# Emulating a Lagrangian Atmospheric Dispersion Model With U-Nets

Chayton Bouwmeester

May 2025

## 1 Abstract

Machine learning methods can significantly reduce the computational burden of simulating certain physical phenomena through the construction of emulators. In this project I attempt to use a U-Net structure to emulate a physical model in the simulation of a fixed dispersion event of a neutral tracer gas. While the emulator was able to produce the accumulated concentration field in approximately 1.5 seconds compared to the 4 minutes required by the physical model, it demonstrated limited ability to accurately replicate the outputs of the physical model. The emulator was only able to achieve a correlation of 0.36 for grid cells with non-zero concentration and a mean intersection-over-union score of 0.3 on the test set. Measuring the permutation importance of the variable revealed the wind variables to have the largest influence on the resulting change in loss, followed by atmospheric boundary layer height and then with surface pressure as the least important. Despite these results and the limitations of this project, it does demonstrate the potential of U-Nets or similar deep-learning architectures to emulate dispersion models and can serve as a foundation for further development. Suggestions for improving the model presented here include increasing the size of the training dataset and the use of an attention term to help overcome the sparsity of the concentration data used to train the emulator.

## 2 Introduction

Atmospheric dispersion models, also known as atmospheric transport models (ATMs), are models which simulate the transport of pollutants through the atmosphere as a result of meteorological conditions. Such models have many applications, including the modelling of chemical accidents, volcanic ash, smoke from wildfires, air pollution, and the transport of radionuclides. The Severe Nuclear Accident Programme (SNAP) [1] is a Lagrangian dispersion model developed at the Norwegian Meteorological Institute and is used operationally for modelling the long range transport of radionuclides and volcanic ash. In emergency situations where time is limited, it is of utmost importance to be able to

provide decision makers with accurate information as fast as possible in order for safety measures to be effectively implemented.

Machine learning methods present a compelling avenue for increasing the speed at which these dispersion calculations can be performed. One such method is the use of emulators, which are trained on the inputs and outputs of physical models with the aim of reproducing output of the physical model at a reduced computational cost. There are primarily two different approaches to emulating dispersion models. Firstly, one can treat each grid cell of the model as a regression problem, thereby creating a large number of independent models which produce predictions for each specific grid cell [4] [3]. The other approach is to use techniques similar to those used in image analysis, where the inputs are passed into the model as grids where then a number of downsampling and upsampling steps are performed, which then finally produces a concentration field of the same dimensions as the physical model. Commonly, encoder-decoder type neural networks such as U-Nets and variational autoencoders are used in these contexts [6] [5] [2].

In this project, I aim to train a U-Net to replicate the output of the SNAP model for a fixed source term and time period. U-Nets are a variation of convolutional neural networks (CNNs) defined by an encoder-decoder structure with skip-sections linking the the decoder layers to encoder layers. The skip-connections allow the decoder to access the high resolution feature maps of the encoder, which helps to preserve higher resolution features otherwise lost during the downsampling steps in the encoder [8]. This symmetrical structure where the is often represented in a U-shape diagram which gives these models the name.

This project generally follows the approach of the Footnet emulator developed for greenhouse gas flux inversions [5]. To limit the scope and make it feasible for this project time frame, I aim to create an emulator which can produce dispersion calculations for a fixed source term from a single location. Furthermore, the model will be trained to predict just a single variable, being the accumulated concentration of a tracer gas, and will be limited to predicting the concentration field 6 hours after the beginning of the release.

The structure of this project is as follows: First in the methods section there is a brief description of U-Nets in the context of the implementation used in this project. Then I describe the set-up of the problem, the data sources, and the data pre-processing, training and any relevant error metrics used. Then in the results and discussion I present the performance of the emulator and explore the relative importance of the predictor variables used. I then include a limitations and future research section where I identify weaknesses in the project, provide suggestions for how these could be rectified, and ideas for how the emulator could be extended. Finally I conclude by highlighting the key takeways and some select results from the project.

# 3 Methods

## 3.1 U-Net Structure

The structure of the U-Net is illustrated in Figure 1. The model takes in an array with 8 channels, representing the 8 meteorological variables, each of which with dimensions 336x336. This array then passes through the encoder path, where it goes through 3 successive blocks which each consist of first applying two 3x3 convolution operations, each followed by a ReLU, and then a 2x2 max pooling operation with stride 2. The convolution operations double the number of feature channels in the array which allows the model to capture more complex feature information, while the max pooling operations then reduce the spatial resolution of the array. After these three blocks, the bottom of the U-Net structure is known as the bottleneck, which connects the encoder to the decoder and performs an additional two 3x3 convolution operations and ReLU steps without the subsequent downsampling. Then the model moves into the decoding path. Here, the three blocks successively upsample the array through first applying a 2x2 up-convolutional layer followed by two 3x3 convolutional layers. Additionally, after each upsampling convolutional step, the saved symmetrical array of the same dimensions from the encoder is concatenated to the array. These steps, known as skip connections, help to preserve the spatial information lost in the downsampling steps in the encoder. The final step of the decoder also differs in that it has just one channel, as in this case the output is the concentration field 6 hours after the release. As the model trains, the loss function is used to adjust the weights in the various convolutional steps via backpropagation.

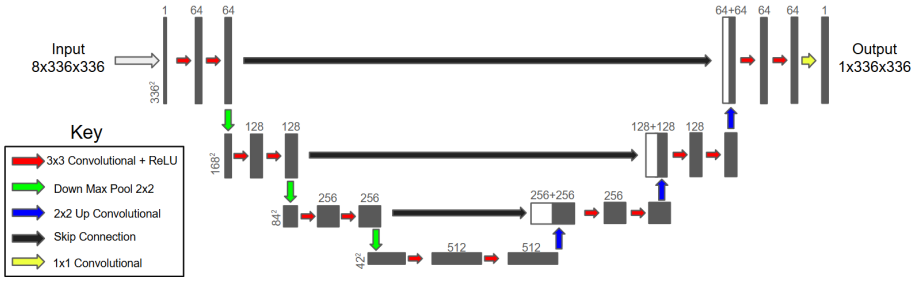


Figure 1: Diagram of U-Net structure used in this project. The horizontal numbers indicate the number of feature channels in the array at different points, and the vertical numbers show the spatial resolution. The various operations performed are shown by the arrows.

## 3.2 Training Data

ATMs are driven by meteorological datasets, which provide the relevant atmospheric variables for determining the path of the Lagrangian particles. For this

project I use forecasts from the AROME-MetCoOp numerical weather prediction (NWP) model, which is used operationally for providing 2.5km resolution forecasts for Finland and Scandinavia [7]. The full model domain encompasses all of Fennoscandia, the Baltic states, as well as some other small parts of Northern Europe. The physical model is run using the whole domain, however for the training process a smaller subset of the domain is selected to reduce computational cost and also to reduce the sparsity of the concentration data (Figure 2).

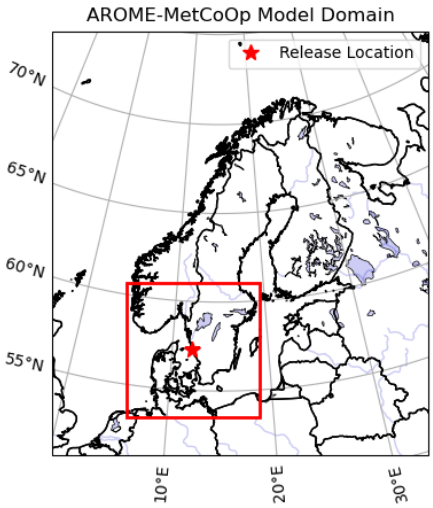


Figure 2: Map of the full model domain of the AROME-MetCoOp weather model. The area of data used in this study is highlighted by the red square. The red star indicates the release location.

To generate the training dataset, I ran the dispersion model a total of 1092 times, each initialised from a different forecast dataset separated by 6 hours between the 1st of January 2023 and the 30th of September 2023. In each run, a release of perfluoromethylcyclohexane (PMCH) occurs at a rate of 100 g/s for a total of 1 hour beginning at the 6th hour of the simulation. This delay is to account for any spin-up anomalies which can occur in the first few hours of an NWP model. The simulation then continues for another 6 hours after the end of the release for a total simulation length of 13 hours. In every run the release occurs at coordinates 57.26006°, 12.11076°, which was chosen as it is close to the centre of the model domain and is the site of Ringhals Nuclear Power Plant, a potential application of the SNAP model. PMCH was chosen as it is a neutral tracer gas, and is therefore neither subject to radioactive decay nor wet and dry deposition. This reduces the complexity of the problem, and means fewer variables need to be used for training.

A total of 8 predictor variables were used for training, being the 10 metre

wind in the x direction, the 10 metre wind in the y direction, atmospheric boundary layer height, and surface pressure. All of these variables are given at the time of release (hereafter time $T$) and at 6 hours after the beginning of the release (hereafter time $T + 6$), totalling 8 variables. The winds and pressure are taken from the meteorological dataset, whereas the ABL height is taken as the computed values from the SNAP model. These 8 variables are then passed into the model, which then produces an output of the same resolution, being the accumulated concentration of the gas at time $T + 6$. The accumulated concentration refers to the time integrated concentration for a given location and time. This is chosen as it is often used when discussing the exposure of populations to a pollutant and is highly relevant in emergency modelling.

Prior to the training process, the concentrations are taken as a natural logarithm, as the concentration values can span a very wide range of values across orders of magnitude. Then, values of less than -20 were filtered out and 20 was added to all values, thus making the minimum of the concentration values 0. This is done because the concentration values are generally very small, which could be difficult for the model to learn to learn from and make predictions for. The meteorological variables also span quite different orders of magnitude, and therefore the surface pressure and boundary layer heights are divided by 1000 and the wind speeds were multiplied by 10 in order to bring the variables on a more similar scale.

### 3.3 Training Process and Evaluation

The training dataset was first split into 80% training, 10% validation and 10% for testing. Loss was measured using the mean squared error (MSE) between the predicted accumulated concentration field at time $T + 6$ and the true value produced by the physical model. The Adam algorithm was used to optimise the weights in the convolutional layers with a learning rate of 0.001. The training loop was run for 100 epochs, at the end of which the model which produced the lowest validation loss was saved.

In order to evaluate the accuracy of the spatial extent of the plume, the intersection-over-union (IoU), also known as the Jaccard index, is used. IoU is commonly used to evaluate machine models in the context of object detection. This is defined as:

$$IoU = \frac{A \cap B}{A \cup B} \tag{1}$$

Where A is the prediction and B is the truth. In this context when calculating the IoU, all grid cells with a concentration above zero are taken as 1, and grid cells with no gas concentration are zero. This statistic then measures the ability of the model to capture the spatial extent of the plume. To evaluate the ability of the model to predict the concentration values, the Pearson correlation coefficient is also used.

To estimate the importance of each of the predictor variables in the emulator, the permutation importance is used. This method involves making pre-

dictions using the trained model while shuffling a particular variable. If the loss increases after shuffling the chosen variable compared to the baseline of unshuffled data, then this variable is 'important' to the models ability to make predictions. Therefore the higher the increase in the loss, the more important the variable is. While this method is rather simple and does not account for correlations between variables, it can be used without retraining the model and gives an indication as to the most influential variables.

# 4 Results and Discussion

The convergence plot shows that the training process appeared to stagnate at after about 40 epochs, at which point the training and validation losses remained relatively constant (Figure 4). The model which produced the lowest validation loss was then used for evaluation. Figure 5 shows three randomly selected examples from the training set to illustrate the predictions of the U-Net model compared to the 'true' results produced by the SNAP model. A qualitative inspection of a few randomly selected results from the test set shows that the model is generally able to capture the direction and approximate size of the plume in these cases, indicating it was able to learn some relationships between plume development and the meteorological variables. However, it can also be seen that the predictions for the concentration values within the plume tend to be much lower in the U-Net model compared to SNAP. Furthermore, as shown in test sample 49, the U-Net model is prone to creating spurious patches of concentration disconnected from the main plume.

Using the full testing dataset, the correlation between the concentration values from the U-Net and the SNAP model are shown in Figure 6. The correlation of the concentration values between the U-net and the SNAP model was reasonable at 0.57. However, as indicated from the heatmap, this seems to be mostly inflated by the large amount of grid cells with zero concentration, which when excluded reduces the correlation significantly to just 0.36. From the plot, it is clear that the U-Net model quite consistently underestimated the true concentration values. A reason for this could be the sparsity of the concentration data. The gas concentration in large majority of grid cells in the model domain is zero, which when simply using MSE as the loss function would likely lead to the model tending to guess smaller values as to minimise the penalty when predicting a concentration in a grid cell than in truth is empty. As for the shape of the plume, the mean IoU score for the testing dataset was found to be 0.3. While some of the testing samples indicate some ability to capture the spatial extent of the dispersion, there is still clearly significant room for improvement.

The poor accuracy of the emulator could be due to the training dataset being too small. The total dataset consisted of 1092 samples, 20% of which is set aside for validation and testing, leaving 874 samples for training. It is quite possible that this is insufficient to effectively learn the patterns in the distribution of the gas over the shape of the plume. Larger datasets would likely be required to properly assess the overall merits of this approach, however storage and time

limitations in this project resulted in the relatively small training set used.

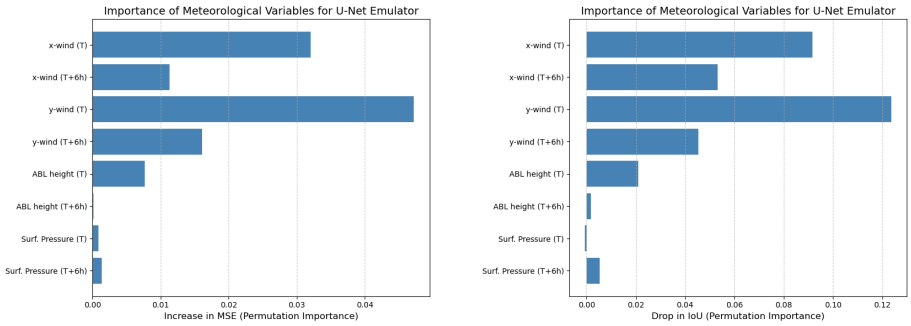


Figure 3: Variable importance as calculated from the permute and predict method. The two figures show the change in MSE (left) and the change in IoU (right).

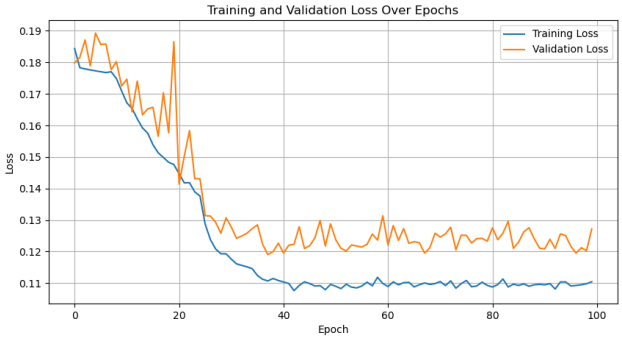


Figure 4: Training and validation loss as measured by mean squared error over training epochs.

The relative importance of the predictor variables used in the emulator as measured by the permutation importance method are shown in Figure 3. When measured either by increase in MSE or decrease in IoU, the influence of permuting a given variable on the emulator's ability to predict the accumulated concentration field is largely the same. Unsurprisingly, the wind speed variables are the most important in determining the characteristics of the plume after 6 hours. The surface pressure by comparison appears to have very little impact on the predictive ability of the emulator. The x-wind, y-wind, and ABL height all indicate that the wind at $T$ is much more important than the same variable at $T + 6$. Also of note is that difference in the importance of the height of the ABL at time $T$ and the ABL height at $T + 6$ is much large than for the other

variables. A reason for this could be that the height of the boundary layer can have a large influence on the initial rise of the plume, which then has strong implications for the transport of the gas later in time.

Although the results of the emulator are not particularly accurate, they are, as expected, able to be produced significantly faster. The speed at which a Lagrangian dispersion model can run is of course highly dependent on the number of Lagrangian particles used and the hardware being used. However as a point of comparison, the U-Net was able to calculate a concentration field at time $T + 6$ in approximately 1.5 core seconds whereas a run of the SNAP model to calculate the same concentration field on the same hardware took approximately 4 core minutes. While 4 minutes in itself is not a significant amount of time, this amount of speedup can be beneficial when large amounts of model runs are required, for instance in applications of source localisation, or in the generation of ensembles to measure uncertainty [5], [4].

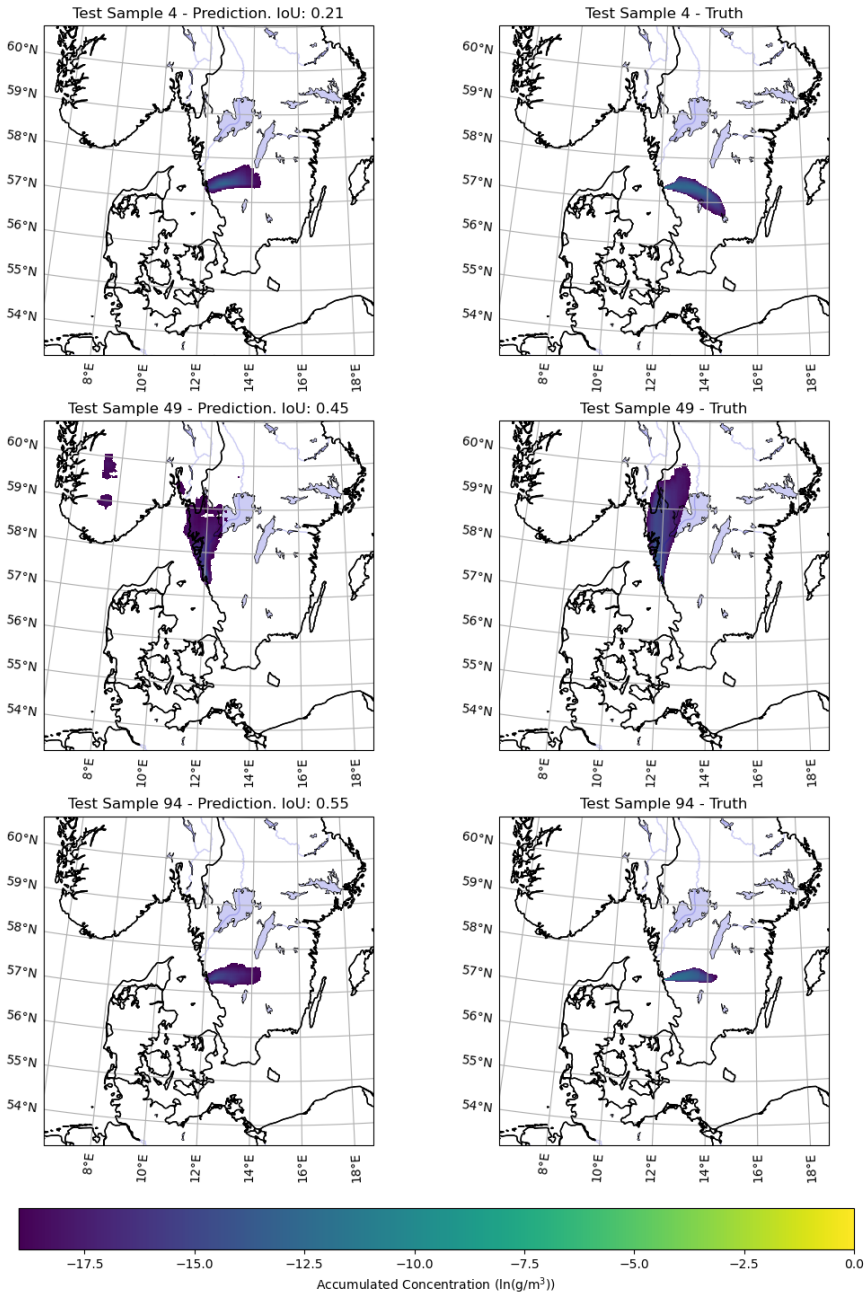


Figure 5: Plotted accumulated concentrations for three different randomly selected samples from the test dataset. Predictions from the U-Net emulator are shown on the left, and the 'true' concentrations fields produced by the SNAP model are on the right. Also shown for each predicted concentration field is the calculated IoU value for that sample.

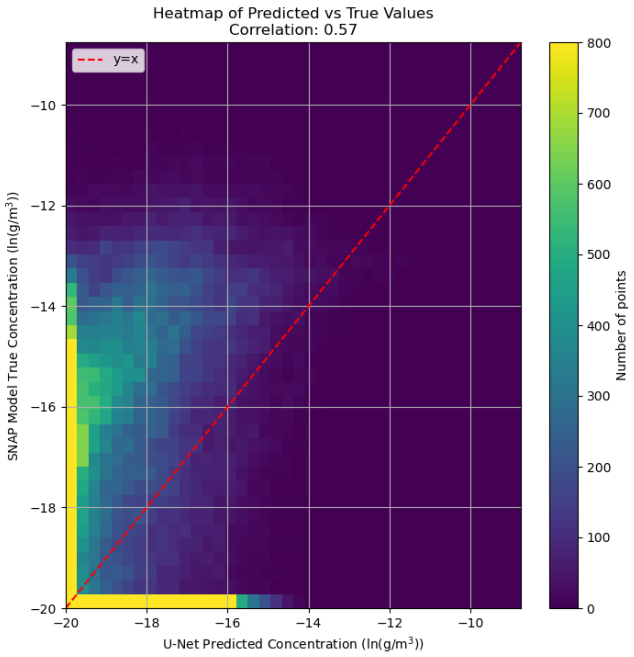


Figure 6: 2D Histogram showing the correlation of each grid cell in the test set between the predicted concentration values from the U-Net model on the x axis and the physical SNAP model on the y axis. The colour reflects the number of grid cells from the whole test set. The red dashed line shows a correlation of 1 for reference.

# 5 Future Work and Limitations

Due to the time constraints of the project, the emulator here was designed to be for essentially the simplest possible case, with a fixed source term, no wet or dry deposition, and no decay of the tracer. Therefore there many avenues for expanding and improving upon this approach. The first priority would of course be improving on the simple case presented here in order to achieve more accurate results.

It would likely be beneficial to include a much larger training dataset, to see if more training examples could improve the ability of the model to learn these patterns. The relatively limited training dataset in this project may not have contained enough information for the model to learn the intricacies of the relationship between the meteorology and the patterns in the dispersion. With the model largely underestimating concentration values, it would likely be beneficial to include some sort of attention term which weights the cells within

the plume higher than the zero values outside of it. It was also not investigated in this project if the model is consistently performing poorly for certain types of weather conditions. It is also possible that the time period from which the data is taken does not capture a wide enough range of meteorological situations which could result in poor performance for underrepresented conditions. Furthermore, the impact of changes to the model architecture and hyperparameters was not explored in this project, and tuning of these would likely result in improved model performance.

If good results for the simple case are able to be achieved, one could then look to extend the model to handle more complex dispersion patterns. For instance, precipitation data could be included in the training to see if the model can capture the patterns of wet deposition. Additionally, variables describing of the source term (e.g. release rate or release height) as training variables to allow it to predict a broader range of releases. Also of interest would be to evaluate the generalisabiltiy of the model to different geographical locations. The model here is simply trained on a single location, however the same physical principles apply in terms of the dispersion patterns. It could be possible to extend the ability of the emulator to predict releases from other locations after the addition of more training examples to help capture any local details which may affect dispersion.

# 6 Conclusions

Overall the emulator produced in this project was not able to replicate the output of the SNAP dispersion model with a high level of accuracy. After convergence, the model was only able to achieve a correlation of 0.36 on the test set and a mean IoU score of 0.3. The permutation importance also seemed to follow intuitive understandings of how the physical dispersion model operates with the wind variables being most important, suggesting that the U-Net was learning meaningful relationships between meteorological input variables and the resulting concentration fields. The significantly increased speeds at which these calculations can be performed also demonstrates the potential benefit of machine learning based emulators, if a higher degree of accuracy is able to be achieved. The limitations in performance are likely due to the limited training data and lack of optimisation in the model structure and hyperparameters. Despite these limitations, the project demonstrates the potential feasibility of using U-Nets and similar deep-learning structures to approximate dispersion models, and it provides a foundation for future improvements.

# References


[1] Jerzy Bartnicki, Hilde Haakenstad, and Øystein Hov. Operational SNAP Model for Remote Applications From NRPA. December 2011.

[2] Laura Cartwright, Andrew Zammit-Mangion, and Nicholas M. Deutscher. Emulation of greenhouse-gas sensitivities using variational autoencoders. *Environmetrics*, 34(2):e2754, March 2023.

[3] Elena Fillola, Raul Santos-Rodriguez, Alistair Manning, Simon O'Doherty, and Matt Rigby. A machine learning emulator for Lagrangian particle dispersion model footprints: a case study using NAME. *Geoscientific Model Development*, 16(7):1997–2009, April 2023.

[4] Nipun Gunawardena, Giuliana Pallotta, Matthew Simpson, and Donald D. Lucas. Machine Learning Emulation of Spatial Deposition from a Multi-Physics Ensemble of Weather and Atmospheric Transport Models. *Atmosphere*, 12(8):953, July 2021.

[5] Tai-Long He, Nikhil Dadheech, Tammy M. Thompson, and Alexander J. Turner. FootNet v1.0: development of a machine learning emulator of atmospheric transport. *Geoscientific Model Development*, 18(5):1661–1671, March 2025.

[6] Mouhcine Mendil, Sylvain Leirens, Patrick Armand, and Christophe Duchenne. Hazardous atmospheric dispersion in urban areas: A Deep Learning approach for emergency pollution forecast. *Environmental Modelling & Software*, 152:105387, June 2022.

[7] Malte Müller, Mariken Homleid, Karl-Ivar Ivarsson, Morten A. Ø. Køltzow, Magnus Lindskog, Knut Helge Midtbø, Ulf Andrae, Trygve Aspelien, Lars Berggren, Dag Bjørge, Per Dahlgren, Jørn Kristiansen, Roger Randriamampianina, Martin Ridal, and Ole Vignes. AROME-MetCoOp: A Nordic Convective-Scale Operational Weather Prediction Model. *Weather and Forecasting*, 32(2):609–627, April 2017.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].