SURROGATE CFD MODELLING

Modelling wind flow velocity over coastal dune terrain using convolutional neural networks

4 JULI 2024

UTRECHT UNIVERSITY
Thijs Modderman 1321072

Table of Contents

1. Abstract

Dunes play a crucial role in protecting coastal areas from flooding, erosion and supporting their fragile ecosystems. Coastal dune management is important to protect these coastal areas. Dunes are formed by various factors, among which the wind magnitude and direction are critical. Traditionally, Computational Fluid Dynamics (CFD) methods are used to model the wind flow over coastal dune terrains. However, these methods are computationally expensive, which limits their application for large and complex aeolian transfer models. This research proposes the implementation of Convolutional Neural Networks (CNNs) for CFD surrogate modelling to predict the wind velocity vectors over coastal dune terrain. This approach aims to reduce the computational cost while trading off some accuracy. Various CNN architectures and backbones are evaluated. The research found that the combination of the Feature Pyramid Network (FPN) architecture and densenet121 backbone provided the best performance, significantly reducing the prediction time compared to traditional CFD simulations. While the model shows some consistent errors in certain upwind and downwind regions, the results show the potential of CNN surrogate modelling to enhance coastal management by offering a faster alternative to CFD simulations. Further research should focus on expanding the dataset to assess the model's generalizability and on exploring backbones and ensemble methods to further improve the model's robustness and accuracy.

2. Introduction

Dunes play a crucial role in coastal areas, acting as natural flood and coastal erosion defences and providing essential ecosystem services (Husemann, Romão, Lima, Costas, & Coelho, 2024). Coastal areas are important for both people and ecosystems. Although they occupy less than 15% of Earth's land area (European Environment Agency, 2008), 40% of the population lives within 100 kilometres of the coast (Maul & Duedall, 2019) and 5% of the population of coastal cities are currently at risk of flooding, which is expected to double by the end of the century (United Nations Development Programme, 2023). It is important to manage dunes to protect coastal areas, as effective dune management can prevent floods, mitigate coastal erosion, and safeguard fragile ecosystems that require significant time to recover if destroyed. Coastal dunes are accumulated by aeolian sediment transportation and are located along the beach. They are shaped by various factors, namely sand supply, amount and granularity of the sand, dune topography, vegetation, and especially the properties of wind flow, such as its magnitude and direction (NSW department of Land and Water Conservation, 2001). These wind flow properties are critical as they determine the transportation and deposition of sand, impacting the shape and size of the dunes.

Traditionally, wind flow has been simulated using Computational Fluid Dynamics (CFD) models, which utilize numerical methods to compute fluid flow. These methods formulate governing equations that describe the fluid flow, including the continuity equation (conservation of mass) and the Navier-Stokes equation (conservation of momentum) (simscale, 2023). To solve these equations within a defined grid, they need to be discretized using techniques such as the Finite Difference Method (Ashgriz & Mostaghimi, 2002) and the Finite Volume Method (Moukalled, Mangani, & Darwish, 2016). While CFD methods have been successfully implemented to simulate fluid mechanics problems, including wind simulation around dunes (Hesp & Smyth, 2021; Smyth 2016), they have one main drawback. This main issue is their high computational cost, resulting in long simulation times, especially for large and complex systems (quadco engineering, sd).

To overcome the drawback mentioned above, research has been performed regarding surrogate Convolutional Neural Network (CNN) models. Some studies have trained CNN surrogate models for pixelwise regression to increase computational efficiency (Baumann, Roßberg, & Schmitt, 2023). Convolutional Neural Networks (CNNs) are a class of deep learning models which have demonstrated exceptional performance in learning patterns from spatial dependent data. Suitable for different fields of computer vision and natural language processing (Ghosh, Sufian, Sultana, Chakrabarti, & De, 2020), CNNs have shown the ability to learn high-level features from spatial dependencies that are informative for supervised tasks (Lecun, Bottou, Bengio, & Haffner, 1998). This is because of the multilayer architecture of CNNs, which allows for the encoding of image-specific features making them more suited for image-focused tasks than typical Artificial Neural Networks (O'Shea & Nash, 2015). There are multiple different CNN architectures and backbones which impact learning rate, computational power, model performance and generalizability (Alzubaidi, et al., 2021).

CNNs have been well researched in segmentation tasks (Bizopoulos, Vretos, & Daras, 2020), however they have been implemented in a variety of applications like pixelwise regression for medical image estimation (Wang, Mattie, Berger, & Levman, 2021) or uncertainty estimation in Machine Learning (Baumann, Roßberg, & Schmitt, 2023), showing their applicability across domains. CNNs have recently been applied to physical simulations and surrogate modelling, one study successfully predicts the velocity field around randomly shaped obstacles in a 2D space, achieving significant speedups with minimal error rates compared to traditional CFD simulations (Ribeiro, Ahmed, Dengel , & Rehman, 2020). Another study successfully implemented a CNN surrogate model for modelling the flow and geomorphic heterogeneity induced by vegetation (Chen, Luo, Li, & Zhang, 2024). CNN

surrogate models also showed efficient estimation of modelling the steady laminar wind flow around alternative geometric representations (Guo, Li, & Iorio, 2016).

The studies mentioned above (Baumann, Roßberg, & Schmitt, 2023; Chen, Luo, Li, & Zhang, 2024; Guo, Li, & Iorio, 2016; Ribeiro, Ahmed, Dengel, & Rehman, 2020; Wang, Mattie, Berger, & Levman, 2021) all utilized the same type of CNN architecture for their surrogate model, namely the U-net architecture with pixelwise regression, with some implementing an output of 3 channels. However, all were modelled within a 2D space, e.g. a side representation of a car (Guo, Li, & Iorio, 2016). The U-net architecture seems to be a popular architecture type for CFD surrogate models, but is not the only architecture to have been implemented in surrogate CNN modelling. In one study, a shared encoder and separated decoder architecture was implemented which effectively captured the geometric features and fluid dynamics and presented acceptable results (Bhatnagar, Afshar, Pan, Duraisamy, & Kaushik, 2019), showing CFD surrogate modelling can be achieved with different types of CNN architectures. However, to the best knowledge of the author, no studies have been found that compare different possible architecture types or backbones for solving a regression task for CFD. This is only the case for different segmentation tasks (Comprehensive Comparison of Deep Learning Models, 2020). Additionally, no studies have been found that implement CNN surrogate models for modelling wind flow in coastal dunes.

A long-term goal of the geoscience department of the University of Utrecht is to model the aeolian transport over time, to be able to achieve this goal, the wind flow should be computed. Computing the wind flow using traditional CFD methods requires too much computing power, which is why the main objective of this research is to serve as a proof of concept for using a CNN approach as a surrogate model for CFD simulations of wind velocities on coastal dune terrains. The main research question of this research is: Which CNN model architecture and backbone most accurately models the wind velocities over coastal dune terrain with varying wind directions compared to a traditional CFD approach? The following sub-questions assist in answering the main research question.

- Model architecture
    - To what extent do the different CNN architectures and backbones impact model performance?
- Model performance
    - How accurately does the model predict the wind velocity at various properties of the combined topography and wind direction within the coastal terrain?
    - How does the computational cost of the model compare to traditional CFD simulations?

3. Data

In this chapter, the generated wind velocity data and the model input data will be discussed. The velocity data is generated using CFD simulation on OpenFoam open-source software (OpenCFD Ltd., 2019). While the specific parameters used for these CFD simulations are out of scope for this research, it is important to note that they remain constant for each simulation, except for the inlet wind direction, which represents the angle of the incoming wind relative to the west.

The input data consists of the spatial coordinates of the coastal dunes and the cosine of the difference between the inlet wind direction and the surface normal vector. The latter parameter is visually explained, on top of the 2D mapped y-spatial coordinates, in Figure 1. This figure shows that the angle difference, theta, between the wind inlet angle and the bed surface normal vector in the x direction is computed, the parameter is the cosine of this angle difference. The interpretation of the parameter is the degree to which the specific grid point is downwind, this parameter will from now on be called the cosine difference.

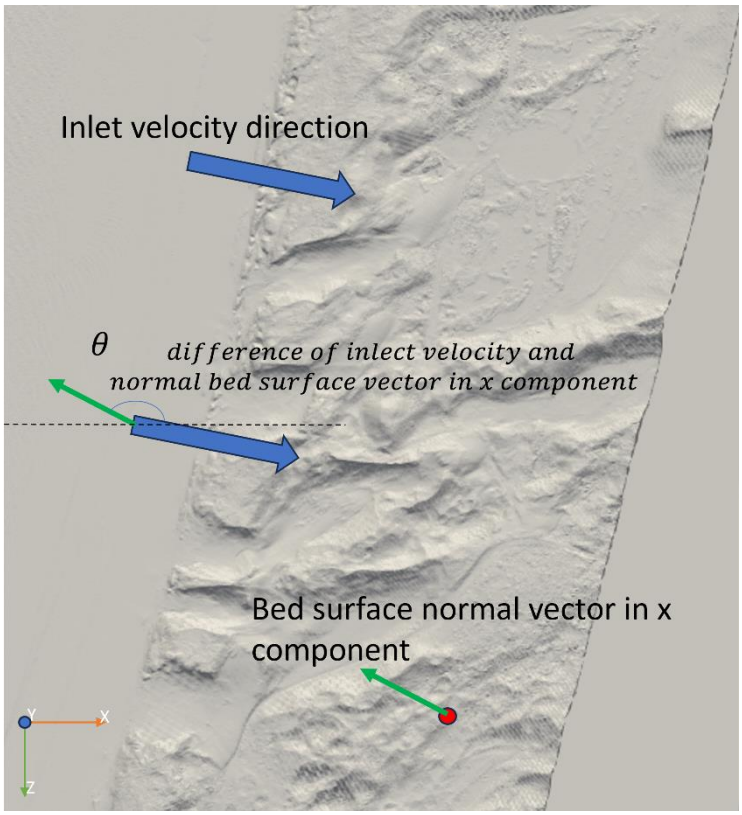


Figure 1: Visual explanation of the cosine difference between inlet angle and surface normal vector on top of the dune topography

The output data consists of three parameters called U_0, U_1 and U_2, these parameters represent the wind velocity in the x, y and z directions one meter above the surface. Due to the ground geometry, among other things, different pressure points within the coastal dunes can occur, causing various circular wind patterns. These patterns can be seen in in Figure 2, which shows a heatmap of the U_0 parameter with the arbitrary wind inlet angle of 20 degrees.

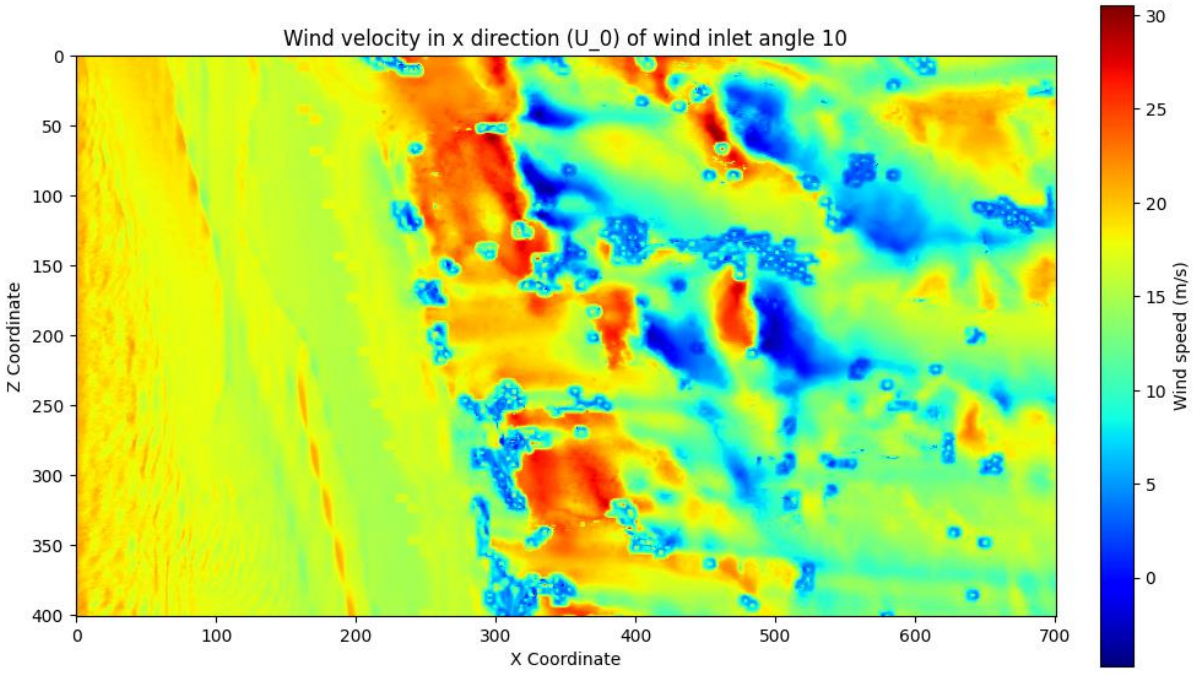


Figure 2: Sample heatmap of U_0 with an inlet angle of 10, derived from the CFD data

An overview of all the different parameters can be found in Table 1 below. Here a parameter called U_magnitude can be found as well, this is essentially the total velocity of parameters U_0, U_1 and U_2 combined. The PointID is used to be able to correctly map all the parameters in the same way.

Column name	***Description***
PointID	Point ID on the grid
X	Spatial coordinate in the x-direction
Y	Spatial coordinate in the y-direction
Z	Spatial coordinate in the z-direction
U_0	Wind velocity in direction x
U_1	Wind velocity in direction y
U_2	Wind velocity in direction z
U_Magnitude	Overall intensity or strength of the wind velocity vector. It gives a scalar measure of how fast the wind is moving in any direction, using the Pythagorean theorem
Wind inlet angle	Represents the angle of the incoming wind from the west
Cosine difference	The cosine of the difference between the inlet wind direction and the bed surface normal vector in x direction

Table 1: Meta data of all utilized parameters; every parameter is briefly explained

Some assumptions and a simplification were made regarding the data in this research. The first assumption is that the CFD output data accurately represents the real-world scenario. This assumption is made because the actual data is not available to such an extent that the wind velocity is known for each grid point. It is worth noting that the accuracy of the CFD model is validated by comparing its results with field-measured data at several sampling locations (Jim Regtien, Saeb Faraji Gargari, Gerben Ruessink, Michiel van den Broeke, 2024). Another assumption is that the vegetation present in the dunes do not affect the wind velocities. This is assumed because not vegetation data is available. Additionally, rough estimates for vegetation data used in CFD simulations shows minimal impact on the wind velocity. The simplification made within this research is that model only receives wind from the west direction. This simplification provides a more controlled environment, as this research is meant as a proof of concept, while still being able to provide relevant results as winds from the west promote sand buildup at the back dunes.

4. Methods

In this chapter, the conceptual model for implementing CNN surrogate models for CFD simulations of the wind velocities in coastal dune terrains, which consists of three distinct parts, is introduced. Afterwards, the process undertaken to eventually train, test and evaluate the models is outlined and the various CNN architectures are explained.

The three distinct parts of the conceptual model are: data generation, model training and model evaluation. The first part is data generation using traditional CFD simulations. The CFD approach simulates the wind velocity vector over a specific coastal dune topography for various scenarios of wind inlet angles. The data generated by the CFD simulation is then converted to a uniform grid, in order to be implemented. The second part, model training, consists of using the input data (coastal dune terrain topography and the cosine difference) and the generated CFD output data (wind velocity vector) to train the model, these data parameters have been explained in further detail in Chapter 3. The third and last part is model evaluation, which employs a Cross-Validation (CV) approach to correctly capture the model's performance and robustness over the entire dataset. An overview of the entire conceptual model can be found in Figure 3.

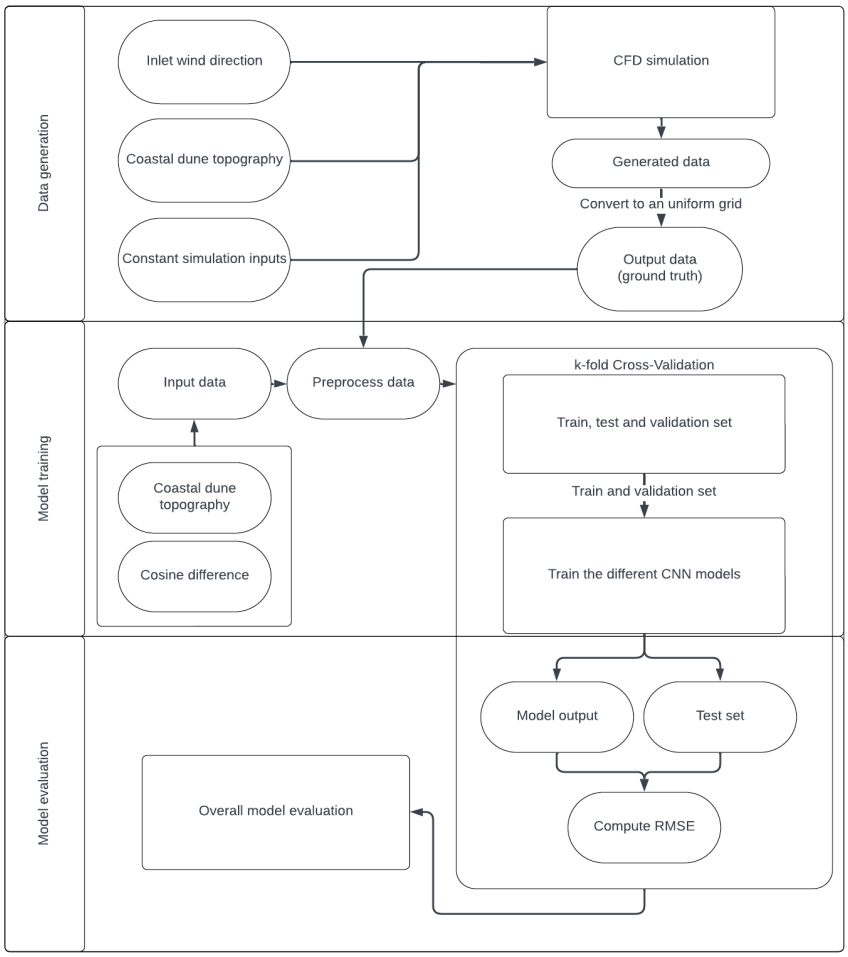


Figure 3: Conceptual model overview, consisting of three distinct parts: Data generation, Model training and Model evaluation. Every part contributes an important step into reliability training, assessing and comparing the different CNN surrogate models

The process undertaken to train, test and evaluate the various models consists of data (pre)processing, implementation of the various CNN architectures and backbones and model evaluation. The final step ensures that the models can be compared quantitatively. Data (pre)processing is an important step in ensuring that the input data is in the correct format and quality to optimize model performance. This process consists of multiple steps. The first step is data standardization, which ensures the data is within a smaller scale, improves the convergence rate during training and can improve model performance. In the second step, the different input and output parameters are mapped into the same size 2D grid format, allowing the parameters to be processed by the CNN models. The last step is to store all data and separate them using folders based on the wind inlet angles. This way, they can be easily accessed for the Cross-Validation (CV) step, which will be discussed later in this chapter.

Various CNN architectures and backbones are implemented and compared in this research. The implemented architectures are the U-net, LinkNet and Feature Pyramid Network (FPN), and the implemented backbones are the resnet152, densenet121 and efficientnetb7. In the next few paragraphs, the CNN architectures and backbones will be briefly explained.

The first CNN architecture is one of the most popular architectures in surrogate modelling, as illustrated by the many examples in the Introduction, namely the U-net architecture. This architecture consists of a shared encode and decoder. For each encoding, or down sampling step, the number of feature channels are doubled and for every decoding step, or up sampling step, the

number of feature channels are halved. At the decoder step, a concatenation of the correspondingly cropped feature map of the encoder step, called a skip connection, is added. In Figure 4 below, an example of the U-net architecture for a 32x32 pixels resolution is shown. Here every blue box represents a multi-channel feature map (Ronneberger, Fischer, & Brox, 2015).

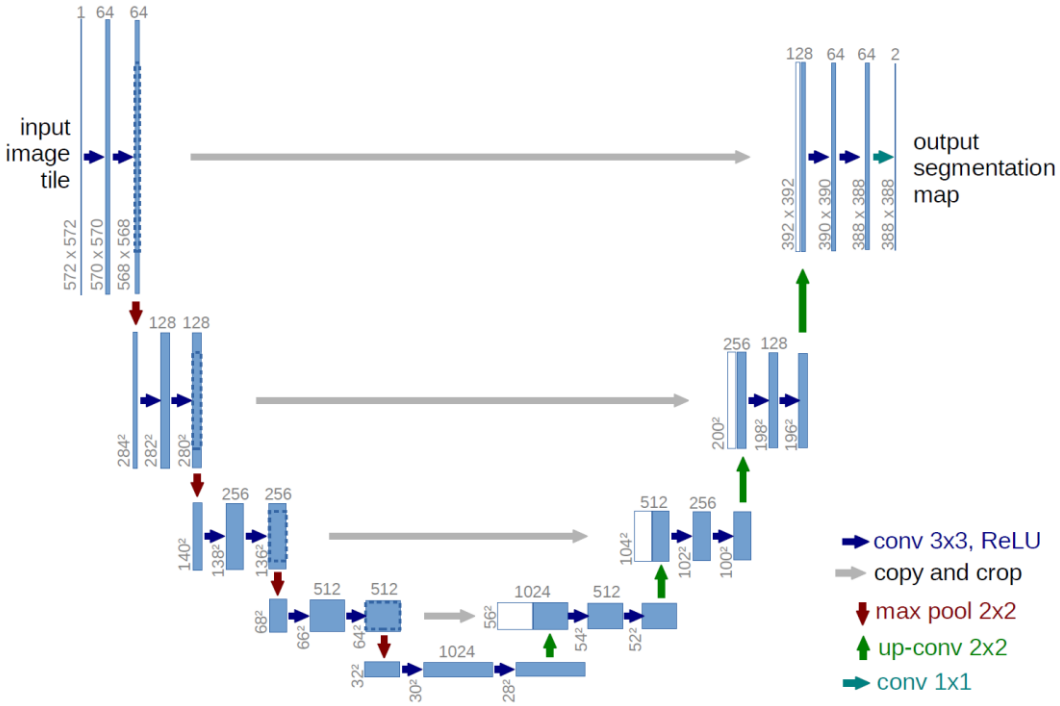


Figure 4: U-net architecture design, where every blue box corresponds to a multi-channel feature map with the number of channels denoted on top of the box. White boxes represent the copied feature maps. The arrows shows the different operations (Ronneberger, Fischer, & Brox, 2015).

The LinkNet architecture also consists of a shared decoder and encoder, with the same steps undertaken for the encoder as with the U-net architecture. The decoder also follows the same steps, except for the fact that the skip connection adds the correspondingly cropped encoders feature map instead of concatenating (Chaurasia & Culurciello, 2017). A global overview of the LinkNet architecture is shown in Figure 5, below.

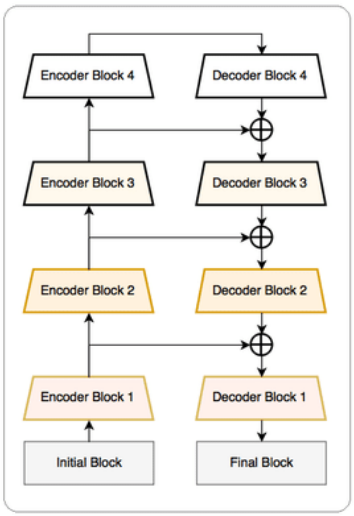


Figure 5: LinkNet architecture design, where the left side represents the encoder part while the right side represents decoder part. The circular icon with a plus represent the summation of the encoder feature map with the decoder feature map (Chaurasia & Culurciello, 2017).

The FPN architecture incorporates the assembling of the different decoder feature maps to form the image prediction. This combines adding the skip connections of the encoders feature map to the decoders one, down sampling this output to a fixed number of feature maps. These feature maps are then combined together using up sampling to form the final model prediction (Lin, et al., 2017). Figure 6 below, shows an example of the FPN architecture with three channel inputs (Seferbekov, Iglovikov, Buslaev, & Shvets, 2018).

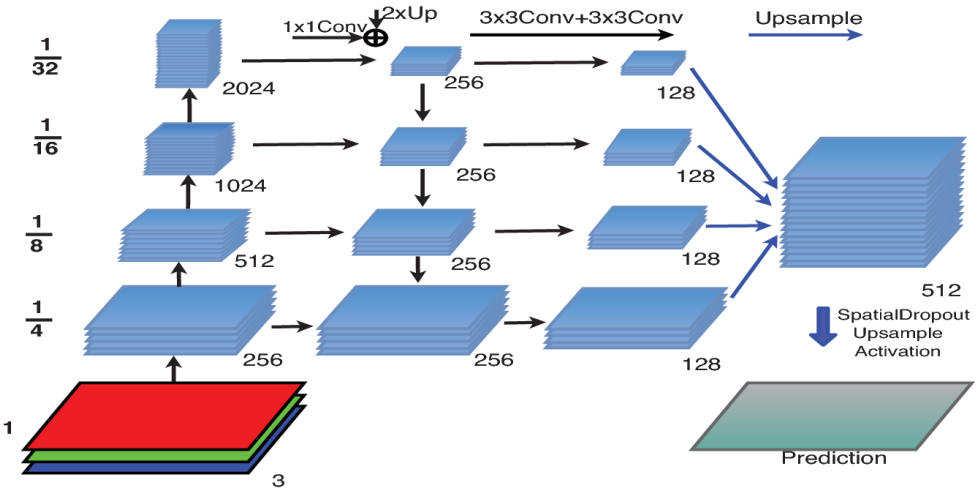


Figure 6: FPN architecture design, for the encoder at the left side the number of channels increases stage by stage while the size of feature maps decreases. The arrows on top of feature maps represent transformations implemented between the layers. For the final step, the feature maps are up sampled to the same size and concatenated and then down sampled to the number of output maps/classes and the resulting image is up sampled to the original image size (Seferbekov, Iglovikov, Buslaev, & Shvets, 2018).

A backbone is a specific network for encoding the feature maps within a CNN architecture. Choosing a different backbone for a CNN architecture can result in extracting different kind or levels of features. The resnet152 backbone utilizes a structure called residual learning unit, for alleviating the degradation of Deep Neural Networks. The residual learning unit's structure is a feedforward network that has a shortcut connection which adds input into the network to generate outputs (Nguyen, Lin, Lin, & Cao, 2018). The densenet121 structure simplifies these connections between layers and attempts to solve the vanishing gradient problem by reusing features through a connection of each block with one another (Jee, et al., 2023). The efficientnetb7 structure consists of 7 blocks, with the MBConv as fundamental building block of the structure, for feature extracting and is proven to be useful for transfer learning (Umut & İlhan, 2022). The structures, together with a brief explanation of these different backbone structures are shown in appendix 8.1.

Because of the model tasks and the kind of data on which these different backbones have been trained differs from our data and task, this research also implements the mentioned CNN architectures without the pre-trained backbone weights. This means the models are trained from scratch with randomly initialized weights. This approach is added because the feature encoding weights of the different backbones, trained for classification and segmentation tasks, might not be able to extract the relevant features for this research, which performs a regression task.

Model evaluation is done using the k-fold Cross-Validation (CV) approach to accurately capture the model's overall performance and robustness across the different wind inlet angles. K-fold CV is a robust method for assessing a model's generalization across the entire dataset. For this approach, the dataset is split into k folds of equal size. The model performance is tested iteratively for each fold, using the remaining folds as training data. After testing the model for each fold, the average evaluation metric for all folds is computed, resulting in the model's overall performance. For this research, the dataset will be split on the wind inlet angles. With three inlet angles of -30, 0 and 30 consistently left out as validation sets and a chosen k of 5, each fold consists of 2 inlet angles since

the dataset has a total of 13 inlet angles. The evaluation metric employed is the Root Mean Squared Error (RMSE) because it is easy to interpret the model performance, as the RMSE reflects the same units as the dependent parameter (Hodson, 2022).

5. Results

In total, nine surrogate models are trained, which are the three Convolutional Neural Network (CNN) architectures combined with three different backbones, discussed in Chapter 4. The averaged RMSEs from the k-fold Cross-Validation (CV) approach, folded across the wind inlet angles, are shown in Table 2 below. The RMSE standardized on the wind velocity is indicated in brackets for easier comparison between the different wind directions. The densenet121 backbone consistently returns the best performance for all CNN architectures while requiring the least amount of training time. Efficientnetb7 performs slightly worse and requires more computing time, while resnet152 has the worst performance and longest computing time. Among the different CNN architectures, the FPN architecture has the best performance, though it does not differ a lot from the U-net architecture, which required slightly less computing time. After performing a two-way ANOVA, the effect of the backbone is statistically significant with a p-value of 0,0205; while the effect of the CNN architecture is not statistically significant with a p-value of 0,148. For modelling the wind velocities in coastal dune terrain, the combination of the FPN model architecture and densenet121 backbone returns the best performance.

CNN architecture	Backbone	Average RMSE of (std) U_0	Average RMSE of (std) U_1	Average RMSE of (std) U_2	Average overall (std) RMSE	Average training time (minutes)	Average prediction time (seconds)
FPN	Densenet121	**2,91 (0,78)**	**0,71 (0,56)**	**3,26 (0,81)**	**2,29 (0,71)**	16,68	1,9
	Efficientnetb7	3,5 (0,92)	0,96 (0,76)	3,89 (0,96)	2,79 (0,88)	26,06	2,1
	Resnet152	3,8 (1,02)	1,53 (1,26)	5,36 (1,20)	3,56 (1,16)	28,94	2,2
LinkNet	Densenet121	3,68 (0,97)	1,2 (0,96)	3,95 (0,97)	2,94 (0,97)	16,1	1,9
	Efficientnetb7	3,78 (0,99)	1,25 (0,99)	3,87 (0,96)	2,97 (0,98)	25,83	2,1
	Resnet152	5,39 (1,56)	2,05 (1,64)	7,34 (1,63)	4,93 (1,61)	25,8	2,1
U-net	Densenet121	3,23 (0,87)	1,01 (0,81)	3,28 (0,83)	2,51 (0,86)	**15,8**	1,8
	Efficientnetb7	3,89 (1,01)	1,23 (0,96)	3,79 (0,95)	2,97 (0,97)	20,81	2,0
	Resnet152	4,64 (1,26)	1,68 (1,37)	4,39 (1,03)	3,57 (1,22)	25,86	2,1

Table 2: Comparison of model performance across the entire dataset, using k-fold CV with 5 total folds

All the models are trained for 75 epochs, using RMSE as metric and MSE as loss function, for equal comparison. Figure 7 below shows a training graph of the best model with an FPN architecture and densenet121 backbone. The orange line indicates the validation set, used for tuning the parameters and prevent overfitting, and the blue line indicates the training set. From this graph, it can be seen that the validation set has not yet converged, meaning the model could have been trained for more

epochs to improve performance. The fact that the model training was cut short decreases the potential model performance on the dataset.

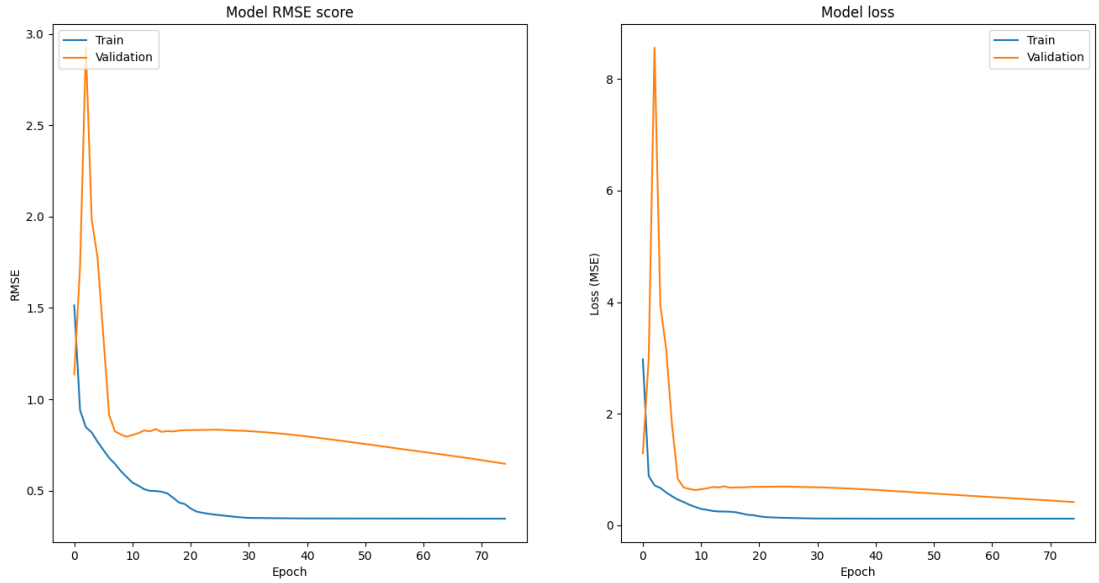


Figure 7: Sample training graph of the FPN densenet121 model with test fold of angle -50 and 20; with RMSE as metric (left) and MSE as loss function (right)

To assess the overall performance of the FPN densenet121 model, the RMSEs of the standardized wind velocities for every wind inlet angle fold are shown in Figure 8, below. Each bar colour represents a distinct wind direction and every inlet angle shown is unseen data for the model. The RMSE for the U_2 (z) direction is consistently higher than the other directions, for inlet angles of -40 or higher. Showing the model struggles with predicting the velocity for this wind direction, which can be a potential model improvement. This can be because the model training did not yet converge, meaning the model should be trained for more epochs. The model has a consistent performance pattern across the different inlet angles, except for inlet angle -40 where it begins to struggle with the U_2 direction. This shows that the models performance is stable and not heavily influenced by specific subsets of data.

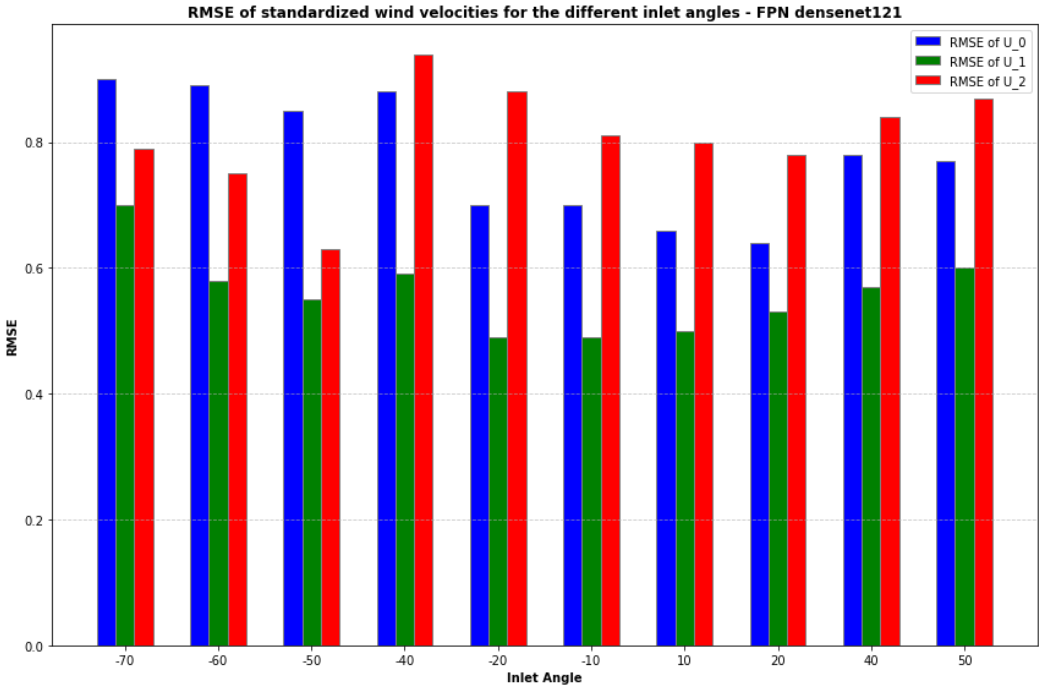


Figure 8: Model performance for each wind direction of every fold; all bars represent unseen test data - FPN architecture in combination with the densenet121 backbone

To gain more insight into the model's performance, Figure 9 below shows the model performance of one of the folds of the entire dataset. The orange bars with diagonal hatches represent the test set, the blue bars with squared hatches represent the validation set and the blue bars without hatches represent the training set. This graph validates that the model has a consistent performance pattern, also when comparing the performance on train, test or validation sets. This also shows the model does not overfit on the training data and the models ability to interpolate between the different wind inlet angles.

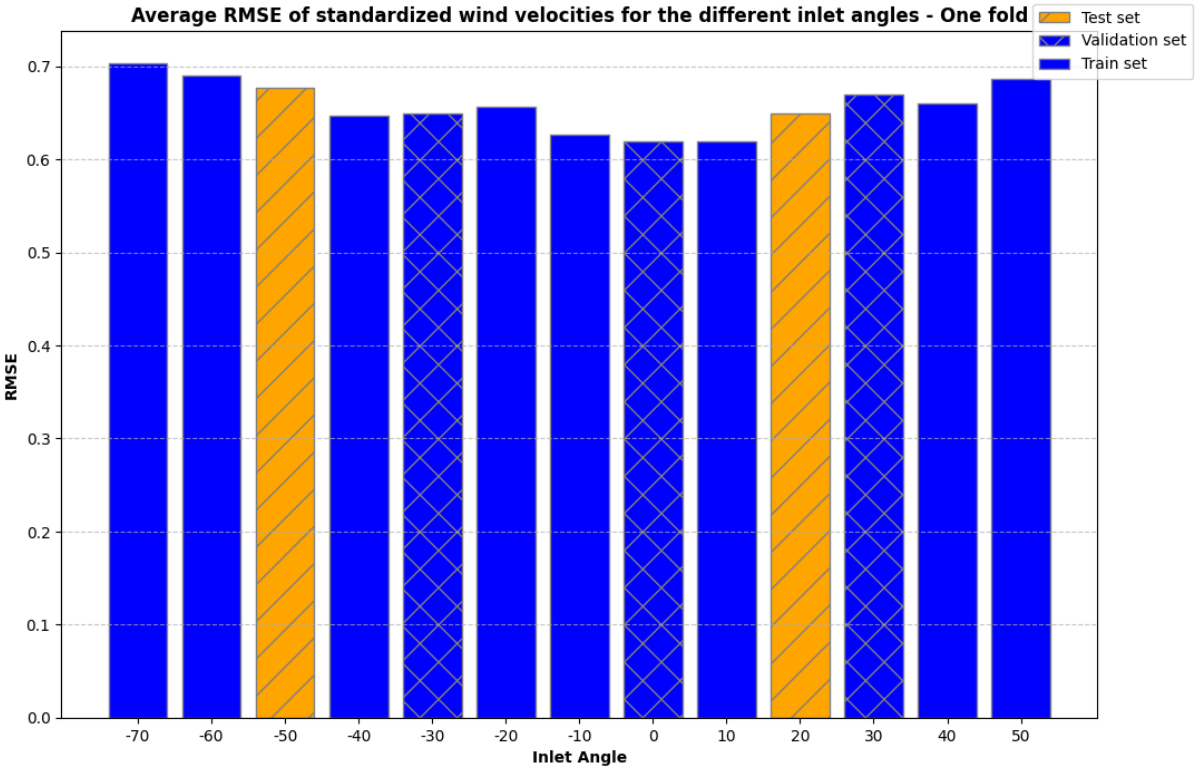


Figure 9: Model performance for each wind direction for one fold; with the orange bars indicating the test sets, the squared bars indicating the validation sets and the blue bar without hatches the training sets - FPN architecture in combination with the densenet121 backbone

An arbitrary prediction example of the FPN densenet121 model is shown in Figure 10, below. The first row on plots is the ground truth, or CFD simulation data, of the different wind directions, the second row is the prediction of the CNN model and the last row is the error between these two. According to the obtained results, the model can predict the general wind velocity vector patterns. However, the model predictions are less detailed than the CFD generated ground truths, as it predicts a more smoothed out wind velocity pattern rather than the actual wind flows. The model seems to not predict the various circular wind patterns, discussed in Chapter 3, at all but rather ignores them.

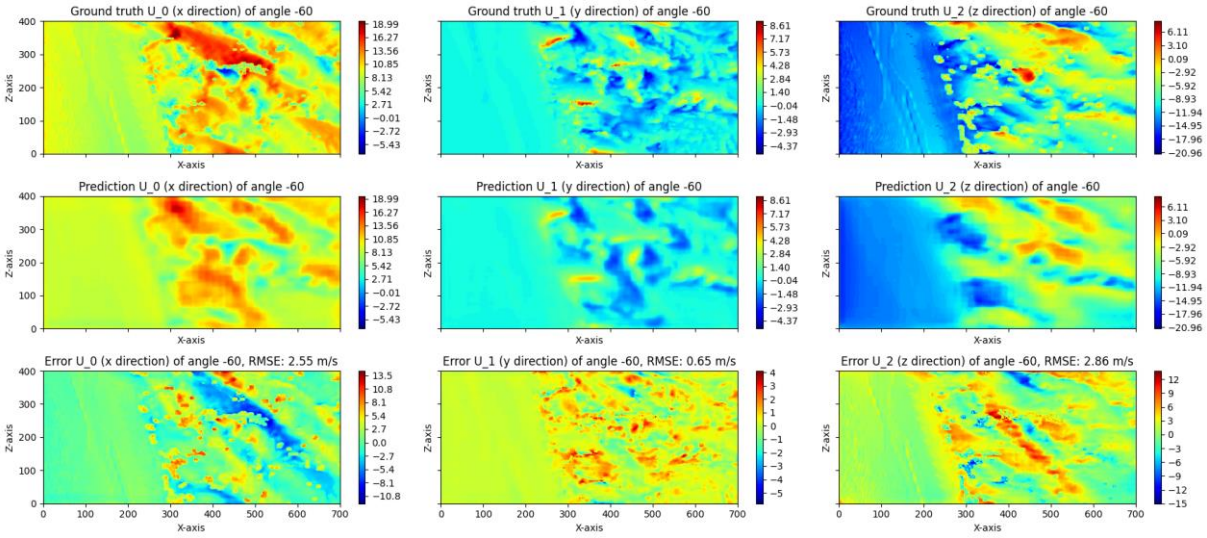


Figure 10: Sample plot of wind inlet angle -60 derived from the CFD generated data as reference (first row), the CNN prediction (middle row) and the error between these two (last row) – FPN architecture combined with the densenet121 backbone

To investigate to location of different errors, Figure 11 below is shown. In this figure, the error of the wind magnitude is presented on the left and the error of the angle on the right, both are overlaid on the topography of the coastal dune relative to the minimum value, of an arbitrarily chosen wind inlet angle. The left graph indicates that most errors lie within the circular wind patterns mentioned above with negative errors, at the sides on top of the different hills with positive errors and in a region on the more right side of the dunes, which is on a lower terrain, with negative errors. This last region of error can be explained by examining the right graph, which shows two large errors that emerged in this lower terrain. The region with large errors in angle differences on the right graph is the same region as the large negative errors on the left graph, excluding the circular patterns. This is also the case for some other wind inlet angles, a few examples are shown in appendix 8.2.

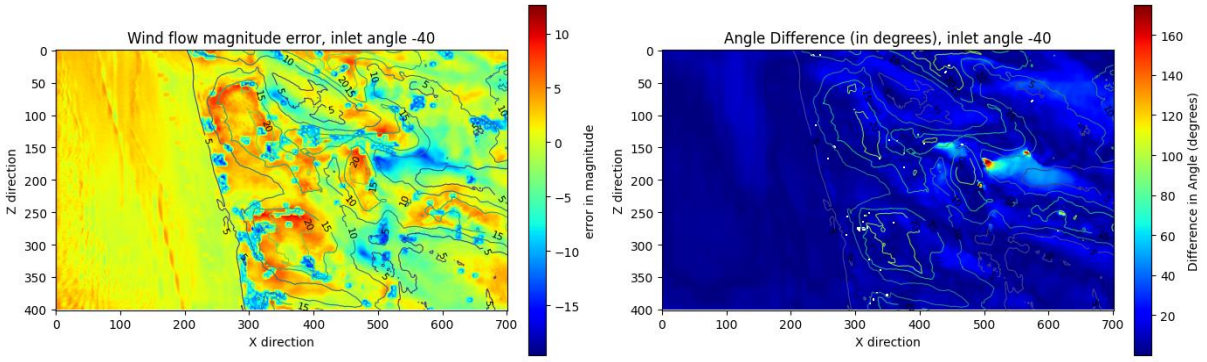


Figure 11: Sample plot of wind inlet angle -40; A contour plot relative to the minimal value of the topography with the error of the wind magnitude (left) and the angle difference in degrees (right) on top of it

The fact that most positive errors lie at the tops of the various dunes while the negative errors lie in the circular patterns and regions where the angle difference is larger is interesting. Because of this, the average error relative to the cosine difference for all data folds is shown in Figure 12 below. This graph shows that the larger positive errors emerge when the cosine difference is lower than -0,2; while the larger negative errors will be at regions with a cosine difference between 0,0 and 0,7. This means the model is underpredicting at regions that are upwind and overpredicting the regions that are closer to downwind.

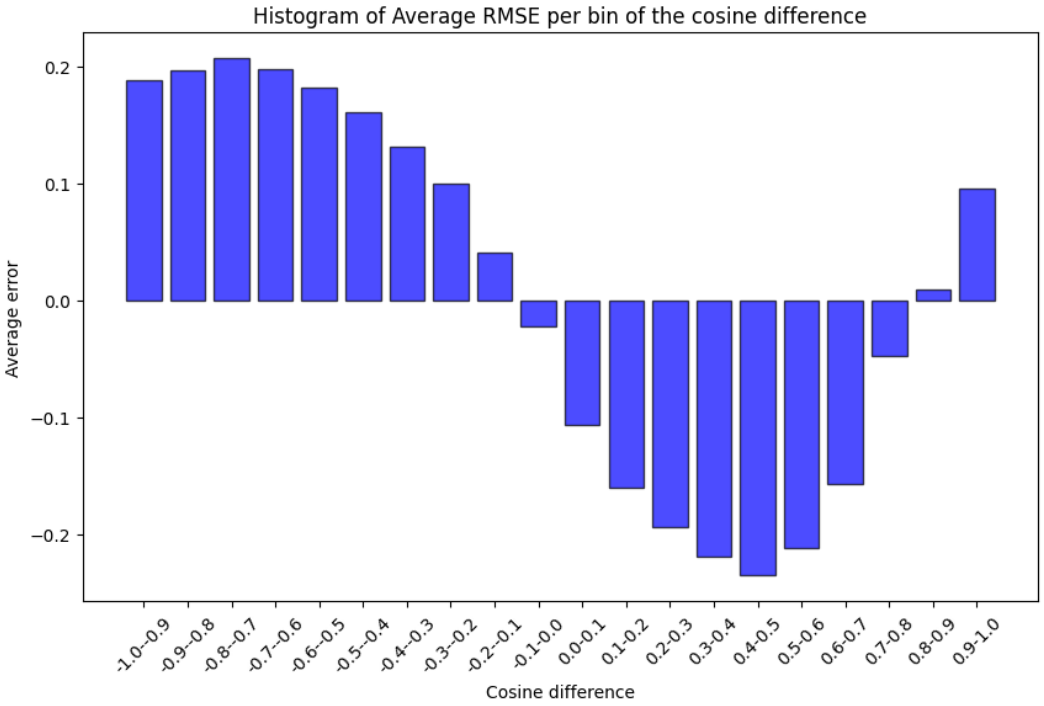


Figure 12: Histogram of the average error relative to the cosine difference for all wind inlet angles

To investigate whether it is plausible to assume for the model to predict the various circular wind patterns, the wind magnitude data on an arbitrary wind inlet angle is compared at different timesteps of the CFD simulation. To be able to compare to distributions of the data of the circular wind patterns, this research zoomed in on a region with a few of them combined, this specific region is visually shown in appendix 8.3. Recall that this research assumed that the wind flow in the coastal dune is in a steady state. To statistically compare the distributions of these different timesteps, the Kolmogorov–Smirnov test is employed and the cumulative distributions are plotted on top of each other for visually inspection. The cumulative distributions can be found in Figure 13 below, which shows all the different distributions are plotted on top of each other, indicating that they have the same distribution. The computed p-value of the Kolmogorov–Smirnov test resulted in a 0,99; which means that the different datasets have the same distribution.

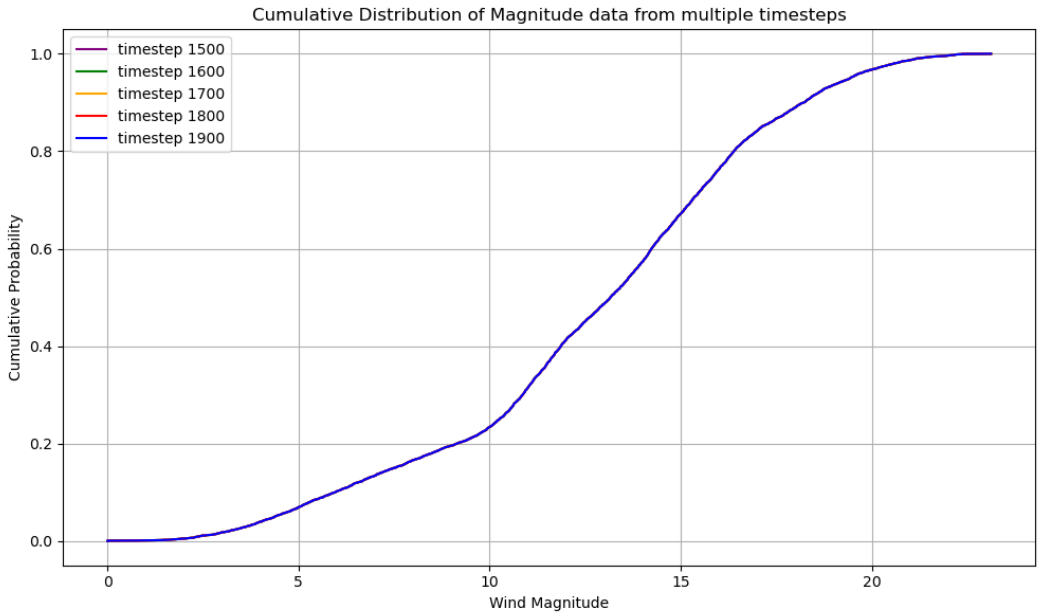


Figure 13: Cumulative distributions of an arbitrary wind inlet angle around a region with various circular wind patterns; timestep: 1500, 1600, 1700, 1800 and 1900

This research project was performed in parallel with another Applied Data Science research project that explored a different modelling approach to predict the velocity vector over a coastal dune terrain. The other project found the best performing model to be a Fully Connected Neural Network (FCNN) (Quigley, 2024). To compare the best model from both research projects, each research trained their best performing model on some predetermined wind inlet angles and shared the results. This research employed the FPN architecture combined with the densenet121 backbone and trained for a total of 150 epochs. The comparison of this model with the FCNN model is shown in Figure 14 below. In the figure, the blue bars represent the performance of the FPN densenet121 model and the red bars represent the performance of the FCNN model. All the hatched bars represent the testing sets. Due to the specifically chosen test sets, the models' interpolation and (deep)extrapolation performance can be observed. The graph shows that the CNN model consistently outperforms the FCNN model. However, the variation between the train and test performance is greater for the CNN model than for the FCNN model. Additionally, the flow between the wind inlet angles is smoother for the FCNN model than for the CNN model, this can be because of the CNN model not being trained until convergence.

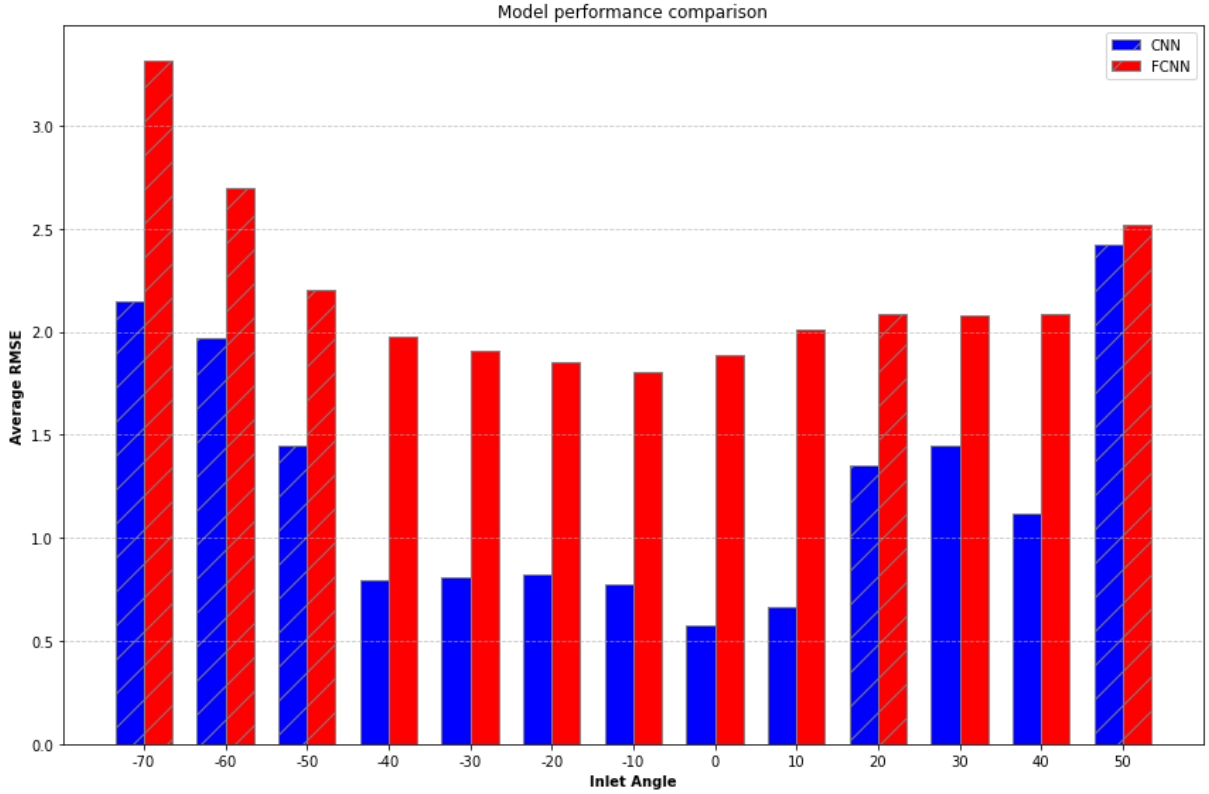


Figure 14: Model comparison of the two Applied Data Science theses, with the hatched bars representing the test inlet angles, the other inlet angles are used for model training; The FPN densenet121 of this research and the Fully Connected Neural Network of

6. Discussion & Conclusion

The main objective of this research was to serve as a proof of concept for using a Convolutional Neural Network (CNN) approach for Computational Fluid Dynamics (CFD) surrogate modelling of wind flow on coastal dune terrain. This study aimed to answer the following main research question: Which CNN model architecture and backbone most accurately models the wind velocities over coastal dune terrain with varying wind directions compared to a traditional CFD approach? The following sub-questions assisted in answering the main research question.

- Model architecture
    - To what extent do the different CNN architectures and backbones impact model performance?
- Model performance
    - How accurately does the model predict the wind velocity at various properties of the combined topography and wind direction within the coastal terrain?
    - How does the computational cost of the model compare to traditional CFD simulations?

Nine different models were implemented and evaluated using k-fold Cross-Validation (CV) on the entire dataset, split on the wind inlet angles. These nine models consist of three different CNN architectures and three different backbone combinations. Among the three CNN architectures, FPN showed the best results, however, after performing a two-way ANOVA, the effect of a CNN architecture was not found to be statistically significant. This is probably because of the small sample size, since large differences are observed between the FPN and U-net architectures compared to the LinkNet architecture. Among the three backbones, the densenet121 showed the best results and, after performing a two-way ANOVA, was found to have a significant effect on the model performance. These findings proof that the implementation of different backbones can significantly improve model performance, while varying CNN architectures also show potential for model performance improvement. The FPN architecture in combination with the densenet121 backbone resulted in the best overall model performance in predicting the wind velocity vector over the coastal dune terrain.

The consistent performance of the FPN densenet121 model across the different wind inlet angles show that the model does not overfit on the training data and is not heavily influenced by specific subsets of data, showing its robustness. The consistent performance also suggest that the model captures the spatial dependencies and complex relationships to some extent.

The FPN densenet121 model provides good predictions of the overall wind flow over the coastal dune terrain. However, some systematic errors do occur. Different circular wind flow patterns, occurring due to pressure point caused by the bumpy topography, consistently results in negative errors in the model's wind magnitude predictions. Lower located regions behind the dunes, where the model's angle prediction error is higher, also show negative errors in wind magnitude predictions. While the higher regions on the sides at the top of the various dunes often exhibit positive prediction errors in the wind magnitude prediction. This result coincides with the finding that the model is underpredicting in upwind regions and overprediction in the regions that are closer to downwind.

Additionally to these systematic errors, the model's prediction of the overall wind flow is not able to follow the finer details of the CFD simulation output. The prediction velocity vector follows a more smoothed out result. This is most likely for the model to prevent overfitting and reduce the impact of possible noise, which Machine Learning methods are known for (Mahoney, 2021). A possible way to remedy this can be to shift the CNN task from pixelwise regression to super-resolution, a well-researched area (Zhao, Gallo, Frosio, & Kautz, 2016). Which is done by a previous study, which

modelled a super-resolution assisted rapid high-fidelity CNN model, that converts a low-fidelity CFD output to high-fidelity resolutions (Hu, Yin, Hamrani, Leon, & McDaniel, 2024). This, of course, would then again increase the computational time of the model prediction, as the low-fidelity CFD output first needs to be simulated, assuming the CNN model is already trained. This means there is always a trade-off between performance and computation time. The CNN surrogate model for this research required, on average, 1,9 seconds to predict the wind velocity vectors of one wind inlet angle, using one GPU. Compared to the traditional CFD simulation, which uses 12 CPUs and takes around 5 minutes to simulate the wind velocity vectors for one wind inlet angle, this is a significant decrease in inference time.

Despite the promising results, several limitations need to be addressed. First, the research was conducted on the dataset of only one coastal dune, which does not fully capture the various real-world coastal dunes, limiting the generalizability of the model. Expanding the dataset to include a larger range of coastal dunes could improve model generalizability and robustness. Additionally, while the densenet121 backbone showed the best performance, experimenting with other backbones might further enhance the model's performance. This research also did not implement a hyperparameters search to tune the model or employ ensemble methods to improve model performance. As the combination of multiple models, using ensemble methods, might decrease the obvious errors at the up- or downwind regions for this research's model.

In conclusion, different CNN architectures and backbones impact model performance, with backbones having significant effect. The best model combination is the FPN architecture with the densenet121 backbone. The model underpredicts the wind velocity vector in upwind regions and overpredicts it in the regions that are closer to downwind. Additionally, the computational cost is significantly lower than traditional CFD simulations, with a prediction time of 1,9 seconds compared to 5 minutes for the CFD simulations. The results of this research demonstrates the feasibility and effectiveness of using CNNs for surrogate modelling of airflow over coastal dune terrains. The findings show the potential of these models to enhance coastal management practices and provide a foundation for future research aimed at improving model robustness and efficiency. The implementation of such CNN surrogate models involves a trade-off between prediction accuracy and computational speed.

7. Bibliography

Alzubaidi, L., Zhang, J., Humaid, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., . . . Farhan , L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 53. Retrieved from https://doi.org/10.1186/s40537-021-00444-8

Ashgriz, N., & Mostaghimi, J. (2002). An Introduction to Computational Fluid Dynamics. *Fluid Flow Handbook*, 1-49. Retrieved from https://www2.mie.utoronto.ca/labs/mussl/cfd20.pdf

Baumann, A., Roßberg, T., & Schmitt, M. (2023). Probabilistic MIMO U-Net: Efficient and Accurate Uncertainty Estimation for Pixel-wise Regression. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Retrieved from https://arxiv.org/pdf/2308.07477

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., & Kaushik, S. (2019). Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics 64*, 525-545. Retrieved from https://link.springer.com/article/10.1007/s00466-019-01740-0

Bizopoulos, P., Vretos, N., & Daras, P. (2020). Comprehensive Comparison of Deep Learning Models. *arXiv preprint*. doi:2009.06412

Chaurasia, A., & Culurciello, E. (2017, December). Linknet: Exploiting encoder representations for efficient semantic segmentation. *IEEE visual communications and image processing (VCIP)*, 1-4.

Chen, Z., Luo, F., Li, R., & Zhang, C. (2024). Modeling the Flow and Geomorphic Heterogeneity Induced by Salt Marsh Vegetation Patches Based on Convolutional. *Journal of Geophysical Research: Earth Surface*. doi:2023JF007336

European Environment Agency. (2008). Coastal and Marine Zones. *Environment in the European Union at the turn of the century*, 356-375. Retrieved from https://www.eea.europa.eu/publications/92-9157-202-0/3.14.pdf/view

European Evironment Agency. (2008). Coastal and Marine Zones. *Environment in the European Union at the turn of the century*, 356-375. Retrieved from https://www.eea.europa.eu/publications/92-9157-202-0/3.14.pdf/view

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, 519-567. doi:10.1007/978-3-030-32644-9_36

Guo, X., Li, W., & Iorio, F. (2016). Convolutional Neural Networks for Steady Flow Approximation. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 481-490. Retrieved from https://doi.org/10.1145/2939672.2939738

Hesp, P., & Smyth, T. (2021). CFD flow dynamics over model scarps and slopes. *Physical Geography*, 1-24. doi:https://doi.org/10.1080/02723646.2019.1706215

Hodson, T. (2022). Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development Discussions*, 1-10.

Hu, B., Yin, Z., Hamrani, A., Leon, A., & McDaniel, D. (2024). Super-resolution-assisted rapid high-fidelity CFD modeling of data centers. *Building and Environment 247*, 111036. Retrieved from https://www.sciencedirect.com/science/article/pii/S0360132323010636?ref=pdf_download&fr=RR-2&rr=8897338cfd1b9ffa

Husemann, P., Romão, F., Lima, M., Costas, S., & Coelho, C. (2024). Review of the Quantification of Aeolian Sediment Transport in Coastal Areas. *Journal of Marine Science and Engineering*, 755. Retrieved from https://doi.org/10.3390/jmse12050755

Jee, G., Gm, H., Gourisaria, M., Singh, V., Rautaray, S., & Pandey, M. (2023). Efficacy determination of various base networks in single shot detector for automatic mask localisation in a post COVID setup. *Journal of Experimental & Theoretical Artificial Intelligence*, 345-364.

Jim Regtien, Saeb Faraji Gargari, Gerben Ruessink, Michiel van den Broeke (2024). Computational Fluid Dynamic Modelling of Airflow in Foredune Blowouts, Utrecht University, Climate Physics

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2278-2324. doi:10.1109/5.726791

Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117-2125.

Mahoney, M. (2021, January 18). *Model averaging methods: how and why to build ensemble models.* Retrieved from mm218: https://www.mm218.dev/posts/2021/01/model-averaging/

Maul, G. A., & Duedall, I. W. (2019). Demography of Coastal Populations. *Encyclopedia of Earth Sciences Series*, 692-700. Retrieved from https://doi.org/10.1007/978-3-319-93806-6_115

Moukalled, F., Mangani, L., & Darwish, M. (2016). The Finite Volume Method. In *The Finite Volume Method in Computational Fluid Dynamics* (pp. 103-135). Springer International Publishing. doi:10.1007/978-3-319-16874-6

Nguyen, L., Lin, D., Lin, Z., & Cao, J. (2018, May). Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation. *IEEE international symposium on circuits and systems (ISCAS)*, 1-5.

NSW department of Land and Water Conservation. (2001). *Coastal Dune Management: A Manual of Coastal Dune Management and Rehabilitation Techniques.* Newcastle: NSW Government publication.

O'Shea, K., & Nash, R. (2015, December 2). An Introduction to Convolutional Neural Networks. Retrieved from https://arxiv.org/pdf/1511.08458

OpenCFD Ltd. (2019). *About OpenFOAM*. Retrieved from openfoam: https://www.openfoam.com/documentation/guides/latest/doc/index.html#main-about-openfoam

quadco engineering. (n.d.). *Advantages and disadvantages of Computational Fluid Dynamics*. Retrieved from quadco engineering: https://www.quadco.engineering/en/know-how/cfd-advantages-and-disadvantages.htm

Quigley, James (2024). Hybrid Computational Fluid Dynamics (CFD) and Machine Learning for Airflow Simulation Over Coastal Terrain. Utrecht University

Ribeiro, M., Ahmed, S., Dengel , A., & Rehman, A. (2020). DeepCFD: Efficient steady-state laminar flow approximation with deep convolutional neural networks. *arXiv preprint arXiv:2004.08826*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference*, 234-241.

Seferbekov, S., Iglovikov, V., Buslaev, A., & Shvets, A. (2018). Feature pyramid network for multi-class land segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 272-275.

simscale. (2023, December 7). *What is CFD | Computational Fluid Dynamics?* Retrieved from simscale: https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/

Smyth, T. (2016). A review of computational fluid dynamics (CFD) airflow modelling over aeolian landforms. *Aeolian Research*, 153-164. doi:https://doi.org/10.1016/j.aeolia.2016.07.003

Umut, M., & İlhan, A. (2022). A soft voting ensemble learning-based approach for multimodal sentiment analysis. *Neural Computing and Applications*, 18391-18406.

United Nations Development Programme. (2023, November 28). *Climate change's impact on coastal flooding to increase five times over this century.* Retrieved from undp: https://hdr.undp.org/content/climate-changes-impact-coastal-flooding-increase-five-times-over-century

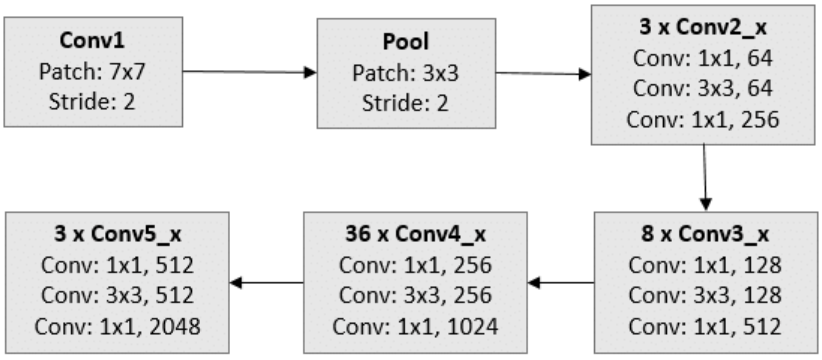
Wang, J., Mattie, D., Berger, D., & Levman, J. (2021). Multichannel input pixelwise regression 3D U-Nets for medical image estimation with 3 applications in brain MRI. *Medical Imaging with Deep Learning*. Retrieved from https://openreview.net/pdf?id=JG895xlWsfA

Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging 3.1*, 47-57. Retrieved from https://ieeexplore.ieee.org/document/7797130

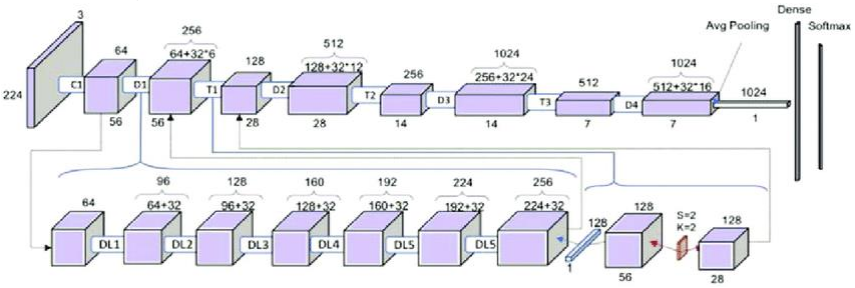
8. Appendix

8.1. The different backbone structures

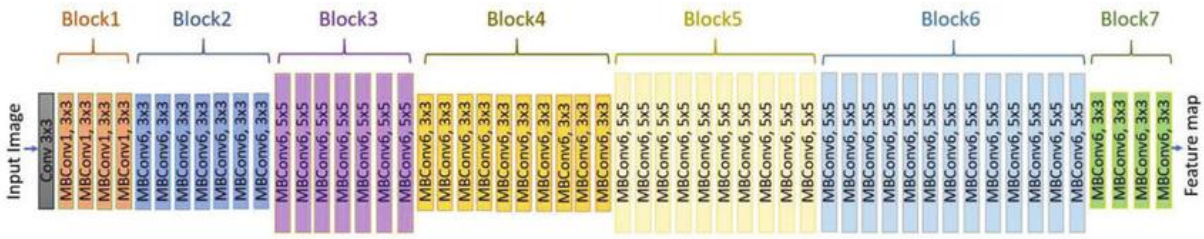
The basic structure of the resnet152 structure (Nguyen, Lin, Lin, & Cao, 2018):



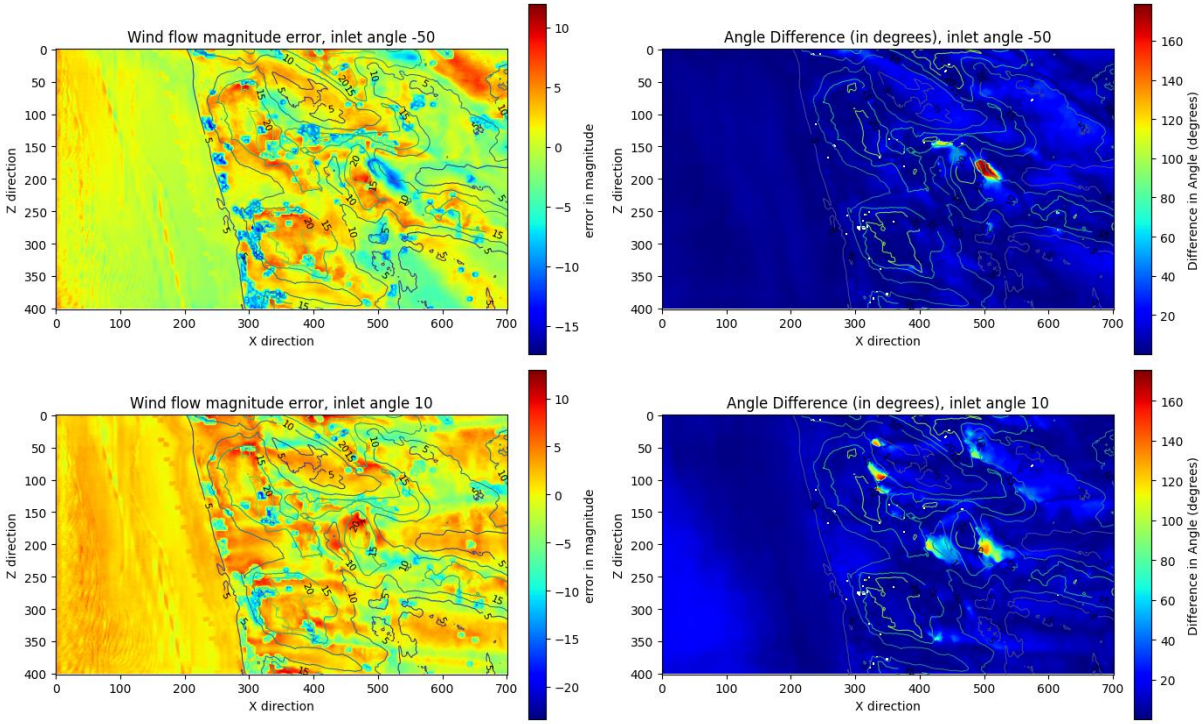
The structure of denseNet121 with Dense block (D) Transition blocks (T) and Dense Layers (DL) (Jee, et al., 2023):



The structure of efficientnetb7, the depth, width, resolution and model size increases as you are walking through the different blocks. The MBConv is the fundamental building block of the structure, the feature map consists of a single vector of features (Umut & İlhan, 2022):



8.2. Few examples of the contour plots with errors



8.3. Zoomed in area to check wind flow steady state