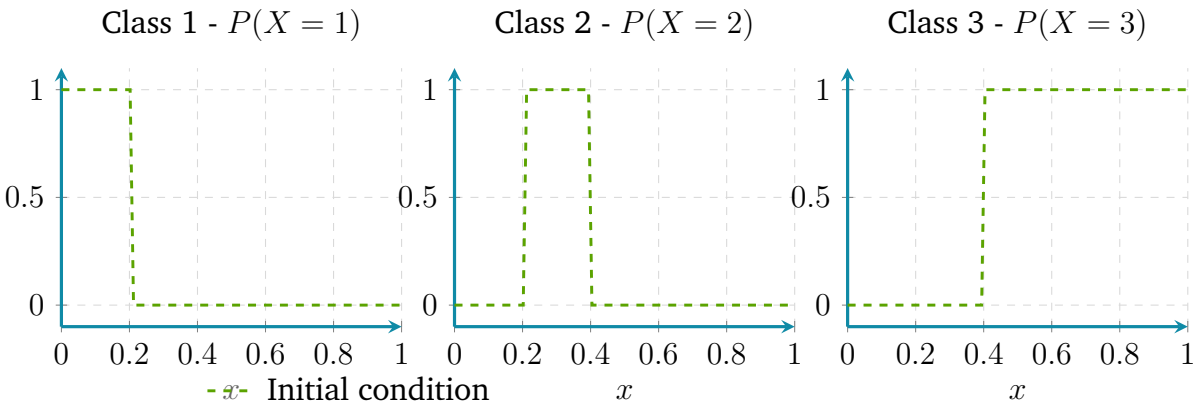


Figure 6.3: Initial condition of the probability fields

Let's consider the advection of three probability fields as follows:

$$\begin{aligned} \frac{\partial P_X^1}{\partial t} + \mathbf{v} \cdot \nabla P_X^1 &= 0, \\ \frac{\partial P_X^2}{\partial t} + \mathbf{v} \cdot \nabla P_X^2 &= 0, \\ \frac{\partial P_X^3}{\partial t} + \mathbf{v} \cdot \nabla P_X^3 &= 0. \end{aligned} \tag{6.2}$$

The Figure 6.3 shows an example of the initial condition. At the initial time, the necessary probabilistic properties are satisfied, i.e.:

1. Non-negativity: $P(\mathbf{x}, t) \geq 0$ for all $\mathbf{x}$ and t, with $\mathbf{x} = (x, y)$, which ensures that the probabilities remain non-negative.

2. Bound preservation: $P(\mathbf{x}, t) \leq 1$ for all $\mathbf{x}$ and t, which ensures that no probability exceeds 1.

3. Probability conservation: $\sum_{i=1}^{C} P_X^i(\mathbf{x}, t) = 1$ for all $\mathbf{x}$ and t, with $C = 12$ is the total number of cloud types. This property guarantees that the sum of all probabilities is equal to 1.

These properties must be preserved during the advection process.

6.4.1 Mass conservation

The probability conservation property is crucial in the context of advection of probability fields. It ensures that the total probability mass remains constant over time, i.e. the sum of the probabilities of all classes at each grid point is equal to 1. Let's consider a general case where the advection of the probability fields is described by the following system of equations:

$$\partial_t P_j + \mathcal{L}\left(P_j\right) = 0 \quad \forall j \in \{1, 2, \ldots, C\}\,, \tag{6.3}$$

where $\mathcal{L}$ represents a linear differential operator with non-zero positive derivative orders.

Proposition 1. *The probability conservation property is ensured if $\mathcal{L}$ is a linear differential operator with non-zero positive spatial derivative orders.*

Proof. Let's sum the three equations in Eq. (6.2):

$$\sum_{i=1}^{3} \partial_t P_X^i(x,t) + \mathcal{L}\left(P_X^i(x,t)\right) = 0$$
$$\partial_t \sum_{i=1}^{3} P_X^i(x,t) = -\sum_{i}^{3} \mathcal{L}\left(P_X^i(x,t)\right),$$

For the specific case where $\mathcal{L}$ is a linear differential operator with non-zero positive spatial derivative orders. Assuming $\sum_{i=1}^{3} P_X^i(x,t_0) = 1$, the linearity property of $\mathcal{L}$ allows us to interchange the summation and the operator, resulting as follows:

$$\begin{aligned}\sum_{i}^{3} \mathcal{L}\left(P_X^i(x,t_0)\right) &= -\mathcal{L}\left(\sum_{i=1}^{3} P_X^i(x,t_0)\right)\\ &= -\mathcal{L}(1)\\ &= 0\end{aligned}$$

$\mathcal{L}(1) = 0$ as $\mathcal{L}$ have only derivatives with positive non-zero orders.
Applying and summing the first order Taylor expansion at t_0 on each of the time derivatives of Eq. (6.2) give

$$\begin{aligned}\sum_{i}^{3} \frac{P_i(x,t_0+\delta t) - P_i(x,t_0)}{\delta t} + \mathcal{o}(1) &= -\sum_{i}^{3} \mathcal{L}\left(P_X^i(x,t)\right)\\ &= 0\end{aligned}$$

$$\sum_{i}^{3} P_i(x,t_0+\delta t) = \sum_{i}^{3} P_i(x,t_0) + \mathcal{o}(\delta t),$$

when δt is small enough, $\sum_{i}^{3} P_i(x,t_0+\delta t) = 1$.
Iteratively, starting from t_0, $\forall t$

$$\sum_{i}^{3} P_i(x,t) = \sum_{i}^{3} P_i(x,t_0) = 1$$

■

In this study, we consider the advection equation using the same velocity field for all

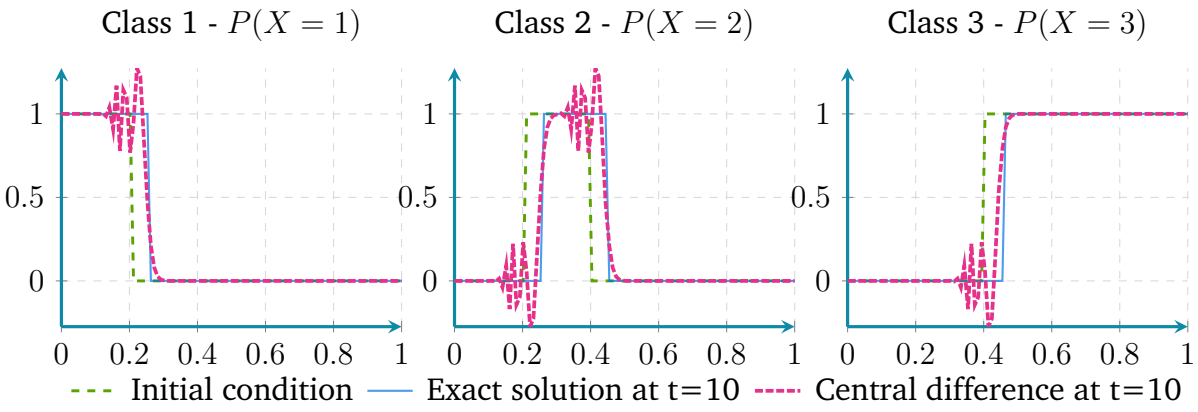


Figure 6.4: The advection of probabilities using central finite differences discretisation presents a dispersion effect

probability maps, where the operator $\mathcal{L}$ is written as follows:

$$\mathcal{L}\left(P_i\right) = v \cdot \partial_x P_i, \quad i \in \{1, 2, \cdot, C\}.$$

This differential operator is linear and have non-zero positive derivative order. Therefore, the sum of probabilities is conserved over time and remains equal to the initial value. This property is illustrated numerically in Fig. 6.5 and Fig. 6.7, and holds even in scenarios where the discretisation scheme introduces some diffusion or dispersion effects during the resolution process.

6.4.2 Non-negativity and bound preservation

The Section 3.2 presents details about two differents spatial discretisation schemes, the central finite difference and the first-order upwind scheme. The first one is a second order accurate scheme but suffers from instability issues as it introduces a negative difusion expressed by the $-\kappa\partial_x^2 f$ term and a dispersion effect expressed by the $\frac{\delta x^2}{6} v\partial_x^3 f$ term in the following equation:

$$\partial_t f + \tilde{v}\partial_x f = -\kappa\partial_x^2 f + \frac{\delta x^2}{6} v\partial_x^3 f + \mathcal{O}(\delta x^2), \tag{6.4}$$

where $\tilde{v} = v - \frac{\delta t}{2}\partial_t v + \frac{\delta t}{2} v\partial_x v$ is the effective velocity field, $\kappa = \frac{\delta t}{2} v^2$ is the diffusion coefficient. The Fig. 6.4 shows these effects. While the later is a first order accurate scheme, it is more robust at the discontinuities, even if it presents some diffusion effects represented by the $v_{num}\partial_x^2 f$ term in the following equation:

$$\partial_t P_i + \tilde{v}\partial_x P_i = v_{num}\partial_x^2 P_i + \mathcal{O}(\delta t^2, \delta x^2), \tag{6.5}$$

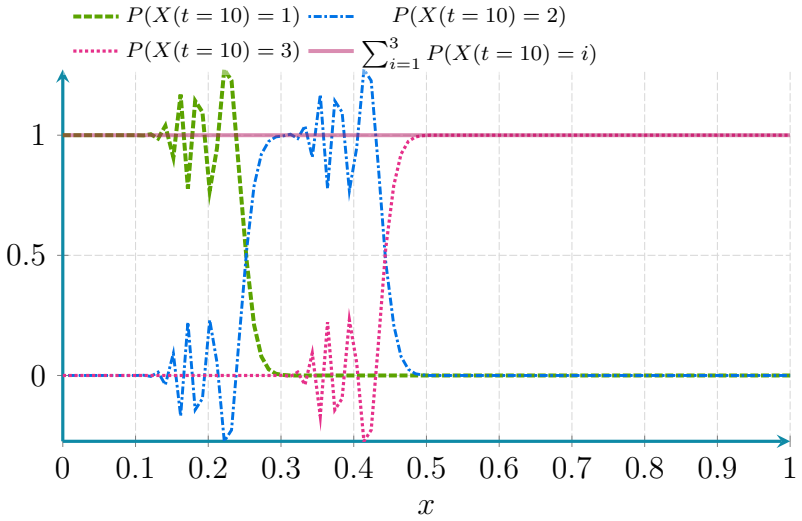


Figure 6.5: The probability conservation property is maintained even in presence of dispersion effects.

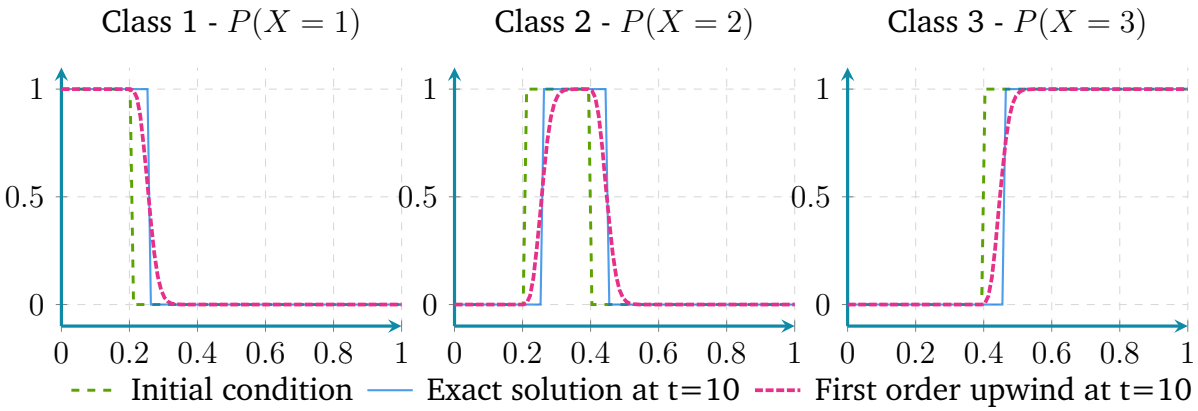


Figure 6.6: The advection of probabilities using first order upwind discretisation presents a diffusion effect

where $\tilde{v} = v - \frac{\delta t}{2}\partial_t v + \frac{\delta t}{2} v \partial_x v$, and $v_{num} = \frac{v}{2}\left(sign(v)\delta x - v\delta t\right)$. The Fig. 6.6 shows these diffusion effects.

Other schemes have been explored in practice, such as the second-order upwind scheme, which uses a three-point stencil instead of a two-point stencil as in the first-order upwind scheme (see Eq. (3.9)). The spatial derivative is approximated as follows:

$$\partial_x f(t, x_i) = \frac{1}{2\delta x} \begin{cases} 3f(t, x_i) - 4f(t, x_{i-1}) + f(t, x_{i-2}) & \text{if } v \geq 0, \\ -f(t, x_{i+2}) + 4f(t, x_{i+1}) - 3f(t, x_i) & \text{if } v < 0, \end{cases} \tag{6.6}$$

where v is the velocity field. This scheme is a second-order accurate scheme that is more robust than the central finite difference scheme. However, it introduces the same level of dispersion as the central finite difference scheme.

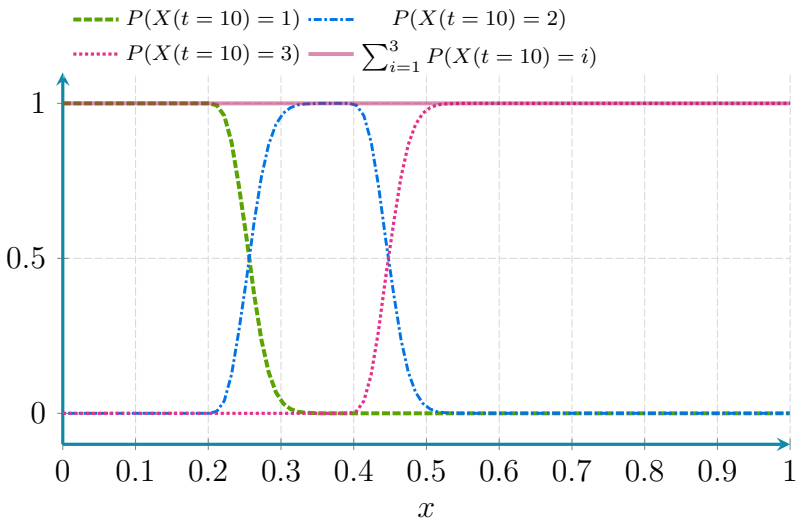


Figure 6.7: The probability conservation property is maintained even in presence of diffusion effects

To summarise, we have shown that the probability conservation property is maintained even in the presence of dispersion or diffusion effects introduced by the spatial discretisation schemes. We have also shown that the non-negativity and bound preservation properties are not always satisfied depending on the discretisation scheme used. In practice, the first-order upwind scheme is preferred due to its robustness at discontinuities, even if it introduces some diffusion effects.

For time integration, the fourth-order Runge-Kutta method is preferred due to its accuracy and stability properties, even if it is computationally more expensive than the Euler method.

During the time integration process, we perform the integration by subdividing the time step $\Delta t = 1$ (representing 15 minutes) into smaller steps $\delta t = 0.1$ to satisfy the CFL condition, which ensures the stability of the numerical solution (see Section 3.5.2 for details).

6.5 Training

The training dataset consisted of approximately three years of satellite imagery data, spanning from 2017 to 2019, with a total of 105,120 images. To refine the dataset, images representing zero cloud cover were removed, and sequences of 12 consecutive images were assembled to form the training samples. After this data cleaning step, the dataset was randomly split into training and validation sets, with 8,224 sequences used for training and 432 sequences reserved for validation. The test set was drawn from a separate dataset from the same geographic region but from 2021, ensuring temporal separation and reducing the risk of data leakage.

To enhance the diversity of the training set and mitigate potential overfitting to the cloud patterns typical of Western Europe, we applied a series of random data augmentation techniques. These included simple transformations such as rotations of 90, 180, and 270 degrees. This approach increased the effective size of the training dataset and improved the model's ability to generalise by learning a broader range of cloud motion patterns.

The models were implemented using the PyTorch framework, leveraging its flexibility and performance advantages for deep learning tasks. The training process used the cross-entropy loss function (defined in Eq. (2.35)).

Training was conducted using gradient-based optimisation methods, specifically the Adam optimizer [Kingma and Ba, 2015], configured with a learning rate of 10^{-3}. A batch size of 4 with 16 gradient accumulation steps was employed, effectively simulating a batch size of 64, allowing for efficient use of GPU memory. The model was trained over 30 epochs on a single Nvidia A100 GPU, providing the computational power necessary to handle the extensive dataset and the complex training process.

6.6 Experimental setup

To benchmark our models, we included established baselines for comparison. One key baseline is the classical U-Net [Ronneberger et al., 2015] (refer to Section 2.10.1 for details on the U-Net architecture), which serves as a common reference in the literature for image segmentation tasks [e.g. Ayzel et al., 2020, Berthomier et al., 2020, Trebing et al., 2021]. The U-Net architecture features a contracting path of convolutional and pooling layers that extract hierarchical features, followed by an expansive path of convolutional and upsampling layers that reconstruct the segmentation map. The U-Net was applied iteratively to predict future states from the past observations, as represented in Fig. 6.8, generating sequential predictions for multiple future time steps.

We also compared our models against EXIM (Extrapolated Imagery), a kinematic extrapolation product developed by EUMETSAT within their NWCSAF/GEO suite [García-Pereda et al., 2019]. EXIM leverages motion vectors derived from satellite imagery to forecast future cloud positions, offering a practical benchmark for extrapolative techniques in cloud cover forecasting.

Additionally, we evaluated the models against the "Persistence" baseline, which predicts future cloud cover based on the most recent observation. This simple yet effective approach leverages the relatively slow evolution of cloud formations, making it a strong competitor in short-term nowcasting scenarios.

All models were tested on 1,000 satellite image samples collected over France from January to October 2021, allowing for a robust comparison across different seasons and

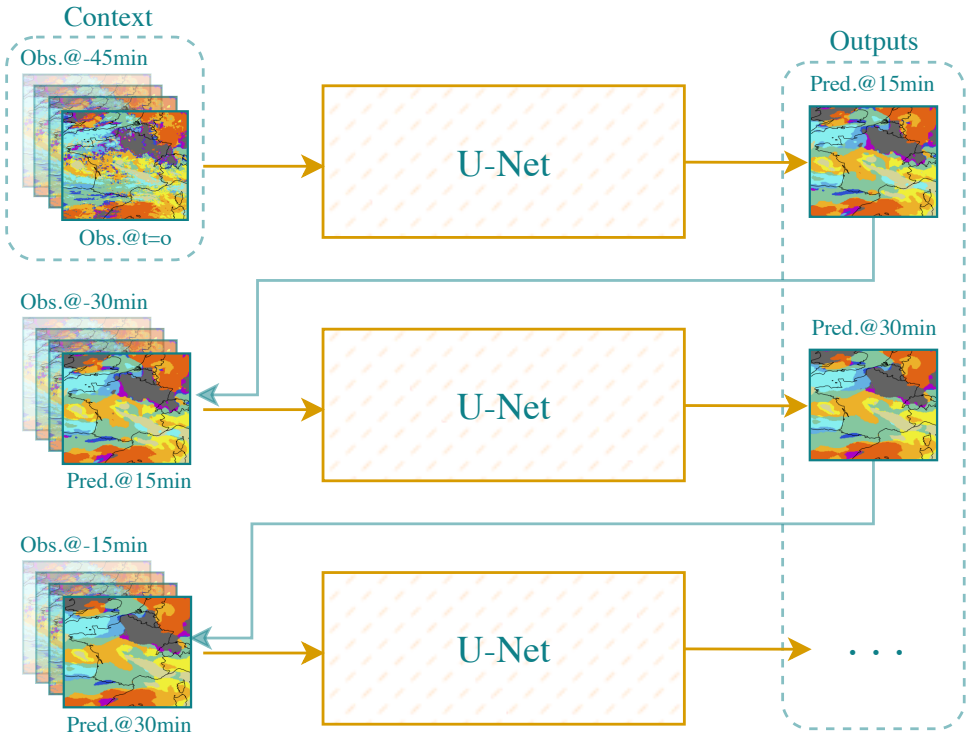


Figure 6.8: The U-Net-based architecture considered in the comparison. A U-Net of type Fig. 2.11 is applied iteratively to predict the next state given the previous ones.

cloud patterns. The selected period and region ensure that the evaluation captures a wide range of meteorological conditions, from clear skies to dense cloud cover, providing a fair assessment of each model's forecasting capabilities.

6.7 Standard classification metrics

To evaluate the performance of the models, we used standard classification metrics. These metrics included accuracy, precision, recall, F1 score and critical success index (CSI), also known as Intersection over Union (IoU) or Jaccard Index [Gilbert, 1884]. These metrics provide insights into different facets of the model's performance:

- Accuracy measures the overall proportion of correct predictions. It is computed for each class j using the following formula:

$$\text{Accuracy}_j = \frac{\text{TP}_j + \text{TN}_j}{\text{TP}_j + \text{TN}_j + \text{FP}_j + \text{FN}_j},$$

- Precision quantifies the proportion of correct positive predictions relative to the

total positive predictions made by the model:

$$\text{Precision}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FP}_j},$$

- Recall evaluates the proportion of actual positives correctly identified by the model:

$$\text{Recall}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FN}_j},$$

- F1 Score provides a harmonic mean of Precision and Recall, balancing the two:

$$\text{F1}_j = \frac{2 \times \text{Precision}_j \times \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j},$$

- CSI measures the overlap between the predicted and actual classes, offering a measure of similarity between the two:

$$\text{CSI}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FP}_j + \text{FN}_j}.$$

These metrics are calculated per class, with TP (True Positives), TN (True Negatives), FP (False Positives), and FN (False Negatives) representing the counts of correctly and incorrectly classified instances.

To assess overall performance, we calculate the overall accuracy as follows:

$$\text{Accuracy} = \frac{\sum_j \text{TP}_j}{\text{Total number of cases}}.$$

For the other metrics, both macro-averaging (arithmetic mean of the class-wise metrics) and micro-averaging (aggregating over all instances) can be used. Due to the highly imbalanced label distribution in our dataset, the macro-average was preferred, as it treats all classes equally, thus mitigating the influence of dominant classes [Fernandes et al., 2020, Wang et al., 2021].

6.8 HyPhAICC-1: results

After training the HyPhAICC-1 model, we evaluated its performance on the test set, comparing it against the U-Net, EXIM, and Persistence baselines.

6.8.1 Visual impressions

Figures 6.9 shows two examples of the model's predictions, illustrating the ability to capture cloud motion patterns and predict future cloud cover states.

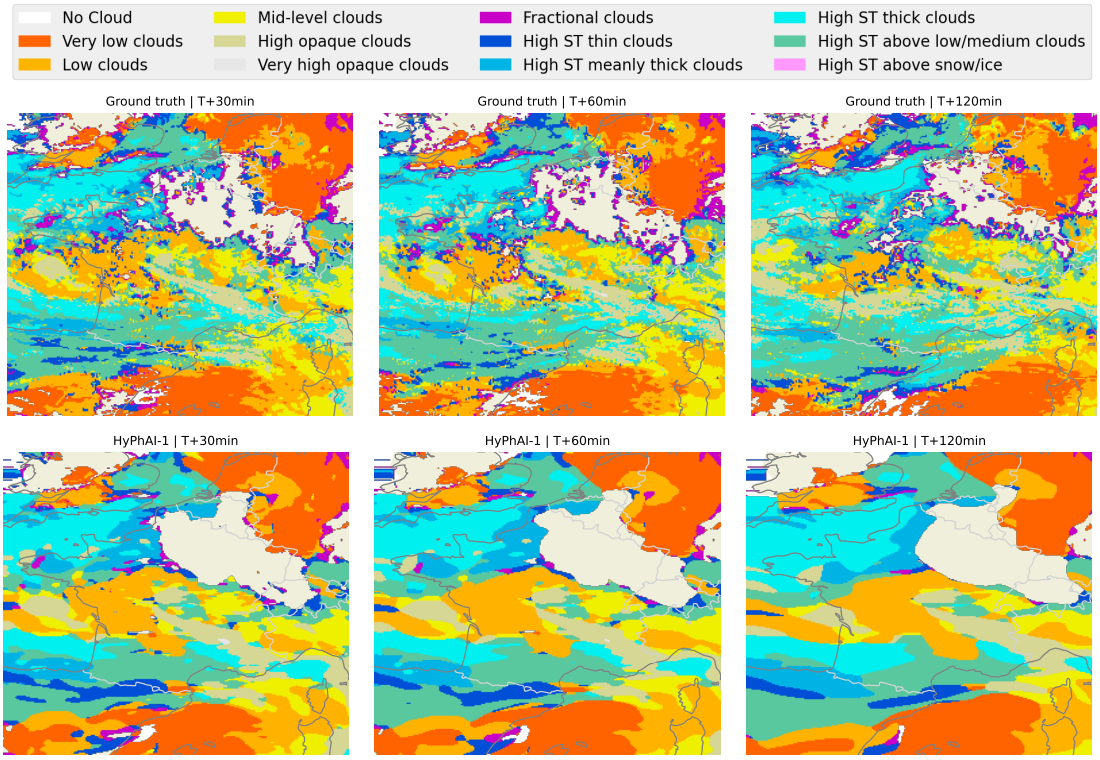


Figure 6.9: Example of the HyPhAICC-1 model's predictions. The top row shows the observations and the second row shows the model's predictions at 30, 75, and 120 minutes ahead.

In Figure 6.10, we present the estimated velocity field generated by the HyPhAICC-1 model, illustrated in Fig. 6.2. This field exhibits a high level of coherence with the observed cloud cover displacements, with exceptions in cloud-free areas, as expected. It is important to emphasize that this velocity field is derived exclusively from cloud cover images, without relying on external wind data or similar sources. This aspect adds a layer of interest, especially in the context of other applications beyond the cloud cover nowcasting. However, it is worth noting that this velocity field is not eaxctly an estimation of the wind field, regardless of the altitude, but rather a representation of the cloud motion patterns observed in the satellite images, even if it may be correlated or similar to the wind field. This question has not been deeply investigated in this study, as it would require wind data to be used as a reference for comparison, which is beyond the scope of this work. This aspect could be the subject of a future study to explore the potential of cloud motion patterns as a proxy for wind fields in meteorological applications.

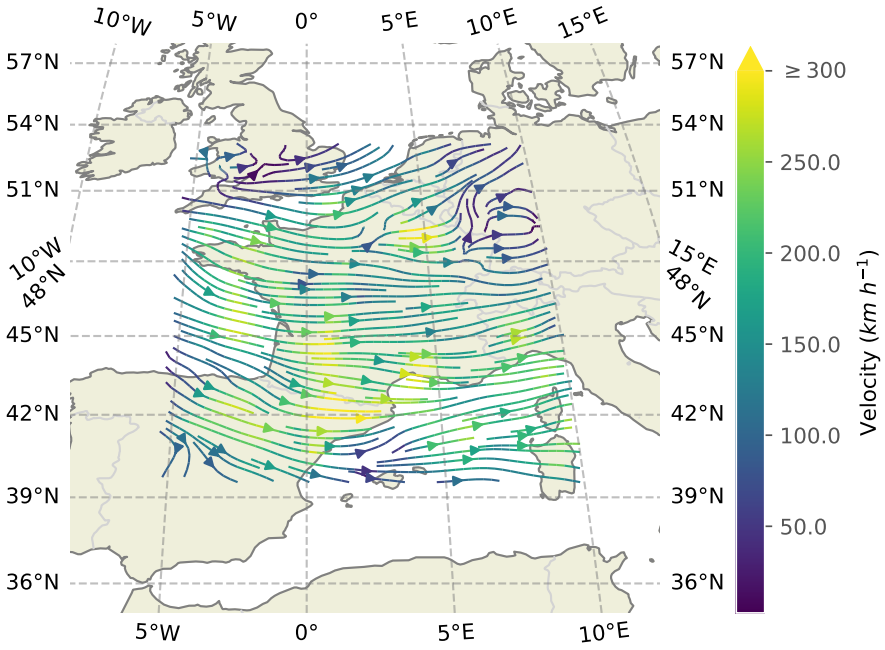


Figure 6.10: Estimated velocity field by the U-Net Xception-style used in the HyPhAICC-1 model

6.8.2 Quantitative evaluation

To quantitatively assess the model's performance, we calculated the macro-averaged F1 score, overall accuracy, and macro-averaged Jaccard index (IoU) for lead times ranging from 15 to 120 minutes. The results are presented in Figure 6.11a, Figure 6.11b, and Figure 6.11c, respectively.

The results confirms the visual impressions presented in Section 6.8.1 as the HyPhAICC-1 produces consistent predictions. In particular, we observe a consistent superiority of data-driven models, HyPhAICC-1 and U-Net, over the physics-based EXIM and Persistence baselines. Regarding our hybrid model, HyPhAICC-1, it outperforms all other models across all lead times in terms of F1 score and CSI (Jaccard index). However, the U-Net model exhibits a slightly higher accuracy than HyPhAICC-1, especially at longer lead times. This discrepancy can be attributed to the U-Net being purely data-driven and free from any physical constraints, thus giving more weight to the dominant classes at the expense of the other classes, resulting in a good accuracy but also a higher false positive rate.

In addition to the performance metrics, we also evaluated the computational efficiency of the HyPhAICC-1 model in comparaison to the U-Net, as discussed in the following sections.

6.9 Time efficiency

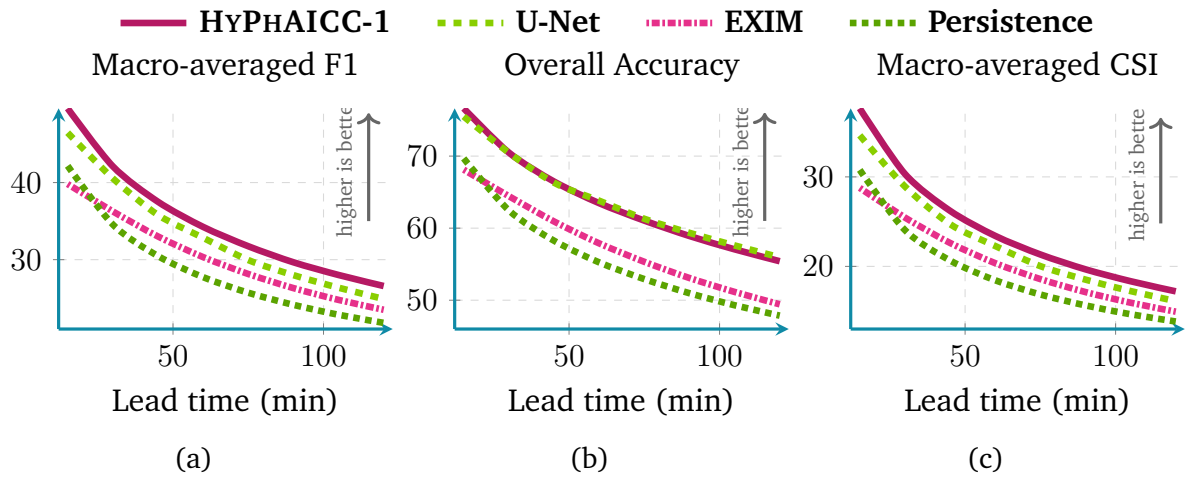


Figure 6.11: **Performance comparison between HyPhAICC-1, U-Net, EXIM, and the Persistence**. Using five metrics including averaged F1 score(%), accuracy(%) and CSI(%). These scores were computed over 1000 random samples covering France in 2021.

By including physical constraints into these hybrid models, we expect a decrease in training time compared to that of the U-Net. Indeed, Fig. 6.12 illustrates the evolution of the validation F1 score for both the U-Net and the HyPhAICC-1 model across epochs. HyPhAICC-1 converges faster than the U-Net, indeed, its convergence occurs after just about 10–15 epochs. Each epoch of the HyPhAICC-1 training takes approximately 55 minutes using a single Nvidia A100 GPU, the entire training over 30 epochs takes 27h. On the other hand, the U-Net necessitates up to 200 epochs for achieving similar performance, with each epoch taking around 23 minutes using the same hardware, which corresponds to thereabout three days of training. This difference implies that training the U-Net is significantly more expensive compared to the HyPhAICC-1. In inference mode, the hybrid models and the U-Net generate predictions within a few seconds, while EXIM's predictions are produced within 20 minutes [Berthomier et al., 2020], which is one of the main drawbacks of this product.

6.10 Data efficiency

To delve deeper into the efficiency of the proposed HyPhAICC-1 model, we carried out various experiments using different training data sizes. In each experiment, both HyPhAICC-1 and the U-Net were trained with 70 %, 50 %, 30 % and 10 % of the available training data (Fig. 6.12, Fig. 6.13). In particular, we observed a more significant performance drop for the U-Net compared to HyPhAICC-1. Interestingly, the

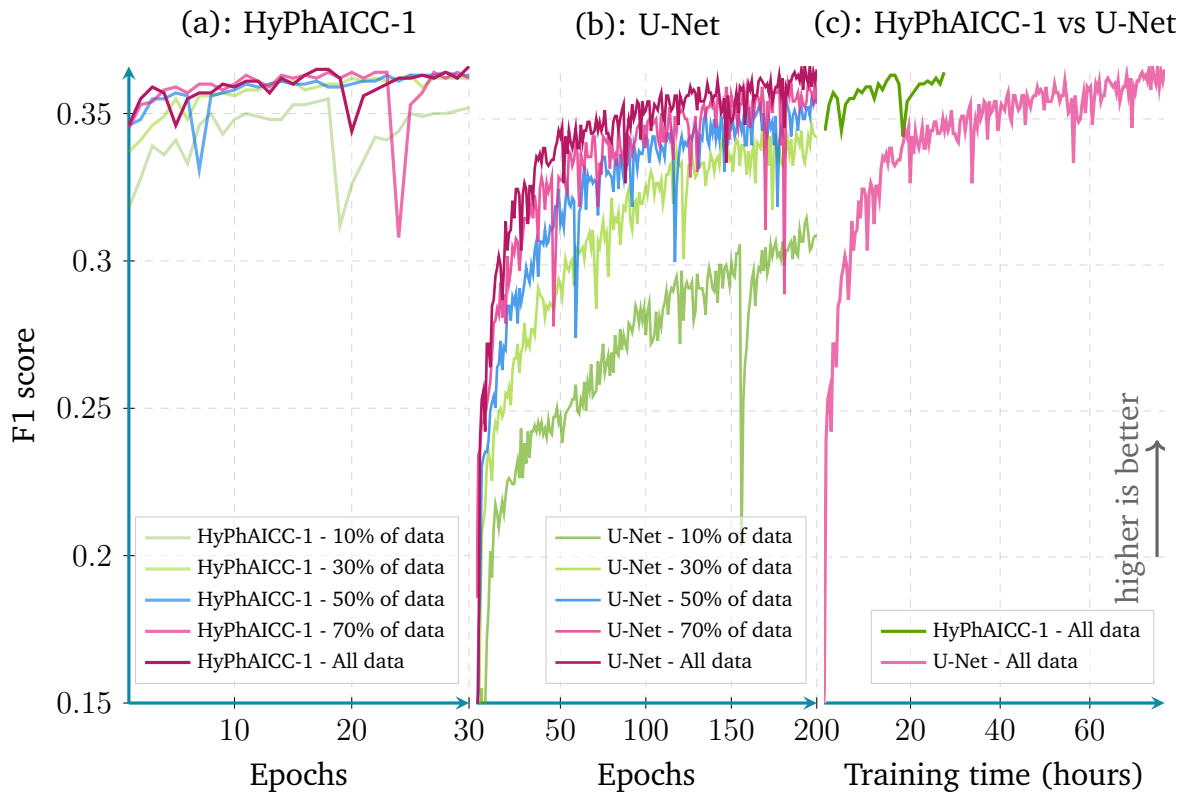


Figure 6.12: **Per epoch validation F1 score comparison between HyPhAICC-1 and the U-Net.** Scores were calculated from 100 random samples covering France (averaged over all lead times).

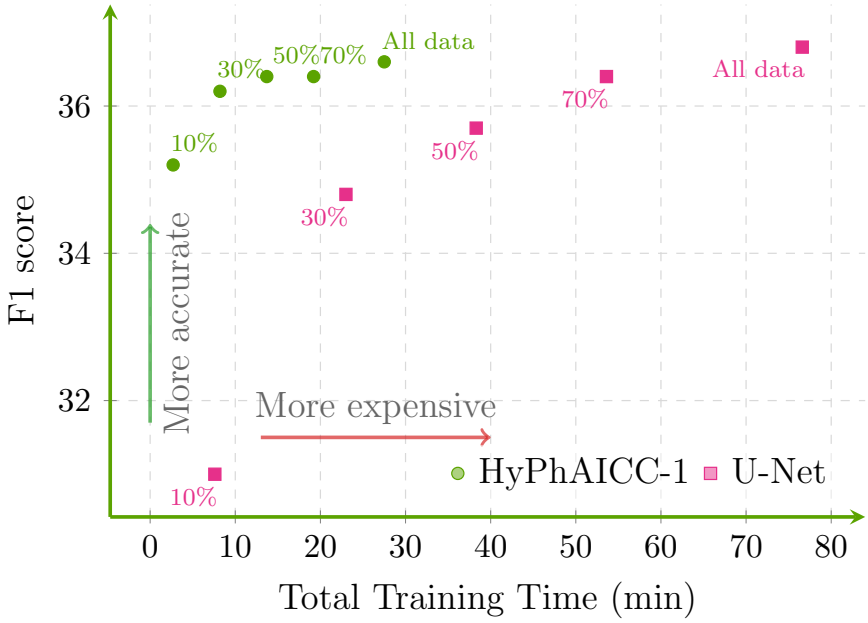


Figure 6.13: Total training time and maximum validation F1 scores over the last 5 epochs for the U-Net and HyPhAICC-1 using different training data sizes (averaged over all the lead times).

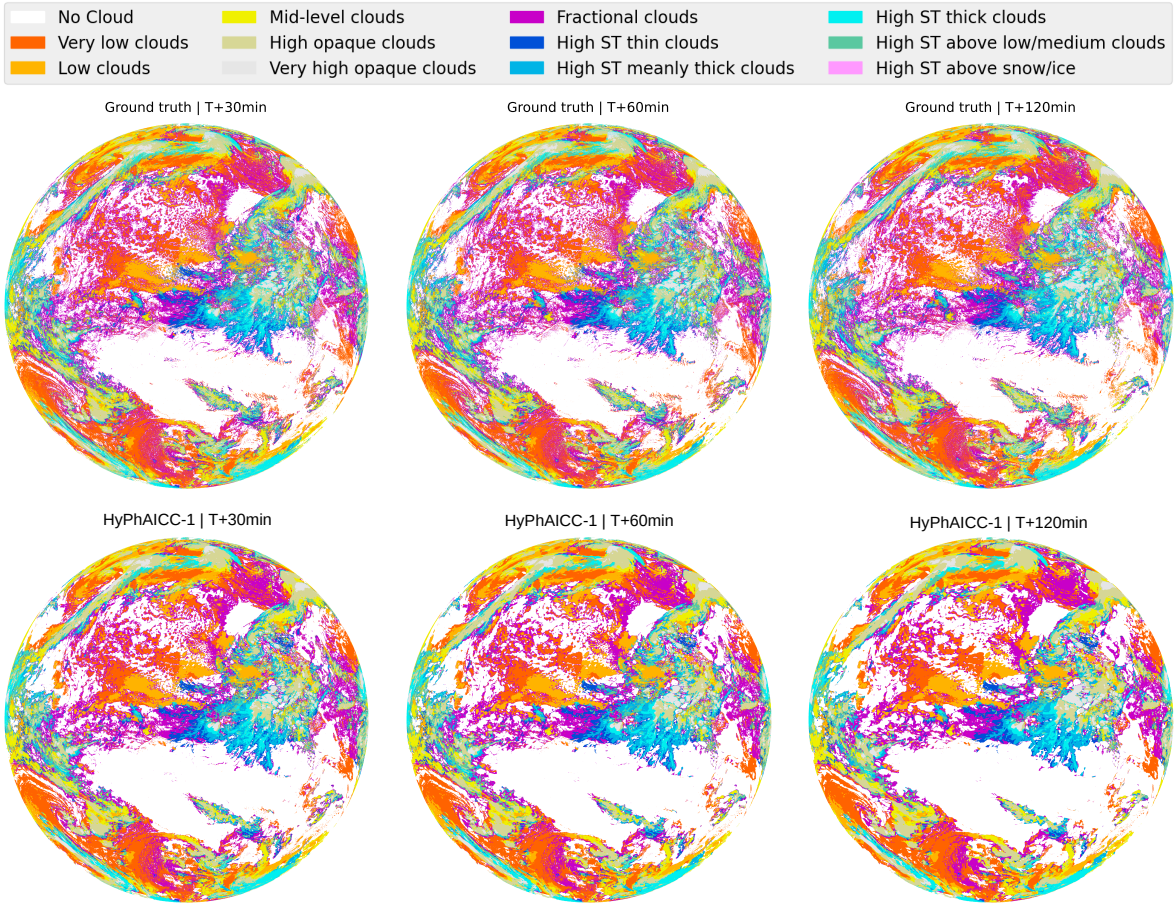


Figure 6.14: **Full disk cloud cover nowcasting predictions**. The predictions were generated by our model without any specific training on the full disk data (of size 3712×3712).

hybrid model exhibited similar performance with only 30 % of the training data as it did with the entire dataset (Fig. 6.12). This finding indicates that this hybrid model is remarkably data-efficient, capable of delivering satisfactory performance even with limited training data, which has been highlighted by other studies [Schweidtmann et al., 2024, Cheng et al., 2023]. This quality is very important, particularly for tasks with insufficient provided data.

In the next section, we extend the evaluation of the HyPhAICC-1 model by testing it on a larger domain to examine its generalisation capabilities.

6.11 Application on Earth's full disk

To check HyPhAICC-1's capabilities on broader scales after training it on a small region, we tested it on a much larger domain, an entire hemisphere of the Earth - also called a full disk - centred at 0 degrees longitude. The satellite observations of this expansive full-disk domain are of size 3712×3712, which is 210.25 times larger than the train-

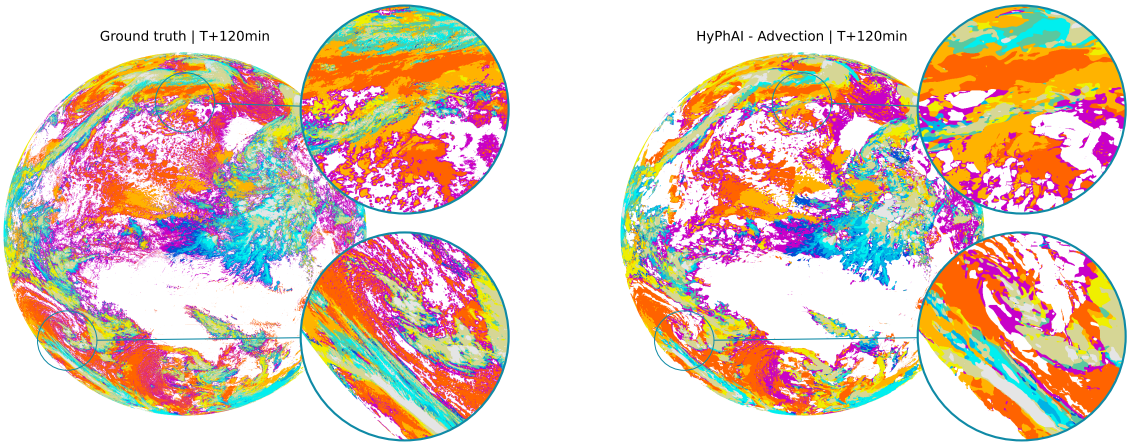


Figure 6.15: **Full disk cloud cover nowcasting predictions**. Zoomed-in views of the 120-minute observation and prediction.

ing ones. It has diverse meteorological conditions and includes projection deformations when mapped onto a two-dimensional plane, while the extreme deformations at the edge of the disk make this data less useful for operation purposes, it still provides an interesting testing ground for HyPhAICC-1's generalisation ability. In this analysis, we focus only on visual aspects. Despite the significant differences between the training domain and the full disk, we observed good qualitative forecasts of the HyPhAICC-1 model on this new domain without any specific training on it (see Fig. 6.15 and Fig. 6.14). The cloud motion estimation on the full disk was found to be visually consistent (see Fig. 6.16).

This successful transferability of the model highlights its potential robustness and suggests that the underlying principles of cloud motion captured during training are applicable across different domain sizes and different projections. Refer to Appendix 9.3.1 for more details on the robustness of the model.

Note that the model requires a data size divisible by 2^d, where d is the number of the encoder blocks within the U-Net-Xception model. Indeed, the possibility to run a model using different data sizes is one of the advantages of fully convolutional networks (FCN) as the convolution operation is independent of the input size.

Overall, HyPhAICC-1 offers an effective and cheaper approach compared to EXIM, with higher efficiency, requiring fewer data compared to the U-Net, with the potential to outperform existing models and enable more accurate and efficient weather forecasting. The ability of the HyPhAICC-1 to adapt and perform well on the full-disk data, despite being trained on a smaller domain, demonstrates the generalisation capabilities of this hybrid model. This is an important property for weather forecasting models, as it is not always possible to train a model on full-disk data due to the high computational cost.

In the next section, we try to assess the visual quality of the predictions made by the different models.

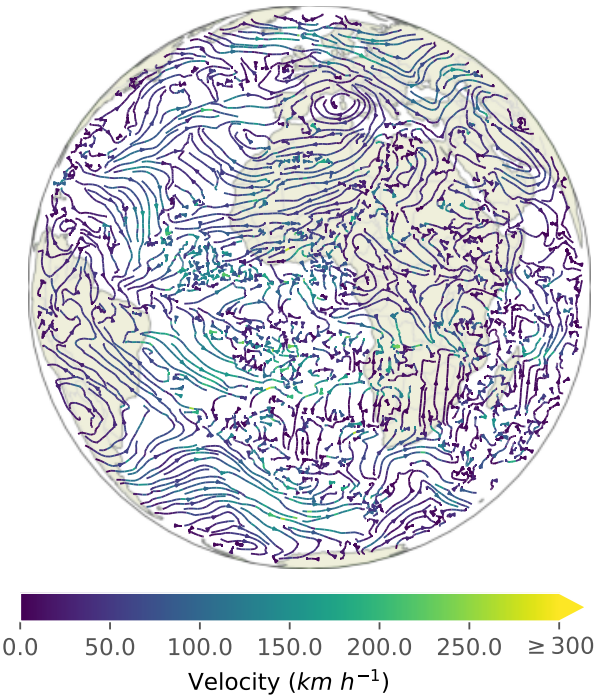


Figure 6.16: Estimated velocity field by the U-Net Xception-style used in the HyPhAICC-1 model.

6.12 Visual quality assessment

Among the other evaluation aspects that are considered when assessing the performance of weather forecasting models, the visual quality and realism of the predictions. Fig. 6.17 shows a case study of the predictions made by the different models in comparison to the ground truth. We observe that EXIM and the Persistence baseline keep the same level of details as in the ground truth, this due to the fact that Persistence is simply the last observation, while EXIM is using a kinematic extrapolation technique without any detail loss. On the other hand, the U-Net has the most loss of details. The HyPhAICC-1 model, however, maintains a good level of details even if it present some diffusiveness over time due to the first upwind spatial discretisation scheme which introduces some diffusion effects.

These observations are qualitative and subjective, and a more quantitative evaluation is needed to assess the visual quality of the predictions. However, there is no standard metric that can be used to evaluate the visual quality of the predictions. In the next section, we propose a modified version of the Hausdorff distance to assess the spatial similarity between the predictions and the ground truth and also to penalise the details loss.

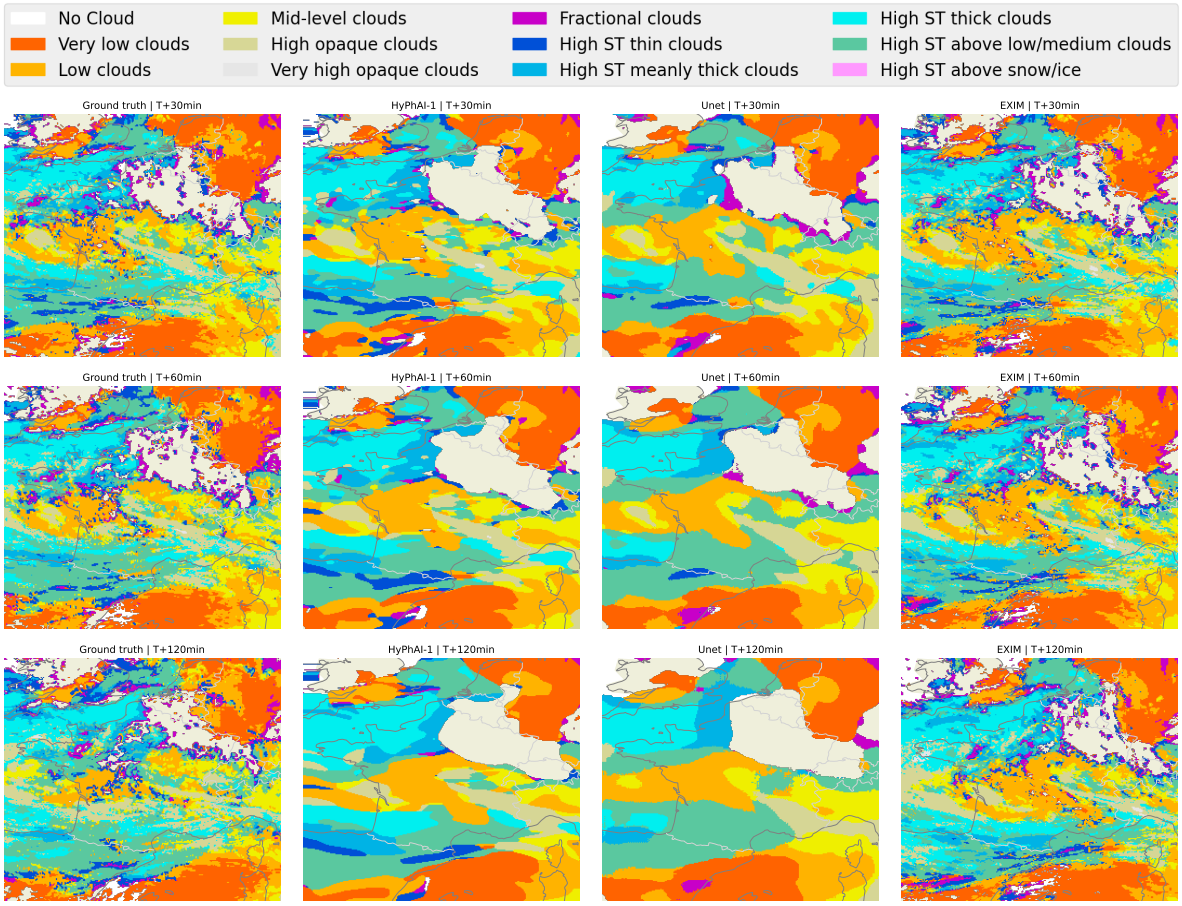


Figure 6.17: **Case study of different models' forecasts**. Left column: ground truth at different time steps; middle columns: HyPhAICC-1 and the U-Net's predictions, respectively; right column: EXIM's predictions. The light beige colour corresponds to the land areas, and 'ST' abbreviation in the legend stands for 'Semi Transparent'.

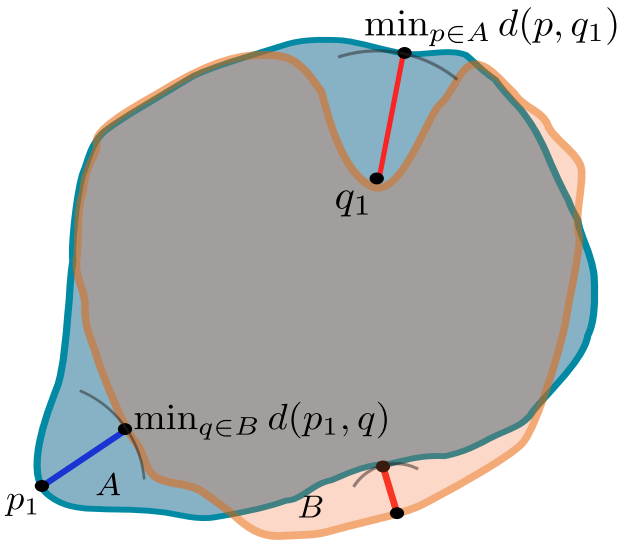


Figure 6.18: Illustration of the $\min_{p\in A} d(p, q_1)$ and $\min_{q\in B} d(p_1, q)$ quantities used to compute the Hausdorff distance; for each point, we look for the closest point in the other region.

6.12.1 Hausdorff distance

The Hausdorff distance is a widely used metric for medical image segmentation [e.g. Karimi and Salcudean, 2020, Aydin et al., 2021], this metric measures the similarity between the predicted region and the ground truth region, by comparing structures, rather than just individual pixels. It can be expressed using either Eq. (6.7) or Eq. (6.8) described as follows:

$$h^1(A, B) = \frac{1}{|A|} \sum_{p\in A} \min_{q\in B} d(p, q), \tag{6.7}$$

$$h^2(A, B) = \max_{p\in A} \min_{q\in B} d(p, q), \tag{6.8}$$

where $d(p, q)$ is the Euclidean distance between p and q. The former computes the mean distance between each point A and the closest point in B, providing an overall measure of similarity. The latter measures the maximum distance between a point in A and the closest point in B (Fig. 6.18), this formulation is a more conservative measure that focuses on the largest discrepancies between the sets.

Both formulations exhibit sensitivity to the loss of small structures. Specifically, when small regions in the ground truth are non-empty while their corresponding regions in the prediction are empty, the search area expands, which increases the overall distance. We opt to limit this search region to the maximum distance traversable by a cloud. Consequently, we introduce the *restricted Hausdorff distance* (rHD) defined as

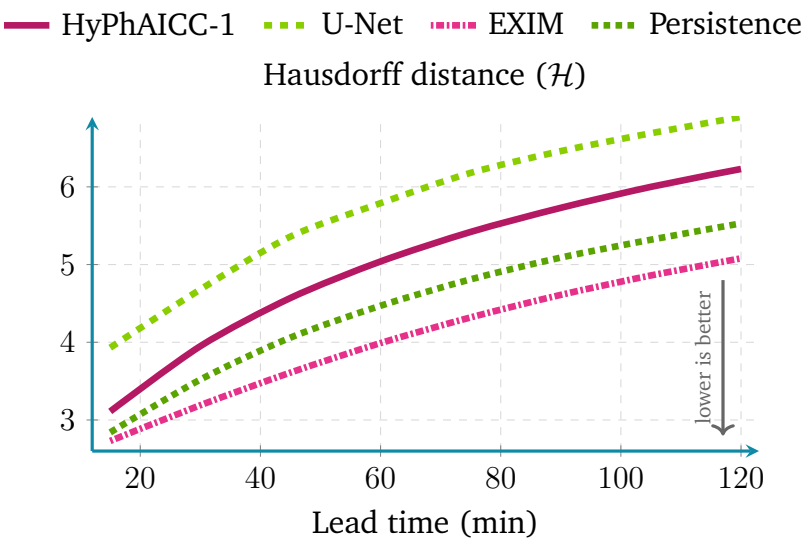


Figure 6.19: Hausdorff distance ($\mathcal{H}$) comparison between HyPhAICC-1, U-Net, EXIM, and the Persistence.

follows:

$$h^3(A,B) = \frac{1}{|A|}\sum_{p\in A}\min_{q\in \mathbf{B}_r(p)} d(p,q), \tag{6.9}$$

where $\mathbf{B}_r(p)$ is the ball of radius r centred at p. In our experiments, we set r to 10 pixels, which corresponds to a radius of approximately 45-50 km, corresponding to the maximum distance crossed by clouds in one time step, considering 200 $km\ h^{-1}$ as the cloud's maximum speed. This means that for each pixel in the first set, we compute the distance to the closest pixel in the second set, but only if it is within a radius of 10 pixels. This allows us to reduce the impact of small regions in the ground truth that are not present in the prediction, while still rewarding the model if it correctly predicts them.

The Hausdorff distance is a directed metric, i.e. $h^p(A,B) \neq h^p(B,A)$, thus, we consider the maximum of the two directed distances as follows:

$$\mathcal{H}(S,\hat{S}) = \max\left(h^3(S,\hat{S}), h^3(\hat{S},S)\right) \tag{6.10}$$

where S and $\hat{S}$ are the coordinates of positive pixels in the ground truth and prediction, respectively.

6.12.2 Results

Fig. 6.19 presents the rHD for the various models, confirming the qualitative observations made in Section 6.12. The Persistence model exhibits a lower rHD compared to HyPhAICC-1 and U-Net, while the EXIM model shows the highest rHD. This is because EXIM keeps the same level of details as Persistence, while providing a more accurate

prediction than just the last observation. Additionally, as observed, the HyPhAICC-1 model outperforms the U-Net in terms of rHD.

This metric demonstrates that visual aspects can be quantitatively assessed, which is crucial for evaluating weather forecasting models. However, this metric is not without flaws; a more thorough analysis is necessary to understand its limitations, improve its accuracy, and determine the optimal value for the radius parameter r.

6.13 Discussion

HyPhAICC-1's performance is not perfect. It can not predict the formation and dissipation of clouds, as well as the transition between different cloud types, even if some of these effects are captured by the velocity field; concentrated arrows in the velocity field indicate the dissipation of clouds, while diverging arrows indicate the formation of clouds. It is also worth mentioning that the data present some imperfections, as some clouds are not correctly classified, whose classification change over time and incoherently, especially at sunrise and sunset.

In order to improve the model's performance, we investigate in the next chapter the use of a source term in the advection equation to account for the formation and dissipation of clouds, as well as the transition between different cloud types.

Chapter 7 Extending the physical modelling

After presenting the first version of the hybrid model, HyPhAICC-1 and the evaluation procedure in the previous chapter. In this chapter, we present extended versions of the first hybrid model. HyPhAICC-2 includes a simple trainable source term. HyPhAICC-3 and HyPhAICC-4 include more complex source terms based on markovian transitions.

7.1 Heuristic-based source term: HyPhAICC-2

In order to take into account non-advective effects, such as the formation or dissipation of clouds, we propose to add a source term to the advection equation. There are different ways to model the source term, one possible way is to add a simple/raw source term of the following form to the advection equation:

$$\partial_t P_j = \tanh(S_j) \quad \forall j \in \{1, 2, \dots, C\}\,, \tag{7.1}$$

where S_j is a 2D map. The hyperbolic tangent activation function ($\tanh$) is used to keep the values of the source term in a range of $[-1, 1]$, preventing it from exploding. In the second version of our hybrid model, we use this simple source term, see Fig. 7.1. Therefore, the advection equation with the source term can be written as follows:

$$\partial_t P_j + \overrightarrow{V} \cdot \overrightarrow{\nabla} P_j = \tanh(S_j) \quad \forall j \in \{1, 2, \dots, C\}\,, \tag{7.2}$$

where S_j is estimated using a U-Net model (see Fig. 7.1), this U-Net model the same inputs as U-Net Xception-style model used to estimate the velocity field, i.e. the last four observations stacked on the channel axis. Its output are C 2D maps stacked on the channel axis, where each map represents the source term for the corresponding equation.

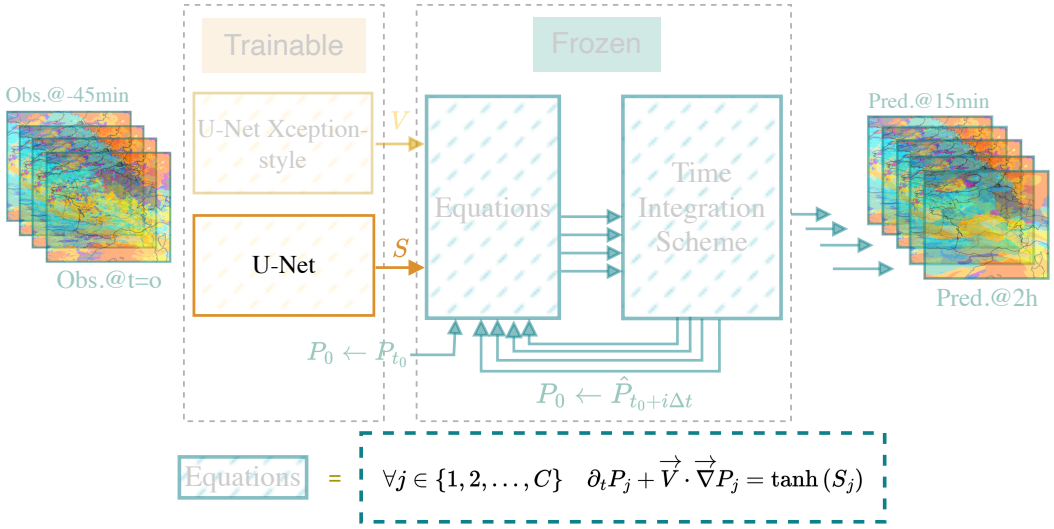


Figure 7.1: HYPHAICC-2: The second version of the proposed hybrid model. It consists of a U-Net Xception-style to estimate the velocity field and a second U-Net to estimate the source term from the last observations. We highlighted the additional parts compared to Fig. 6.2 and faded the unchanged ones.

7.2 HyPhAICC-2: results

Table 7.1: Score comparison at the 120-minute lead time (↑: higher is better, ↓: lower is better). The best scores are indicated in **bold** font.

Model	↑ F1	↑ Precision	↑ Recall	↑Accuracy	↑ CSI	↓ rHD ($\mathcal{H}$)
HYPHAICC-1	**26.6 %**	**27.5 %**	**25.9 %**	55.4 %	**17.2 %**	6.23
HYPHAICC-2	**26.5 %**	**27.6 %**	25.7 %	**57.3 %**	**17.1 %**	6.54
U-Net	24.9 %	25.6 %	24.5 %	56.0 %	16.1 %	6.90
EXIM	23.5 %	23.5 %	23.6 %	49.4 %	14.9 %	**5.08**
Persistence	21.8 %	21.9 %	21.8 %	47.9 %	13.8 %	5.53

Table 7.1 and Fig. 7.2 show that HyPhAICC-2 is slightly better in terms of precision and accuracy than HyPhAICC-1, and both of these hybrid models significantly outperform U-Net in terms of F1 score, precision, and CSI, and perform similarly in terms of accuracy and recall.

Figure 7.3 presents a case study comparing HyPhAICC-1, HyPhAICC-2, U-Net, EXIM, Persistence, and the ground truth. It is evident that HyPhAICC-2 loses some details in its predictions compared to HyPhAICC-1, though it still retains more detail than the U-Net model. This loss is attributed to the inclusion of a learned statistical component—the source term—which follows a similar trend as the U-Net by favouring the most probable cloud types in the training dataset.

This observation is supported by the rHD metric, as shown in Fig. 7.2 and Table 7.1.

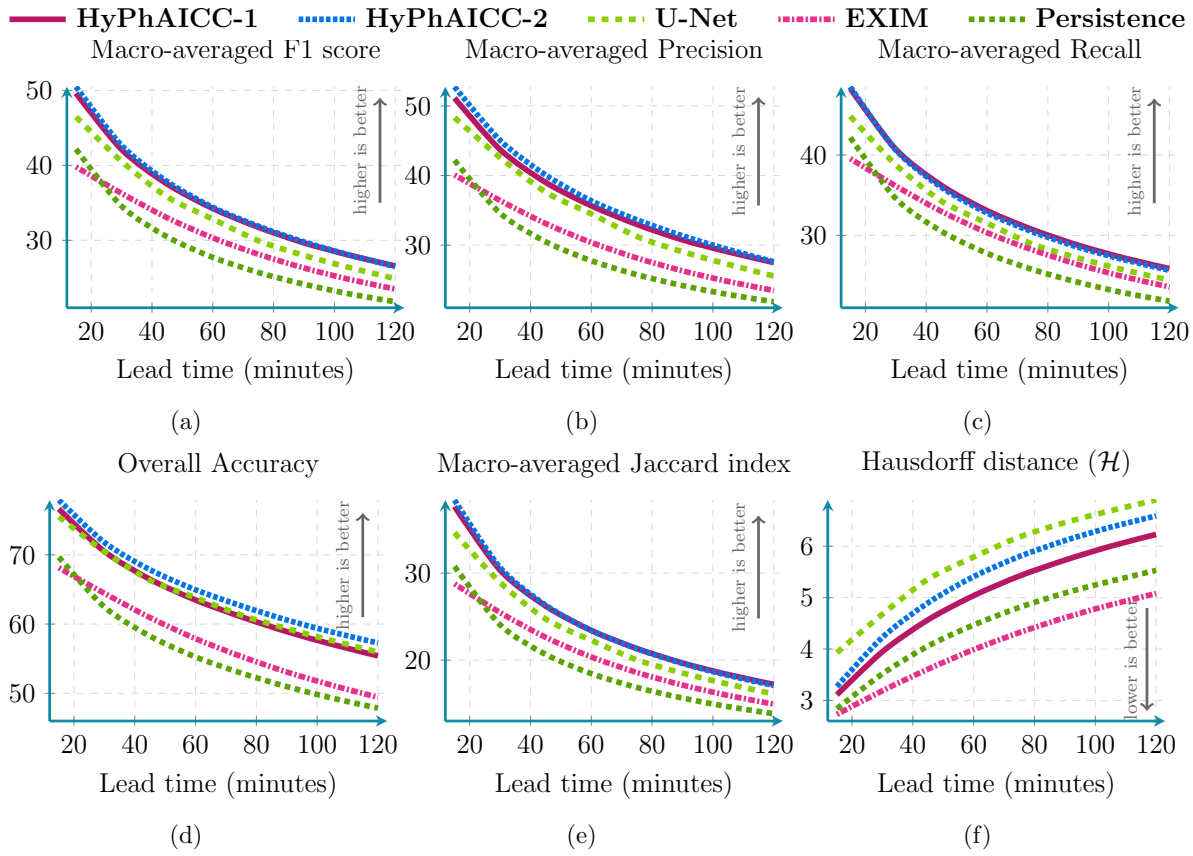


Figure 7.2: **Performance comparison between HyPhAICC-1, HyPhAICC-2, U-Net, EXIM, and the Persistence baseline**. Using five metrics including averaged F1 score(%), precision(%), recall(%), accuracy(%), CSI(%) and the rHD (defined in Eq. (6.10)). These scores were computed over 1000 random samples covering France in 2021. See Fig. 9.2 for confidence intervals.

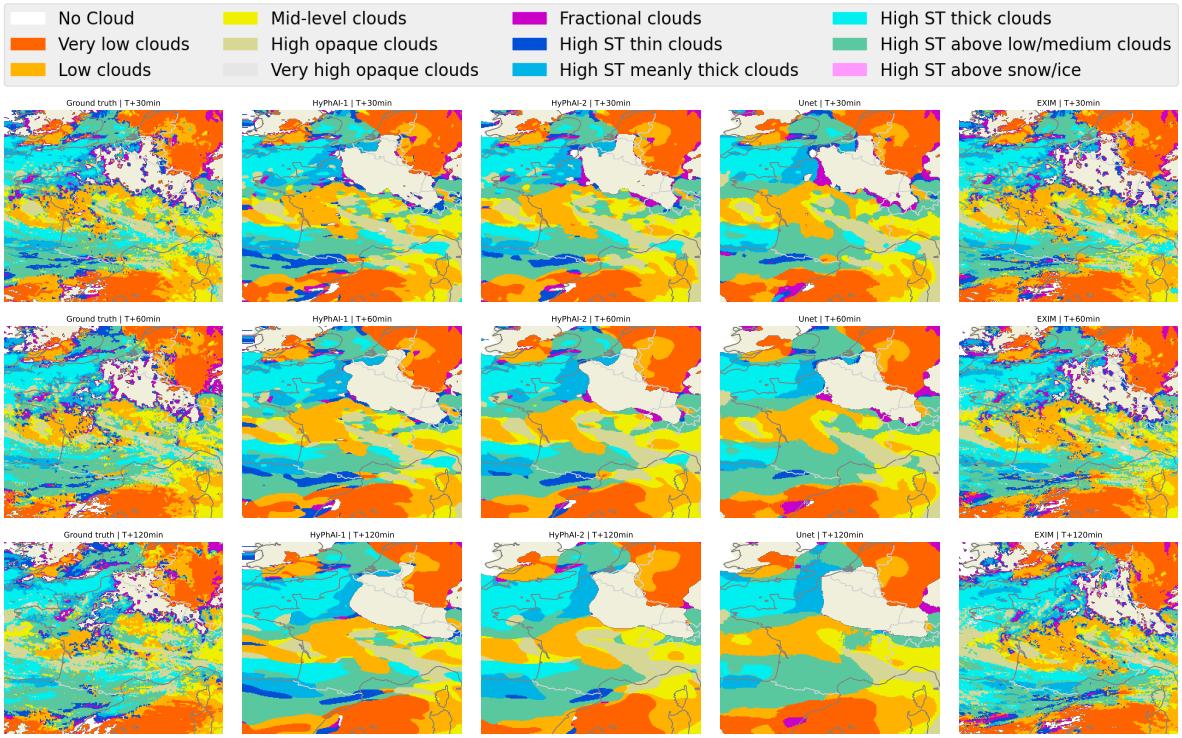


Figure 7.3: **Case study of different models' forecasts**. Left column: ground truth at different time steps; middle columns: HyPhAICC-1, HyPhAICC-2 and the U-Net's predictions, respectively; right column: EXIM's predictions. The light beige colour corresponds to the land areas, and 'ST' abbreviation in the legend stands for 'semi transparent'.

Ultimately, the HyPhAICC-2 model sacrifices some detail in exchange for improved quantitative performance, a trade-off commonly observed in machine learning models when probabilistic loss functions such as cross-entropy and MSE are used.

While this simple source term can capture some of the non-advective effects, it does not take into account the probabilistic nature of the cloud cover data. In particular, it does not preserve properties discussed in Sect. 6.4. Even if we still managed to train the model, we believe that it is important to model a more adapted and 'clean' source term.

To address this issue, we propose to model the source term using Markovian transition matrices between the different types of cloud cover. This allows the transfer of probabilities between the different types of cloud cover, including from and to the class "No cloud".

7.3 Markov-based modelling of the source term

7.3.1 Fundamentals of Markov chains

Markov chains are foundational tools in the study of stochastic processes, providing a way to model systems that transition between states in a probabilistic manner. A Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ that represent the states of a system at discrete time steps. The defining feature of a Markov chain is the *Markov property*, which states that the probability of transitioning to the next state depends only on the current state, not on the sequence of events that preceded it. This is mathematically expressed as:

$$P(X_{n+1} = s_j \mid X_1 = s_1, X_2 = s_2, \ldots, X_n = s_i) = P(X_{n+1} = s_j \mid X_n = s_i) \tag{7.3}$$

where s_i and s_j are possible states, or values, of the random variable X, the set of all possible states is denoted by S.

The state space S of a Markov chain could be finite or countably infinite. For a Markov chain with a finite state space $S = \{s_1, s_2, \ldots, s_N\}$, the transitions between states are typically represented using a transition matrix Π. The transition matrix P is an $N \times N$ matrix where each entry $\Pi_{i,j}$ represents the probability of transitioning from state x_i to state x_j in the next time step. The transition matrix is defined as follows:

$$\Pi = \begin{pmatrix} P(X_{n+1} = s_1 \mid X_n = s_1) & P(X_{n+1} = s_2 \mid X_n = s_1) & \ldots & P(X_{n+1} = s_N \mid X_n = s_1) \\ P(X_{n+1} = s_1 \mid X_n = s_2) & P(X_{n+1} = s_2 \mid X_n = s_2) & \ldots & P(X_{n+1} = s_N \mid X_n = s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(X_{n+1} = s_1 \mid X_n = s_N) & P(X_{n+1} = s_2 \mid X_n = s_N) & \ldots & P(X_{n+1} = s_N \mid X_n = s_N) \end{pmatrix}$$

Each row of the transition matrix must sum to 1, as they represent the total probability of transitioning from a given state to any other state in the next time step:

$$\sum_{j=1}^{N} \Pi_{i,j} = 1 \quad \forall i$$

The behaviour of a markovian process depends on the initial distribution, denoted π. The initial distribution is typically represented as a row vector:

$$\pi^{(0)} = [P(X_1 = s_1), P(X_1 = s_2), \ldots, P(X_1 = s_N)]$$

In case where the initial starting state is known then the initial distribution is a one-hot vector (refer to 2.9).

Using the transition matrix, the probability distribution at time n and the markov

property (Eq. (7.3)), the probability distribution at time $n+1$ can be computed as follows:

$$\pi^{(n+1)} = \pi^{(n)} P \tag{7.4}$$

It is important to note that the row vector-matrix multiplication is the usual notation in mathematics, it is equivalent to the following matrix-column vector multiplication:

$$\pi^{(n+1)^\intercal} = \Pi^\intercal \pi^{(n)^\intercal}$$

which will be used in the numerical implementation in the next section (7.3.2).

Definition 1. *A Markov chain is said to be irreducible if it is possible to reach any state from any other state in a finite number of steps. Which means that the chain cannot be divided into separate graphs or components that are unreachable from each other. Irreducibility ensures that the chain can explore all possible states over time.*

Definition 2. *A Markov chain is aperiodic if the greatest common divisor of the lengths of all cycles in the chain is 1. In other words, we cannot predict when the chain will return to a given state. Aperiodicity ensures that the chain does not get stuck in a loop, allowing it to explore the state space freely.*

Definition 3. *A Markov chain is said to have a stationary distribution if there exists a probability distribution π^∞ over the states such that, after applying the transition matrix, the distribution remains unchanged:*

$$\pi^\infty \Pi = \pi^\infty$$

This stationary distribution represents the long-term behaviour of the Markov chain, where the probabilities of being in each state stabilize over time.

Proposition 2. *If a Markov chain is irreducible and aperiodic, then it has a unique stationary distribution.*

Proposition 3. *Given the initial distribution $\pi^{(0)}$ and the transition matrix P, the distribution of the states after n steps, $\pi^{(n)}$, can be found by repeated multiplication:*

$$\pi^{(n)} = \pi^{(0)} \Pi^n$$

As n increases, $\pi^{(n)}$ will converge to the stationary distribution π, assuming the chain is ergodic.

Example: Consider a simple weather model with two states: sunny and cloudy. The transition matrix might look like this:

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

This matrix indicates that if it is sunny today, there is an 80% chance it will be sunny tomorrow and a 20% chance it will be cloudy. If it is cloudy today, there is a 70% chance it will continue to be cloudy and a 30% chance it will be sunny tomorrow.

If the initial distribution is $\pi^{(0)} = (0.5, 0.5)$, meaning there is an equal chance of it being sunny or cloudy today, then the distribution after one time step can be computed as follows:

$$\pi^{(1)} = \pi^{(0)} P = (0.5, 0.5) \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix} = (0.55, 0.45)$$

Over time, this distribution will converge to the stationary distribution, which represents the long-term probabilities of sunny and cloudy days in this model:

$$\Pi^{\infty} = \lim_{n \to \infty} \Pi^n = \begin{pmatrix} \frac{7}{9} & \frac{2}{9} \\ \frac{7}{9} & \frac{2}{9} \end{pmatrix}$$

applied to the initial distribution, the stationary distribution is:

$$\pi^{\infty} = \pi^{(0)} \Pi^{\infty} = (0.5, 0.5) \begin{pmatrix} \frac{7}{9} & \frac{2}{9} \\ \frac{7}{9} & \frac{2}{9} \end{pmatrix} = \left(\frac{7}{9}, \frac{2}{9} \right)$$

i.e. in the long run, there is a 78% probability of sunny days and a 22% probability of cloudy days.

In the next section, we will use this concept to model the transitions between different types of cloud cover in the source term of the advection equation.

7.3.2 Markov-based source term: HyPhAICC-3

In this section, we propose a third version of the hybrid model, called HyPhAICC-3, which uses Markov chains to model the source term in the advection equation. This source term is expressed as follows:

$$\partial_t P_j = \sum_{i=1}^{C} \mathbf{\Lambda}_{j,i} P_i \quad \forall j \in \{1, 2, \ldots, C\}, \tag{7.5}$$

where

$$D^2 \mapsto \mathbb{R}$$
$$x \mapsto \mathbf{\Lambda}_{j,i}(x)$$

is a map representing the transition rates between i and j, D is spatial domain, here it represents the pixels of an image of size 256×256.

For a small fixed time step Δt, Eq. (7.5) can be approximated as:

$$P(t+\Delta t) \approx P(t) + \Delta t \mathbf{\Lambda} P(t),$$
$$P(t+\Delta t) \approx (I + \Delta t \mathbf{\Lambda})\, P(t),$$

in order to model a markovian process, the matrix $(I + \Delta t \mathbf{\Lambda})^\intercal$ involved in the transition step should be stochastic, this is the transition matrix $\mathbf{\Pi}$ defined in Section 7.3.1. Therefore, the matrix $\mathbf{\Lambda}$ can be computed as follows:

$$\Lambda(\mathbf{x}) = \frac{\mathbf{\Pi}(\mathbf{x})^\intercal - I}{\Delta t}.$$

The matrix $\mathbf{\Lambda}$ represents what is known as the infinitesimal generator of a Markov process; it defines the transition rates between the various states of the process.

This modelling ensures that the probabilistic properties are maintained over time.

Let's consider the following example of a source term for a three-class problem on a single point domain without considering the advection term:

$$\begin{aligned}
\frac{\partial P_X^1}{\partial t} &= \Lambda_{1,1} P_X^1 + \Lambda_{2,1} P_X^2 + \Lambda_{3,1} P_X^3, \\
\frac{\partial P_X^2}{\partial t} &= \Lambda_{1,2} P_X^1 + \Lambda_{2,2} P_X^2 + \Lambda_{3,2} P_X^3, \\
\frac{\partial P_X^3}{\partial t} &= \Lambda_{1,3} P_X^1 + \Lambda_{2,3} P_X^2 + \Lambda_{3,3} P_X^3.
\end{aligned} \tag{7.6}$$

Let's consider the following transition rates:

$$\Lambda_{1,1} = \left[\frac{-0.5}{\Delta t}\right] \qquad \Lambda_{2,1} = \left[\frac{0.5}{\Delta t}\right] \qquad \Lambda_{3,1} = [0.0]$$
$$\Lambda_{1,2} = [0.0] \qquad \Lambda_{2,2} = [0.0] \qquad \Lambda_{3,2} = [0.0]$$
$$\Lambda_{1,3} = [0.0] \qquad \Lambda_{2,3} = [0.0] \qquad \Lambda_{3,3} = [0.0]$$

The Figures 7.4 and 7.5 show the evolution of the probabilities using Eq. (7.6) with $\Delta t = 10$.

Physically, $\mathbf{\Lambda}_{j,i}(\mathbf{x})$ represents, in the context of cloud cover, the transition rate from cloud type i to cloud type j at grid point $\mathbf{x}$ and Δt represents the time step, and $I(\mathbf{x})$ denotes the identity matrix. This third version of the hybrid model,HyPhAICC-3, (see Fig. 7.6), uses this source term combined with the advection as showed in the following equations:

$$\partial_t P_j + \overrightarrow{V} \cdot \overrightarrow{\nabla} P_j = \sum_{i=1}^{C} \mathbf{\Lambda}_{j,i} P_i \quad \forall j \in \{1, 2, \ldots, C\}, \tag{7.7}$$

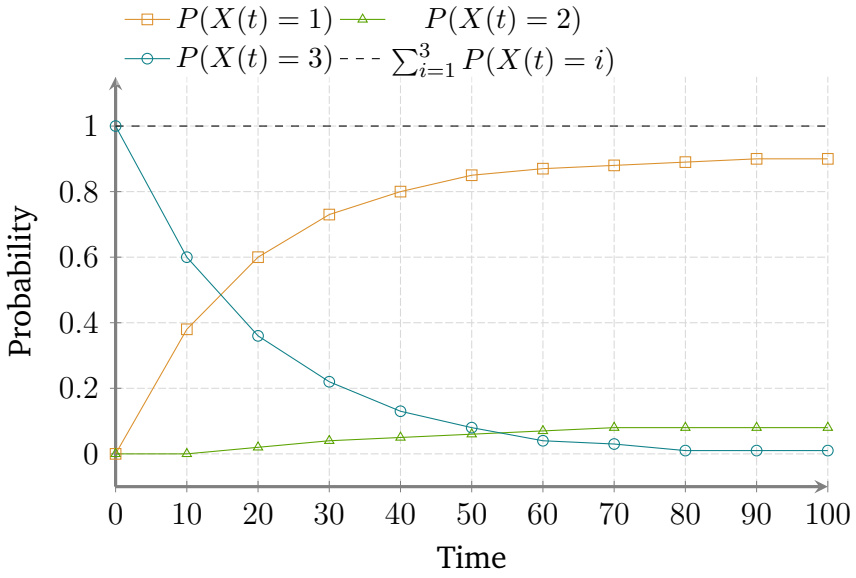
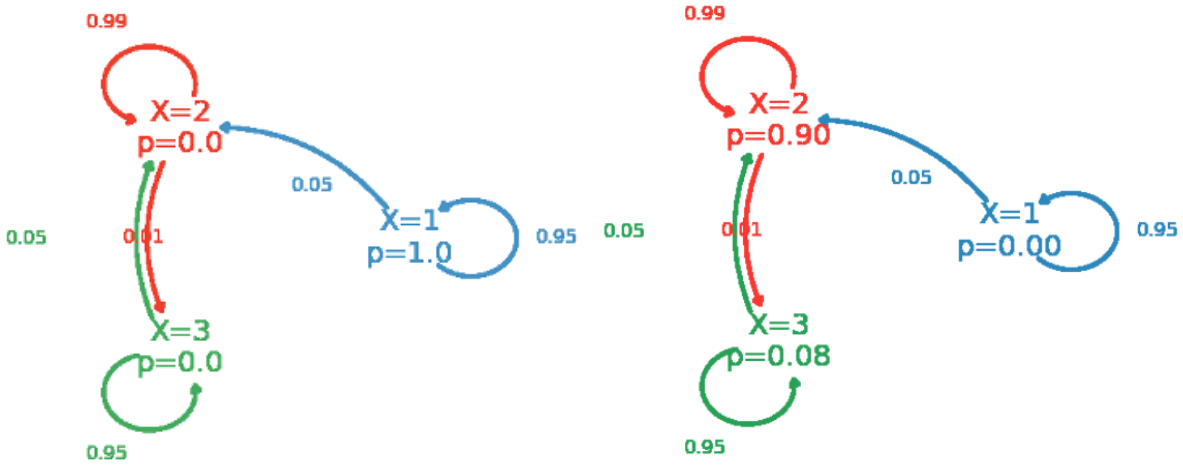


Figure 7.4: Probability evolution in the case of inter-class transitions.



(a) Initial state

(b) Final state

Figure 7.5: Graphs showing the transition rates and the class probabilities at the initial and final states.

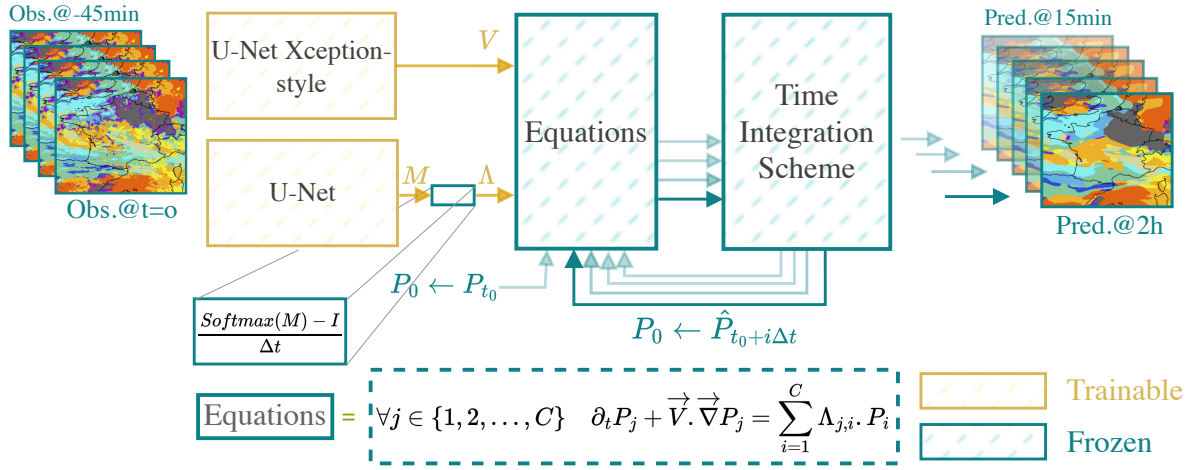


Figure 7.6: HYPHAICC-3: The third version of the proposed hybrid model. It consists of a U-Net Xception-style to estimate the velocity field and a second U-Net to estimate the per-pixel transition matrices from the last observations.

where the stochastic property of $\mathbf{\Pi}$ is ensured by construction using the *Softmax* function as follows:

$$\Pi_{i,k} = \text{Softmax}\left(M_i\right)_k = \frac{e^{M_{i,k}}}{\sum_{j=1}^{C} e^{M_{i,j}}},$$

where the matrix M is generated using a U-Net.

This representation of cloud cover dynamics offers a comprehensive description of cloud formation and dissipation. However, it increases the output dimension size of the U-Net, as a $C \times C$ transition matrix is generated for each pixel. This makes the U-Net model poorly constrained, which led to provideding the same results as the HyPhAICC-1 model using only the advection. In fact, during training, the Λ matrices in Eq. (7.7) are consistently estimated as zeros meaning that no inter-class transitions were captured. Furthermore, in our experiments, we faced difficulties in training the model due to the high dimensionality of the output space. We also noticed an increased memory usage during the training process and a significant increase in the training time.

7.3.3 It is not Fokker-Planck equation!

The Eq. (7.5) presented in the previous section is a form of the so-called master equation, which describes the evolution of the probability distribution of a discrete-time Markov chain. In the continuous limit, the master equation converges to another PDE called the Fokker-Planck equation, which describes the evolution of the probability density function of a continuous-state Markov process. Although the modelling presented in Eq. (7.5) share similarities with the Fokker-Planck equation, both equations are fundamentally different. We believe that it is important to clarify this distinction to avoid any confusion. In this section, we provide a detailed derivation of the Fokker-Planck

equation to highlight the differences between the two approaches.

Following the same path as in [Øksendal, 2010], we consider the pricing of a financial asset, such as a stock. A simple model for the stock price $S(t)$ over time might assume constant growth:

$$\frac{dS}{dt} = \mu S(t)$$

Here, $S(t)$ is the stock price at time t, and μ represents the rate of return. This model assumes that the stock price increases smoothly at a fixed rate.

However, in reality, stock prices fluctuate due to market volatility and random events. To capture this randomness, we adjust the rate of return:

$$\mu(t) = \mu + \text{"random fluctuations"}$$

In this form, the random fluctuations, let's note them as $\xi(t)$, represent the unpredictable changes in the stock price. The exact nature of these fluctuations isn't known. The modified equation becomes:

$$\frac{dS}{dt} = \mu S(t) + \xi(t)S(t) \tag{7.8}$$

The solution to this modified equation describes not a single path for the stock price, but a set of possible future price paths, reflecting the uncertainty in how the market will evolve over time. Thus, it is impossible to predict the exact future price of the stock, but we still use some statistical tools to model its behaviour.

In general, PDEs similar to Eq. (7.8) presenting random terms are called stochastic differential equations (SDEs). SDEs are part of the stochastic calculus field, which deals with systems containing some sort of randomness. The goal of this section is not to provide a comprehensive and rigorous introduction to Fokker-Planck equation, but to give a general idea of this equation in order to highlight the differences with Eq. (7.5).

In the stochastic theory, an Ito diffusion process is a markovian process of continuous state and that is almost surely continuous in time, and that is defined using Ito integrals as follows [Øksendal, 2010]:

$$X_t = X_0 + \int_0^t \mu(X_s)ds + \int_0^t \sigma(X_s)dB_s, \tag{7.9}$$

or in a differential form:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dB_t, \tag{7.10}$$

where X_t is the state of the system at time t, $\mu(X_t)dt$ is called the drift term, $\sigma(X_t)dB_t$ is called the martingale term representing random fluctuations where B_t is a Brownian motion, i.e., a stochastic process with independent and normally distributed increments

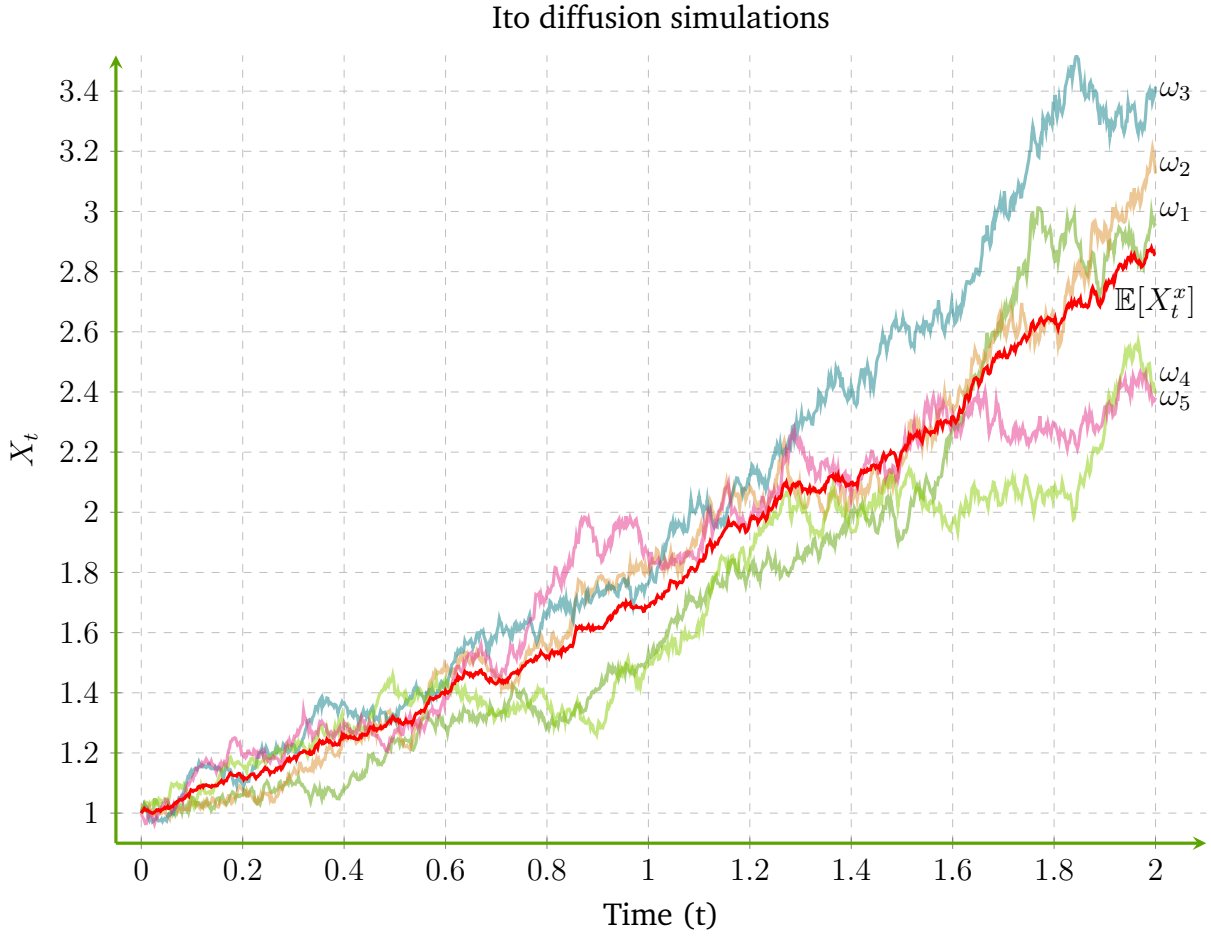


Figure 7.7: Simulation of multiple trajectories of an Ito diffusion process, using $\mu(X_t) = 0.1X_t$, $\sigma(X_t) = 0.2$, a time step $\Delta t = 2 \cdot 10^{-3}$, an initial condition $x = 1$, 1000 time steps and 5 realisations. Each curve represents a different realisation of the stochastic process over time and noted ω_i. The red line indicates the expectation of the process, calculated as the average across all trajectories. The simulation was performed using the Euler-Maruyama method (see Appendix 9.2.1 for more details).

with mean zero and variance dt [Øksendal, 2010, chap. 2] (Appendix 9.2.1 provides more details and Fig. 9.3 shows some realisations of a Brownian motion), μ and σ are in $L^2(\mathbb{R}^n)$.

This equation models the evolution of the deterministic evolution $\frac{dx}{dt} = \mu(x)$ in the presence of random fluctuations (white noise). The Fig. 7.7 shows an example of a simulation of an Ito diffusion process with $\mu(x) = 0.5x$ and $\sigma(x) = 0.2$. The goal is to determine the evolution of the probability distribution of the system over time.

Let us consider a function $g \in C_0^2(\mathbb{R}^n)$ (the space of twice-differentiable functions with compact support), called an observable, and let $Y_t = g(X_t)$. If X_t is an Ito diffusion, then Y_t is also an Ito diffusion. The Ito formula provides the shift and the martingale

terms of Y_t[Øksendal, 2010, chap. 4], and states that the SDE of Y_t is given by:

$$dY_t = \left(\mu(X_t)\partial_x g(X_t) + \frac{1}{2}\left(\sigma(X_t)\right)^2 \partial_x^2 g(X_t)\right) dt + \sigma(X_t)\partial_x g(X_t) dB_t. \tag{7.11}$$

In the stochastic frame, the observable dynamics is described through the expectation of the observable at time t given an initial condition $X_0 = x$:

$$g_t(x) = \mathbb{E}[g(X_t) \mid X_0 = x] = \mathbb{E}[g(X_t^x)], \tag{7.12}$$

Using Eq. (7.11), and the Kolmogorov backward equation, we can show that the expectation of the observable satisfies the following partial differential equation:

$$\partial_t g_t = \mathcal{L} g_t, \tag{7.13}$$

where $\mathcal{L}$ is the infinitesimal generator of the process, defined as:

$$\mathcal{L} = \mu(x)\partial_x + \frac{1}{2}\sigma^2(x)\partial_x^2. \tag{7.14}$$

Assuming the initial probability distribution $\mathbb{P}_0 = \mathbb{P}(X_0)$ has a density $x \mapsto P_0(x)$, meaning $\mathbb{P}(X_0 \in A) = \int_A P_0(x)\, dx$, we demonstrate in Appendix 9.2.2 that the expectation of the observable g_t with respect to the initial probability density, i.e.

$$\mathbb{P}_0[g_t] = \int_{\mathbb{R}^n} g_t(x) P_0(x)\, dx, \tag{7.15}$$

is equivalent to an expectation of the initial observable g_0 under a new probability density $x \mapsto P_t(x)$ at time t, i.e.

$$\mathbb{P}_t[g_0] = \int_{\mathbb{R}^n} g_0(x) P_t(x)\, dx, \tag{7.16}$$

where $x \mapsto P_t(x)$ is defined as a measure of the probability density of the system at time t, i.e.

$$\mathbb{P}_t[g_0] \equiv \mathbb{P}_0[g_t], \tag{7.17}$$

and that the probability density function $x \mapsto P_t(x)$ satisfies the following partial differential equation:

$$\partial_t P = \mathcal{L}^\star P, \tag{7.18}$$

where $\mathcal{L}^\star$ is the formal adjoint of $\mathcal{L}$. Using simple integration by parts as explained in Appendix 9.2.3, we can show that the formal adjoint operator $\mathcal{L}^\star$ is given by:

$$\mathcal{L}^\star(\cdot) = -\partial_x\left(\mu\cdot\right) + \frac{1}{2}\partial_x^2\left(\sigma^2\cdot\right) \tag{7.19}$$

Thus, Eq. (7.18) can be rewritten as:

$$\partial_t P + \nabla \cdot (\mu P) = \frac{1}{2}\Delta\left(\sigma^2 P\right) \tag{7.20}$$

This equation is known as the Fokker-Planck equation.

It should be noted that the variable x in ∂_x and ∂_x^2 in Eq. (7.19) does not represent the spatial dimension, but the state of the system. For example, in the context of climate modelling, the temperature fluctuations in a given location could be modelled using the Fokker-Planck equation to take into account both the deterministic evolution of the temperature (seasonal variations, etc.) and the unpredictable weather/climate perturbations. In this case, $\mathcal{L}^\star$ would be written as:

$$\mathcal{L}^\star = -\partial_T\left(\mu\cdot\right) + \frac{1}{2}\partial_T^2\left(\sigma^2\cdot\right),$$

where $\mu : T \mapsto \mu(T)$ and $\sigma : T \mapsto \sigma(T)$ are functions of the temperature random variable T.

In summary, the Fokker-Planck equation describes the evolution of the probability distribution of a stochastic process over time. This equation is different from the equation presented in Eq. (7.5), which describes the evolution of the probability distribution of a discrete-time Markov chain.

7.3.4 Reducing the training time: which convolution to use?

In this part, we delve into the implementation details of the proposed hybrid models, and we discuss the strategies used to reduce the training time.

The numerical scheme is the most computationally expensive part of the model, as it requires the resolution of the advection equation at each time step. The resolution of the advection equation involves the computation of the spatial derivatives of each of the C equations. This process is repeated ten times for each time step, as the time step is subdivided into ten smaller steps to satisfy the CFL condition. Thus, it is crucial to carefully consider how spatial derivatives are calculated.

As discussed in Section 3.5, the first-order upwind scheme is preferred over the central finite difference scheme due to its robustness at discontinuities. One of the drawbacks of this scheme among those already discussed is that it requires two times more computations than the central finite-difference scheme. This is because the upwind scheme requires computing the spatial derivative in both directions. This adds a significant computational overhead; however, we decided to keep this scheme in the absence of a better alternative. Therefore, the only way to reduce the training time is to effectively implement the convolutional part.

7.3.4.1 Using 2D convolutions

One simple approach is to use C convolutional layers, along x and y axis and for the two cases of the upwind scheme ($C \times 2 \times 2$ convolutional layers in total). These layers take as input a tensor of size $(B, 1, H, W)$ where:

- B is the batch size,
- 1 is the number of input channels,
- $H = 256$ is the height of the input tensor,
- $W = 256$ is the width of the input tensor.

Each of these layers uses a kernel of size $1 \times 1 \times 3 \times 3$ (Fig. 7.8 explains the shape of the convolution kernels on PyTorch). This approach is the slowest as it does not take advantage of the parallelism of the GPU and was not used in our experiments.

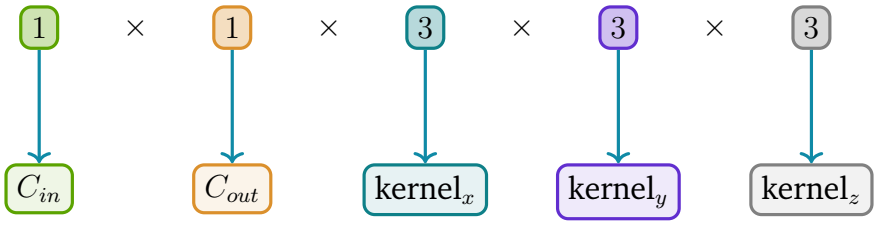


Figure 7.8: The shape of 3D convolution kernels on PyTorch. The same principle applies to 2D and 1D convolutions.

7.3.4.2 Using 3D convolutional layers

The second approach is the one initially used in our experiments. It consists of using four 3D convolutional layers for both axes and the two cases of the upwind scheme. Each of these layers takes as input a tensor of size $(B, 1, H, W, C)$ and uses a kernel of size $1 \times 1 \times 3 \times 3 \times 1$ (number of input channels, number of output channels, kernel size along the x axis, kernel size along the y axis, kernel size along the z axis). Indeed, it is a 3D convolution that does not take into account the third dimension. This approach is faster than the first one, as we remove the sequential part of the computation. However, it is not the most efficient. This approach was the first one used in our experiments.

7.3.4.3 Using depthwise 2D convolutions

The third approach is the one we used in the final version of the model. In this approach, we use four 2D convolutional layers, for both axes and the two cases of the upwind scheme, taking as input a tensor of size (B, C, H, W). Each of these layers uses C kernels of size $1 \times 1 \times 3 \times 3$ concatenated along the output channel axis, resulting in a

	CPU		GPU	
Operation	Time	Speedup	Time	Speedup
Conv2D	0.560ms	-	6.777ms	-
Conv3D	0.049ms	11.43	6.262ms	1.08
Depthwise Conv2D	0.019ms	29.47	1.561ms	4.34

Table 7.2: Average time taken by each operation on both the CPU and the GPU (NVIDIA V100) over 1000 run. The speed-up is calculated as the ratio of the time taken by the Conv2D operation (as a baseline) to the time taken by the other operations.

kernel of size $C \times 1 \times 3 \times 3$ each of these 3×3 kernels is applied separately to only one of the C input channels, hence the second dimension of the kernel is 1 even if the output tensor has C channels. This operation is known as depthwise convolution. Using this approach, we take advantage of the parallelism of the GPU, as the convolutional layers are applied to the C channels in parallel. The following table shows the time taken by each of these operations on both the CPU and the GPU. The depthwise convolution is the fastest operation on both the CPU and the GPU, however the speedup is more significant on the GPU, which is expected as explained before. The 3D convolution has the same performance as the 2D convolution on the CPU, as the computation is still done sequentially.

7.3.5 Limited regimes-based source term: HyPhAICC-4

Using the depthwise 2D convolutional layers, we were able to reduce the training time compared to the other approaches, however, the memory usage is still high due to the high dimensionality of the output space. To address this issue, we propose a fourth version of the hybrid model, HyPhAICC-4. In this model, we assume a limited number of transition regimes, each representing one of the most frequent transitions. For instance, in the case of two regimes, the source term is described as follows:

$$\partial_t P_j = \alpha^1 \cdot \sum_{i=1}^{C} \Lambda^1_{j,i} P^i + \alpha^2 \cdot \sum_{i=1}^{C} \Lambda^2_{j,i} P^i,$$

where Λ^1 and Λ^2 are the transition matrices, α^1 and α^2 are positive factors, these factors determine which regime to consider at each pixel, with the constraint that $\alpha^1 + \alpha^2 \leq 1$. This source term combined with the advection equation is expressed as follows:

$$\partial_t P_j + \overrightarrow{V} \cdot \overrightarrow{\nabla P_j} = \alpha^1 \cdot \sum_{i=1}^{C} \Lambda^1_{j,i} P^i + \alpha^2 \cdot \sum_{i=1}^{C} \Lambda^2_{j,i} P^i, \tag{7.21}$$

where, α^1 and α^2 are estimated using a U-Net, and Λ^1 and Λ^2 are learned as model parameters (see Fig. 7.9).

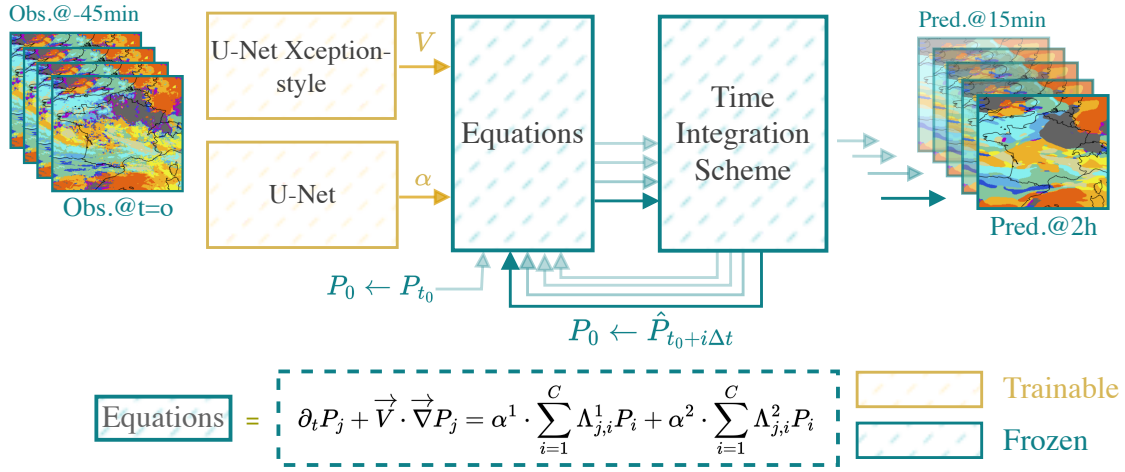


Figure 7.9: HyPhAICC-4: The fourth version of the proposed hybrid model. It consists of a U-Net Xception-style to estimate the velocity field and a second U-Net to estimate the α factors from the last observations, these factors are used to choose which transition regime to consider for each pixel.

This model reduces memory usage and training time compared to previous versions, as the number of parameters is reduced. However, it adds a new hyperparameter to be tuned, the number of regimes.

Unfortunately, this model did not outperform the previous version, HyPhAICC-3, as the transition matrices were consistently learned as zeros during training. The training process began with random values for the matrices, but the model quickly converged on the strategy of ignoring transitions altogether to avoid penalties from false transitions. Thus, HyPhAICC-3 and HyPhAICC-4 were not considered in the performance evaluation.

Finally, despite not yielding the expected results, both models were valuable to explore, demonstrating the flexibility of the proposed architecture.

Chapter 8 Conclusion

8.1 Conclusion

In this thesis, we introduced the HyPhAICC framework, a hybrid approach that integrates physical principles with neural networks, and applied it to cloud cover nowcasting. The principal objective of this research was to improve the learning, efficiency, and physical consistency of neural networks by incorporating physical dynamics into classical neural network frameworks. Another goal was to enhance cloud cover nowcasting by using a hybrid Physics-AI model. Several key results that emerged from the study show the effectiveness, advantages, and limitations of this approach.

The key elements and components of the architecture were presented, including implementing the derivatives in neural networks, the probability advection dynamics, and additional source terms. Different models were developed for the cloud cover nowcasting task.

The first model, HyPhAICC-1, successfully combined the advection dynamics and a U-Net into a single trainable neural network. This model outperforms the U-Net, traditionally used for similar applications, particularly in qualitative metrics (Section 6.8,6.12). This outcome confirms the potential of hybrid models to enhance the physical consistency of predictions, an important aspect in meteorological applications.

Additionally, the HyPhAICC-1 model demonstrated remarkable data efficiency. They achieved high performance with significantly less training data compared to the U-Net (Section 6.10). This efficiency is particularly valuable in operational settings, where data availability may be limited or costly to acquire.

The proposed hybrid approach also facilitated rapid convergence of HyPhAICC-1 during training (Section 6.9). This is due to the physical constraints guiding the learning process, leading to improved overall model efficiency. We also observed that the

HyPhAICC-1 model was robust to domain change and showed a consistent behaviour over the full-disk data (Section 6.11). It also indirectly provides a way to estimate the velocity field of the clouds (Section 6.8).

Finally, the introduction of a source term in HyPhAICC-2 resulted in the highest accuracy among the models tested (Section 7.2). Although this addition led to some compromises in visual rendering, the trade-off was justified by the gains in predictive accuracy. This provides a second option with a better accuracy to the HyPhAICC-1 model, each could be more suitable than the other depending on the use purpose. Two other versions of the source term were also developed, for HyPhAICC-3 and HyPhAICC-4 models in order to keep the physical consistency of the modelling, but they did not show improvements in the results (Section 7.3.2,7.3.5). This is mainly due to the difficulty of estimating the unknown variables in the source term. This difficulty is probably due to the unbalanced representation of the cloud types in the dataset.

This study highlights the value of interdisciplinary approaches in addressing complex scientific problems. Collaboration between meteorology, computer science, and physics can lead to promising solutions that leverage the strengths of each field. Such collaborations should be encouraged and supported, especially in contexts where each field has its own limitations but still strive to assert superiority. Such collaborations are more crucial than ever in tackling the challenges of climate change and extreme weather events (e.g., heatwaves, floods). Particularly in regions with limited weather monitoring infrastructure to acquire high quality data and perform accurate predictions.

Beyond cloud cover nowcasting, the principles demonstrated in this research could be applied to other areas of weather, climate science and beyond. The flexibility of these types of hybrid approaches opens new avenues for exploration and application.

Despite the promising results, this study revealed several limitations and challenges that need to be addressed. One of the primary issues was model diffusiveness, particularly in HyPhAICC-1. This was mainly attributed to the use of a first-order upwind discretisation scheme, which impacted the precision of predictions, especially in high-contrast scenarios. This suggests a need for future research to explore alternative discretisation methods that could mitigate this issue and enhance the quality of predictions.

Another challenge involved stability concerns. The process of setting appropriate time-step sizes for the advection dynamics, especially given the uncertainty in cloud velocities, posed a risk of instability during model inference. Ensuring robust model performance in varying atmospheric conditions will require careful consideration of stability criteria such as the CFL condition.

Additionally, the study acknowledged limitations in the estimation of certain unknown variables within the physical modelling framework (Section 7.3.2). To address this, future investigations should focus on refining the approach, potentially incorporat-

ing techniques such as class imbalance penalties, to improve the physical consistency and accuracy of the model.

The essential of this work is published in the Geoscientific Model Development journal [El Montassir et al., 2024] at https://doi.org/10.5194/gmd-17-6657-2024. This article is also provided in Appendix 9.4.

The source code of the models presented in this study is available on GitHub at https://github.com/relmonta/hyphai and at https://doi.org/10.5281/zenodo.1151854.

8.2 Discussion and perspectives

Building on the results and challenges identified in this research, several potential avenues for future work emerge. One promising direction is the exploration of advanced discretisation schemes. By investigating higher-order discretisation methods, researchers could improve the accuracy of the HyPhAICC models, particularly by reducing diffusiveness and improving the representation of sharp gradients in cloud structures.

Another significant area for future exploration involves the integration of additional physical processes into the models. Incorporating factors such as moisture dynamics or thermodynamic variables could improve the performance of hybrid models. This integration would allow for a more comprehensive understanding of cloud formation and behaviour, potentially leading to more accurate and reliable predictions.

Another rising approach is to use foundation models, which are large-scale pretrained models designed to be adapted to various tasks [Bishop and Bishop, 2024]. These models excel at capturing patterns in large datasets [Nguyen et al., 2023, Bodnar et al., 2024]. However, the inherently complex and chaotic nature of atmospheric phenomena presents significant challenges. Weather systems are governed by fundamental physical laws that dictate how variables such as temperature, pressure, and moisture interact over time. We believe that, without embedding these physical principles into AI models, there is a risk of developing predictions that, while accurate in specific cases, may lack robustness and fail to generalise across different conditions or geographical regions [Bonavita, 2023].

The HyPhAICC framework exemplifies how integrating physics with AI can lead to more reliable and consistent results. By incorporating physical constraints directly into the model architecture, the HyPhAICC approach ensures that predictions remain physically plausible, even when applied to unseen data. This integration is crucial for maintaining the model's validity and relevance in real-world applications, where understanding the underlying physical processes is as important as the accuracy of the predictions themselves. In the other direction, ML models can also help to discover

new physical laws or to improve the understanding of the existing ones as shown in Raissi et al. [2019a] and Pannekoucke and Fablet [2020], but on a larger scale where large derivative dictionaries are involved or using foundation models as in Herde et al. [2024].

Therefore, as we look toward the future of AI for weather forecasting, it is clear that foundation models alone may not suffice. The integration of physical principles into these models is not just beneficial, but necessary to achieve the accuracy and generalisability required for operational weather forecasting. This approach represents a balanced path forward, leveraging the strengths of both data-driven and physics-based methods to tackle the complexities of atmospheric science.

The results of this hybrid architecture underscores the potential for integrating scientific knowledge with machine learning to improve the accuracy and reliability of weather predictions. As the field continues to evolve, integrating green computing principles into the development of AI models will become increasingly important. Future work should focus on optimising computational efficiency and minimising energy consumption without sacrificing predictive performance. We hope that the principles and results of this study will serve as motivation for further innovations, ultimately contributing to more effective and environmentally responsible approaches to weather forecasting and beyond.

With the advent of foundation models, the future of meteorological modelling holds great promise. However, it is essential to recognise the limitations of purely data-driven approaches and the importance of embedding physical principles into AI models. This way, we guarantee that we can leverage AI advancements efficiently, leading to more accurate, generalisable, and responsible weather forecasting solutions.

Conclusion (French version)

8.3 Conclusion

Dans cette thèse, nous avons introduit l'architecture HyPhAICC, une approche hybride qui intègre des principes physiques aux réseaux de neurones, et l'avons appliqué à la prévision immédiate de la couverture nuageuse. L'objectif principal de cette recherche était d'améliorer l'apprentissage, l'efficacité et la cohérence physique des réseaux de neurones en intégrant les dynamiques physiques dans des cadres de réseaux de neurones classiques. Un autre objectif était d'améliorer la précision de la prévision immédiate de la couverture nuageuse. Plusieurs résultats clés émanant de l'étude montrent l'efficacité, les avantages et les limites de cette approche.

Les éléments et composants clés de l'architecture ont été présentés, y compris l'implémentation des dérivées dans les réseaux de neurones, la dynamique d'advection probabiliste et les termes sources supplémentaires. Différents modèles ont été développés pour la tâche de prévision de la couverture nuageuse.

Le premier modèle, HyPhAICC-1, a réussi à combiner les dynamiques d'advection et un U-Net de style Xception en un seul réseau de neurones entraînable. Ce modèle surpasse le U-Net, traditionnellement utilisé pour des applications similaires, notamment en termes de métriques qualitatives (Section 6.8,6.12). Ce résultat confirme le potentiel des modèles hybrides pour améliorer la cohérence physique des prévisions, un aspect important dans les applications météorologiques.

De plus, le modèle HyPhAICC-1 a démontré une remarquable efficacité des données. Ils ont atteint de bonnes performances avec significativement moins de données d'entraînement par rapport au U-Net (Section 6.10). Cette efficacité est particulièrement précieuse dans les contextes opérationnels, où la disponibilité des données peut être limitée ou coûteuse à acquérir.

L'approche hybride proposée a également facilité une convergence rapide du modèle HyPhAICC-1 lors de l'entraînement (Section 6.9). Cela est dû aux contraintes physiques qui guident le processus d'apprentissage, menant à une amélioration de l'efficacité globale du modèle. Nous avons également observé que le modèle HyPhAICC-1 était robuste face aux changements de domaine et montrait un comportement consistent sur les don-

nées de disque complet (Section 6.11). Il fournit également, de manière indirecte, un moyen d'estimer le champ de vitesse des nuages (Section 6.8).

Enfin, l'introduction d'un terme source dans HyPhAICC-2 a abouti à la plus grande précision parmi les modèles impliqués dans l'evaluation (Section 7.2). Bien que cette addition ait entraîné certains compromis dans le rendu visuel, le compromis était justifié par les gains en précision prédictive. Cela fournit une seconde option avec une meilleure précision par rapport au modèle HyPhAICC-1, chaque modèle pouvant être plus adapté selon l'objectif d'utilisation. Deux autres versions du terme source ont également été développées pour les modèles HyPhAICC-3 et HyPhAICC-4 afin de maintenir la cohérence physique de la modélisation, mais elles n'ont pas montré d'améliorations dans les résultats (Section 7.3.2,7.3.5). Cela est principalement dû à la difficulté d'estimer les variables inconnues dans le terme source, qui est probablement due à la représentation déséquilibrée des types de nuages dans les données d'entraînement.

Cette étude met en lumière la valeur des approches interdisciplinaires pour aborder des problèmes scientifiques complexes. La collaboration entre la météorologie, l'informatique et la physique peut conduire à des solutions prometteuses qui tirent parti des forces de chaque domaine. De telles collaborations devraient être encouragées et soutenues, surtout dans des contextes où chaque domaine a ses propres limites mais essaye d'affirmer sa superiorité. Ces collaborations sont plus cruciales que jamais pour relever les défis liés au changement climatique et aux événements météorologiques extrêmes (par exemple, les canicules ou les inondations).En particulier dans les régions manquant d'infrastructures météorologiques robustes pour acquérir des données de haute qualité et effectuer des prévisions précises.

Au-delà de la prévision de la couverture nuageuse, les principes démontrés dans cette recherche pourraient être appliqués à d'autres domaines de la science météorologique, climatique et au-delà. La flexibilité de ces types d'approches hybrides ouvre de nouvelles avenues d'exploration et d'application.

Malgré les résultats prometteurs, cette étude a révélé plusieurs limites et défis qui doivent être abordés. L'un des principaux problèmes était la diffusivité du modèle, en particulier dans HyPhAICC-1. Cela est due à l'utilisation d'un schéma de discrétisation de premier ordre, ce qui a impacté la précision des prévisions, notamment dans des scénarios à fort contraste. Cela suggère un besoin de recherches futures pour explorer des méthodes de discrétisation alternatives qui pourraient atténuer ce problème et améliorer la qualité des prévisions.

Un autre défi concernait les problèmes de stabilité. Le processus de définition de tailles de pas de temps appropriées pour les dynamiques d'advection, notamment compte tenu de l'incertitude des vitesses des nuages, posait un risque d'instabilité lors de l'inférence du modèle. Assurer des performances robustes du modèle dans des conditions atmosphériques variées nécessitera une attention particulière aux critères de

stabilité tels que la condition CFL.

De plus, l'étude reconnait des limites dans l'estimation de certaines variables inconnues dans le cadre de modélisation physique (Section 7.3.2). Pour remédier à cela, de futures recherches devraient se concentrer sur le raffinement de l'approche, en intégrant potentiellement des techniques telles que les pénalités d'imprégnation des classes, pour améliorer la cohérence physique et la précision du modèle.

L'essentiel de ce travail est publié dans la revue Geoscientific Model Development [El Montassir et al., 2024] à https://doi.org/10.5194/gmd-17-6657-2024. Cet article est également fourni en annexe 9.4.

Le code source des modèles présentés dans cette étude est disponible sur GitHub à https://github.com/relmonta/hyphai.

8.4 Discussion et perspectives

En s'appuyant sur les résultats et les défis identifiés dans cette recherche, plusieurs pistes potentielles pour des travaux futurs émergent. Une direction prometteuse est l'exploration de schémas de discrétisation avancés. En étudiant des méthodes de discrétisation d'ordre supérieur, les chercheurs pourraient améliorer la précision des modèles HyPhAICC, notamment en réduisant la diffusivité et en améliorant la représentation des gradients abrupts dans les structures nuageuses.

Un autre domaine significatif pour de futures explorations concerne l'intégration de processus physiques supplémentaires dans les modèles. L'incorporation de facteurs tels que les dynamiques de l'humidité ou des variables thermodynamiques pourrait améliorer les performances de ces modèles. Cette intégration permettrait une compréhension plus complète de la formation et du comportement des nuages, menant potentiellement à des prévisions plus précises et fiables.

Une autre approche émergente consiste à utiliser des modèles fondation, qui sont des modèles pré-entraînés à grande échelle conçus pour être adaptés à diverses tâches [Bishop and Bishop, 2024]. Ces modèles excellent à capturer des motifs dans de grands ensembles de données [Nguyen et al., 2023, Bodnar et al., 2024]. Cependant, la nature complexe et chaotique des phénomènes atmosphériques présente des défis significatifs. Les systèmes météorologiques sont régis par des lois physiques fondamentales qui dictent comment des variables telles que la température, la pression et l'humidité interagissent au fil du temps. Nous croyons que, sans intégrer ces principes physiques dans les modèles d'IA, il y a un risque de développer des prévisions qui, bien que précises dans des cas spécifiques, peuvent manquer de robustesse et échouer à généraliser à travers différentes conditions ou régions géographiques [Bonavita, 2023].

L'architecture HyPhAICC illustre comment l'intégration de la physique avec l'IA peut

conduire à des résultats plus fiables et cohérents. En intégrant directement des contraintes physiques dans l'architecture du modèle, l'approche HyPhAICC garantit que les prévisions restent physiquement plausibles, même lorsqu'elles sont appliquées à des données non vues. Cette intégration est cruciale pour maintenir la validité et la pertinence du modèle dans les applications réelles, où la compréhension des processus physiques sous-jacents est tout aussi importante que la précision des prévisions elles-mêmes. Dans l'autre sens, les modèles de ML peuvent également aider à découvrir de nouvelles lois physiques ou à améliorer la compréhension des lois existantes, comme le montrent Raissi et al. [2019a] et Pannekoucke and Fablet [2020], mais à une échelle plus grande où de grands dictionnaires de dérivées sont impliqués ou en utilisant des modèles fondation comme dans Herde et al. [2024].

Ainsi, alors que nous envisageons l'avenir de l'IA pour la prévision météorologique, il est clair que les modèles fondation seuls peuvent ne pas suffire. L'intégration de principes physiques dans ces modèles n'est pas seulement bénéfique, mais nécessaire pour atteindre la précision et la généralisabilité requises pour la prévision météorologique opérationnelle. Cette approche représente une voie équilibrée, tirant parti des forces des méthodes basées sur les données et sur la physique pour relever les complexités de la science atmosphérique.

Les résultats de cette architecture hybride souligne le potentiel d'intégrer la connaissance scientifique avec l'apprentissage automatique pour améliorer la précision et la fiabilité des prévisions météorologiques. À mesure que le domaine continue d'évoluer, l'intégration des principes du green computing dans le développement des modèles d'IA deviendra de plus en plus importante. Les travaux futurs devraient se concentrer sur l'optimisation de l'efficacité de l'entraînement et la minimisation de la consommation d'énergie sans compromettre les performances prédictives. Nous espérons que les principes et les résultats de cette étude serviront de motivation pour d'autres innovations, contribuant à des approches plus efficaces et respectueuses de l'environnement dans le domaine des prévisions météorologiques et au-delà.

Avec l'avènement des modèles fondation, l'avenir de la modélisation météorologique semble prometteur. Cependant, il est essentiel de reconnaître les limites des approches purement basées sur les données et l'importance d'intégrer des principes physiques dans les modèles d'IA, en plus d'imaginer des métriques d'évaluation plus adaptés et plus représentatives de la qualité des prévisions. De cette manière, nous garantissons que nous pouvons exploiter efficacement les avancées de l'IA, conduisant à des solutions de prévision météorologique plus précises, généralisables et responsables.

Chapter 9 Appendix

9.1 Confidence intervals

9.1.1 Bootstrapping

Bootstrapping is a statistical method used to estimate the sampling distribution of a statistic by resampling with replacement from the original data. It is widely used to estimate standard error, confidence intervals, and other statistical measures when the theoretical distribution is unknown or difficult to derive. The key idea behind bootstrapping is to treat the observed sample as the best representation of the population and to generate many new samples (called bootstrap samples) by randomly sampling from the observed data with replacement. The steps of the bootstrapping procedure are described in Fig. 9.1.

Bootstrapping is particularly useful because it makes few assumptions about the underlying distribution of the data and can be applied to a wide range of statistics. However, it is important to note that bootstrapping is not a panacea and has limitations. For example, bootstrapping may not be appropriate for small sample sizes or when the data is not independent and identically distributed (i.i.d.).

9.1.2 Scores with confidence intervals

The Fig. 9.2 represents the score comparison showed in the Fig. 7.2, but with additional confidence intervals. These confidence intervals were estimated using Bootstrapping, with a threshold of 99%.

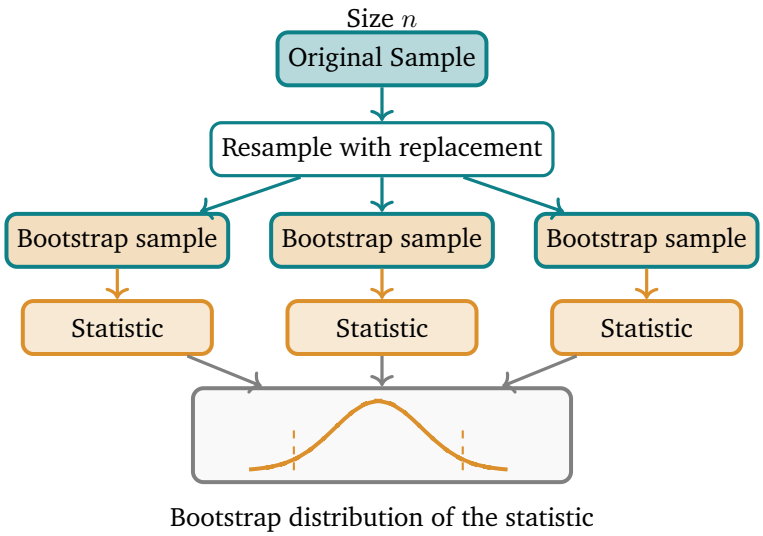


Figure 9.1: Bootstrapping begins with an original sample of data of size n. From this original sample, many bootstrap samples (usually 1,000 or more) are generated by sampling with replacement. Each bootstrap sample is of the same size n as the original sample. For each of these bootstrap samples, the statistic of interest, such as the mean, median, or standard deviation, is calculated. The distribution of these bootstrap statistics is then used to estimate the standard error, construct confidence intervals, or perform hypothesis testing.

9.2 Stochastic differential equations: Fokker-Planck equation

9.2.1 Brownian motion

Brownian motion is a stochastic process that describes the random motion of particles in a fluid. It is named after the botanist Robert Brown, who observed the random motion of pollen grains in water. Brownian motion is a continuous-time stochastic process with independent and stationary increments. It has several properties, including the Markov property, i.e.

$$\mathbb{P}[B_{t+dt} \mid B_t, B_{s<t}] = \mathbb{P}[B_{t+dt} \mid B_t],$$

the paths of the process are almost surely continuous, i.e.,

$$\lim_{dt \to 0} \mathbb{P}[|B_{t+dt} - B_t| > \epsilon] = 0,$$

it has no drift $\mathbb{E}[B_t] = 0$, and the variance of the process grows linearly with time $\mathbb{E}[B_t^2] = t$. This standard Brownian motion is also known as the Wiener process.

In order to simulate the Brownian motion, we use the Euler-Maruyama method, which is a numerical method to solve SDEs. The Euler-Maruyama method is an ex-

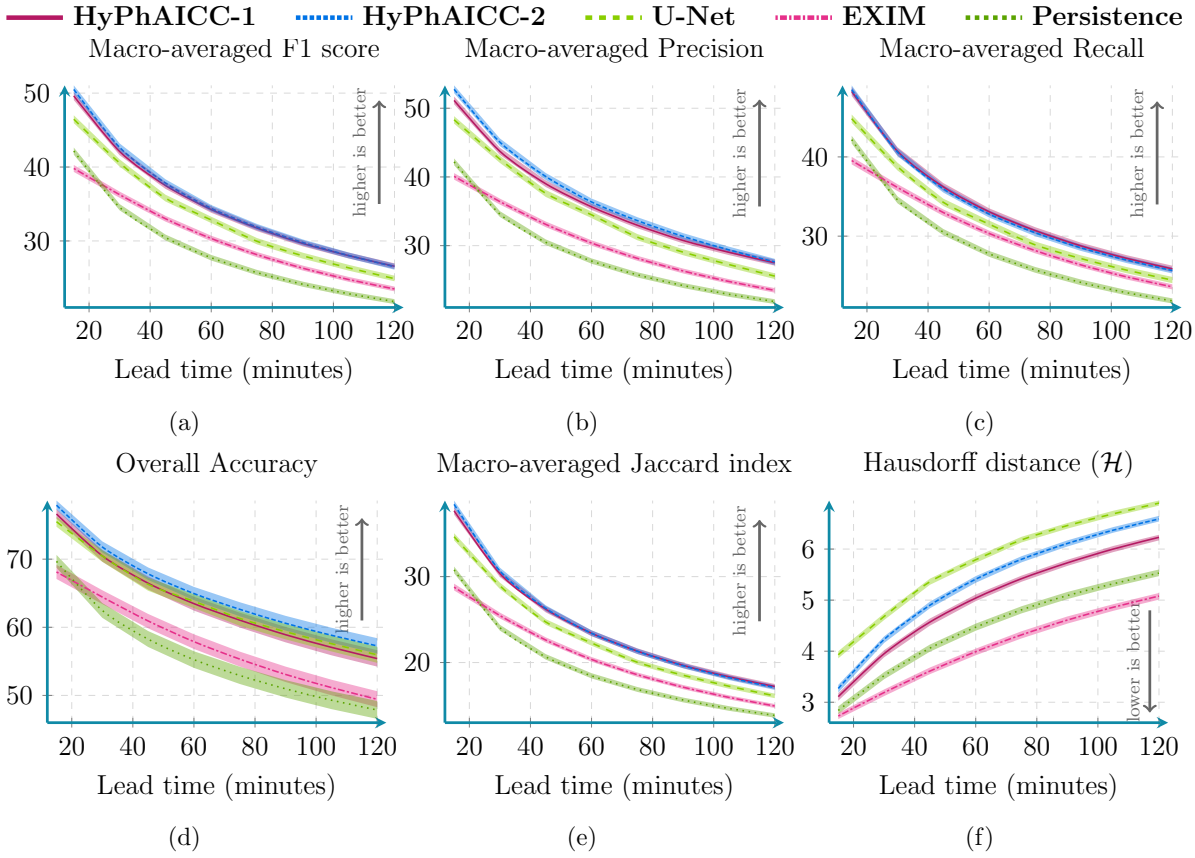


Figure 9.2: **Performance comparison between our HyPhAI-1, U-Net, EXIM, and the Persistence baseline**. Using five metrics including averaged F1 score(%), precision(%), recall(%), accuracy(%), CSI(%) and Hausdorff distance (defined in Eq. (6.10)). These scores were computed over 1000 random samples covering France in 2021. The confidence intervals were estimated using Bootstrapping with a threshold of 99%.

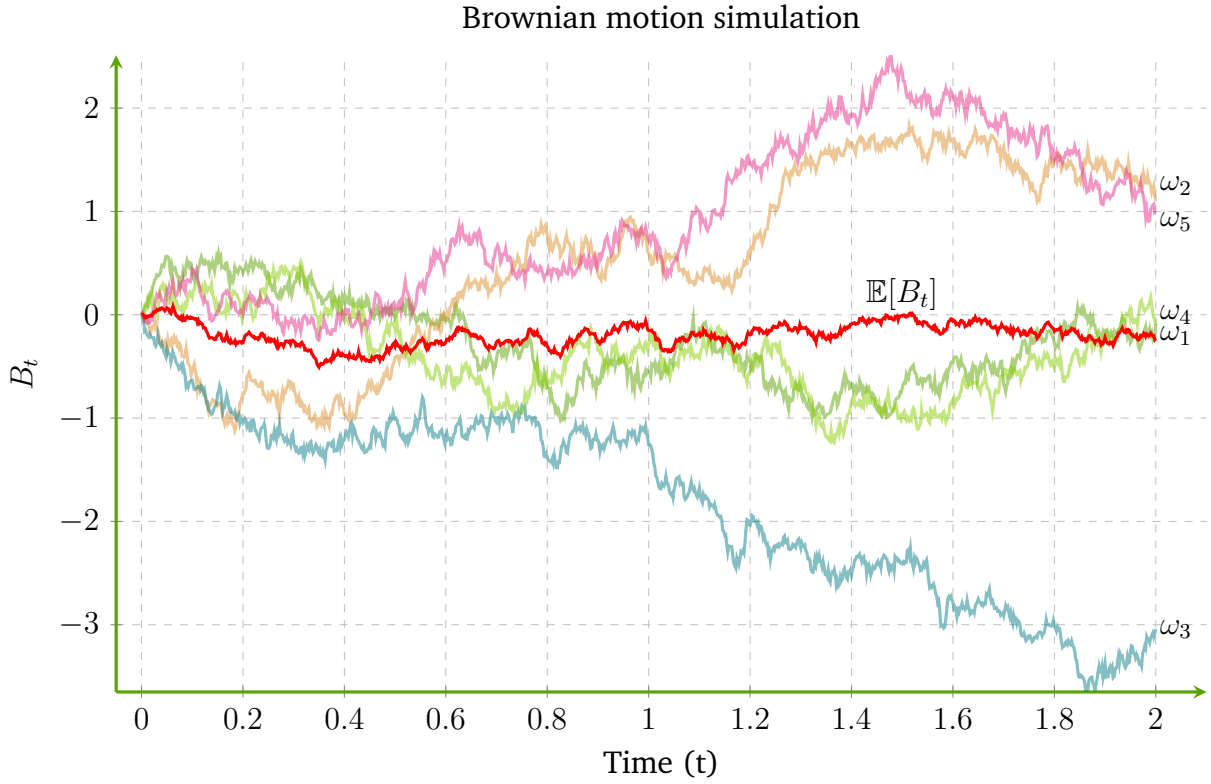


Figure 9.3: Multiple realisations of the standard Brownian motion. The expectation of the process is shown in red.

tension of the Euler method for ODEs. The Euler-Maruyama method for the Brownian motion is given by:

$$\begin{aligned} B_{t+dt} &= B_t + \sqrt{dt}\xi, \\ \xi &\sim \mathcal{N}(0,1), \end{aligned} \tag{9.1}$$

The Fig. 9.3 shows multiple realisations of Brownian motion using the Euler-Maruyama method, each trajectorie is noted as ω_i.

9.2.2 Additional details

In this section, we provide the details about the transition from Eq. (7.15) to Eq. (7.16).

The formal solution of the Eq. (7.13) is given by the exponential of the infinitesimal generator $\mathcal{L}$ acting on the initial observable g_0:

$$g_t(x) = \left(e^{t\mathcal{L}} g_0\right)(x) \tag{9.2}$$

g_t is measurable, and the expectation of g_t under the initial distribution $P_0(x)$ is given by:

$$P_0[g_t] = \int_{\mathbb{R}^n} g_t(x) P_0(x)\, dx$$

This tracks how the average value of g evolves over time based on the initial state of the system.

Replacing g_t by its analytical expression (Eq. (9.2)) gives:

$$\mathbb{P}_0[g_t] = \int_{\mathbb{R}^n} \left(e^{t\mathcal{L}} g_0\right)(x) P_0(x)\, dx \tag{9.3}$$

Using the adjoint operator of $\mathcal{L}$, we can rewrite Eq. (9.3) as:

$$\mathbb{P}_0[g_t] = \int_{\mathbb{R}^n} g_0(x) \left(e^{t\mathcal{L}^\star} P_0\right)(x)\, dx \tag{9.4}$$

where $\mathcal{L}^\star$ is the adjoint operator of $\mathcal{L}$, verifying the inner product property of $\langle \mathcal{L}g, h\rangle = \langle g, \mathcal{L}^\star h\rangle$.

Rewriting Eq. (9.4) gives:

$$\mathbb{P}_0[g_t] = \int_{\mathbb{R}^n} g_0(x) P_t(x)\, dx \tag{9.5}$$

which corresponds to the expectation of the initial observable g_0 under a certain time-dependent measure $P_t(x)$.

$$P_t(x) \triangleq \left(e^{t\mathcal{L}^\star} P_0\right)(x) \tag{9.6}$$

As $\mathcal{L}^\star$ is a linear operator, then $e^{t\mathcal{L}^\star}$ is also a linear operator. The linear transformation of a measure is a measure. Thus, P_t is a measure, that is, a probability density function.

This shows that the expectation of the observable g_t under the initial distribution $\mathbb{P}_0$ (Eq. (9.3)) is equivalent to the expectation of the initial observable g_0 under the distribution $P_t(x)$ (Eq. (9.5)).

Moreover, Eq. (9.6) is a linear flow map, whose dynamics reads as

$$\partial_t P = \mathcal{L}^\star P. \tag{9.7}$$

9.2.3 Adjoint operator

As explained in Section 7.3.3, the operator $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mu \partial_x + \frac{\sigma^2}{2} \partial_x^2 \tag{9.8}$$

It remains to express the formal adjoint operator $\mathcal{L}^\star$. The formal adjoint operator verifies the inner product property of

$$\langle \mathcal{L}f, h\rangle = \langle f, \mathcal{L}^\star h\rangle \quad \forall f, h \in \mathcal{C}_0^2(\mathbb{R}^n)$$

where $\mathcal{C}_c^2(\mathbb{R}^n)$ is the space of smooth functions with compact support. The inner product is defined as

$$\langle f, g \rangle = \int f(x)g(x)\,dx.$$

For the first term of the operator $\mathcal{L}$, $\mu \cdot \partial_x$, we proceed using the integration by parts:

$$\begin{aligned}\langle \mu \cdot \partial_x f, h \rangle &= \int_{\mathbb{R}^n} \mu(x)\partial_x f(x)h(x)\,dx \\ &= [\mu(x)f(x)h(x)] - \int_{\mathbb{R}^n} f(x)\partial_x\left(\mu \cdot h\right)(x)\,dx \\ &= \langle f, -\partial_x\left(\mu \cdot h\right)\rangle\end{aligned}$$

$[\mu(x)f(x)h(x)] = 0$ as f and h are compact support functions. Thus, the formal adjoint operator of $\mu \cdot \partial_x$ is $-\partial_x\left(\mu \cdot .\right)$.

For the second term of the operator $\mathcal{L}$; $\frac{1}{2}\sigma^2\partial_x^2$, we proceed using the integration by parts twice:

$$\begin{aligned}\langle \frac{1}{2}\sigma^2 \cdot \partial_x^2 f, h \rangle &= \int \frac{1}{2}\sigma^2(x)\partial_x^2 f(x)h(x)\,dx \\ &= \left[\frac{1}{2}\sigma^2(x)\partial_x f(x)h(x)\right] - \int \frac{1}{2}\partial_x f(x)\partial_x\left(\sigma^2 \cdot h\right)(x)\,dx \\ &= -\int \frac{1}{2}\partial_x f(x)\partial_x\left(\sigma^2 \cdot h\right)(x)\,dx \\ &= \left[-\frac{1}{2}\sigma^2(x)f(x)h(x)\right] + \int \frac{1}{2}f(x)\partial_x^2\left(\sigma^2 \cdot h\right)(x)\,dx \\ &= \langle f, \frac{1}{2}\partial_x^2\left(\sigma^2 \cdot h\right)\rangle\end{aligned}$$

Thus, the formal adjoint operator of $\frac{1}{2}\sigma^2\partial_x^2$ is $\frac{1}{2}\partial_x^2\left(\sigma^2 \cdot\right)$.

Therefore, the formal adjoint operator of the infinitesimal generator $\mathcal{L}$ is:

$$\mathcal{L}^\star = -\partial_x\left(\mu \cdot\right) + \frac{1}{2}\partial_x^2\left(\sigma^2 \cdot\right)$$

9.3 Additional ressources

9.3.1 Robustness to change of coordinates

In a given coordinate system $x = (x_i)$, the advection of a passive scalar $c(t, x)$ by a velocity field $u = (u_i)$ reads as

$$\partial_t c + u_i \partial_{x_i} c = 0. \tag{9.9}$$

A change of coordinate system from the coordinate system x to the coordinate system $y = (y_j)$ related by $x = x(y)$, remains to the dynamics

$$\partial_t C + v_j \partial_{y_j} C = 0, \tag{9.10}$$

where $C(t, y) = c(t, x(y))$ and where the velocity $v = (v_j)$ is deduced from the chain rule

$$v_j = u_i \partial_{x_i} y_j, \tag{9.11}$$

(using Einstein's summation convention).

Since HyPhAICC-1 architecture estimates a velocity field from the data, that is either u or v, depending on the choice of the coordinate system, it implicitly accounts for the chain rule Eq. (9.11). As a result, the HyPhAICC-1 architecture is not sensitive to the coordinate system and can apply to regional domain as well as global projections. However, numerical effects due to the finite spatio-temporal resolution associated with the discretisation, can lead to abnormal distortion of signals after several time steps of integration, e.g. the disk resulting from an orthographic projection of the Earth may be deformed by the advection near its boundaries unless the velocity field is close to zero, meaning that the apparent displacement is small.

Note that this relative invariance of HyPhAICC-1 to the choice of coordinate is because it only concerns the advection of a scalar field. Covariant transport of vector or tensor fields would imply additional terms (Christoffel symbols, e.g. Nakahara [2003]) that would break the invariance of HyPhAICC-1 as it is formulated here.

9.4 Journal article

Geosci. Model Dev., 17, 6657–6681, 2024
https://doi.org/10.5194/gmd-17-6657-2024

Geoscientific Model Development
Open Access
EGU

HyPhAICC v1.0: a hybrid physics–AI approach for probability fields advection shown through an application to cloud cover nowcasting

Rachid El Montassir[1], **Olivier Pannekoucke**[1,2,3], **and Corentin Lapeyre**[1]

[1]CERFACS, Toulouse, France
[2]INPT-ENM, Toulouse, France
[3]CNRM, Université de Toulouse, Météo-France, CNRS, Toulouse, France

Correspondence: Rachid El Montassir (elmontassir@cerfacs.fr) and Olivier Pannekoucke (olivier.pannekoucke@meteo.fr)

Received: 20 December 2023 – Discussion started: 20 February 2024
Revised: 8 July 2024 – Accepted: 18 July 2024 – Published: 10 September 2024

Abstract. This work proposes a hybrid approach that combines physics and artificial intelligence (AI) for cloud cover nowcasting. It addresses the limitations of traditional deep-learning methods in producing realistic and physically consistent results that can generalise to unseen data. The proposed approach, named HyPhAICC, enforces a physical behaviour. In the first model, denoted as HyPhAICC-1, a multi-level advection dynamics is considered a hard constraint for a trained U-Net model. Our experiments show that the hybrid formulation outperforms not only traditional deep-learning methods but also the EUMETSAT Extrapolated Imagery model (EXIM) in terms of both qualitative and quantitative results. In particular, we illustrate that the hybrid model preserves more details and achieves higher scores based on similarity metrics in comparison to U-Net. Remarkably, these improvements are achieved while using only one-third of the data required by the other models. Another model, denoted as HyPhAICC-2, adds a source term to the advection equation, it impaired the visual rendering but displayed the best performance in terms of accuracy. These results suggest that the proposed hybrid physics–AI architecture provides a promising solution to overcome the limitations of classical AI methods and contributes to open up new possibilities for combining physical knowledge with deep-learning models.

1 Introduction

Meteorological services are responsible for providing accurate and timely weather forecasts and warnings to ensure public safety and mitigate damage to property caused by severe weather events. Traditionally, these forecasts have been based on numerical weather prediction (NWP) models, which provide predictions of atmospheric variables such as temperature, humidity, and wind speed. However, NWP models have inherent limitations in their ability to capture small-scale weather phenomena such as thunderstorms, tornadoes, and localised heavy-rainfall events (Schultz et al., 2021; Matte et al., 2022; Joe et al., 2022).

To address this limitation, the concept of nowcasting has emerged as a valuable tool in meteorology (Lin et al., 2005; Sun et al., 2014). Nowcasting refers to the process of using recently acquired high-resolution observations to generate short-term forecasts of weather conditions, typically on a timescale of minutes to a few hours. Nowcasting techniques exploit various observational data sources, including radar, satellite, lightning, and ground-based observations, to generate real-time estimates of weather conditions and can take advantage of these recent data to significantly outperform NWP on short lead times (Lin et al., 2005; Sun et al., 2014).

Cloud cover nowcasting is a critical component of weather forecasting. It is used to predict the likelihood of precipitation, thunderstorms, and other hazardous weather events. Accurate cloud cover forecasts on a short timescale are particularly important for weather-sensitive applications such as aviation, agriculture, and renewable energy production.

Published by Copernicus Publications on behalf of the European Geosciences Union.

Traditionally, cloud cover forecasting has been done using physics-based methods, relying on the laws of physics to model the evolution of cloud cover, e.g. cloud motion vectors as in Bechini and Chandrasekar (2017) and García-Pereda et al. (2019), optical flow (Wood-Bradley et al., 2012), or NWP-based data assimilation (Ballard et al., 2016). However, with the recent advances in artificial intelligence (AI) and machine learning (ML), data-driven methods have become increasingly popular for these types of tasks (e.g. Espeholt et al., 2022; Ravuri et al., 2021; Trebing et al., 2021; Ayzel et al., 2020; Berthomier et al., 2020; Shi et al., 2015).

Among these data-driven methods, long short-term memory (LSTM) networks, introduced by Hochreiter and Schmidhuber (1997), stand out. LSTMs are a type of recurrent neural network capable of learning long-term dependencies; they are useful for time series predictions, as they can learn from past entries to predict future values. In tasks involving multidimensional data, they are commonly used with convolutional layers, forming what is known as a convolutional LSTM. This neural architecture excels in capturing spatio-temporal correlations compared to fully connected LSTMs (Shi et al., 2015). Spatio-temporal LSTM (Wang et al., 2018) increases the number of memory connections within the network, allowing efficient flow of spatial information. This model was further optimised by adding stacked memory modules (Wang et al., 2019). U-Net is another popular architecture; it was originally designed by Ronneberger et al. (2015) for biomedical image segmentation. Unlike LSTMs, U-Net has no explicit memory modelling, yet it has shown good performance for a binary cloud cover nowcasting task as shown in Berthomier et al. (2020). Furthermore, it has found application in precipitation nowcasting, as highlighted by Ayzel et al. (2020), and a modified version was used for a similar task in Trebing et al. (2021).

Machine learning models hold great promise for addressing scientific challenges associated with processes that cannot be fully simulated due to either a lack of resources or the complexity of the physical process. However, their application in scientific domains faced challenges, including constraints related to large data needs, difficulty in generating physically coherent outcomes, limited generalisability, and issues related to explainability (Karpatne et al., 2017). To overcome these challenges, incorporating physics into ML models is of paramount importance. It leverages the inherent structure and principles of physical laws to improve the interpretability of the model, handle limited labelled data, ensure consistency with known scientific principles during optimisation, and ultimately improve the overall performance and applicability of the models, making them more likely to be generalisable to out-of-sample scenarios. As discussed by Willard et al. (2022) and Cheng et al. (2023), the available hybridisation techniques leverage different aspects of ML models, e.g. the cost function, the design of the architecture, or the weights' initialisation.

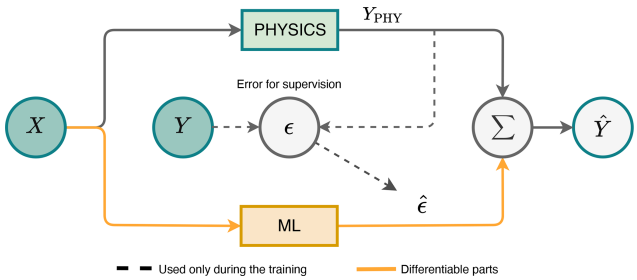


Figure 1. Illustration of error modelling. The physics-based model is used to predict the output, and the ML model is used to predict the residuals. Adapted from Forssell and Lindskog (1997).

A common method to ensure the consistency of ML models with physical laws is to embed physical constraints within the loss function (Karpatne et al., 2017). This involves incorporating a physics-based term weighted by a hyperparameter, alongside the supervised error term. This addition enhances prediction accuracy and accommodates unlabelled data. It has proven to be effective in addressing a range of problems, including uncertainty quantification, parameterisation, and inverse problems (Daw et al., 2021; Jia et al., 2019; Raissi et al., 2019). However, one drawback lies in the challenge of appropriately tuning the hyperparameter.

Given the necessity for an initial choice of model parameters in many ML models, researchers explore ways to inform the initial state of a model with physical insights. One possible way is transfer learning, where a pre-trained model is fine-tuned with limited data (Jia et al., 2021). Additionally, simulated data from physics-based models can be employed for pre-training, akin to methods used in computer vision. This technique has found application in diverse fields, including biophysics (Sultan et al., 2018), temperature modelling (Jia et al., 2019), and autonomous vehicle training (Shah et al., 2017). However, this method requires the assumption that the underlying physics of the simulated data aligns with the real-world data.

To address imperfections in physics-based models, a common strategy is error modelling. Here, an ML model learns to predict the errors (also called residuals) made by the physics-based model (Forssell and Lindskog, 1997). This approach leverages learned biases to correct predictions (see Fig. 1).

A more general approach that does not deal only with errors is to create hybrid models merging physics-based models and ML models. For example, in scenarios where the dynamics of physics are fully defined, the output of a physics-based model can be used as input to an ML model. This approach has demonstrated enhanced predictions in tasks such as lake temperature modelling (Daw et al., 2021). However, in cases where a physical model contains unknown elements requiring coupling with an ML model for joint resolution, a viable strategy involves substituting a segment of a comprehensive physics-based model with a neural network. An

illustrative example is found in sea surface temperature prediction, where de Bezenac et al. (2018) employed a neural network to estimate the motion field. In alignment with this strategy, our study proposes leveraging physical knowledge based on the advection equation to address the cloud cover nowcasting task. This results in simulating the advection of clouds by winds while using a neural network to estimate unknown variables, such as the two components of the velocity field.

Moreover, our study introduces an additional requirement, cloud type classification. Specifically, our dataset contains cloud cover observations with pre-existing categorical classifications based on cloud types (e.g. very low clouds, low clouds). This necessitates adopting a probabilistic approach in our hybrid architecture, which, to the best of our knowledge, has not been explored in geophysics. Indeed, adopting a probabilistic approach with probability maps allows us to account for the inherent variability and uncertainties associated with the model's predictions. This also provides a more natural framework for such a classification problem for further extensions of the modelling beyond the advection.

Rather than using the theoretical solution of the equation as proposed in de Bezenac et al. (2018), our hybrid approach solves a system of partial differential equations (PDEs) within a neural network, which makes the architecture more flexible. However, it poses some implementation challenges, as explained in Appendix B. This paper is organised as follows. Section 2 introduces the hybrid architecture. Section 3 is dedicated to presenting results and performance analysis compared to state-of-the-art models. Finally, in Sect. 4, we draw conclusions based on our findings.

2 Methodology

In this work, we address applications involving dynamics with unknown variables that require estimation. For example, the cloud motion field is one of the unknown variables in the application considered. In such cases, as discussed in Sect. 1, a joint resolution approach is more appropriate. Here, the physical model utilises the neural network outputs to compute predictions, integrating the two models as follows:

$$y = \phi \circ f_\theta(x),$$

where x is the input, f_θ represents the neural network, ϕ denotes the physical model, and y is the output. In this setup, ϕ implicitly imposes a hard constraint on the outputs, potentially accelerating the convergence of the neural network during training.

This method raises some trainability challenges as the physics-based model is involved in the training process, and it should be differentiable, in the sense of automatic differentiation, in order to allow the back-propagation of gradients (refer to Appendix B). We show in Appendix B how spatial derivatives of PDEs can be approximated within a neural network in a differentiable way using convolution operations. This allows us to compute gradients and back-propagate them during the training process. This fundamental knowledge serves as a foundation for our investigation of novel hybrid physics–AI architectures. With these established principles, we present in this section the proposed hybrid architecture, which is applied to cloud cover nowcasting.

In this section, we introduce our hybrid physics–AI architecture, detailed in Sect. 2.1. Section 2.2 explains the different physical modelling approaches investigated in this study. Following that, Sect. 2.3, 2.4, and 2.5 present the training procedure, evaluation metrics, and benchmarking procedure, respectively.

2.1 The HyPhAICC architecture

The proposed hybrid architecture is a dual-component system (see Fig. 2). The first component is composed of one or more classical deep-learning models. These models process the most recent observations, yielding predictions for the physical unknowns of interest. The second block takes as inputs the physical variables, whether known or predicted by the neural networks, along with initial conditions. This second component time integrates one or multiple PDEs to generate the subsequent state of the system. The fourth-order Runge–Kutta (RK4) method is used for time integration. These PDEs encode essential physical knowledge. As discussed in the Appendix B4, the spatial derivatives are approximated using convolutional layers.

The parameters of the first component are trainable; they are optimised during training to estimate the unknown variables. However, we froze the parameters of the second block, as it represents already-known operations. This ensures that the second block maintains its fixed structure, representing the known physical knowledge encoded in the equations, while the trainable block focusses on learning and predicting the unknown aspects of the system. This architecture combines the physical knowledge encoded in the equations with the pattern extraction capabilities of neural networks.

In the following, we employ this architecture for cloud cover nowcasting, with different models being implemented, each using a different physical modelling approach.

2.2 Physical modelling

Before delving into the details of the proposed models, let us first establish the essential characteristics of the data at hand. In this work, we investigate cloud cover nowcasting over France using cloud cover satellite images captured by the Meteosat Second Generation (MSG) satellite at 0° longitude. The spatial resolution of the data over France is ≈ 4.5 km and the time step is 15 min, and each image is of size 256×256. These images have been processed by EUMETSAT (García-Pereda et al., 2019), classifying each pixel into 16 distinct

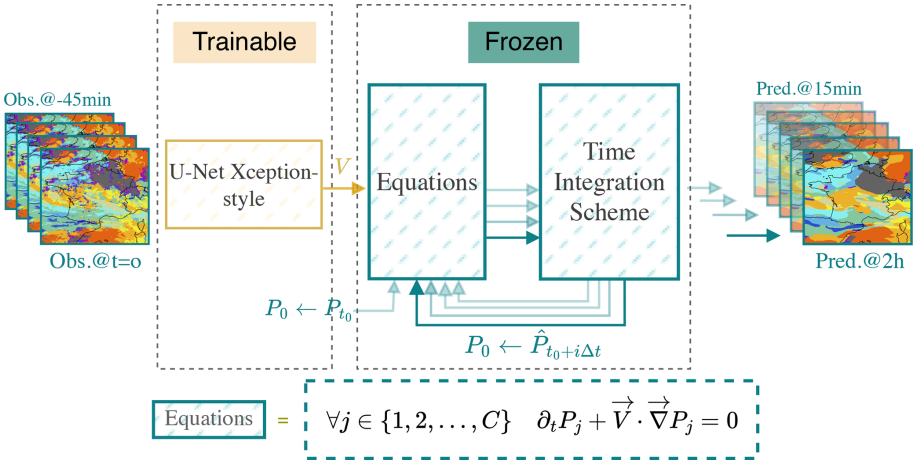


Figure 2. HyPhAICC-1. The proposed hybrid model consists of a U-Net Xception-style model to estimate the velocity field from the last observations; the estimated velocity field is smoothed using a Gaussian filter. The equation is numerically integrated using the fourth-order Runge–Kutta method over multiple time steps. The initial condition (f_0) is updated after each time step to the current state, allowing the computation of the next state.

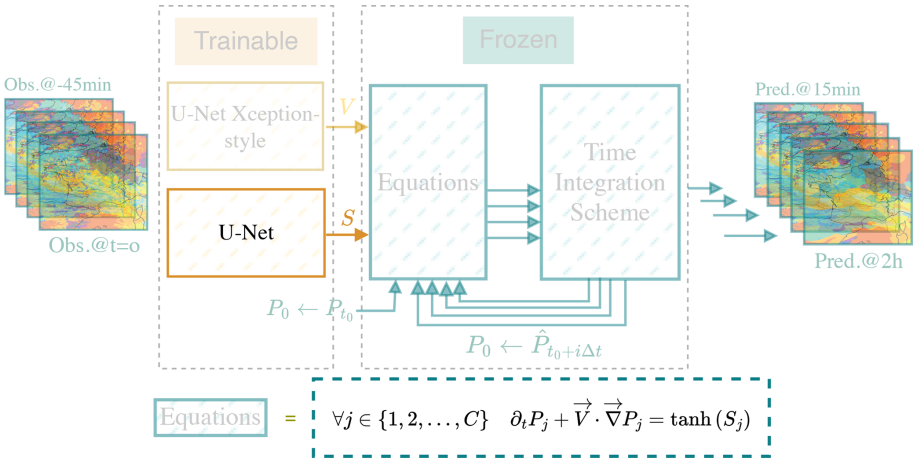


Figure 3. HyPhAICC-2. The second version of the proposed hybrid model. It consists of a U-Net Xception-style model to estimate the velocity field and a second U-Net model to estimate the source term from the last observations. We highlighted the additional parts compared to Fig. 2 and faded the unchanged ones.

categories. We only considered cloud-related categories, 12 in total.

In what follows, we introduce two models: HyPhAICC-1 which uses an advection equation to capture the motion of clouds, and HyPhAICC-2 which extends this by incorporating a simple source term in the advection equation.

2.2.1 Advection equation: HyPhAICC-1

To easily model the advection of these maps with different cloud types, we adopt a probabilistic approach; i.e. rather than representing a single map showing assigned labels, we use 12 maps, each representing the likelihood or probability of a specific cloud type being present at a given location. These maps must satisfy the following properties.

1. *Non-negativity.* $P(\mathbf{x}, t) \geq 0$ for all $\mathbf{x}$ and t, with $\mathbf{x} = (x, y)$, which ensures that the probabilities remain non-negative.

2. *Bound preservation.* $P(\mathbf{x}, t) \leq 1$ for all $\mathbf{x}$ and t, which ensures that no probability exceeds 1.

3. *Probability conservation.* $\sum_{i=1}^{C} P_X^i(\mathbf{x}, t) = 1$ for all $\mathbf{x}$ and t, with $C = 12$ being the total number of cloud types. This property guarantees that the sum of all probabilities is equal to 1.

This approach, known as "one-hot encoding", is more natural to address classification tasks. It involves using 12 distinct advection equations, each corresponding to a specific cloud

type, as described below:

$$\partial_t P_j + \boldsymbol{V} \cdot \boldsymbol{\nabla} P_j = 0 \quad \forall j \in \{1, 2, \ldots, C\}, \tag{1}$$

where $P_j(\mathbf{x})$ represents the classification probability of the jth cloud type and $\boldsymbol{V}(\mathbf{x})$ is the velocity field, which has two components, i.e. $u(\mathbf{x})$ and $v(\mathbf{x})$. Finally, $\boldsymbol{\nabla}$ denotes the gradient operator.

Although one might initially perceive similarities between this modelling and a Fokker–Planck equation (Fokker, 1914; Pavliotis and Stuart, 2008, Chap. 6), the modelling approach presented here deviates significantly from the Fokker–Planck equation. In contrast, the Fokker–Planck equation is typically employed to depict the evolution of probability distributions for time-continuous Markov processes over continuous states, e.g. Brownian motion. On the other hand, Eq. (1) characterises the probability advection for each finite state.

Nevertheless, by employing equations in the following form:

$$\partial_t P_j + \mathcal{L}\left(P_j\right) = 0 \; \forall j \in \{1, 2, \ldots, C\}, \tag{2}$$

where $\mathcal{L}$ represents a differential operator with non-zero positive derivative orders, we demonstrate in Appendix D that the probability conservation property is maintained over time. This assertion holds even in scenarios where the discretisation scheme introduces some diffusion or dispersion effects during the resolution process (see Appendix D2 and Appendix E). However, the non-negativity and bound preservation properties are compromised when a discretisation scheme with dispersion effects is used, unlike the diffusive schemes. Consequently, we opt for the first-order upwind diffusive discretisation scheme (see Appendix E2 for details about the equivalent equation) along with the RK4 for time integration. During the time integration process, we perform the integration by subdividing the time step $\Delta t = 1$ (representing 15 min) into smaller steps $\delta t = 0.1$ to satisfy the Courant–Friedrichs–Lewy (CFL) condition (Courant et al., 1928); this condition ensures the stability of the numerical solution.

In the first hybrid model, denoted as HyPhAICC-1, we use a U-Net Xception-style model (Tamvakis et al., 2022) inspired by the Xception architecture (Chollet, 2017). It takes the last four observations stacked on the channel axis and estimates the velocity field (see Fig. 2). This model will be guided during training by the advection equation to learn the cloud motion patterns.

2.2.2 Advection with source term: HyPhAICC-2

As the advection alone does not take into account other physical processes, especially class change, appearance, and disappearance of clouds, we propose adding a trainable source term to capture them. In this first attempt, we use a simple source term:

$$\partial_t P_j = \tanh(S_j) \; \forall j \in \{1, 2, \ldots, C\}, \tag{3}$$

where S_j is a 2D map. The hyperbolic tangent activation function (tanh) is used to keep the values of the source term in a range of $[-1, 1]$, preventing it from exploding.

The second version of the hybrid model, denoted as HyPhAICC-2, adds this source term to the advection. This modelling is described in the following equation:

$$\partial_t P_j + \boldsymbol{V} \cdot \boldsymbol{\nabla} P_j = \tanh(S_j) \; \forall j \in \{1, 2, \ldots, C\}, \tag{4}$$

where S_j is estimated using a second U-Net model (see Fig. 3).

While the previous modelling describes the missing physical process in the advection, it does not satisfy the probability conservation property. Thus, this modelling does not conserve the probabilistic nature of P over time. To ensure the appropriate dynamics of probability, a robust framework is provided by continuous-time Markov processes across finite states (Pavliotis and Stuart, 2008, Chap. 5). In this framework, the probability trend is controlled by linear dynamics, ensuring the bound preservation, positivity, and probability conservation. Two other models based on this framework, named HyPhAICC-3 and HyPhAICC-4, are presented in Appendix A1 and Appendix A2, respectively. However, these models did not show any performance improvement compared to the simpler HyPhAICC-1.

Indeed, beyond the performance aspect, this hybridisation framework is flexible, is not limited to the advection, and can be extended to other physical processes.

2.3 Training procedure

The training was carried out on a dataset containing about 3 years of data from 2017 to 2019, with a total of 105 120 images. The images with zero cloud cover were removed, then we assembled all the sequences with 12 consecutive images. After this cleaning step, we randomly split the dataset, 8224 sequences were used for training, and 432 for validation. The test set was performed on a separate dataset from the same region but from 2021.

To improve the diversity of the training set and take into account a possible overfitting on the typical movements of clouds in the western Europe region, we randomly applied simple transformations to the images. More precisely, we applied rotations of 90, 180, and 270°, which increased the dataset size and improved the model's ability to learn various cloud motion patterns.

PyTorch framework is used to implement the models, and the cross-entropy loss function is employed for training. This function is given by

$$l(Y, p) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} Y_{i,j} \log(p_{i,j}), \tag{5}$$

where N represents the total number of pixels, C denotes the number of classes, $p_{i,j}$ is the predicted probability of the ith pixel belonging to the jth class, and Y_i corresponds to the

one-hot encoded ground truth at the ith pixel, i.e. $Y_{i,j} = 0$, except for the correspondent cloud type, where $Y_{i,j} = 1$.

The training of the model parameters is achieved through gradient-based methods. Here, an Adam optimiser (Kingma and Ba, 2017) is used with a learning rate of 10^{-3} and a batch size of 4 with 16 accumulation steps, allowing us to simulate a batch size of 64. The training was performed using a single Nvidia A100 GPU for 30 epochs.

You can find the source code for our project on GitHub at https://github.com/relmonta/hyphai (last access: 7 June 2024).

2.4 Performance metrics

To evaluate the performance of competing models in this study, we employed various metrics. Firstly, standard classification metrics are used to evaluate the statistical aspect, then the Hausdorff distance is introduced to evaluate the qualitative aspect of the results.

2.4.1 Classic classification metrics

The selected metrics include accuracy, precision, recall, F1 score, and critical success index (CSI, Gilbert, 1884), also called intersection over union (IoU) or Jaccard index. These metrics offer multiple insights into different aspects of model performance. Accuracy measures the proportion of correct predictions, while precision quantifies the proportion of correct positive predictions relative to the total number of positive predictions. Recall evaluates the proportion of correct positive predictions relative to the total number of positive cases. The F1 score provides a balance between precision and recall. The CSI measures the overlap between prediction and ground truth, providing a measure of similarity.

To compute these metrics for the jth class, we use the following formulas:

$$\text{Accuracy}_j = \frac{\text{TP}_j + \text{TN}_j}{\text{TP}_j + \text{TN}_j + \text{FP}_j + \text{FN}_j},$$
$$\text{Recall}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FN}_j},$$
$$\text{Precision}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FP}_j},$$
$$\text{F1}_j = \frac{2 \times \text{Precision}_j \times \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j},$$
$$\text{CSI}_j = \frac{\text{TP}_j}{\text{TP}_j + \text{FP}_j + \text{FN}_j}.$$

These metrics are calculated separately for each class, where TP denotes instances correctly identified as positive cases, TN refers to instances correctly identified as negative cases, FP represents cases misclassified as positives, and FN is the number of positive cases that are classified as negative.

To obtain an overall performance evaluation of the accuracy, we use the following formula:

$$\text{Accuracy} = \frac{\sum_j \text{TP}_j}{\text{Total number of cases}}.$$

For the remaining metrics, we can calculate two types of average: the macro-average and the micro-average. The macro-average is the arithmetic mean of the metric scores computed for each class, while the micro-average considers all classes as a single entity (Takahashi et al., 2022). Given the highly imbalanced distribution of labels in our dataset, we adopted the macro-average to evaluate the models' performance (Fernandes et al., 2020; Wang et al., 2021). The macro-averaged F1 is defined as in Sokolova and Lapalme (2009) as follows:

$$\text{F1}_{\text{macro}} = \frac{2 \times \text{Precision}_{\text{macro}} \times \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}},$$

where the macro-averaged precision and recall are defined as follows:

$$\text{Precision}_{\text{macro}} = \frac{1}{C} \sum_{j=1}^{C} \text{Precision}_j.$$

$$\text{Recall}_{\text{macro}} = \frac{1}{C} \sum_{j=1}^{C} \text{Recall}_j.$$

We define the macro-averaged CSI following the same method as follows:

$$\text{CSI}_{\text{macro}} = \frac{1}{C} \sum_{j=1}^{C} \text{CSI}_j.$$

These pixel-wise metrics are commonly used for evaluating image segmentation tasks or more generally classification tasks, but it is important to note the limitations of these metrics and evaluation approaches. Although selected metrics provide valuable insights, they do not capture all aspects of model performance, for instance, because they do not take into account the spatial correspondence between predicted and ground-truth cloud structures. This means that a model can statistically perform well using pixel-wise metrics but still have poor performance in identifying the correct cloud structures or miss a significant amount of detail. As a result, evaluating cloud cover forecasting models based solely on pixel-wise metrics may not be sufficient to ensure their effectiveness in real-world applications.

2.4.2 Hausdorff distance

The Hausdorff distance is a widely used metric for medical image segmentation (e.g. Karimi and Salcudean, 2019; Aydin et al., 2021). This metric measures the similarity between the predicted region and the ground-truth region by comparing structures rather than just individual pixels. It can be expressed using either Eq. (6) or Eq. (7), which are described

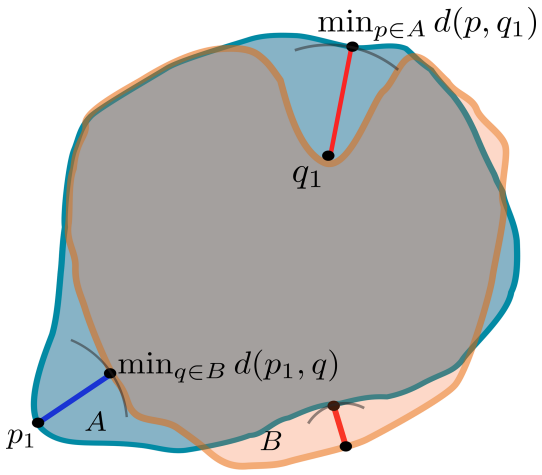


Figure 4. Illustration of the $\min_{p\in A} d(p, q_1)$ and $\min_{q\in B} d(p_1, q)$ quantities used to compute the Hausdorff distance; for each point, we look for the closest point in the other region.

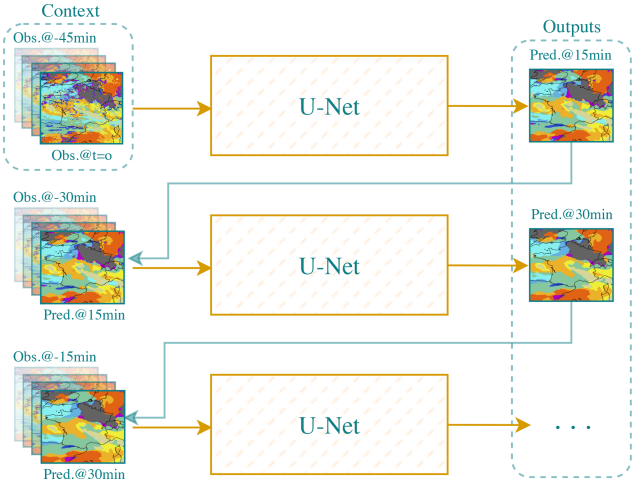


Figure 5. The U-Net architecture considered in the comparison. U-Net is iteratively used to predict the next state given the previous ones.

as follows:

$$h^1(A, B) = \frac{1}{|A|} \sum_{p\in A} \min_{q\in B} d(p, q), \tag{6}$$

$$h^2(A, B) = \max_{p\in A} \min_{q\in B} d(p, q), \tag{7}$$

where $d(p, q)$ is the Euclidean distance between p and q. The former computes the mean distance between each point A and the closest point in B, providing an overall measure of similarity. The latter measures the maximum distance between a point in A and the closest point in B (Fig. 4), this formulation is a more conservative measure that focuses on the largest discrepancies between the sets. Both formulations exhibit sensitivity to the loss of small structures. Specifically, when small regions in the ground truth are non-empty while their corresponding regions in the prediction are empty, the search area expands, which increases the overall distance. We opt to limit this search region to the maximum distance traversable by a cloud. Consequently, we introduce the restricted Hausdorff distance (rHD), which is defined as follows:

$$h^3(A, B) = \frac{1}{|A|} \sum_{p\in A} \min_{q\in \mathbf{B}_r(p)} d(p, q), \tag{8}$$

where $\mathbf{B}_r(p)$ is the ball of radius r centred at p. In our experiments, we set r to 10 pixels, which corresponds to a radius of approximately 45–50 km, corresponding to the maximum distance crossed by clouds in one time step, considering 200 km h^{-1} as the cloud's maximum speed. This means that for each pixel in the first set, we compute the distance to the closest pixel in the second set, but we only do this if it is within a radius of 10 pixels. This allows us to reduce the impact of small regions in the ground truth that are not present in the prediction, while still rewarding the model if it correctly predicts them.

The Hausdorff distance is a directed metric, i.e. $h^p(A, B) \neq h^p(B, A)$; thus, we consider the maximum of the two directed distances as follows:

$$\mathcal{H}(S, \hat{S}) = \max\left(h^3(S, \hat{S}), h^3(\hat{S}, S)\right), \tag{9}$$

where S and $\hat{S}$ are the coordinates of positive pixels in the ground truth and prediction, respectively.

2.5 Benchmarking procedure

To assess the performance of the proposed models, we consider established benchmarks. In the comparative evaluation, we included the well-known U-Net (Ronneberger et al., 2015). This classical U-Net is different from the one used to estimate the velocity in the proposed hybrid models (refer to Figs. 2 and 3). The choice of this classical U-Net for comparison is justified by the fact that it is the most widely used in the literature for the same task (e.g. Ayzel et al., 2020; Berthomier et al., 2020; Trebing et al., 2021). U-Net architecture is structured with a contracting path and an expansive path connected by a bottleneck layer. The contracting path comprises four levels of convolutional layers, each followed by a max pooling layer. The number of filters we used in these convolutional layers progressively increases from 32 to 64, 128, and finally 256. On the other hand, the expansive path consists of four sets of convolutional layers, each followed by an upsampling layer. These layers help in the reconstruction and expansion of the feature maps to match the original input size. We iterate over U-Net, as illustrated in Fig. 5, to generate predictions for multiple future time steps.

In addition to U-Net, we consider in our comparison a product called EXIM (for extrapolated imagery), devel-

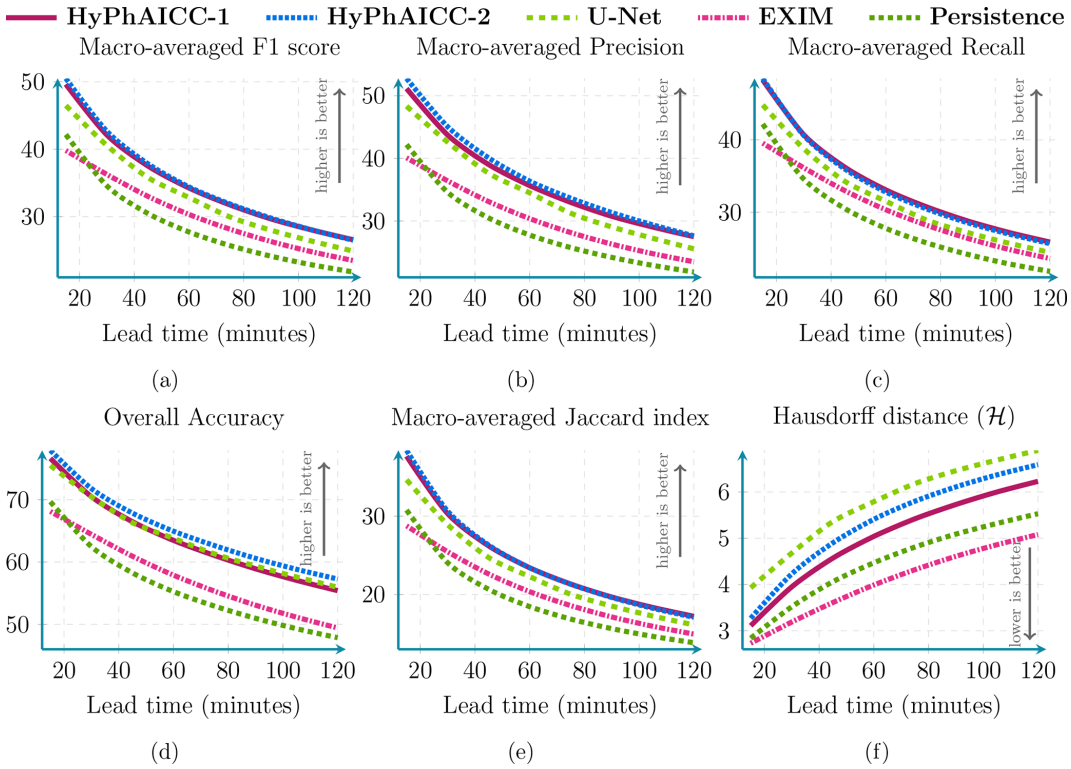


Figure 6. Performance comparison between our HyPhAICC-1, U-Net, EXIM, and the persistence baseline using five metrics: averaged F1 score (%), precision (%), recall (%), accuracy (%), CSI (%), and Hausdorff distance (defined in Eq. 9). These scores were computed over 1000 random samples covering France in 2021. See Fig. A3 for confidence intervals.

oped by EUMETSAT as part of their NWCSAF/GEO products (García-Pereda et al., 2019). This product involves applying the atmospheric motion vector field multiple times to a current image to produce forecasts. Each pixel's new location is calculated using the motion vector, and this process is repeated assuming a constant displacement field. For continuous variables like brightness temperature, the method uses weighted contributions to forecast pixel values, ensuring that there are no gaps by interpolating values from adjacent pixels if necessary. For categorical variables such as cloud type, the pixel value is directly assigned to the new location, and conflicts are resolved by overwriting. If a pixel is not touched by any trajectory, the value is determined by the majority class of its nearest neighbours (García-Pereda et al., 2019) (https://www.nwcsaf.org/exim_description, last access: 4 July 2024). This approach is also called kinematic extrapolation.

We also included a commonly used meteorological baseline method known as "persistence". This method predicts future time steps by simply using the last observation, a relevant approach in nowcasting since weather changes occur slowly, making the last observation a strong prediction, which makes the persistence baseline challenging to outperform.

We tested the competing models using 1000 satellite images samples captured over France from January 2021 to October 2021.

3 Experiments and results

We trained the hybrid models, in addition to the U-Net model used for comparison, on 3 years of data. The models were designed to predict a 2 h forecast at 15 min intervals.

3.1 Quantitative analysis

Diving into the numerical evaluations, here we present a comparative analysis based on standard metrics used in image classification tasks. Figure 6 shows a score comparison using different metrics over multiple lead times up to 2 h. The confidence intervals, indicating statistical significance, are computed using a resampling method called bootstrap, which is a statistical technique that involves repeatedly sampling from a single dataset to generate numerous simulated samples (Efron, 1979). Through this method, standard errors, confidence intervals, and hypothesis testing can be computed. Table 1 and Fig. 6 show that HyPhAICC-2 is slightly better in terms of precision and accuracy than the model using advection equation without source term (HyPhAICC-1),

Table 1. Score comparison at the 120 min lead time (↑: higher is better; ↓: lower is better). The best scores are indicated in bold font.

Model	↑ F1 score	↑ precision	↑ recall	↑ accuracy	↑ CSI	↓ Hausdorff distance ($\mathcal{H}$)
HyPhAICC-1	**26.6 %**	**27.5 %**	**25.9 %**	55.4 %	**17.2 %**	6.23
HyPhAICC-2	**26.5 %**	**27.6 %**	25.7 %	**57.3 %**	**17.1 %**	6.54
U-Net	24.9 %	25.6 %	24.5 %	56.0 %	16.1 %	6.90
EXIM	23.5 %	23.5 %	23.6 %	49.4 %	14.9 %	**5.08**
Persistence	21.8 %	21.9 %	21.8 %	47.9 %	13.8 %	5.53

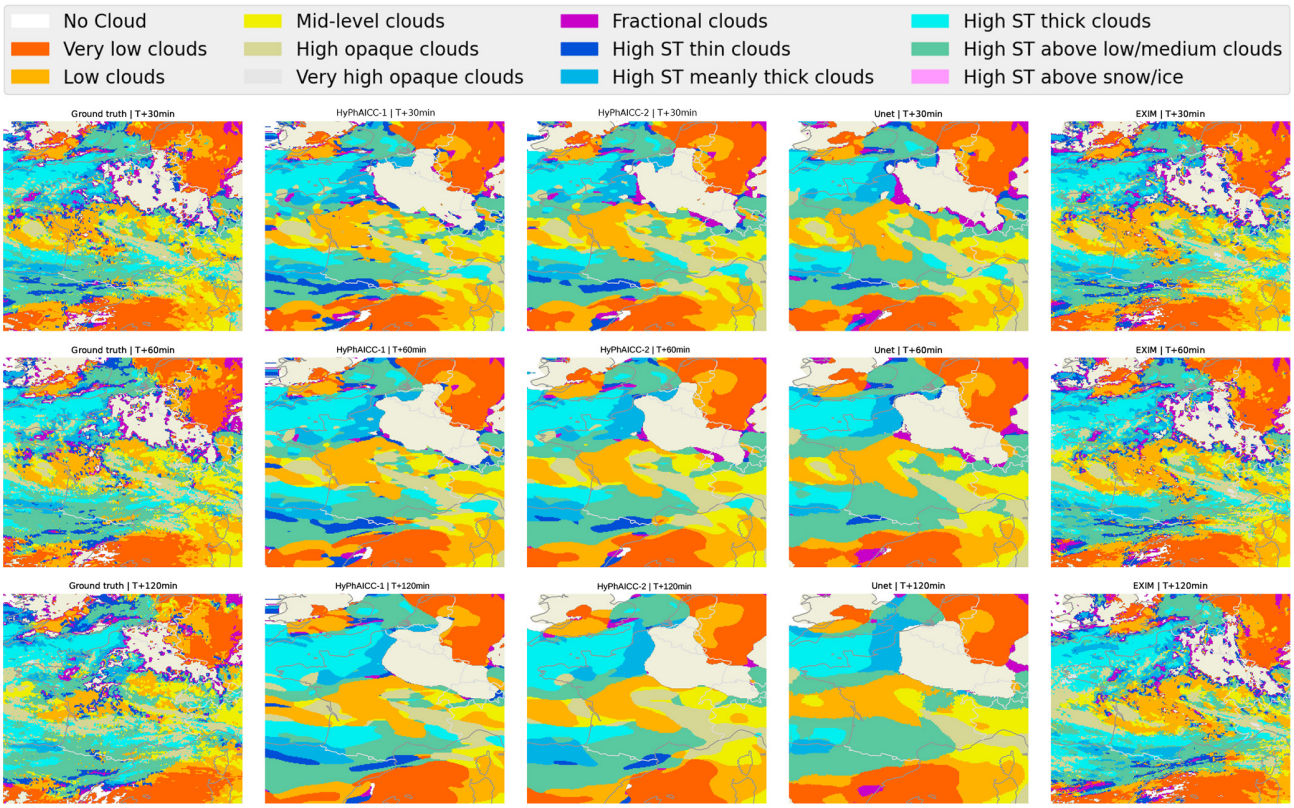


Figure 7. Case study of different models' forecasts. The left column shows ground truth at different time steps. The middle columns show, from left to right, HyPhAICC-1, HyPhAICC-2, and the U-Net predictions, respectively. The right column shows EXIM's predictions. The light beige colour corresponds to the land areas, and the "ST" abbreviation in the legend stands for "semi-transparent".

and both of these hybrid models significantly outperform the U-Net model in terms of F1 score, precision, and CSI and perform similarly in terms of accuracy and recall. This is because the U-Net model tends to give more weight to the dominant classes at the expense of the other classes, resulting in a higher false positive rate.

Although quantitative performance metrics offer a numerical assessment of a model's ability to predict weather states, providing crucial insights into the reliability and precision of forecasts, they are not sufficient on their own. Qualitative aspects also play a significant role, including the visual interpretation of model predictions and an assessment of its capability to capture complex atmospheric patterns and phenomena.

3.2 Qualitative analysis

Figure 7 presents a case study involving multiple models, highlighting that HyPhAICC-1 produces more realistic and less blurry forecasts compared to the U-Net model. To substantiate this claim, we used the restricted Hausdorff distance (rHD), described in Eq. (8), to assess the sharpness of predicted cloud boundaries. Both HyPhAICC-1 and HyPhAICC-2 models outperformed the U-Net model in this metric, as shown in Fig. 6. EXIM and the persistence baseline exhibit superior results in terms of the Hausdorff metric, and the gap between them and the other models increases with the lead time, which is visually expected. The reason behind this result is that the hybrid models, especially HyPhAICC-

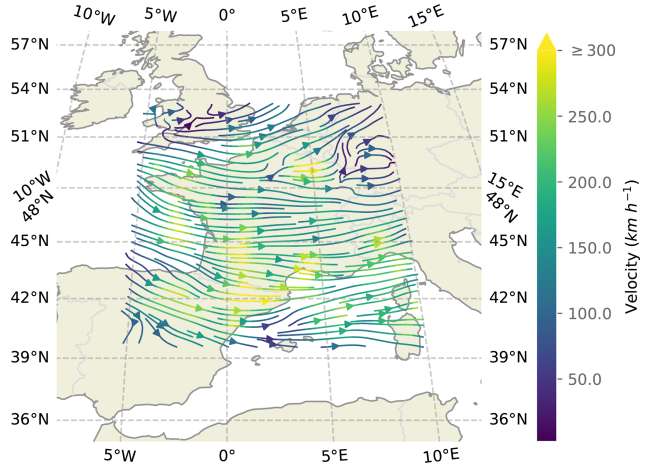


Figure 8. Estimated velocity field by the U-Net Xception-style architecture used in the HyPhAICC-1 model.

1, preserve more details compared to the U-Net model. The lost details in HyPhAICC-1's predictions are only due to the numerical scheme. In ideal conditions, HyPhAICC-1 should preserve the same details during the advection process, and there is no other trainable part in between that can smooth the predictions; however, the upwind discretisation used a scheme that adds a numerical diffusion and crushing the small cloud cells (refer to Appendix E for more details). In contrast, U-Net focuses more on dominant structures and labels, which are more likely to persist over time, which is statistically relevant. Nevertheless, EXIM and the persistence baseline still outperform the other models in this regard. This observation aligns with the fact that the persistence uses the last observation as its predictions, and EXIM is advecting the last observation using the kinematic extrapolation, which keeps the same level of details without diffusion effects (García-Pereda et al., 2019). However, EXIM is slightly more accurate, compared to persistence, in terms of predicted cloud positions.

In Fig. 8, we present the estimated velocity field generated by the HyPhAICC-1 model, illustrated in Fig. 2. This field exhibits a high level of coherence with the observed cloud cover displacements, with exceptions in cloud-free areas, as expected. It is important to emphasise that this velocity field is derived exclusively from cloud cover images, without relying on external wind data or similar sources. This aspect adds a layer of interest, especially in the context of other applications beyond the cloud cover nowcasting.

3.3 Time efficiency

In what follows, we focus only on the HyPhAICC-1 model. By including physical constraints into these hybrid models, we expect a decrease in training time compared to that of the U-Net model. Indeed, Fig. 9 illustrates the evolution of the validation F1 score for both the U-Net and HyPhAICC-1 models across epochs. HyPhAICC-1 converges faster than U-Net. Its convergence does indeed occur after just about 10–15 epochs. Each epoch of the HyPhAICC-1 training takes approximately 55 min using a single Nvidia A100 GPU, while the entire training over 30 epochs takes 27 h. On the other hand, U-Net necessitates up to 200 epochs for achieving similar performance, with each epoch taking around 23 min using the same hardware, which corresponds to about 3 d of training. This difference implies that training U-Net is significantly more expensive compared to HyPhAICC-1.

In inference mode, the hybrid models and U-Net generate predictions within a few seconds, while EXIM's predictions are produced within 20 min (Berthomier et al., 2020), which is one of the main drawbacks of this product.

3.4 Data efficiency

To delve deeper into the efficiency of the proposed HyPhAICC-1 model, we conducted various experiments using different training data sizes. In each experiment, both HyPhAICC-1 and U-Net were trained with 70 %, 50 %, 30 %, and 10 % of the available training data (Figs. 9, 10). Notably, we observed a more significant performance drop for U-Net compared to HyPhAICC-1. Interestingly, the hybrid model exhibited a similar performance using only 30 % of the training data to it when it used the entire dataset (Fig. 9). This finding indicates that this hybrid model is remarkably data efficient, capable of delivering satisfactory performance even with limited training data, which has been highlighted by other studies (Schweidtmann et al., 2024; Cheng et al., 2023). This quality is very important, particularly for tasks with insufficient provided data.

3.5 Application to Earth's full disc

To check HyPhAICC-1's capabilities on broader scales after training it on a small region, we tested it on a much larger domain, an entire hemisphere of the Earth – also called a full disc – centred at 0° longitude. The satellite observations of this expansive full-disc domain are of size 3712×3712, which is 210.25 times larger than the size of the training ones. It has diverse meteorological conditions and includes projection deformations when mapped onto a two-dimensional plane, while the extreme deformations at the edge of the disc make this data less useful for operation purposes, it still provides an interesting testing ground for HyPhAICC-1's generalisation ability. In this analysis, we focus only on visual aspects. Despite the significant differences between the training domain and the full disc, we observed good qualitative forecasts of the HyPhAICC-1 model on this new domain without any specific training on it (see Figs. 11 and A4). The cloud motion estimation on the full disc was found to be visually consistent (a Video Supplement is provided at https://doi.org/10.5281/zenodo.10375284, El Montassir et al., 2023b).

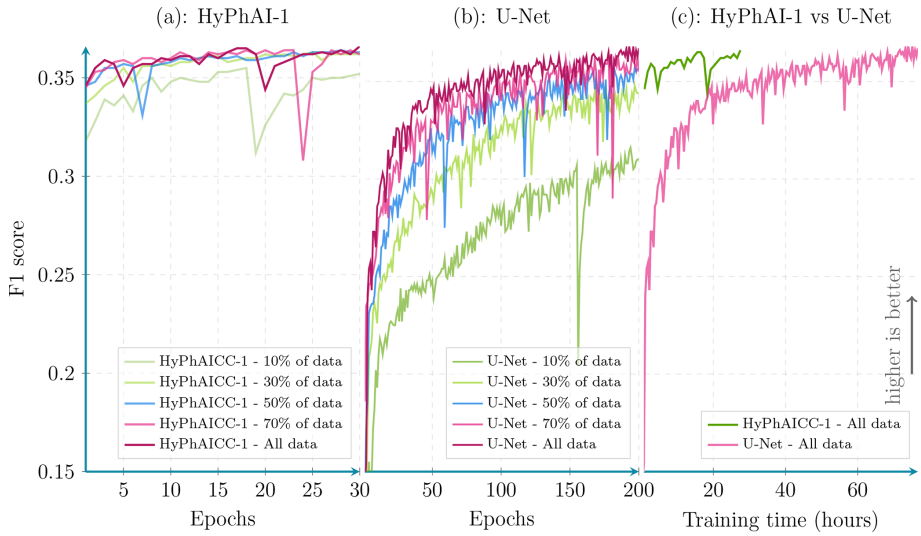


Figure 9. Per epoch validation F1 score comparison between HyPhAICC-1 and U-Net. Scores were calculated from 100 random samples covering France (averaged over all lead times).

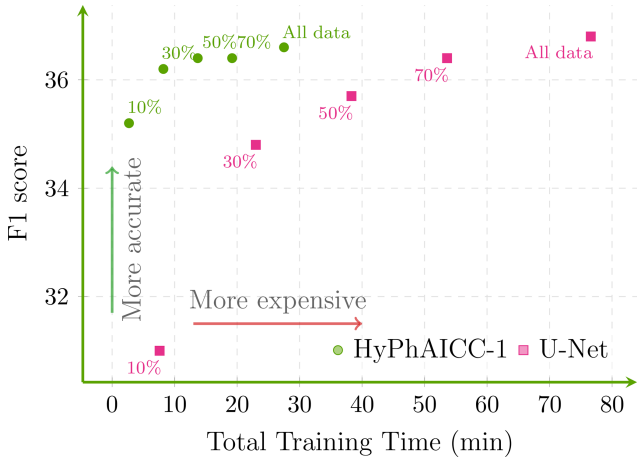


Figure 10. Total training time and maximum validation F1 scores over the last five epochs for U-Net and HyPhAICC-1 using different training data sizes (averaged over all lead times).

This successful transferability of the model highlights its potential robustness and suggests that the underlying principles of cloud motion captured during training are applicable across different domain sizes and different projections (see Appendix C for a formal explanation). Note that the model requires a data size divisible by 2^d, where d is the number of the encoder blocks within the U-Net-Xception model. Indeed, the possibility of running a model using different data sizes is one of the advantages of fully convolutional networks (FCNs) as the convolution operation is independent of the input size.

Overall, HyPhAICC-1 offers an effective and cheaper approach compared to EXIM, with higher efficiency, requiring fewer data compared to U-Net, with the potential to outperform existing models and enable more accurate and efficient weather forecasting. The ability of HyPhAICC-1 to adapt and perform well on the full-disc data, despite being trained on a smaller domain, demonstrates the generalisation capabilities of this hybrid model. This is an important property for weather forecasting models, as it is not always possible to train a model on full-disc data due to the high computational cost.

4 Conclusions

In this study, we introduced a hybrid physics–AI framework that combines the insights from partial differential equations, representing physical knowledge, with the pattern extraction capabilities of neural networks. Our primary focus was on applying this hybrid approach to the task of cloud cover nowcasting, also involving cloud type classification. To leverage continuous physical advection phenomena for this discrete classification task, we proposed a probabilistic modelling strategy based on the advection of probability maps. This flexible approach was easy to adapt to include the prediction of source terms, demonstrating its versatility.

The first model, HyPhAICC-1, leverages the advection equation and slightly outperforms the widely used U-Net model in the quantitative metrics, while showing a significantly better performance in the qualitative aspect. This hybrid model requires a significantly lower amount of data and converges faster, cutting down the training time, which is expected as the physical modelling implicitly imposes a constraint on the trainable component. Notably, the estimated velocity field demonstrated high accuracy compared to actual cloud displacements. This accuracy suggests that this architecture could find utility in diverse tasks, such as wind

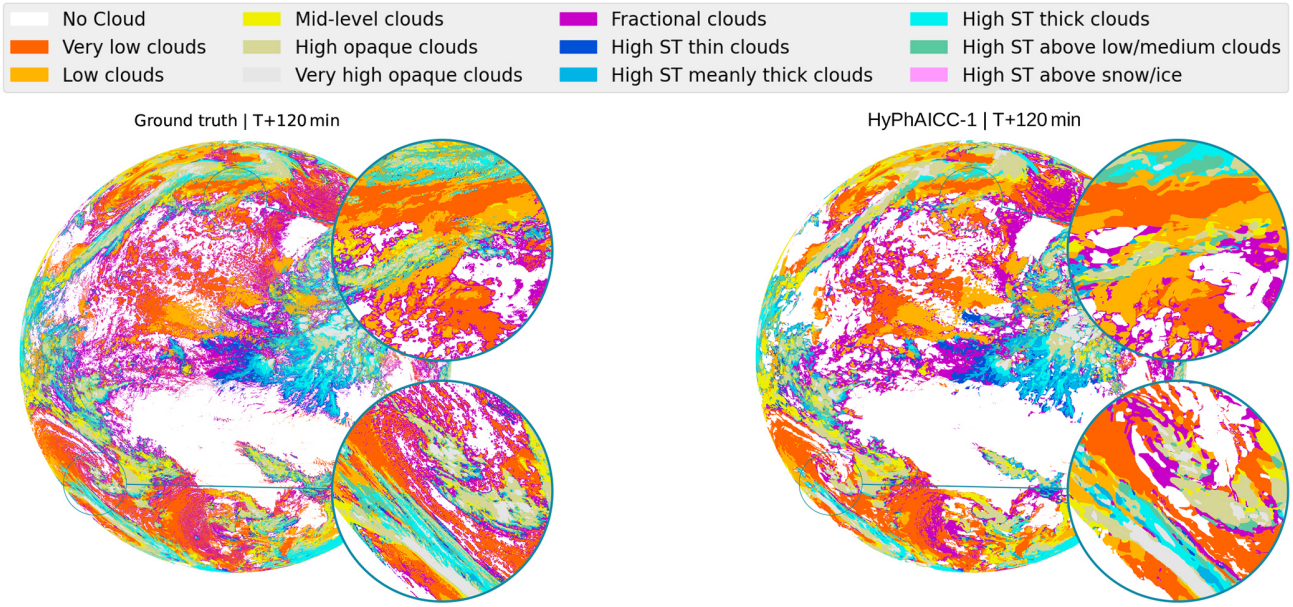


Figure 11. Full-disc cloud cover nowcasting predictions. Zoomed-in views of the 120 min observation and prediction.

speed estimation using only satellite observations. The second version, HyPhAICC-2, adds a source term to the advection equation. This model impaired the visual rendering but displayed the best performance in terms of accuracy.

The HyPhAICC architecture demonstrated an effective path towards uniting the strengths of a continuous physics-informed phenomenon with a data-driven approach in the context of a discrete classification task.

Despite these successes, the models still exhibit some diffusiveness. However, in the case of HyPhAICC-1, it is only attributed to the use of the first-order upwind discretisation scheme. Exploring less diffusive schemes could potentially mitigate this effect, especially in inference mode, where there is no differentiability constraint.

The CFL condition is designed to guarantee stability by imposing a restriction on the time step size relative to the maximum velocity in the system. However, in our scenario, where the maximum velocity of the cloud is unknown, setting the time step becomes challenging. This uncertainty may lead to stability issues if the time step is not small enough, particularly if the predicted velocity turns out to be unexpectedly high, highlighting the importance of carefully considering and addressing potential instability concerns in such cases.

While HyPhAICC-3 (see Appendix A1) and HyPhAICC-4 (see Appendix A2) presented interesting modelling variations, the study acknowledges limitations in not obtaining the desired variables. We suggest that modifying the approach to estimate these variables may lead to improved results, e.g. penalising the dominant classes.

As we move forward, the integration of green computing principles into AI research becomes crucial. The success of the HyPhAICC models in achieving these results with low data requirement and rapid convergence encourages further exploration of energy-efficient AI models and methodologies. This emphasises the importance of balancing computational power with environmental responsibility in the pursuit of scientific advancements.

Appendix A: Additional resources

A1 Advection with source term: HyPhAICC-3

We introduced another version of the HyPhAICC models using a source term based on Markov inter-class transitions. This preserves the probabilistic properties as discussed in Sect. 2.2.2. This dynamics is expressed using the following equations:

$$\partial_t P_j = \sum_{i=1}^{C} \mathbf{\Lambda}_{j,i} P_i \quad \forall j \in \{1, 2, \ldots, C\},$$

where

$$\Lambda(\mathbf{x}) = \frac{\mathbf{\Pi}(\mathbf{x})^{\intercal} - I}{\Delta t},$$

with $\mathbf{\Pi}(\mathbf{x})$ being a stochastic matrix, i.e. a non-negative square matrix where the sum of each row is equal to one. This constraint ensures that the probabilistic properties are maintained over time.

Physically, $\mathbf{\Lambda}_{j,i}(\mathbf{x})$ represents the transition rate from cloud type i to cloud type j at grid point $\mathbf{x}$, Δt represents the time step, and $I(\mathbf{x})$ denotes the identity matrix.

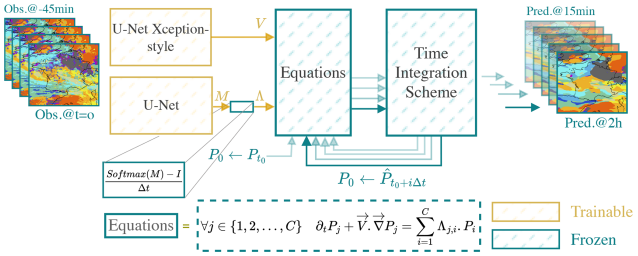


Figure A1. HyPhAICC-3. The third version of the proposed hybrid model. It consists of a U-Net Xception-style model to estimate the velocity field and a second U-Net model to estimate the per-pixel transition matrices from the last observations.

The third version of the hybrid model (see Fig. A1), denoted as HyPhAICC-3, uses this source term combined with the advection as shown in the following equations:

$$\partial_t P_j + \boldsymbol{V} \cdot \nabla P_j = \sum_{i=1}^{C} \boldsymbol{\Lambda}_{j,i} P_i \quad \forall j \in \{1, 2, \ldots, C\}, \tag{A1}$$

where the stochastic property of $\boldsymbol{\Pi}$ is ensured by construction using the Softmax function as follows:

$$\Pi_{i,k} = \text{Softmax}(M_i)_k = \frac{e^{M_{i,k}}}{\sum_{j=1}^{C} e^{M_{i,j}}},$$

where the matrix $\mathbf{M}$ is generated using a U-Net model.

This representation of cloud cover dynamics offers a comprehensive description of cloud formation and dissipation. However, it increases the output dimension size of U-Net, as a $C \times C$ transition matrix is generated for each pixel. This makes the U-Net model poorly constrained. Furthermore, in our experiments, we noticed an increased memory usage during the training.

A2 Advection with source term: HyPhAICC-4

To reduce the number of values output by U-Net, we assume a limited number of transition regimes, each representing one of the most frequent transitions. For instance, in the case of two regimes, the source term is described as follows:

$$\partial_t P_j = \alpha^1 \cdot \sum_{i=1}^{C} \Lambda^1_{j,i} P^i + \alpha^2 \cdot \sum_{i=1}^{C} \Lambda^2_{j,i} P^i,$$

where Λ^1 and Λ^2 are the transition matrices and α^1 and α^2 are positive factors. These factors determine which regime to consider at each pixel, with the constraint that $\alpha^1 + \alpha^2 \leq 1$.

The fourth version of the hybrid model, denoted as HyPhAICC-4, uses this source term in addition to the advection as described in the following equations:

$$\partial_t P_j + \boldsymbol{V} \cdot \nabla \boldsymbol{P_j} = \alpha^1 \cdot \sum_{i=1}^{C} \Lambda^1_{j,i} P^i + \alpha^2 \cdot \sum_{i=1}^{C} \Lambda^2_{j,i} P^i, \tag{A2}$$

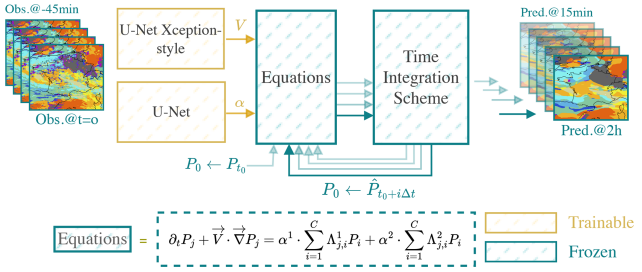


Figure A2. HyPhAICC-4. The fourth version of the proposed hybrid model. It consists of a U-Net Xception-style model to estimate the velocity field and a second U-Net model to estimate the α factors from the last observations. These factors are used to choose which transition regime to consider for each pixel.

where α^1 and α^2 are estimated using U-Net and Λ^1 and Λ^2 are learned as model parameters (see Fig. A2).

HyPhAICC-3 and HyPhAICC-3 are trained using the same dataset and training procedure as for HyPhAICC-1 and HyPhAICC-2. However, during training the Λ matrices in Eqs. (A1) and (A2) are consistently estimated as zeros. In other words, no inter-class transitions were captured.

A3 Scores

Figure A3 represents the score comparison shown in Fig. 6 but with additional confidence intervals. These confidence intervals were estimated using bootstrapping with a threshold of 99 %.

A4 Full-disc predictions

Figure A4 shows predictions of the HyPhAICC-1 model on the Earth's full disc centred at 0° longitude.

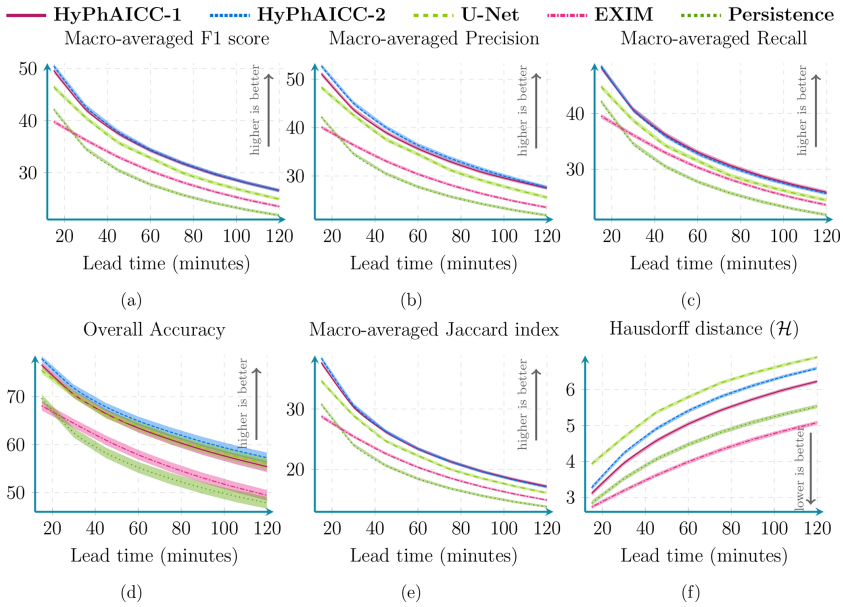


Figure A3. Performance comparison between HyPhAICC-1, U-Net, EXIM, and the persistence baseline using five metrics, including averaged F1 score (%), precision (%), recall (%), accuracy (%), CSI (%), and Hausdorff distance (defined in Eq. 9). These scores were computed over 1000 random samples covering France in 2021. The confidence intervals were estimated using bootstrapping with a threshold of 99 %.

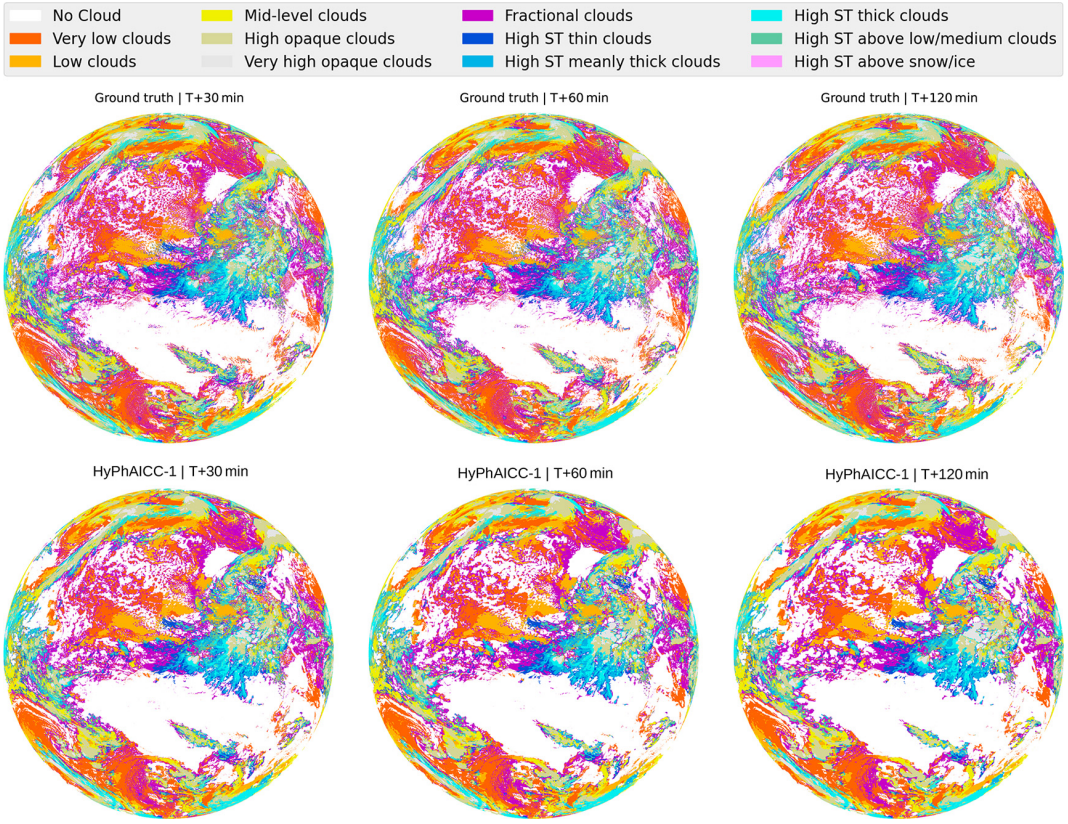


Figure A4. Full-disc cloud cover nowcasting predictions. The predictions were generated by our model without any specific training on the full disc data (of size 3712×3712).

Appendix B: Bridging neural networks and numerical modelling

In this section, we present fundamental components for implementing the proposed hybrid architecture. In Sect. B1 we explore the integration of physics within a neural network. We then explain the trainability challenges associated with this architecture in Sect. B2. Following this, in Sect. B3 we provide a brief introduction to numerical methods for solving PDEs. Finally, in Sect. B4 and B5, we present the method used to approximate derivatives and perform time integration within a neural network.

B1 Combining neural networks and physics

An artificial neural network is a function f_θ parameterised by a set of parameters θ. It results from the composition of a sequence of elementary non-linear parameterised functions called layers that process and transform input data x into output predictions y as follows:

$$y = f_\theta(x). \tag{B1}$$

Physics-based models aim to represent the underlying physical processes or equations that govern the behaviour of a system. To incorporate physics into the neural network, one possible approach involves feeding the output of the physics-based model as an input to the neural network as follows:

$$y = f_\theta\left(x, \phi\left(x_{\text{Phy}}\right)\right), \tag{B2}$$

where x_{Phy} are the inputs of the physics-based model ϕ. This method could be effective when the physics-based model is self-contained and its components are explicitly known. However, it becomes impractical in scenarios where the physics-based model presents unknown variables that need to be estimated. This is the case in the application considered in this work, where the cloud motion field is unknown. In such instances, a more suitable approach is to pursue a joint resolution. Here, the physical model takes the outputs of the neural network and computes the predictions, resulting in a composition of f_θ and ϕ as follows:

$$y = \phi \circ f_\theta\left(x, x_{\text{Phy}}\right). \tag{B3}$$

In this approach, ϕ implicitly applies a hard constraint on these outputs. This might contribute to accelerate the convergence of the neural network during the training process.

Unlike the first method (Eq. B2), where the physics-based model ϕ is passive and not involved in the training procedure, the second method raises some challenges concerning the trainability of the architecture.

B2 Training a neural network

Neural networks learn to minimise a loss function $\mathcal{L}_\theta$ by adjusting its set of parameters θ using data. The loss function measures the error between the predicted outcomes $\hat{y}$ and the ground truth y. It is expressed as

$$\mathcal{L}_\theta = \frac{1}{N}\sum_{k=1}^{N} l(y_k, f_\theta(x_k)), \tag{B4}$$

where N is the sample size and l is a measure of the discrepancy between the ground truth y_i and the model's production associated with the input x_i, i.e. $f_\theta \circ \phi(x_i)$. For instance, using $l(a,b) = \| a-b\|^2$ is the measure used to calibrate a regression model, and $\mathcal{L}_\theta$ is then the so-called mean-squared error (MSE).

During this training process, an algorithm called backpropagation is used to optimise model parameters. Backpropagation involves computing the gradient of the loss function with respect to the network's parameters. It indicates how much each weight contributed to the error. This gradient is then used to update the parameters in the direction that minimises $\mathcal{L}_\theta$, following a sequential optimisation algorithm such as gradient descent, as described below:

$$\theta_{\text{updated}} = \theta_{\text{old}} - \gamma \nabla \mathcal{L}_{\theta_{\text{old}}}, \tag{B5}$$

where γ is the magnitude of the descent. In order to perform the backpropagation, we assume that the gradient of the loss function with respect to the model's parameters could be calculated using the chain rule. This assumption is called differentiability. Indeed, neural networks rely on activation functions and operations that are differentiable, allowing the gradients to be propagated backward through the network layers.

In this proposed hybrid approach, PDEs are solved to produce model predictions. If the PDE solver includes non-differentiable steps, the chain rule breaks down, making it impossible to compute gradients within the standard deep-learning frameworks. In what follows, we describe our strategy for modelling and solving PDEs using basic differentiable operations commonly employed in neural networks.

B3 Numerical methods for partial differential equations

Numerical weather prediction involves addressing equations of the form

$$\partial_t f = \mathcal{F}\left(f, \partial_x f, \partial_x^2 f, \ldots\right), \tag{B6}$$

governing the evolution of a univariate or multivariate field f over time. Computers cannot directly solve symbolic PDEs, and a common approach involves a two-stage process to transform the PDE into a mathematical formulation more suitable for computational handling. This process begins by discretising the partial derivatives with respect to spatial coordinates, resulting in an ordinary differential equation. Subsequently, a temporal integration describes the evolution of the system over time.

Spatial discretisation can be performed using several methods, e.g. finite volumes, finite elements, or spectral

methods. However, the simplest one, the finite-difference method, consists of replacing spatial derivatives of f with quantities that depend on the values of f on a grid. For example, on a 1D periodic domain $[0, L]$ of coordinate x, discretised in N grid points $(x_i)_{[0,n-1]}$ ($x_n = x_0$), the central difference method of the first-order spatial derivative reads

$$\partial_x f(t, x_i) \approx \frac{f(t, x_{i+1}) - f(t, x_{i-1})}{2\delta x}, \tag{B7}$$

where $\delta x = x_{i+1} - x_i$ represents the grid resolution. Following spatial discretisation, Eq. (B6) can be written as an ordinary differential equation as follows:

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}t} = \hat{F}(\mathbf{f}), \tag{B8}$$

where $\mathbf{f}(t)$ is the discretised form of f over the spatial domain, i.e. the vector of grid-point values of f at time t, giving $\mathbf{f}(t) = (f(t, x_i))_i$ in the 1D domain mentioned above.

For the time integration, various methods can also be employed, e.g. Euler's method or a fourth-order Runge–Kutta method (RK4) (Runge, 1895; Kutta, 1901). These methods differ in their accuracy, stability, and computational cost. An explicit Euler time integration of Eq. (B8) reads

$$\mathbf{f}_{q+1} = \mathbf{f}_q + \delta t \hat{F}(\mathbf{f}_q), \tag{B9}$$

where $\mathbf{f}_q = \mathbf{f}(t_q)$ and $t_q = q\delta t$ is the discretised time of time step δt.

For the sake of illustration, we consider the advection over the above-mentioned 1D periodic domain, given by the following equation:

$$\partial_t f + u \partial_x f = 0, \tag{B10}$$

where u is a velocity field, whose values on the grid are denoted as $(u_i)_{i\in[0,n-1]}$. Applying central difference and an Euler scheme discretisation yields the following sequential evolution:

$$f_{q+1,i} = f_{q,i} - \frac{\delta t}{2\delta x} u_i \left(f_{i+1} - f_{i-1}\right). \tag{B11}$$

This example illustrates the integration of the advection equation over time using a simple explicit method. However, depending on the problem characteristics and requirements, other time integration schemes may be more suitable.

In this study, we propose to model and solve PDEs within a neural network, e.g. equations of the form Eq. (B6). This is achieved by describing the equivalent of spatial and temporal discretisation in the frame of neural network layers, i.e. how it can be implemented in a deep-learning (DL) framework as TensorFlow (Abadi et al., 2016) or PyTorch (Paszke et al., 2019).

B4 Finite-difference methods and convolutional layers

To implement a finite-difference discretisation, one viable approach is to employ the convolution operation. For instance, the 1D convolution associated with Eq. (B7) can be

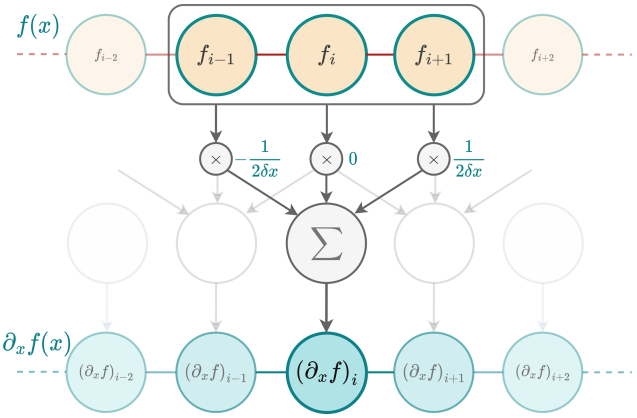


Figure B1. In order to calculate the numerical derivative of f, a kernel K^1 is used to slide across an input vector, which is a discretised version of f with N elements, multiplying values element-wise within its window and summing the results to approximate the derivative at each position. The result is a new vector of size $N-2$ containing the numerical derivative of f (padding at the bounds with zeros or duplicated values in the input vector can be applied to produce an output vector of size N). This is equivalent to a convolution between K^1 and f and can be reproduced using a 1D convolutional layer with K^1 as a kernel.

mathematically written as follows:

$$(K^1 \cdot f)[i] = \sum_{m=0}^{M-1} K^1[i]\, f[m+i], \tag{B12}$$

where K^1 is the kernel or filter used for the convolution and expressed as

$$K^1 = \begin{bmatrix} \frac{-1}{2\delta x} & 0 & \frac{1}{2\delta x} \end{bmatrix},$$

and f represents the input data. The variable M corresponds to the size of the kernel. It is the number of finite-difference coefficients, also called stencil size. In this case, a three-point stencil is considered ($M = 3$). Finally, $*$ is the convolution operator.

This leads to an interesting interaction with DL frameworks. Indeed, convolutional neural networks (CNNs) rely on the operation

$$\mathrm{ConvLayer}(f)[i] = \sigma\left(\sum_{m=0}^{M-1} K[m]\, f[m+i] + b\right),$$

where σ is called activation function and b is a parameter representing the bias. Observing that using data where sigma is equal to identity and b is equal to 0 leads to the same operation as in Eq. (B12), one can leverage deep-learning frameworks to approximate derivatives, which enables derivative-based operations in neural networks, as shown in Fig. B1. The same principle applies to higher derivative orders. For any positive integer α, we can write the approximation of the

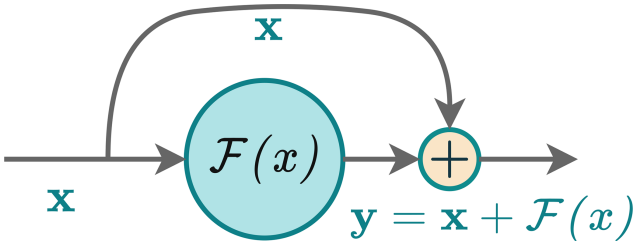


Figure B2. Illustration of a residual block.

αth derivative of f as

$$\partial^{\alpha} f \approx K^{\alpha} * f, \tag{B13}$$

where K^{α} are the finite-difference coefficients for the αth derivative.

Finally, using convolutions is a straightforward method to model the spatial term of a PDE, also called the trend, as follows

$$\hat{F}(f) = \mathcal{N}(f). \tag{B14}$$

This results in a neural network that can be used for time integration.

B5 Temporal schemes and residual networks

The time integration expressed in Eq. (B9) can be written using the neural network implementation $\mathcal{N}$ of the trend as

$$f_{n+1} = f_n + \Delta t \mathcal{N}(f_n). \tag{B15}$$

Interestingly, this formulation is very similar to that of a building block commonly used in deep neural networks called a residual block (see Fig. B2), proposed in the ResNet architecture (He et al., 2016). It is formulated as follows:

$$y = x + \mathcal{F}(x), \tag{B16}$$

where x is the input to the block, y is the output, and $\mathcal{F}$ is called a residual function and is made up of multiple neural layers. These layers represent the difference between the input and output. This function aims to capture the additional information or adjustments needed to transform the input into the desired output. This similarity between residual blocks and time schemes, also observed in Ruthotto and Haber (2020), Chen et al. (2018), and Fablet et al. (2017), suggests that the time integration step can be done inside a neural network. All we need is the residual function, which can be modelled using convolutional layers, as shown previously. Pannekoucke and Fablet (2020) proposed a general framework (called PDE-NetGen (https://github.com/opannekoucke/pdenetgen, last access: 7 June 2024) to model a PDE in a neural network form using this method. Residual blocks were originally designed to address vanishing gradient issues in image classification tasks. Intriguingly, these blocks proved to function similarly to time schemes, where they introduce small changes over incremental time steps. This challenges the traditional black box perception of neural networks, although full interpretability remains a distant goal.

Appendix C: Robustness of hybrid formulation to changes in coordinates

In a given coordinate system $x = (x_i)$, the advection of a passive scalar $c(t, x)$ by a velocity field $u = (u_i)$ reads as follows:

$$\partial_t c + u_i \partial_{x_i} c = 0. \tag{C1}$$

A change in coordinate system from the coordinate system x to the coordinate system $y = (y_j)$ related by $x = x(y)$ changes this to the following equation:

$$\partial_t C + v_j \partial_{y_j} C = 0, \tag{C2}$$

where $C(t, y) = c(t, x(y))$ and the velocity $v = (v_j)$ is deduced from the chain rule

$$v_j = u_i \partial_{x_i} y_j, \tag{C3}$$

(using Einstein's summation convention).

Since HyPhAICC architecture estimates a velocity field from the data that is either u or v depending on the choice of the coordinate system, it implicitly accounts for the chain rule Eq. (C3). As a result, the HyPhAICC architecture is not sensitive to the coordinate system and can apply to regional domain and global projections. However, numerical effects due to the finite spatio-temporal resolution associated with the discretisation can lead to abnormal distortion of signals after several time steps of integration; e.g. the disc resulting from an orthographic projection of the Earth may be deformed by the advection near its boundaries unless the velocity field is close to zero, meaning that the apparent displacement is small.

Note that this relative invariance of HyPhAICC to the choice of coordinate is because it only concerns the advection of a scalar field. Covariant transport of vector or tensor fields would imply additional terms (Christoffel symbols, e.g. Nakahara, 2003) that would break the invariance of HyPhAICC as it is formulated here.

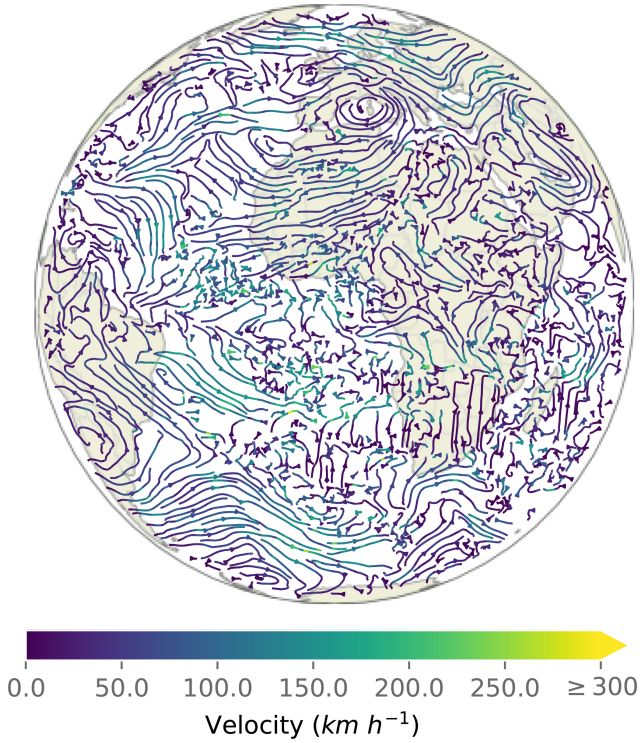


Figure C1. Estimated velocity field from the U-Net Xception-style architecture used in the HyPhAICC-1 model.

Appendix D: Probability advection

Here, we are considering a three-class problem where we have a discrete random variable X with values in the set $1, 2, 3$, and we denote $X(t, x)$ using the value of X at time t and space x, with $t \in [0, T]$ and $x \in [0, L]$. We are interested in studying the evolution of the state probabilities of X with respect to t and x. For this purpose, we define a vector $\mathcal{P}$ as

$$\mathcal{P} = \begin{bmatrix} P_X^1 \\ P_X^2 \\ P_X^3 \end{bmatrix}.$$

Here, $P_X^c(t, x)$ represents the probability of the c^{th} class:

$$P_X^c(t, x) = P(X(t, x) = c).$$

For the sake of simplicity, a 1D problem is considered, but the same analysis applies to the 2D case and for N-class problems with $N \geq 2$. Let us consider the following partial differential equation governing the evolution of $\mathcal{P}(x, t)$:

$$\partial_t \mathcal{P}(x, t) + \mathcal{L}(\mathcal{P}(x, t)) = 0, \tag{D1}$$

where $\mathcal{L}$ is a differential operator. This equation can be written component-wise as follows:

$$\begin{cases} \partial_t P_X^1(x, t) + \mathcal{L}\left(P_X^1(x, t)\right) = 0 \\ \partial_t P_X^2(x, t) + \mathcal{L}\left(P_X^2(x, t)\right) = 0 \\ \partial_t P_X^3(x, t) + \mathcal{L}\left(P_X^3(x, t)\right) = 0 \end{cases}. \tag{D2}$$

As already discussed in Sect. 2.2.1, three properties should be checked in order to ensure the probabilistic nature of P.

1. *Non-negativity.* $P(\mathbf{x}, t) \geq 0$ for all values of $\mathbf{x}$ and t, with $\mathbf{x} = (x, y)$, which ensures that the probabilities remain non-negative.
2. *Bound preservation.* $P(\mathbf{x}, t) \leq 1$ for all values of $\mathbf{x}$ and t, which ensures that no probability exceeds 1.
3. *Probability conservation.* $\sum_{i=1}^{C} P_X^i(\mathbf{x}, t) = 1$ for all values of $\mathbf{x}$ and t, with $C = 12$ being the total number of cloud types. This property guarantees that the sum of all probabilities is equal to 1.

D1 Probability conservation

Property. The probability conservation property is ensured if $\mathcal{L}$ is a linear differential operator with non-zero positive spatial derivative orders.

Proof. Let us sum the three equations in Eq. (D2) for the specific case where $\mathcal{L}$ is a linear differential operator with non-zero positive spatial derivative orders.

$$\sum_{i=1}^{3} \partial_t P_X^i(x, t) + \mathcal{L}\left(P_X^i(x, t)\right) = 0$$

$$\partial_t \sum_{i=1}^{3} P_X^i(x, t) = -\sum_{i}^{3} \mathcal{L}\left(P_X^i(x, t)\right).$$

Assuming $\sum_{i=1}^{3} P_X^i(x, t_0) = 1$, the linearity property of $\mathcal{L}$ allows us to interchange the summation and the operator, resulting in the following equation:

$$\begin{aligned} \sum_{i}^{3} \mathcal{L}\left(P_X^i(x, t_0)\right) &= -\mathcal{L}\left(\sum_{i=1}^{3} P_X^i(x, t_0)\right) \\ &= -\mathcal{L}(1) \\ &= 0. \end{aligned}$$

$\mathcal{L}(1) = 0$ as $\mathcal{L}$ only has derivatives with positive non-zero orders.

Applying and summing the first-order Taylor expansion at t_0 on each of the time derivatives of Eq. (D2) gives

$$\begin{aligned} \sum_{i}^{3} \frac{P_i(x, t_0 + \delta t) - P_i(x, t_0)}{\delta t} + \mathcal{O}(1) &= -\sum_{i}^{3} \mathcal{L}\left(P_X^i(x, t)\right) \\ &= 0 \end{aligned}$$

$$\sum_{i}^{3} P_i(x, t_0 + \delta t) = \sum_{i}^{3} P_i(x, t_0) + \mathcal{O}(\delta t),$$

when δt is small enough, $\sum_{i}^{3} P_i(x, t_0 + \delta t) = 1$.

Iteratively, starting from t_0, $\forall t$

$$\sum_{i}^{3} P_i(x, t) = \sum_{i}^{3} P_i(x, t_0) = 1.$$

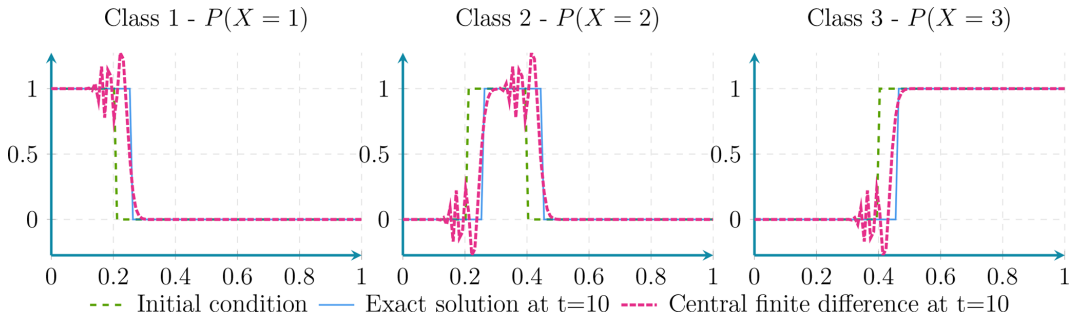


Figure D1. Here it can be seen that the advection of probabilities using central finite-difference discretisation presents a dispersion effect.

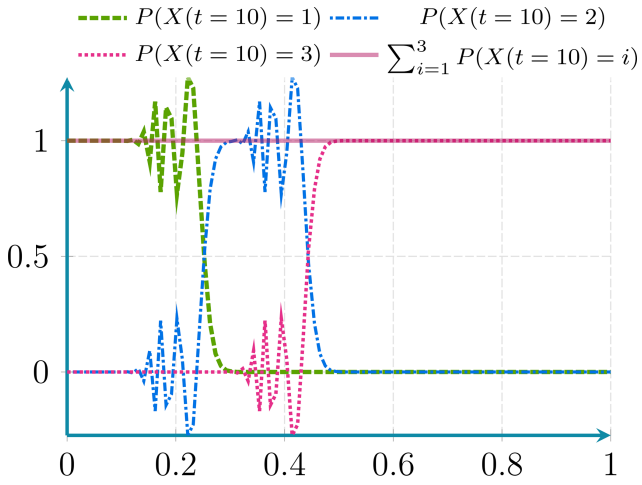


Figure D2. Here the probability conservation property is maintained even in presence of dispersion effects.

In this study, we consider the advection equation using the same velocity field for all probability maps, where the operator $\mathcal{L}$ is written as follows:

$$\mathcal{L}(P_i) = u \cdot \partial_x P_i, \quad i \in \{1, 2, \cdot, 12\}.$$

This differential operator is linear and has a non-zero positive derivative order. Therefore, the sum of probabilities is conserved over time and remains equal to the initial value. This property is illustrated numerically in Figs. D2 and D4, and it is maintained independently of the discretisation scheme.

D2 Non-negativity and bound preservation

In order to check the two other properties, we need to study the discretisation schemes.

Out of the four numerical schemes studied (central finite differences, semi-Lagrangian, and first- and second-order upwind), only the semi-Lagrangian and the first-order upwind discretisation satisfy the first and second properties. The remaining two schemes exhibit some form of dispersion.

Details about central finite difference and first-order upwind scheme are given in Sect. E.

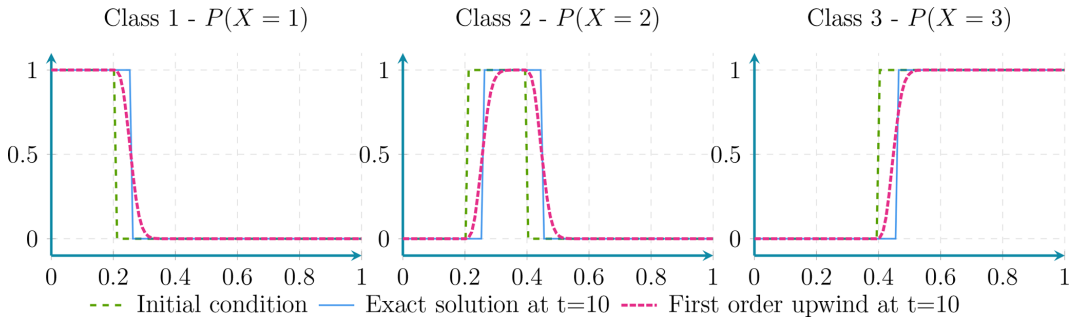


Figure D3. Here the advection of probabilities using first-order upwind discretisation presents a diffusion effect.

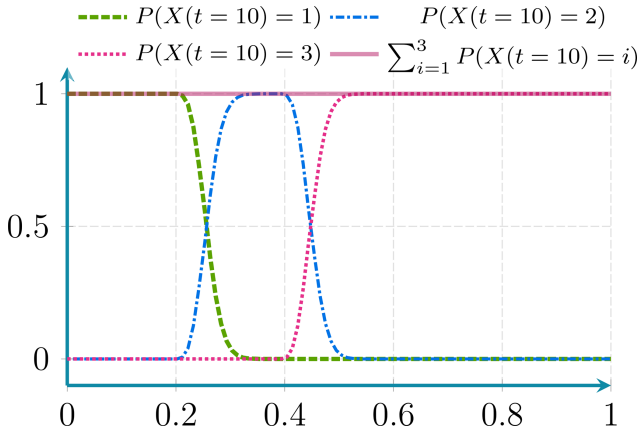


Figure D4. Here the probability conservation property is maintained even in presence of diffusion effects.

Appendix E: Discretisation schemes

Here, we will derive the equivalent equation of central differences and upwind scheme applied to the following advection equation:

$$\frac{\partial F(x,t)}{\partial t} + u\frac{\partial F(x,t)}{\partial x} = 0. \tag{E1}$$

E1 Central differences: equivalent equation

We consider the second-order central discretisation in space and a first-order explicit forward difference in time applied to the advection equation.

$$\frac{F_i^{n+1} - F_i^n}{\Delta t} + u_i \frac{F_{i+1} - F_{i-1}}{2\Delta x} = 0 \tag{E2}$$

Using the Taylor formulas in Eq. (E2), we get

$$\partial_t F + \frac{\Delta t}{2}\partial_t^2 F + \mathcal{O}(\Delta t^2) = -u\left(\partial_x F - \frac{\Delta x^2}{6}\partial_x^3 F + \mathcal{O}(\Delta x^2)\right). \tag{E3}$$

However, when we only require a first-order expansion in time, we can replace the second-order time derivative with another term coming from a Taylor first-order expansion of Eq. (E2) :

$$\partial_t(\partial_t F) + \mathcal{O}(\Delta t) = -\partial_t(u\partial_x F) + \mathcal{O}(\Delta x), \tag{E4}$$

which leads to

$$\partial_t^2 F = -\partial_t u \partial_x F - u\partial_{xt}^2 F + \mathcal{O}(\Delta t, \Delta x).$$

Using the same approach as in Eq. (E4), the derivative $\partial_{xt}^2 F$ can be computed as follows:

$$\partial_x(\partial_t F) = -\partial_x u \partial_x F - u\partial_x^2 F + \mathcal{O}(\Delta t, \Delta x).$$

We replace the derivative $\partial_{xt}^2 F$ in the last equation as follows:

$$\partial_t^2 F = -\partial_t u \partial_x F - u\left(-\partial_x u \partial_x F - u\partial_x^2 F\right) + \mathcal{O}(\Delta t, \Delta x) \quad . \tag{E5}$$

Finally, we replace the second-order derivative in Eq. (E3) with the expression in Eq. (E5):

$$\partial_t F + \frac{\Delta t}{2}\left(-\partial_t u \partial_x F - u\left(-\partial_x u \partial_x F - u\partial_x^2 F\right)\right) = -u\left(\partial_x F - \frac{\Delta x^2}{6}\partial_x^3 F\right) + \mathcal{O}\left(\Delta t^2, \Delta x^2\right).$$

Hence,

$$\partial_t F + \tilde{u}\partial_x F = -\frac{\Delta t}{2}u^2\partial_x^2 F + \frac{\Delta x^2}{6}u\partial_x^3 F + \mathcal{O}\left(\Delta t^2, \Delta x^2\right), \tag{E6}$$

where $\tilde{u} = u - \frac{\Delta t}{2}\partial_t u + \frac{\Delta t}{2}u\partial_x u$.

E2 First-order upwind scheme: equivalent equation

Now let us consider the first-order upwind discretisation of the spatial term given by

$$\begin{cases} \frac{F_i^{n+1} - F_i^n}{\Delta t} + u\frac{F_i - F_{i-1}}{\Delta x} = 0 & \text{if } u \geq 0 \\ \frac{F_i^{n+1} - F_i^n}{\Delta t} + u\frac{F_{i+1} - F_i}{\Delta x} = 0 & \text{if } u < 0 \end{cases}.$$

These two equations can be written as follows:

$$\frac{F_i^{n+1} - F_i^n}{\Delta t} + \left(u_i^+\frac{F_i - F_{i-1}}{\Delta x} + u_i^-\frac{F_{i+1} - F_i}{\Delta x}\right) = 0, \tag{E7}$$

where $u_i^+ = \max(u_i, 0)$ and $u_i^- = \min(u_i, 0)$.

Considering the case of $u \geq 0$ of Eq. (E7), using the Taylor formulas, we get:

$$\partial_t F + \frac{\Delta t}{2}\partial_t^2 F + \mathcal{O}\left(\Delta t^2\right) = -u\left(\partial_x F - \frac{\Delta x}{2}\partial_x^2 F + \mathcal{O}\left(\Delta x^2\right)\right). \tag{E8}$$

As in the case of the central differences, we replace the second-order derivative $\partial_t^2 F$ in Eq. (E8) with the expression in Eq. (E5).

$$\partial_t F + \frac{\Delta t}{2}\left(-\partial_t u \partial_x F - u\left(-\partial_x u \partial_x F - u\partial_x^2 F\right)\right) = -u\left(\partial_x F - \frac{\Delta x}{2}\partial_x^2 F\right) + \mathcal{O}\left(\Delta t^2, \Delta x^2\right)$$

Hence,

$$\partial_t F + \tilde{u}\partial_x F = v_{\text{num}}\partial_x^2 F + \mathcal{O}\left(\Delta t^2, \Delta x^2\right), \tag{E9}$$

where $\tilde{u} = u - \frac{\Delta t}{2}\partial_t u + \frac{\Delta t}{2}u\partial_x u$ and $v_{num} = \frac{u}{2}(\Delta x - u\Delta t)$.

The equivalent equation of the second case of Eq. (E7) (case $u \leq 0$) is written as follows:

$$\partial_t F + \tilde{u}\partial_x F = v_{\text{num}}\partial_x^2 F + \mathcal{O}\left(\Delta t^2, \Delta x^2\right), \tag{E10}$$

where $v_{\text{num}} = \frac{u}{2}(-\Delta x - u\Delta t)$

From Eqs. (E9) and (E10) we can write the equivalent equation as follows:

$$\partial_t F + \tilde{u}\partial_x F = v_{\text{num}}\partial_x^2 F + \mathcal{O}\left(\Delta t^2, \Delta x^2\right), \tag{E11}$$

where $\tilde{u} = u - \frac{\Delta t}{2}\partial_t u + \frac{\Delta t}{2}u\partial_x u$ and $v_{\text{num}} = \frac{u}{2}(\text{sign}(u)\Delta x - u\Delta t)$..

E3 Conclusion

It should be noted that the finite central difference scheme exhibits instability due to the presence of negative diffusion in the second term in Eq. (E6). However, when using a temporal scheme of higher order than two, the negative diffusion term

in Δt can be eliminated, rendering the scheme stable. Nevertheless, the scheme becomes dispersive due to the third-order spatial derivative term, resulting in oscillations during the propagation of sharp signals, such as a front or Heaviside function.

Alternatively, the first-order upwind scheme offers stability but introduces numerical diffusion, affecting the accuracy of the solution, this diffusion is due to the second-order derivative term in Eq. (E11).

Finally, the choice of numerical scheme depends on the specific requirements of the problem, such as the desired accuracy and stability of the solution. To respect the properties described above, we use the first-order upwind scheme, as it does not introduce oscillations in the solution. The first-order upwind scheme is also easy to implement in a differentiable mode. Despite the limitation on the time step linked to the CFL condition, we consider it to be a more appropriate scheme to integrate probability advection in a neural network.

Code and data availability. The code used in this study is available at https://github.com/relmonta/hyphai (last access: 7 June 2024) and at https://doi.org/10.5281/zenodo.11518540 (El Montassir, 2024). The weights of the pre-trained HyPhAICC-1, HyPhAICC-2, and U-Net are available at https://doi.org/10.5281/zenodo.10393415 (El Montassir et al., 2023a). The training data are not provided as they are proprietary data from EUMETSAT. However, data can be obtained from EUMETSAT for research purposes. A sample of the test data used in this study is available on the GitHub repository, and a sample of the training data is available at https://doi.org/10.5281/zenodo.10642094 (European Organisation for the Exploitation of Meteorological Satellites, 2024).

Interactive computing environment. Three Jupyter notebooks are provided at https://github.com/relmonta/hyphai/tree/main/examples (last access: 7 June 2024) or at https://github.com/relmonta/hyphai/tree/main/examples (last access: 7 June 2024) and at https://doi.org/10.5281/zenodo.11518540 (El Montassir, 2024). Each notebook corresponds to an example of the use of HyPhAICC-1, HyPhAICC-2, and the baseline U-Net model.

Video supplement. A video supplement of a 2 h forecast is available at https://doi.org/10.5281/zenodo.10375284 (El Montassir et al., 2023b).

Author contributions. REM implemented the code, performed the experiments, and wrote the first draft of the manuscript. CL and OP supervised the work. All authors collaborated on the design of the models and contributed to the manuscript's writing.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. While Copernicus Publications makes every effort to include appropriate place names, the final responsibility lies with the authors.

Acknowledgements. We extend our gratitude to CERFACS for funding this work and providing computing resources and to EUMETSAT and Météo France for providing essential data. We express our sincere thanks to Luciano Drozda, Léa Berthomier, Bruno Pradel, and Pierre Lepetit for providing constructive discussions and feedback and to Isabelle d'Ast for technical support. We also thank the editor, Travis O'Brien, and the two reviewers, Alban Farchi and the anonymous reviewer, for their valuable comments and suggestions.

Financial support. This research has been supported by CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique).

Review statement. This paper was edited by Travis O'Brien and reviewed by Alban Farchi and one anonymous referee.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X.: TensorFlow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, OSDI'16, pp. 265–283, USENIX Association, USA, ISBN 978-1-931971-33-1, 2016.

Aydin, O. U., Taha, A. A., Hilbert, A., Khalil, A. A., Galinovic, I., Fiebach, J. B., Frey, D., and Madai, V. I.: On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking, European Radiology Experimental, 5, 4, https://doi.org/10.1186/s41747-020-00200-2, 2021.

Ayzel, G., Scheffer, T., and Heistermann, M.: RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting, Geosci. Model Dev., 13, 2631–2644, https://doi.org/10.5194/gmd-13-2631-2020, 2020.

Ballard, S. P., Li, Z., Simonin, D., and Caron, J.-F.: Performance of 4D-Var NWP-based nowcasting of precipitation at the Met Office for summer 2012, Q. J. Roy. Meteor. Soc., 142, 472–487, https://doi.org/10.1002/qj.2665, 2016.

Bechini, R. and Chandrasekar, V.: An Enhanced Optical Flow Technique for Radar Nowcasting of Precipitation and Winds, J. Atmos. Ocean. Tech., 34, 2637–2658, https://doi.org/10.1175/JTECH-D-17-0110.1, 2017.

Berthomier, L., Pradel, B., and Perez, L.: Cloud Cover Nowcasting with Deep Learning, in: 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA), 1–6, https://doi.org/10.1109/IPTA50016.2020.9286606, 2020.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K.: Neural Ordinary Differential Equations, in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., https://doi.org/10.48550/arXiv.1806.07366, 2018.

Cheng, S., Quilodrán-Casas, C., Ouala, S., Farchi, A., Liu, C., Tandeo, P., Fablet, R., Lucor, D., Iooss, B., Brajard, J., Xiao, D., Janjic, T., Ding, W., Guo, Y., Carrassi, A., Bocquet, M., and Arcucci, R.: Machine Learning With Data Assimilation and Uncertainty Quantification for Dynamical Systems: A Review, IEEE/CAA Journal of Automatica Sinica, 10, 1361–1387, https://doi.org/10.1109/JAS.2023.123537, 2023.

Chollet, F.: Xception: Deep Learning with Depthwise Separable Convolutions, IEEE Computer Society, ISBN 978-1-5386-0457-1, https://doi.org/10.1109/CVPR.2017.195, 2017.

Courant, R., Friedrichs, K., and Lewy, H.: Über die partiellen Differenzengleichungen der mathematischen Physik, Mathematische Annalen, 100, 32–74, https://doi.org/10.1007/BF01448839, 1928.

Daw, A., Karpatne, A., Watkins, W., Read, J., and Kumar, V.: Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling, in: Knowledge Guided Machine Learning, Chapman and Hall/CRC, https://doi.org/10.1201/9781003143376-15, 2021.

de Bezenac, E., Pajot, A., and Gallinari, P.: Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge, J. Stat, Mech., 2019, 124009, https://doi.org/10.1088/1742-5468/ab3195, 2018.

Efron, B.: Bootstrap Methods: Another Look at the Jackknife, Ann. Stat., 7, 1–26, https://doi.org/10.1214/aos/1176344552, 1979.

El Montassir, R.: relmonta/hyphai: Update paper information (v1.1.1), Zenodo [code], https://doi.org/10.5281/zenodo.11518540, 2024.

El Montassir, R., Pannekoucke, O., and Lapeyre, C.: Pre-trained HyPhAICCast-1, HyPhAICCast-2 and U-Net's weights, Zenodo [code], https://doi.org/10.5281/zenodo.10393415, 2023a.

El Montassir, R., Pannekoucke, O., and Lapeyre, C.: HyPhAICCast-1 2-hour forecast on 01/01/2021 at 12:00 p.m., Zenodo [video], https://doi.org/10.5281/zenodo.10375284, 2023b.

Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., Gazen, C., Carver, R., Andrychowicz, M., Hickey, J., Bell, A., and Kalchbrenner, N.: Deep learning for twelve hour precipitation forecasts, Nat. Commun., 13, 5145, https://doi.org/10.1038/s41467-022-32483-x, 2022.

European Organisation for the Exploitation of Meteorological Satellites: A sample of the training data used in the paper "A Hybrid Physics-AI (HyPhAI) approach for probability fields advection: Application to cloud cover nowcasting", Zenodo [data set], https://doi.org/10.5281/zenodo.10642094, 2024.

Fablet, R., Ouala, S., and Herzet, C.: Bilinear residual Neural Network for the identification and forecasting of dynamical systems, 26th European Signal Processing Conference (EUSIPCO), Rome, Italy, 2018, pp. 1477–1481, https://doi.org/10.23919/EUSIPCO.2018.8553492, 2017.

Fernandes, B., González-Briones, A., Novais, P., Calafate, M., Analide, C., and Neves, J.: An Adjective Selection Personality Assessment Method Using Gradient Boosting Machine Learning, Processes, 8, 618, https://doi.org/10.3390/pr8050618, 2020.

Fokker, A. D.: Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld, Annalen der Physik, 348, 810–820, https://doi.org/10.1002/andp.19143480507, 1914.

Forssell, U. and Lindskog, P.: Combining Semi-Physical and Neural Network Modeling: An Example ofIts Usefulness, IFAC Proceedings Volumes, 30, 767–770, https://doi.org/10.1016/S1474-6670(17)42938-7, 1997.

García-Pereda, J., Fernandez-Serdan, J. M., Alonso, O., Sanz, A., Guerra, R., Ariza, C., Santos, I., and Fernández, L.: NWCSAF High Resolution Winds (NWC/GEO-HRW) Stand-Alone Software for Calculation of Atmospheric Motion Vectors and Trajectories, Remote Sens., 11, 2032, https://doi.org/10.3390/rs11172032, 2019.

Gilbert, G. K.: Finley's tornado predictions, Am. Meteorol. J., 1, 166–172, 1884.

He, K., Zhang, X., Ren, S., and Sun, J.: Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, https://doi.org/10.1109/CVPR.2016.90, iSSN: 1063-6919, 2016.

Hochreiter, S. and Schmidhuber, J.: Long Short-term Memory, Neural computation, 9, 1735–80, https://doi.org/10.1162/neco.1997.9.8.1735, 1997.

Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., and Kumar, V.: Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles, in: Proceedings of the 2019 SIAM International Conference on Data Mining (SDM), Proceedings, 558–566, Society for Industrial and Applied Mathematics, https://doi.org/10.1137/1.9781611975673.63, 2019.

Jia, X., Willard, J., Karpatne, A., Read, J. S., Zwart, J. A., Steinbach, M., and Kumar, V.: Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles, ACM/IMS Transactions on Data Science, 2, 20:1–20:26, https://doi.org/10.1145/3447814, 2021.

Joe, P., Sun, J., Yussouf, N., Goodman, S., Riemer, M., Gouda, K. C., Golding, B., Rogers, R., Isaac, G., Wilson, J., Li, P. W. P., Wulfmeyer, V., Elmore, K., Onvlee, J., Chong, P., and Ladue, J.: Predicting the Weather: A Partnership of Observation Scientists and Forecasters, in: Towards the "Perfect" Weather Warning: Bridging Disciplinary Gaps through Partnership and Communication, edited by: Golding, B., 201–254, Springer International Publishing, Cham, ISBN 978-3-030-98989-7, https://doi.org/10.1007/978-3-030-98989-7_7, 2022.

Karimi, D. and Salcudean, S. E.: Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks, in: IEEE Transactions on Medical Imaging, 39, 499–513, https://doi.org/10.1109/TMI.2019.2930068, 2019.

Karpatne, A., Atluri, G., Faghmous, J., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V.: Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data, IEEE T. Knowl. Data En., 29, 2318–2331, https://doi.org/10.1109/TKDE.2017.2720168, 2017.

Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings, ArXiv [preprint], https://doi.org/10.48550/arXiv.1412.6980, 2017.

Kutta, W.: Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, Zeitschrift für Mathematik und Physik, 46, 435–453, 1901.

Lin, C., Vasić, S., Kilambi, A., Turner, B., and Zawadzki, I.: Precipitation forecast skill of numerical weather prediction models and radar nowcasts, Geophys. Res. Lett., 32, 14, https://doi.org/10.1029/2005GL023451, 2005.

Matte, D., Christensen, J. H., Feddersen, H., Vedel, H., Nielsen, N. W., Pedersen, R. A., and Zeitzen, R. M. K.: On the Potentials and Limitations of Attributing a Small-Scale Climate Event, Geophys. Res. Lett., 49, e2022GL099481, https://doi.org/10.1029/2022GL099481, 2022.

Nakahara, M.: Geometry, Topology and Physics (Second Edition), Taylor & Francis, ISBN 0-7503-0606-8, 2003.

Pannekoucke, O. and Fablet, R.: PDE-NetGen 1.0: from symbolic partial differential equation (PDE) representations of physical processes to trainable neural network representations, Geosci. Model Dev., 13, 3373–3382, https://doi.org/10.5194/gmd-13-3373-2020, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: an imperative style, high-performance deep learning library, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 8026–8037, Curran Associates Inc., Red Hook, NY, USA, 2019.

Pavliotis, G. and Stuart, A.: Multiscale Methods: Averaging and Homogenization, vol. 53, Springer, New York, NY, ISBN 978-0-387-73828-4, https://doi.org/10.1007/978-0-387-73829-1, 2008.

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E.: Deep Learning of Vortex Induced Vibrations, J. Fluid Mech., 861, 119–137, https://doi.org/10.1017/jfm.2018.872, 2019.

Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., Prudden, R., Mandhane, A., Clark, A., Brock, A., Simonyan, K., Hadsell, R., Robinson, N., Clancy, E., Arribas, A., and Mohamed, S.: Skilful precipitation nowcasting using deep generative models of radar, Nature, 597, 672–677, https://doi.org/10.1038/s41586-021-03854-z, 2021.

Ronneberger, O., Fischer, P., and Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation, in: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Lecture Notes in Computer Science, edited by: Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., vol. 9351, 234–241, Springer International Publishing, Cham, ISBN 978-3-319-24573-7 978-3-319-24574-4, https://doi.org/10.1007/978-3-319-24574-4_28, 2015.

Runge, C.: Ueber die numerische Auflösung von Differentialgleichungen, Mathematische Annalen, 46, 167–178, https://doi.org/10.1007/BF01446807, 1895.

Ruthotto, L. and Haber, E.: Deep Neural Networks Motivated by Partial Differential Equations, J. Math. Imaging Vis., 62, 352–364, https://doi.org/10.1007/s10851-019-00903-1, 2020.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., and Stadtler, S.: Can deep learning beat numerical weather prediction?, Philos. T. A, 379, 20200097, https://doi.org/10.1098/rsta.2020.0097, 2021.

Schweidtmann, A. M., Zhang, D., and von Stosch, M.: A review and perspective on hybrid modeling methodologies, Digital Chemical Engineering, 10, 100 136, https://doi.org/10.1016/j.dche.2023.100136, 2024.

Shah, S., Dey, D., Lovett, C., and Kapoor, A.: AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, in: Field and Service Robotics. Springer Proceedings in Advanced Robotics, edited by: Hutter, M. and Siegwart, R., vol 5, Springer, Cham, https://doi.org/10.1007/978-3-319-67361-5_40, 2017.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c.: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, in: Advances in Neural Information Processing Systems, vol. 28, Curran Associates, Inc., https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html (last access: 7 June 2024), 2015.

Sokolova, M. and Lapalme, G.: A systematic analysis of performance measures for classification tasks, Inform. Process. Manage., 45, 427–437, https://doi.org/10.1016/j.ipm.2009.03.002, 2009.

Sultan, M. M., Wayment-Steele, H. K., and Pande, V. S.: Transferable Neural Networks for Enhanced Sampling of Protein Dynamics, J. Chem. Theor. Comput., 14, 1887–1894, https://doi.org/10.1021/acs.jctc.8b00025, 2018.

Sun, J., Xue, M., Wilson, J. W., Zawadzki, I., Ballard, S. P., Onvlee-Hooimeyer, J., Joe, P., Barker, D. M., Li, P.-W., Golding, B., Xu, M., and Pinto, J.: Use of NWP for Nowcasting Convective Precipitation: Recent Progress and Challenges, B. Am. Meteorol. Soc., 95, 409–426, https://doi.org/10.1175/BAMS-D-11-00263.1, 2014.

Takahashi, K., Yamamoto, K., Kuchiba, A., and Koyama, T.: Confidence interval for micro-averaged F1 and macro-averaged F1 scores, Applied Intelligence, 52, 4961–4972, https://doi.org/10.1007/s10489-021-02635-5, 2022.

Tamvakis, P. N., Kiourt, C., Solomou, A. D., Ioannakis, G., and Tsirliganis, N. C.: Semantic Image Segmentation with Deep Learning for Vine Leaf Phenotyping, IFAC-PapersOnLine, 55, 83–88, https://doi.org/10.1016/j.ifacol.2022.11.119, 2022.

Trebing, K., Stanczyk, T., and Mehrkanoon, S.: SmaAt-UNet: Precipitation nowcasting using a small attention-UNet architecture, Pattern Recogn. Lett., 145, 178–186, https://doi.org/10.1016/j.patrec.2021.01.036, 2021.

Wang, S.-H., Nayak, D. R., Guttery, D. S., Zhang, X., and Zhang, Y.-D.: COVID-19 classification by CCSHNet with deep fusion using transfer learning and discriminant correlation analysis, Information Fusion, 68, 131–148, https://doi.org/10.1016/j.inffus.2020.11.005, 2021.

Wang, Y., Gao, Z., Long, M., Jianmin Wang, Wang, J., Yu, P. S., and Philip S. Yu: PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning, ICML, pp. 5110–5119, aRXIV_ID: 1804.06300 MAG ID: 2963326684 S2ID: d718941506d2adabc4792cb13d49e6336957e52e, 2018.

Wang, Y., Zhang, J., Zhu, H., Long, M., Wang, J., and Yu, P. S.: Memory in Memory: A Predictive Neural Network for Learning Higher-Order Non-Stationarity From Spatiotemporal Dynamics, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 9146–9154, https://doi.org/10.1109/CVPR.2019.00937, 2019.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V.: Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems, ACM Computing Surveys, 55, 1–37, https://doi.org/10.1145/3514228, 2022.

Wood-Bradley, P., Zapata, J., and Pye, J.: Cloud tracking with optical flow for short-term solar forecasting, in: Proceedings of 50th Annual AuSES Conference (Solar 2012), Australian Solar Energy Society, https://openresearch-repository.anu.edu.au/handle/1885/28800 (last access: 17 November 2022), 2012.

Chapter 10

Bibliography

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: a system for large-scale machine learning. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, OSDI'16, pages 265–283, USA, Nov. 2016. USENIX Association. ISBN 978-1-931971-33-1. 41, 71

O. U. Aydin, A. A. Taha, A. Hilbert, A. A. Khalil, I. Galinovic, J. B. Fiebach, D. Frey, and V. I. Madai. On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking. European Radiology Experimental, 5(1):4, Jan. 2021. ISSN 2509-9280. doi: 10.1186/s41747-020-00200-2. URL https://doi.org/10.1186/s41747-020-00200-2. 104

G. Ayzel, T. Scheffer, and M. Heistermann. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. Geoscientific Model Development, 13(6):2631–2644, June 2020. ISSN 1991-959X. doi: 10.5194/gmd-13-2631-2020. URL https://gmd.copernicus.org/articles/13/2631/2020/. Publisher: Copernicus GmbH. 50, 78, 80, 93

J. Baño-Medina, R. Manzanas, and J. M. Gutiérrez. On the suitability of deep convolutional neural networks for continental-wide downscaling of climate change projections. Climate Dynamics, 57(11):2941–2951, Dec. 2021. ISSN 1432-0894. doi: 10.1007/s00382-021-05847-0. URL https://doi.org/10.1007/s00382-021-05847-0. 78

L. Berthomier, B. Pradel, and L. Perez. Cloud Cover Nowcasting with Deep Learning. In 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA), pages 1–6, Nov. 2020. doi: 10.1109/IPTA50016.2020.9286606. URL http://arxiv.org/abs/2009.11577. arXiv:2009.11577 [cs]. 25, 50, 78, 93, 98

C. Besombes, O. Pannekoucke, C. Lapeyre, B. Sanderson, and O. Thual. Producing realistic climate data with generative adversarial networks. Nonlinear Processes in Geophysics, 28(3):347–370, July 2021. ISSN 1023-5809. doi: 10.5194/

npg-28-347-2021. URL https://npg.copernicus.org/articles/28/347/2021/. Publisher: Copernicus GmbH. 78

T. Beucler, S. Rasp, M. Pritchard, and P. Gentine. Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling, June 2019. URL http://arxiv.org/abs/1906.06622. arXiv:1906.06622 [physics]. 80, 82

K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian. Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast, Nov. 2022. URL http://arxiv.org/abs/2211.02556. arXiv:2211.02556 [physics]. 79, 80, 81

C. M. Bishop and H. Bishop. Transformers. In Deep Learning: Foundations and Concepts, pages 357–406. Springer International Publishing, Cham, 2024. ISBN 978-3-031-45468-4. doi: 10.1007/978-3-031-45468-4_12. URL https://doi.org/10.1007/978-3-031-45468-4_12. 126, 130

M. Blondel and V. Roulet. The Elements of Differentiable Programming, Mar. 2024. URL http://arxiv.org/abs/2403.14606. arXiv:2403.14606 [cs]. 70

M. Bode, M. Gauding, Z. Lian, D. Denker, M. Davidovic, K. Kleinheinz, J. Jitsev, and H. Pitsch. Using Physics-Informed Super-Resolution Generative Adversarial Networks for Subgrid Modeling in Turbulent Reactive Flows, Nov. 2019. URL http://arxiv.org/abs/1911.11380. arXiv:1911.11380 [physics, stat]. 66

C. Bodnar, W. P. Bruinsma, A. Lucic, M. Stanley, J. Brandstetter, P. Garvan, M. Riechert, J. Weyn, H. Dong, A. Vaughan, J. K. Gupta, K. Tambiratnam, A. Archibald, E. Heider, M. Welling, R. E. Turner, and P. Perdikaris. Aurora: A Foundation Model of the Atmosphere, May 2024. URL http://arxiv.org/abs/2405.13063. arXiv:2405.13063 [physics]. 79, 126, 130

M. Bonavita. On some limitations of data-driven weather forecasting models, Nov. 2023. URL http://arxiv.org/abs/2309.08473. arXiv:2309.08473 [physics, stat]. 126, 130

J. Boyd. Chebyshev and Fourier Spectral Methods, volume 49 of Lecture Notes in Engineering. Springer Berlin, Sept. 1989. ISBN 978-3-540-51487-9. URL https://link.springer.com/book/9783540514879. 61

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford,

I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners, July 2020. URL http://arxiv.org/abs/2005.14165. arXiv:2005.14165 [cs]. 31

R. Cang, H. Li, H. Yao, Y. Jiao, and Y. Ren. Improving Direct Physical Properties Prediction of Heterogeneous Materials from Imaging Data via Convolutional Neural Network and a Morphology-Aware Generative Model, Dec. 2017. URL http://arxiv.org/abs/1712.03811. arXiv:1712.03811 [cond-mat, physics:physics]. 66

A.-L. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. In Œuvres complètes, volume 10, pages 399–402. Gauthier-Villars, 1847. URL https://gallica.bnf.fr/ark:/12148/bpt6k90190w. 35

J. G. Charney, R. Fjörtoft, and J. v. Neumann. Numerical Integration of the Barotropic Vorticity Equation. Tellus A: Dynamic Meteorology and Oceanography, 2(4), Jan. 1950. ISSN 1600-0870. doi: 10.3402/tellusa.v2i4.8607. URL https://a.tellusjournals.se/articles/10.3402/tellusa.v2i4.8607. 17, 23

K. Chen, T. Han, J. Gong, L. Bai, F. Ling, J.-J. Luo, X. Chen, L. Ma, T. Zhang, R. Su, Y. Ci, B. Li, X. Yang, and W. Ouyang. FengWu: Pushing the Skillful Global Medium-range Weather Forecast beyond 10 Days Lead, Apr. 2023. URL http://arxiv.org/abs/2304.02948. arXiv:2304.02948 [physics]. 79, 80

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. doi: 10.48550/arXiv.1806.07366. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html. 73

S. Cheng, C. Quilodrán-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, D. Xiao, T. Janjic, W. Ding, Y. Guo, A. Carrassi, M. Bocquet, and R. Arcucci. Machine Learning With Data Assimilation and Uncertainty Quantification for Dynamical Systems: A Review. IEEE/CAA Journal of Automatica Sinica, 10(6):1361–1387, 2023. ISSN 2329-9266. doi: 10.1109/JAS.2023.123537. URL https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2023.123537. Publisher: IEEE/CAA Journal of Automatica Sinica. 100

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Sept. 2014. URL http://arxiv.org/abs/1406.1078. arXiv:1406.1078 [cs, stat]. 51

F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1800–

1807. IEEE Computer Society, July 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.195. URL https://www.computer.org/csdl/proceedings-article/cvpr/2017/0457b800/12OmNqFJhzG. ISSN: 1063-6919. 86

R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. Mathematische Annalen, 100(1):32–74, Dec. 1928. ISSN 1432-1807. doi: 10.1007/BF01448839. URL https://doi.org/10.1007/BF01448839. 63

A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, and A. Karpatne. Physics-Guided Architecture (PGA) of Neural Networks for Quantifying Uncertainty in Lake Temperature Modeling. Technical Report arXiv:1911.02682, arXiv, Nov. 2019. URL http://arxiv.org/abs/1911.02682. arXiv:1911.02682 [physics, stat] type: article. 82

A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (PGNN): An application in lake temperature modeling. In Knowledge Guided Machine Learning, volume 1, pages 353–372. Taylor & Francis, 1 edition, 2022. ISBN 978-1-00-314337-6. URL https://pubs.usgs.gov/publication/70237341. 66, 68

E. de Bézenac, A. Pajot, and P. Gallinari. Deep learning for physical processes: incorporating prior scientific knowledge. Journal of statistical mechanics, 2019(12): 124009–, 2019. ISSN 1742-5468. doi: 10.1088/1742-5468/ab3195. 69, 82

R. Dechter. Learning while searching in constraint-satisfaction-problems. In Proceedings of the Fifth AAAI National Conference on Artificial Intelligence, AAAI'86, pages 178–183, Philadelphia, Pennsylvania, Aug. 1986. AAAI Press. 33

B. Després. Neural Networks and Numerical Analysis. De Gruyter, Aug. 2022. ISBN 978-3-11-078318-6. doi: 10.1515/9783110783186. URL https://www.degruyter.com/document/doi/10.1515/9783110783186/html. 72

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. 31

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is

Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs]. 53

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. The Journal of Machine Learning Research, 12:2121–2159, 2011. doi: 10.5555/1953048.2021068. 37

F. Dupuy, Olivier Mestre, Olivier Mestre, O. Mestre, M. Serrurier, Mathieu Serrurier, M. C. Bakkay, V. K. Burdá, N. C. Cabrera-Gutiérrez, J.-C. Jouhaud, M.-A. Mader, G. Oller, and M. Zamo. ARPEGE Cloud Cover Forecast Postprocessing with Convolutional Neural Network. Weather and forecasting, 36(2):567–586, 2020. doi: 10.1175/waf-d-20-0093.1. 25

Y. El Mghouchi, E. Chham, E. M. Zemmouri, and A. El Bouardi. Assessment of different combinations of meteorological parameters for predicting daily global solar radiation using artificial neural networks. Building and Environment, 149:607–622, Feb. 2019. ISSN 0360-1323. doi: 10.1016/j.buildenv.2018.12.055. URL https://www.sciencedirect.com/science/article/pii/S0360132318308138. 78

R. El Montassir, O. Pannekoucke, and C. Lapeyre. HyPhAICC v1.0: a hybrid physics–AI approach for probability fields advection shown through an application to cloud cover nowcasting. Geoscientific Model Development, 17(17):6657–6681, Sept. 2024. ISSN 1991-959X. doi: 10.5194/gmd-17-6657-2024. URL https://gmd.copernicus.org/articles/17/6657/2024/. Publisher: Copernicus GmbH. 126, 130

L. Espeholt, S. Agrawal, C. Sønderby, M. Kumar, J. Heek, C. Bromberg, C. Gazen, R. Carver, M. Andrychowicz, J. Hickey, A. Bell, and N. Kalchbrenner. Deep learning for twelve hour precipitation forecasts. Nature Communications, 13(1):5145, Sept. 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-32483-x. URL https://www.nature.com/articles/s41467-022-32483-x. Number: 1 Publisher: Nature Publishing Group. 78

R. Fablet, S. Ouala, and C. Herzet. Bilinear residual Neural Network for the identification and forecasting of dynamical systems. In 2018 26th European Signal Processing Conference (EUSIPCO), pages 1477–1481. IEEE, Sept. 2018. doi: 10.23919/EUSIPCO.2018.8553492. URL https://hal.science/hal-01686766. 73

B. Fernandes, A. González-Briones, P. Novais, M. Calafate, C. Analide, and J. Neves. An Adjective Selection Personality Assessment Method Using Gradient Boosting Machine Learning. Processes, 8(5):618, May 2020. ISSN 2227-9717. doi: 10.3390/pr8050618. URL https://www.mdpi.com/2227-9717/8/5/618. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. 95

U. Forssell and P. Lindskog. Combining Semi-Physical and Neural Network Modeling: An Example ofIts Usefulness. IFAC Proceedings Volumes, 30(11):767–770, July 1997. ISSN 1474-6670. doi: 10.1016/S1474-6670(17)42938-7. URL https://www.sciencedirect.com/science/article/pii/S1474667017429387. 9, 67, 68

R. Frostig, M. J. Johnson, and C. Leary. Compiling machine learning programs via high-level tracing. Systems for Machine Learning, 4(9), 2018. URL https://mlsys.org/Conferences/doc/2018/146.pdf. Publisher: SysML. 41

J. García-Pereda, J. M. Fernandez-Serdan, O. Alonso, A. Sanz, R. Guerra, C. Ariza, I. Santos, and L. Fernández. NWCSAF High Resolution Winds (NWC/GEO-HRW) Stand-Alone Software for Calculation of Atmospheric Motion Vectors and Trajectories. Remote Sensing, 11(17):2032, Jan. 2019. ISSN 2072-4292. doi: 10.3390/rs11172032. URL https://www.mdpi.com/2072-4292/11/17/2032. Number: 17 Publisher: Multidisciplinary Digital Publishing Institute. 84, 93

G. K. Gilbert. Finley's tornado predictions. American Meteorological Journal, 1:166–172, Sept. 1884. 94

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks, June 2014. URL http://arxiv.org/abs/1406.2661. arXiv:1406.2661 [cs, stat]. 31

J. M. Han, Y. Q. Ang, A. Malkawi, and H. W. Samuelson. Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements. Building and Environment, 192:107601, Apr. 2021. ISSN 0360-1323. doi: 10.1016/j.buildenv.2021.107601. URL https://www.sciencedirect.com/science/article/pii/S0360132321000160. 78

H. A. Hazen. The Origin and Value of Weather Lore. The Journal of American Folklore, 13(50):191–198, 1900. ISSN 0021-8715. doi: 10.2307/533883. URL https://www.jstor.org/stable/533883. Publisher: University of Illinois Press. 16, 22

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. URL https://ieeexplore.ieee.org/document/7780459. ISSN: 1063-6919. 52

M. Herde, B. Raonić, T. Rohner, R. Käppeli, R. Molinaro, E. de Bézenac, and S. Mishra. Poseidon: Efficient Foundation Models for PDEs, May 2024. URL http://arxiv.org/abs/2405.19101. arXiv:2405.19101 [cs]. 127, 131

C. Hirsch. Numerical computation of internal and external flows. Volume 2 : Computational methods for inviscid and viscous flows. John Wiley & Sons, Chichester, 1990. ISBN 978-0-471-92452-4. 57

C. Hirsch. Numerical computation of internal and external flows: fundamentals of computational fluid dynamics. Elsevier/Butterworth-Heinemann, Oxford ; Burlington, MA, 2nd ed edition, 2007. ISBN 978-0-7506-6594-0. OCLC: ocn148277909. 61

C. W. Hirt. Heuristic stability theory for finite-difference equations. Journal of Computational Physics, 2(4):339–355, June 1968. ISSN 0021-9991. doi: 10.1016/0021-9991(68)90041-7. URL https://www.sciencedirect.com/science/article/pii/0021999168900417. 61

S. Hochreiter and J. Schmidhuber. Long Short-term Memory. Neural computation, 9: 1735–80, Dec. 1997. doi: 10.1162/neco.1997.9.8.1735. 30, 51

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, Jan. 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208. 42

X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles. In Proceedings of the 2019 SIAM International Conference on Data Mining (SDM), Proceedings, pages 558–566. Society for Industrial and Applied Mathematics, May 2019. doi: 10.1137/1.9781611975673.63. URL https://epubs.siam.org/doi/10.1137/1.9781611975673.63. 66, 67, 82

X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar. Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles. ACM/IMS Transactions on Data Science, 2(3):20:1–20:26, May 2021. ISSN 2691-1922. doi: 10.1145/3447814. URL https://dl.acm.org/doi/10.1145/3447814. 67, 82

P. Joe, J. Sun, N. Yussouf, S. Goodman, M. Riemer, K. C. Gouda, B. Golding, R. Rogers, G. Isaac, J. Wilson, P. W. P. Li, V. Wulfmeyer, K. Elmore, J. Onvlee, P. Chong, and J. Ladue. Predicting the Weather: A Partnership of Observation Scientists and Forecasters. In B. Golding, editor, Towards the "Perfect" Weather Warning: Bridging Disciplinary Gaps through Partnership and Communication, pages 201–254. Springer International Publishing, Cham, 2022. ISBN 978-3-030-98989-7. doi: 10.1007/

978-3-030-98989-7_7. URL https://doi.org/10.1007/978-3-030-98989-7_7. 17, 23

D. Karimi and S. E. Salcudean. Reducing the Hausdorff Distance in Medical Image Segmentation With Convolutional Neural Networks. IEEE Trans. Medical Imaging, 39(2):499–513, 2020. doi: 10.1109/TMI.2019.2930068. 104

A. Karpatne, G. Atluri, J. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar. Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data. IEEE Transactions on Knowledge and Data Engineering, 29 (10):2318–2331, Oct. 2017. ISSN 1041-4347. doi: 10.1109/TKDE.2017.2720168. URL http://arxiv.org/abs/1612.08544. arXiv:1612.08544 [cs, stat]. 65, 66

P. Kidger and T. Lyons. Universal Approximation with Deep Narrow Networks. In Proceedings of Thirty Third Conference on Learning Theory, pages 2306–2327. PMLR, July 2020. URL https://proceedings.mlr.press/v125/kidger20a.html. ISSN: 2640-3498. 42, 43

B.-Y. Kim, J. W. Cha, and Y. H. Lee. Estimation of twenty-four-hour continuous cloud cover using ground-based imager with convolutional neural network. Atmospheric Measurement Techniques Discussions, pages 1–19, July 2023. doi: 10.5194/amt-2023-131. URL https://amt.copernicus.org/preprints/amt-2023-131/. Publisher: Copernicus GmbH. 78

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980. 38, 93

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html. 30

R. J. Kuligowski and A. P. Barros. Localized Precipitation Forecasts from a Numerical Weather Prediction Model Using Artificial Neural Networks. Weather and Forecasting, 13(4):1194–1204, Dec. 1998. ISSN 1520-0434, 0882-8156. doi: 10.1175/1520-0434(1998)013<1194:LPFFAN>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/wefo/13/4/1520-0434_1998_013_1194_lpffan_2_0_co_2.xml. Publisher: American Meteorological Society Section: Weather and Forecasting. 78

W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Zeitschrift für Mathematik und Physik, 46:435–453, 1901. 58

R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, S. Ravuri, T. Ewalds, F. Alet, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, J. Stott, O. Vinyals, S. Mohamed, and P. Battaglia. GraphCast: Learning skillful medium-range global weather forecasting, Dec. 2022. URL http://arxiv.org/abs/2212.12794. arXiv:2212.12794 [physics]. 79, 80

S. Lang, M. Alexe, M. Chantry, J. Dramsch, F. Pinault, B. Raoult, M. C. A. Clare, C. Lessig, M. Maier-Gerber, L. Magnusson, Z. B. Bouallègue, A. P. Nemesio, P. D. Dueben, A. Brown, F. Pappenberger, and F. Rabier. AIFS – ECMWF's data-driven forecasting system, Aug. 2024. URL http://arxiv.org/abs/2406.01465. arXiv:2406.01465 [physics]. 79

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539. Number: 7553 Publisher: Nature Publishing Group. 30

G. W. F. v. Leibniz. The Early Mathematical Manuscripts of Leibniz: Translated from the Latin Texts Published by Carl Immanuel Gerhardt with Critical and Historical Notes. Open court publishing Company, 1920. ISBN 978-0-598-81846-1. Google-Books-ID: bOIGAAAAYAAJ. 40

X.-D. Liu, S. Osher, and T. Chan. Weighted Essentially Non-oscillatory Schemes. Journal of Computational Physics, 115(1):200–212, Nov. 1994. ISSN 0021-9991. doi: 10.1006/jcph.1994.1187. URL https://www.sciencedirect.com/science/article/pii/S0021999184711879. 57

Y. Liu, Y. Gao, and W. Yin. An Improved Analysis of Stochastic Gradient Descent with Momentum. In Advances in Neural Information Processing Systems, volume 33, pages 18261–18271. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d3f5d4de09ea19461dab00590df91e4f-Abstract.html. 37

E. N. Lorenz. Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences, 20(2):130–141, Mar. 1963. ISSN 0022-4928, 1520-0469. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml. Publisher: American Meteorological Society Section: Journal of the Atmospheric Sciences. 77

D. Matte, J. H. Christensen, H. Feddersen, H. Vedel, N. W. Nielsen, R. A. Pedersen, and R. M. K. Zeitzen. On the Potentials and Limitations of Attributing a Small-Scale Climate Event. Geophysical Research Letters, 49(16): e2022GL099481, 2022. ISSN 1944-8007. doi: 10.1029/2022GL099481. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2022GL099481. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2022GL099481. 17, 23

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, Dec. 1943. ISSN 1522-9602. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259. 29

M. Minsky and S. Papert. Perceptrons; an Introduction to Computational Geometry. MIT Press, 1969. ISBN 978-0-262-13043-1. Google-Books-ID: Ow1OAQAAIAAJ. 30

M. Nakahara. Geometry, Topology and Physics (Second Edition). Taylor & Francis, 2003. ISBN 0-7503-0606-8. 138

Y. E. Nesterov. A method of solving a convex programming problem with convergence rate O(1/k^2). Proceedings of the USSR Academy of Sciences, 269(3):543–547, 1983. 37

T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover. ClimaX: A foundation model for weather and climate, Feb. 2023. URL http://arxiv.org/abs/2301.10343. arXiv:2301.10343 [cs]. 54, 79, 126, 130

M. Ossendrijver. Weather Prediction in Babylonia. Journal of Ancient Near Eastern History, 8(1-2):223–258, June 2021. ISSN 2328-9562. doi: 10.1515/janeh-2020-0009. URL https://www.degruyter.com/document/doi/10.1515/janeh-2020-0009/html. Publisher: De Gruyter. 16, 22

S. J. Pan and Q. Yang. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, Oct. 2010. ISSN 1558-2191. doi: 10.1109/TKDE.2009.191. URL https://ieeexplore.ieee.org/document/5288526. Conference Name: IEEE Transactions on Knowledge and Data Engineering. 31

O. Pannekoucke and R. Fablet. PDE-NetGen 1.0: from symbolic partial differential equation (PDE) representations of physical processes to trainable neural network representations. Geoscientific Model Development, 13(7):3373–3382, July 2020. ISSN 1991-959X. doi: 10.5194/gmd-13-3373-2020. URL https://gmd.copernicus.org/articles/13/3373/2020/. Publisher: Copernicus GmbH. 74, 127, 131

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on International Conference

on Machine Learning, 28:1310–1318, 2013. URL https://proceedings.mlr.press/v28/pascanu13.pdf. 30

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: an imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 8026–8037, Red Hook, NY, USA, Dec. 2019. Curran Associates Inc. 41, 71

A. Pinkus. Approximation theory of the MLP model in neural networks. Acta Numerica, 8:143–195, Jan. 1999. ISSN 1474-0508, 0962-4929. doi: 10.1017/S0962492900002919. URL https://www.cambridge.org/core/journals/acta-numerica/article/abs/approximation-theory-of-the-mlp-model-in-neural-networks/18072C558C8410C4F92A82BCC8FC8CF9. Publisher: Cambridge University Press. 42, 43

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5):1–17, Jan. 1964. ISSN 0041-5553. doi: 10.1016/0041-5553(64)90137-5. URL https://www.sciencedirect.com/science/article/pii/0041555364901375. 36

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, Nov. 2017. URL http://arxiv.org/abs/1711.10561. arXiv:1711.10561 [cs, math, stat]. 46

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378: 686–707, Feb. 2019a. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125. 66, 127, 131

M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis. Deep Learning of Vortex Induced Vibrations. Journal of Fluid Mechanics, 861:119–137, Feb. 2019b. ISSN 0022-1120, 1469-7645. doi: 10.1017/jfm.2018.872. URL http://arxiv.org/abs/1808.08952. arXiv:1808.08952 [physics, stat]. 66

S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, R. Prudden, A. Mandhane,

A. Clark, A. Brock, K. Simonyan, R. Hadsell, N. Robinson, E. Clancy, A. Arribas, and S. Mohamed. Skilful precipitation nowcasting using deep generative models of radar. Nature, 597(7878):672–677, Sept. 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03854-z. URL https://www.nature.com/articles/s41586-021-03854-z. Number: 7878 Publisher: Nature Publishing Group. 78, 80, 81

M. Reyniers. Quantitative Precipitation Forecasts based on radar observations: principles, algorithms and operational systems. RMI Publication, Mar. 2016. URL https://orfeo.belnet.be/handle/internal/8780. Accepted: 2016-03-07T16:16:58Z Publisher: IRM. 19, 25

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, volume 9351, pages 234–241. Springer International Publishing, Cham, 2015. ISBN 978-3-319-24573-7 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28. URL http://link.springer.com/10.1007/978-3-319-24574-4_28. Series Title: Lecture Notes in Computer Science. 50, 93

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386–408, 1958. ISSN 1939-1471. doi: 10.1037/h0042519. Place: US Publisher: American Psychological Association. 30, 31

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323(6088):533–536, Oct. 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0. Number: 6088 Publisher: Nature Publishing Group. 30, 39

C. Runge. Ueber die numerische Auflösung von Differentialgleichungen. Mathematische Annalen, 46(2):167–178, June 1895. ISSN 1432-1807. doi: 10.1007/BF01446807. URL https://doi.org/10.1007/BF01446807. 58

L. Ruthotto and E. Haber. Deep Neural Networks Motivated by Partial Differential Equations. Journal of Mathematical Imaging and Vision, 62(3):352–364, Apr. 2020. ISSN 1573-7683. doi: 10.1007/s10851-019-00903-1. URL https://doi.org/10.1007/s10851-019-00903-1. 73

O. San and R. Maulik. Machine learning closures for model order reduction of thermal fluids. Applied Mathematical Modelling, 60:681–710, Aug. 2018a. ISSN

0307904X. doi: 10.1016/j.apm.2018.03.037. URL https://linkinghub.elsevier.com/retrieve/pii/S0307904X18301616. 67

O. San and R. Maulik. Neural network closures for nonlinear model order reduction. Advances in Computational Mathematics, 44(6):1717–1750, Dec. 2018b. ISSN 1572-9044. doi: 10.1007/s10444-018-9590-z. URL https://doi.org/10.1007/s10444-018-9590-z. 67

M. G. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. H. Leufen, A. Mozaffari, and S. Stadtler. Can deep learning beat numerical weather prediction? Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, 379(2194):20200097, Apr. 2021. ISSN 1364-503X. doi: 10.1098/rsta.2020.0097. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7898133/. 17, 23

A. M. Schweidtmann, D. Zhang, and M. von Stosch. A review and perspective on hybrid modeling methodologies. Digital Chemical Engineering, 10:100136, Mar. 2024. ISSN 2772-5081. doi: 10.1016/j.dche.2023.100136. URL https://www.sciencedirect.com/science/article/pii/S2772508123000546. 100

Y. Seity, P. Brousseau, S. Malardel, G. Hello, P. Bénard, F. Bouttier, C. Lac, and V. Masson. The AROME-France Convective-Scale Operational Model. Monthly Weather Review, 139(3):976–991, Mar. 2011. ISSN 1520-0493, 0027-0644. doi: 10.1175/2010MWR3425.1. URL https://journals.ametsoc.org/view/journals/mwre/139/3/2010mwr3425.1.xml. Publisher: American Meteorological Society Section: Monthly Weather Review. 77

T. Selz and G. C. Craig. Can Artificial Intelligence-Based Weather Prediction Models Simulate the Butterfly Effect? Geophysical Research Letters, 50 (20):e2023GL105747, 2023. ISSN 1944-8007. doi: 10.1029/2023GL105747. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2023GL105747. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2023GL105747. 81

S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In M. Hutter and R. Siegwart, editors, Field and Service Robotics, volume 5, pages 621–635. Springer International Publishing, Cham, 2018. ISBN 978-3-319-67360-8 978-3-319-67361-5. doi: 10.1007/978-3-319-67361-5_40. URL http://link.springer.com/10.1007/978-3-319-67361-5_40. Series Title: Springer Proceedings in Advanced Robotics. 67

X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html. 17, 23, 78

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. Nature, 550(7676):354–359, Oct. 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270. Number: 7676 Publisher: Nature Publishing Group. 31

M. M. Sultan, H. K. Wayment-Steele, and V. S. Pande. Transferable Neural Networks for Enhanced Sampling of Protein Dynamics. Journal of Chemical Theory and Computation, 14(4):1887–1894, Apr. 2018. ISSN 1549-9618. doi: 10.1021/acs.jctc.8b00025. URL https://doi.org/10.1021/acs.jctc.8b00025. Publisher: American Chemical Society. 67

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning, pages 1139–1147. PMLR, May 2013. URL https://proceedings.mlr.press/v28/sutskever13.html. ISSN: 1938-7228. 37

C. K. Sønderby, L. Espeholt, J. Heek, M. Dehghani, A. Oliver, T. Salimans, S. Agrawal, J. Hickey, and N. Kalchbrenner. MetNet: A Neural Weather Model for Precipitation Forecasting, Mar. 2020. URL http://arxiv.org/abs/2003.12140. arXiv:2003.12140 [physics, stat]. 81

P. N. Tamvakis, C. Kiourt, A. D. Solomou, G. Ioannakis, and N. C. Tsirliganis. Semantic Image Segmentation with Deep Learning for Vine Leaf Phenotyping. IFAC-PapersOnLine, 55(32):83–88, Jan. 2022. ISSN 2405-8963. doi: 10.1016/j.ifacol.2022.11.119. URL https://www.sciencedirect.com/science/article/pii/S2405896322027525. 86

L. Torrey and J. Shavlik. Transfer Learning: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. In E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J. Serrano López, editors, Handbook of Research on Machine Learning Applications and Trends, pages 242–264. IGI Global, 2010. ISBN 978-1-60566-766-9 978-1-60566-767-6. doi: 10.4018/978-1-60566-766-9.ch011.

URL http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-766-9.ch011. 67

Q.-K. Tran and S.-k. Song. Computer Vision in Precipitation Nowcasting: Applying Image Quality Assessment Metrics for Training Deep Neural Networks. Atmosphere, 10(5):244, May 2019. ISSN 2073-4433. doi: 10.3390/atmos10050244. URL https://www.mdpi.com/2073-4433/10/5/244. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. 80

K. Trebing, T. Stanczyk, and S. Mehrkanoon. SmaAt-UNet: Precipitation nowcasting using a small attention-UNet architecture. Pattern Recognition Letters, 145:178–186, May 2021. ISSN 0167-8655. doi: 10.1016/j.patrec.2021.01.036. URL https://www.sciencedirect.com/science/article/pii/S0167865521000556. 78, 93

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 9, 31, 52, 55

P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. Data-Driven Forecasting of High-Dimensional Chaotic Systems with Long Short-Term Memory Networks. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474(2213):20170844, May 2018. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2017.0844. URL http://arxiv.org/abs/1802.07486. arXiv:1802.07486 [nlin, physics:physics]. 69

Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. PloS One, 13(5):e0197704, 2018. ISSN 1932-6203. doi: 10.1371/journal.pone.0197704. 67

S.-H. Wang, D. R. Nayak, D. S. Guttery, X. Zhang, and Y.-D. Zhang. COVID-19 classification by CCSHNet with deep fusion using transfer learning and discriminant correlation analysis. Information Fusion, 68:131–148, Apr. 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2020.11.005. URL https://www.sciencedirect.com/science/article/pii/S1566253520304073. 95

J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems. ACM Comput. Surv., 55(4):66:1–66:37, Nov. 2022. ISSN 0360-0300. doi: 10.1145/3514228. URL https://dl.acm.org/doi/10.1145/3514228. 65

J. W. Wilson, N. A. Crook, C. K. Mueller, J. Sun, and M. Dixon. Nowcasting Thunderstorms: A Status Report. Bulletin of the American Meteorological Society, 79(10):2079–2100, Oct. 1998. ISSN 0003-0007, 1520-0477. doi: 10.1175/1520-0477(1998)079<2079:NTASR>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/bams/79/10/1520-0477_1998_079_2079_ntasr_2_0_co_2.xml. Publisher: American Meteorological Society Section: Bulletin of the American Meteorological Society. 19, 25

J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, Prabhat, and H. Xiao. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. Journal of Computational Physics, 406:109209, Apr. 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.109209. URL https://www.sciencedirect.com/science/article/pii/S0021999119309143. 66

R. Yang, J. Hu, Z. Li, J. Mu, T. Yu, J. Xia, X. Li, A. Dasgupta, and H. Xiong. Interpretable Machine Learning for Weather and Climate Prediction: A Survey, Mar. 2024. URL http://arxiv.org/abs/2403.18864. arXiv:2403.18864 [physics]. 81

Z. Yang, J.-L. Wu, and H. Xiao. Enforcing Deterministic Constraints on Generative Adversarial Networks for Emulating Physical Systems, Nov. 2020. URL http://arxiv.org/abs/1911.06671. arXiv:1911.06671 [physics, stat]. 66

D. Yarotsky. Error bounds for approximations with deep ReLU networks. Neural Networks, 94:103–114, Oct. 2017. ISSN 0893-6080. doi: 10.1016/j.neunet.2017.07.002. URL https://www.sciencedirect.com/science/article/pii/S0893608017301545. 42

B. Øksendal. Stochastic Differential Equations: An Introduction with Applications. Springer Science & Business Media, 6 edition, Nov. 2010. ISBN 978-3-642-14394-6. URL https://link.springer.com/book/10.1007/978-3-642-14394-6. 117, 118, 119

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Titre : Approche hybride basée sur la physique et l'IA pour l'advection des champs de probabilités. Application à la prévision immédiate de la couverture nuageuse

Mots clés : Apprentissage profond, Modélisation hybride, Couverture nuageuse, Advection de probabilités, Apprentissage machine informé par la physique

Résumé : Au cours des dernières décennies, le réchauffement climatique s'est accéléré, tout comme la fréquence des événements météorologiques extrêmes, affectant considérablement les sociétés et les économies. Ces événements soulignent le besoin croissant de prévisions météorologiques précises. Les modèles traditionnels de prévision numérique du temps, bien qu'efficaces, restent coûteux en termes de calcul et peinent à prédire les phénomènes à petite échelle tels que les orages. Parallèlement, les modèles d'apprentissage profond se sont révélés prometteurs dans les prévisions météorologiques, mais manquent souvent de cohérence physique et de capacités de généralisation.
Cette thèse aborde les limites des méthodes traditionnelles d'apprentissage profond dans la production de résultats réalistes et physiquement cohérents qui peuvent se généraliser à des données non vues. Dans cette thèse, nous explorons des méthodes hybrides qui cherchent à concilier la précision des méthodes de premier principe avec la puissance d'exploitation des données des techniques d'apprentissage, avec une application à la prévision immédiate de la couverture nuageuse. Les données de couverture nuageuse utilisées sont des images satellites avec classification des types de nuages, et l'objectif est de prédire la position de la couverture nuageuse au cours des deux prochaines heures tout en préservant la classification des types de nuages.
L'approche proposée, nommée HyPhAICC, impose un comportement physique basé sur l'advection probabiliste. Dans le premier modèle, dénommé HyPhAICC-1, des dynamiques d'advection multi-niveaux sont utilisées pour guider l'apprentissage d'un modèle U-Net. Cela est réalisé en résolvant l'équation d'advection pour plusieurs champs de probabilité, chacun correspondant à un type de nuage différent, tout en apprenant simultanément le champ de vitesse inconnu.
Nos expériences montrent que la formulation hybride surpasse non seulement le modèle d'imagerie extrapolée d'EUMETSAT (EXIM), mais également le modèle U-Net en termes de métriques standard telles que le score F1, l'indice de succès critique (CSI) et l'accuracy. Nous démontrons également que le modèle HyPhAICC-1 préserve plus de détails et produit des résultats plus réalistes par rapport au modèle U-Net. Pour mesurer quantitativement cet aspect, nous utilisons une version modifiée de la distance de Hausdorf qui est, à notre connaissance, la première fois que cette métrique est utilisée à cette fin dans la littérature. Cette première version montre aussi une convergence remarquablement rapide. Elle a également affiché de meilleures performances par rapport au U-Net lorsqu'elle a été entraînée sur des ensembles de données plus petits, soulignant l'efficacité computationnelle de l'approche proposée.
Un autre modèle, dénommé HyPhAICC-2, ajoute un terme source à l'équation d'advection. Bien que cela ait dégradé le rendu visuel, il a affiché les meilleures performances en termes d'accuracy. Ces résultats suggèrent que l'architecture hybride physique-IA proposée constitue une solution prometteuse pour surmonter les limitations des méthodes d'IA traditionnelles. Cela pourrait motiver des recherches supplémentaires pour combiner les connaissances physiques avec les modèles d'apprentissage profond afin d'améliorer la précision et l'efficacité des prévisions météorologiques.

Title: Hybrid physics- and AI-based approach to probability field advection. Application to cloud cover nowcasting

Key words: Deep Learning, Hybrid modelling, Cloud cover, Probability advection, Physics-informed machine learning

Abstract: During the last decades, as the global warming has accelerated, so has the frequency of extreme weather events, significantly affecting societies and the economies. These events highlight the growing need for accurate weather forecasting. Traditional numerical weather prediction models, while effective, remain computationally expensive and struggle to predict small-scale phenomena such as thunderstorms. Meanwhile, deep learning models have shown promise in weather forecasting but often lack physical consistency and generalisation capabilities.
This thesis addresses the limitations of traditional deep learning methods in producing realistic and physically consistent results that can generalise to unseen data. In this thesis, we explore hybrid methods that seek to reconcile the accuracy of first-principle methods with the data-leveraging power of learning techniques, with an application to cloud cover nowcasting. The cloud cover data used are satellite images with cloud type classification, and the goal is to predict the cloud cover position over the next two hours while preserving the classification of the cloud types.
The proposed approach, named HyPhAICC, enforces physical behaviour based on probability advection. In the first model, denoted HyPhAICC-1, multi-level advection dynamics are used to guide the learning of a U-Net model. This is achieved by solving the advection equation for multiple probability fields, each corresponding to a different cloud type, while simultaneously learning the unknown velocity field.
Our experiments show that the hybrid formulation outperforms not only the EUMETSAT Extrapolated Imagery model (EXIM) but also the U-Net model in terms of standard metrics such as F1 score, Critical Success Index (CSI), and accuracy. We also demonstrate that the HyPhAICC-1 model preserves more details and produces more realistic results compared to the U-Net model. To quantitatively measure this aspect, we use a modified version of the Hausdorff distance which is, to the best of our knowledge, the first time this metric is used for this purpose in the literature. This first version shows also a significant faster convergence. It also performed significantly better compared to the U-Net when trained on smaller datasets, highlighting the computational efficiency of the proposed approach.
Another model, denoted HyPhAICC-2, adds a source term to the advection equation. While this impaired the visual rendering, it displayed the best performance in terms of accuracy.
These results suggest that the proposed hybrid Physics-AI architecture provides a promising solution to overcome the limitations of traditional AI methods. This could motivate further research to combine physical knowledge with deep learning models for more accurate and efficient weather forecasting.