Super-resolution of 3D fluid simulations using neural networks

Thomas Michel
thomas.michel@epfl.ch
Supervisors: Jiannong Fang and Fernando Porté-Agel
Wind Engeneering and Renewable Energy (WIRE)

December 23, 2025

Contents

1 Introduction

Wind predictions allow to forecast the power generated by wind turbines, helping to know the quantity of energy that will be available and manage its pricing. Predictions are obtained with computational fluid dynamics (CFD) simulations around the wind turbines. However those simulations come with a high computational cost, especially because high resolution is necessary and predictions are runned frequently. The goal of this project is to develop a 3D U-net convolutional neural network (CNN) able to downscale, which consists in obtaining a high resolution simulation from the low resolution version of it. The model is built on the work from Jiang et al. [1], modifying the input from two spatial dimensions and one time dimension to an input of three spatial dimensions. The GitHub repository containing all the code used for this report can be found here.

2 Context and motivation

Super-resolution, also known as downscaling, refers to the process of taking a low-resolution simulation and generating a higher-resolution version of the same simulation. Given a downscaling factor c_d, an input snapshot with 3D resolution (n_x, n_y, n_z) is mapped to an output snapshot with resolution ($c_d \cdot n_x$, $c_d \cdot n_y$, $c_d \cdot n_z$). Importantly, the spatial domain itself does not change, only the resolution of the simulated fields increases. An example of ideal downscaling is shown in Fig.1, on the pressure field.

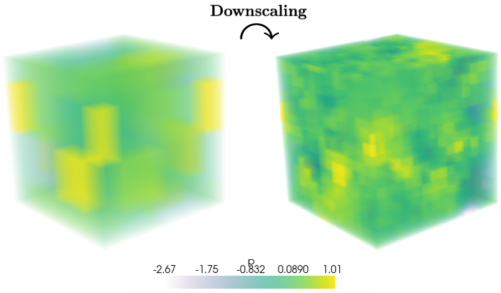


Figure 1: Example of downscaling from low-resolution (4, 4, 4) to high-resolution (16, 16, 16) for the pressure field p.

Multiple super-resolution models exist, the simplest one being interpolation. Interpolation methods will be used as a baseline to evaluate results, but they are not good super-resolution models in the case of turbulent flows as they can't "invent" small scale structures that are important. There also exists other classic approaches such as Proper Orthogonal Decomposition or Spectral Reconstruction, but those methods are also not suited for turbulent flows. Most of the super-resolution models are nowadays using deep learning (DL) in var-

ious forms. The state-of-the-art DL options are either transformers, generative adversarial models or convolutional neural networks (CNN). The later approach is choosed for this project following the work of Jiang et al. [1]. Convolutional networks have proved there capability to extract meaningful features from spatial fields and are particularly effective at modeling local patterns, like eddies. The same filters are used multiple times, which makes them parameter-efficient and capable of recognizing similar turbulent structures everywhere on the domain. Compared to more complex models such as transformers or generative adversarial networks, CNNs offer a good balance of accuracy, stability, and computational cost, making them a well-justified choice.

3 Model

3.1 Dimensions

In the work of Jiang et al. [1], their model is downscaling 2D simulations, which contains two spatial dimensions and one time dimension. In this project the model is adapted to downscale 3D snapshots of simulations, which contains three spatial dimensions but no time dimension. Keeping the same number of dimensions allows relatively small changes in code but permits to test the model on 3D simulations. Note that downscaling three spatial dimensions is more challenging than downscaling two spatial dimensions and one temporal dimension, since the additional spatial dimension contains turbulent structures rather than the temporal coherence typically present along the time axis.
To downscale complete 3D simulations, an easy option would be to apply the model on each snapshots. Moving to 4D downscaling (three spatial dimensions and one time dimension) is more complex and requires a completely new code.

3.2 U-net and Multi-layer Perceptron

The model is made of two neural networks; a U-net that will generate a latent context grid from the simulation snapshot and a Multi-layer Perceptron (MLP) that will generate physical quantities predictions from the latent context grid. U-net is a CNN architecture that was initially develloped for image segmentation. In this project, three-dimensional CNNs are used. A U-net consists of an encoder and a decoder, starting with c_{in} initial channels and outputing c_{out} channels for each point on the grid. For this work, the initial channels are pressure p and the three velocity components u, v and w, giving $c_{in} = 4$. c_{out} is an hyper-parameter that can be adjusted. The encoder and decoder are made of layers containing convolutions, batch normalizations and ReLUs. The encoder contracts the dimensions with max-pooling until reaching a single node with 256 channels. Then the decoder expends back

to the original resolution with nearest neighbor upsampling. Note that the input and output have the same resolution, only the number of channels is changing. The output of this U-net is called *latent context grid* and is expected to contain all the relevant features at each grid point. Then to produce predictions of the four physical variables, the MLP is used. This second neural network takes as input the latent context grid and a position $\mathbf{x}$ and outputs a prediction for the four physical variables at the position $\mathbf{x}$. An unique feature of this model is that the output is continuous, meaning it's possible to query points everywhere in the domain. In this project, the query positions $\mathbf{x}$ will be the high-resolution grid points.

3.3 Data shape

To build the training and evaluation dataset, it requires a high resolution simulation dataset $\mathcal{D}_h$. This dataset has dimensions $(n_t,\ n_x,\ n_y,\ n_z)$, with n_t the number of time-steps and n_i the number of grid-points in axis i, and contains the four physical quantities p, u, v and w. From this high resolution dataset, a low resolution dataset $\mathcal{D}_l$ is produced using linear interpolation. This low resolution dataset $\mathcal{D}_l$ has dimensions $\left(n_t, \frac{n_x}{c_d}, \frac{n_y}{c_d}, \frac{n_z}{c_d}\right)$ with c_d the downscaling factor, and contains the same four physical quantities. Those low-res/high-res pairs of simulation snapshots are used in the supervised learning of the model. The model predicts from the low resolution $l \in \mathcal{D}_l$ the high resolution $\hat{y}$, corresponding to the predicted physical quantities on high-resolution grid. Then ground-truth $y \in \mathcal{D}_h$ is used to evaluate the loss.

3.4 Loss

The particularity of the model from Jiang et al. [1] is that the loss is composed of two terms, the prediction loss $\mathcal{L}_p$ and the equation loss $\mathcal{L}_e$. The prediction loss $\mathcal{L}_p$ for one sample is simply given by:

$$\mathcal{L}_p = \sum_{j \in C} ||y_j - \hat{y}_j||_1 \tag{1}$$

with $C = \{p, u, v, w\}$ the four physical channels, $\hat{y}_j$ the prediction of the model for quantity j on the high-resolution grid, y_j the ground-truth value and $||\cdot||_1$ the L1 Frobenius norm. This prediction loss $\mathcal{L}_p$ is purely the distance between the prediction and the ground-truth. The second loss, the equation loss $\mathcal{L}_e$, corresponds to how much the prediction is satisfying some physical equations. In our case, the only physical equation that can be constrained is the mass conservation as it's not possible to calculate time derivatives. If constant density of the fluid is assumed, this leads to $\nabla \vec{u} = 0$. The corresponding loss is then:

$$\mathcal{L}_e = \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i}. \tag{2}$$

Other physical equations could be constrained depending on the problem. The equation loss term is weighted with a weight factor γ, giving the final expression for the total loss $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_p + \gamma \mathcal{L}_e. \tag{3}$$

3.5 Training

The whole training is done using the PyTorch library and minimizing the loss with Adam optimizer. All hyperparameters, including batch size, number of epochs, learning rate, and others, a tunable before starting the training, and should be adapted to the dataset used. The original code was improved by adding an early-stopping when the evaluation loss is not decreasing for a few epochs. To reduce over-fitting, some regularization and synthetic data were also added. The synthetic data generation is done by imposing random flips to the simulation snapshots, which virtually increases the amount of data available. Finally normalization was corrected and visualization of the model outputs was improved.

4 Results

4.1 3D Isotropic dataset

The dataset used to test and debug the code was found in the Johns Hopkins Turbulence Database [2] (an access is required). The dataset was generated through DNS and has a very high spatial resolution of 1024^3 nodes. The turn-over time is 2 seconds and the total duration of the simulation is 10 seconds. Only a high-resolution basis of 16^3 over the full physical domain was choosed for this project to keep reasonable downloading time. The downscaling factor is set to $c_d = 2$, which means the low-resolution input has shape 8^3. For training set, the time resolution is 400 snapshots (0s to 4s with $\Delta t = 0.01$). For validation set, the time resolution is 40 snapshots (4s to 6s with $\Delta t = 0.05$). The dataset contains the four channels p, u, v and w. The example in Fig.1 is from this dataset. The most important hyperparameters related to the model architecture are `lat_dims`, which defines the number of channels c_{out} in the latent context grid; `unet_nf`, which specifies the number of filters in the convolutional layers of the U-Net; and `imnet_nf`, which determines the number of neurons in the first layer of the MLP. Increasing these values increases the number of trainable parameters, allowing the model to represent more complex features in the data. However, larger models also carry a higher risk of overfitting, particularly when the available training data are limited. The idea is to find the right parameters for a specific resolution, downscaling factor and amount of a data. For simplicity we will always use in this part `lat_dims = unet_nf = imnet_nf = n_f` as this keeps a model structure that is balanced and reduces the notation.

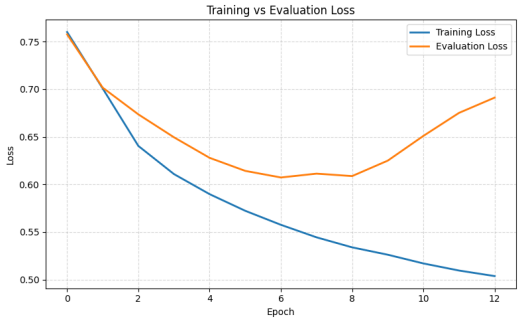
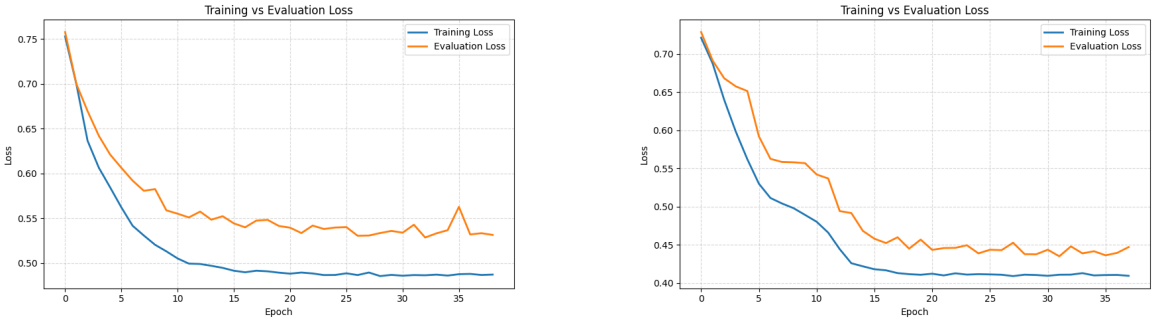


Figure 2: Training and validation loss as the small model is being trained. The model has only $\approx 7'000$ parameters.

4.1.1 Training

Let's start with a model that has $n_f = 2$, which gives 6436 parameters for the U-net and 1064 parameters for the Imnet (MLP). This is a very small model. The training and evaluation loss for each epochs is shown in Fig.2. The main observation is that the model does not perform well on the evaluation data: while the training loss keeps decreasing, the evaluation loss does not. The model is not fed enough data or is under-parametrized, it can't learn patterns that generalize to unseen data. To address this problem, two options are available: training on more data in the hope that the model eventually learns general patterns, or increasing the number of parameters. The first option was tried by downloading twice more data over the same time interval. The results of this second training are shown in Fig.3a.



(a) (b)

Figure 3: Training and evaluation loss for a model with more training data (a) and for a model with more parameters ($\approx 16'000$) (b).

Indeed the evaluation loss decreased and the model seems to have found good patterns with the additional data. The second option was tried by taking $n_f = 3$, which changes the model from $7'000$ parameters to $16'000$ parameters, and the results of training are shown in Fig.3b. This increase in parameters also worked as the evaluation loss decreased. These examples show the importance of choosing the correct hyper-parameters to obtain a stable training.

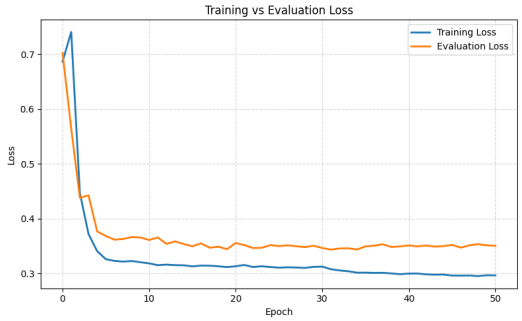


Figure 4: Training and validation loss for a model with $n_f = 100$, with early stopping turned off. The model has more than $\approx 4'100'000$ parameters.

Let's now increase the number of parameters of the model until it reaches full capacity for the amount of data available. The lowest evaluation loss $\mathcal{L}_{eval}$ for each model is consigned in Tab.1.

n_f	# params	$\mathcal{L}_{eval}$
2	7'000	0.61
3	16'000	0.54
4	29'000	0.35
5	44'000	0.34
6	65'000	0.35
30	690'000	0.35
100	4'100'000	0.35

Table 1: Evaluation loss for each model when increasing the number of parameters.

The best model occurs when $n_f = 5$. A model bigger than that seems to have many unused/useless parameters that would cost memory, computational time and lead to overfitting. An example of the training loss for the biggest model ($n_f = 100$) is shown in Fig.4, with early stopping turned off. As observed the model starts to overfit and there is no point to have more parameters if loss is not decreasing. This is why the final choice for this example model is $n_f = 5$, as it offers the best performance for the smallest amount of parameters. Once the model is trained, let's evaluate how well it performs on the validation dataset with examples and other metrics than simply the loss.

4.1.2 Evaluation

The script evaluating a trained model was adapted to produce a report file called `Evaluation_Metrics.txt`. This file contains evaluation metrics such as the mean absolute error (MAE), the kinetic energy error (K), the root-mean-square (RMS) velocity error, and the velocity divergence, all computed with respect to the ground-truth high-resolution fields.

These metrics are reported for both the model predictions and a trilinear interpolation baseline, enabling a direct comparison of performance. The mean absolute error provides a measure of the average deviation of the predictions from the ground truth. The kinetic energy is defined as $K = \frac{1}{2}\overline{\vec{v}^2}$ and the root mean squared velocity as $RMSV = \sqrt{\overline{\vec{v}^2}}$. These two quantities are particularly informative because they characterize the global energy content, allowing an assessment of how well the model preserves physically meaningful flow properties beyond pointwise accuracy. The results for the model with $n_f = 5$ are shown in Tab.2. As

Table 2: Comparison of Model Predictions and Interpolation Baseline

Method	MAE P	MAE U	MAE V	MAE W	Err K (%)	Err RMSV (%)	$\overline{\nabla \cdot \vec{u}}$
Prediction	0.170	0.322	0.330	0.321	49.38	28.86	0.2233
Interpolation	0.191	0.336	0.342	0.332	53.89	32.10	0.2361

observed, the model outperforms only slightly the interpolation method. The improvement in all velocity fields is less than 5%, which seems a bit deceiving. An example of the four predicted channels for one sample is shown in Fig.5. As seen in those examples, the model

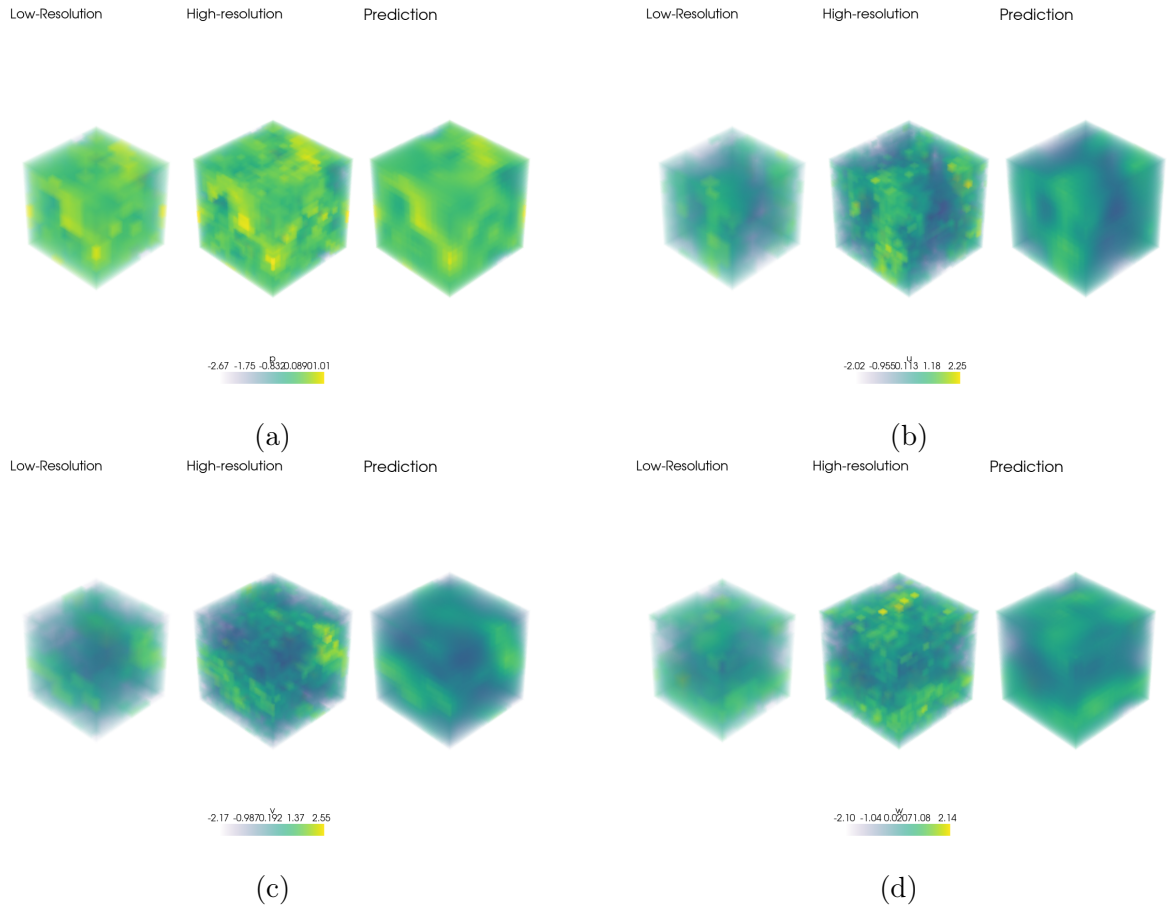


Figure 5: The downscaling model takes as imput the low-resolution snapshots and returns the high-resolution prediction, for the four physical channels. This prediction can then be compared with the ground truth high-resolution.

is more or less doing interpolation. It has not learn any physical patterns. The problem is that the data in itself is not showing turbulent patterns as the resolution 16^3 is too coarse to resolve such patterns. This is assumed to be the reason the model is not performing better. However these results confirm that the overall training and evaluation is working, which is already a good start.
To confirm that the equation loss is working properly aswell, the model was trained with various equation loss weight factor γ. The resulting mean divergences are shown in Tab.3. An increase in γ leads to a lower divergence. These results are reassuring and show that

Table 3: Comparison of models with different weight factor γ.

γ	0.0	0.01	0.05	0.2
$\overline{\nabla \cdot \vec{u}}$	0.27	0.23	0.14	1.3e-06

the equation loss term is efficient to reduce divergence. Note that for $\gamma = 0.2$, the total loss is dominated by this equation loss and thus the model predicts constant fields to ensure zero divergence. This factor γ needs to be set carefully to ensure a good balance between reducing divergence and keeping some signal in the output.

4.2 MHD Dataset

After this test phase on the Isotropic dataset, let's move to another dataset of Magnetohydrodynamics (MHD) simulation from TheWell database [3]. This dataset has time resolution of 600 snapshots (500 training and 100 validation) and spatial resolution 64^3, which is way higher than the previous dataset. The fluid is this time compressible, so we will remove the equation loss term ($\gamma = 0$) in the loss. The training is done in a similar way as for the isotropic dataset. Note that pressure was not available for this dataset, but density ρ was. The pressure channel is thus replaced by the density channel. This shows the capability of the model to adapt easily the training to various physical quantities. The loss curves during training are shown in Fig.6 for a model with total number of parameters of 350'000 and downscaling factor of $c_d = 4$. The loss is greatly reducing before reaching a plateau,

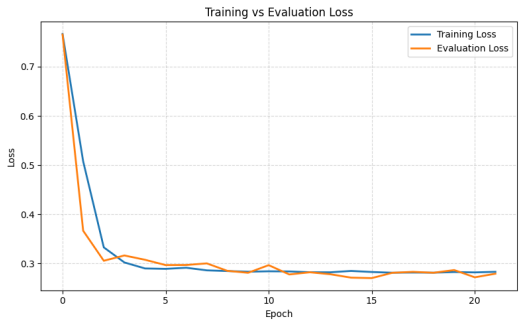


Figure 6: Training and validation loss as the model is being trained on the MHD dataset.

symptoms of a stable training. The evaluation loss is surprisingly close to the training loss, but this is not indicative of overfitting, rather, it reflects the homogeneous nature of the MHD turbulence, for which the model generalizes well across samples. After being trained, the model is evaluated as before. An example of inference on one sample is shown in Fig.7 with the four physical channels (ρ, u, v and w). The evaluation metrics and comparison to

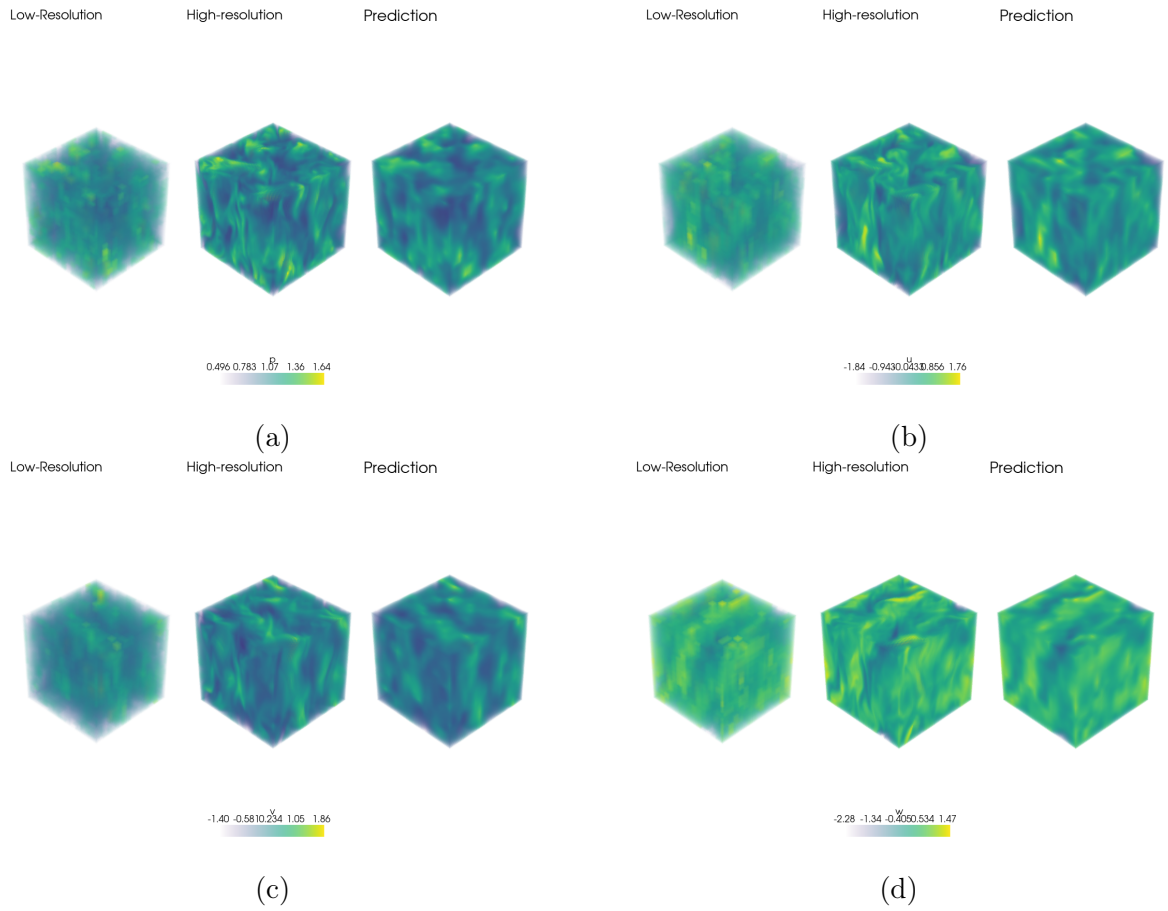


Figure 7: Example of one inference of the model trained on the MHD dataset. The four physical variables are density (a), u (b), v (c) and w (d). Downscaling factor is $c_d = 4$.

the interpolation baseline are shown in Tab. 4.

Table 4: Comparison of Model Predictions and Interpolation Baseline on MHD Dataset

Method	MAE ρ	MAE U	MAE V	MAE W	Err K (%)	Err RMSV (%)	$\overline{\nabla \cdot \vec{u}}$
Prediction	0.0668	0.1569	0.1123	0.1219	21.83	11.59	0.1043
Interpolation	0.0782	0.1957	0.1369	0.1518	35.88	19.93	0.0881

The results on the MHD dataset are significantly better than those obtained on the isotropic dataset. Across all velocity components, the U-Net model achieves a reduction of approximately 20% in MAE compared to the interpolation baseline, and the kinetic energy error is reduced from 35.88% to 21.83%. Similarly, the RMS velocity error decreases from 19.93%

to 11.59%, indicating that the model predicts the overall energy content of the flow field much more accurately than interpolation. These improvements demonstrate that the network successfully learns the multi-scale turbulent structures present in the data, and that it reconstructs physically meaningful high-resolution flow features that are not captured by simple interpolation methods. The predictions give structures that look more like turbulence patterns than simple interpolation. This is a very satisfying result and shows the usefulness of such models.

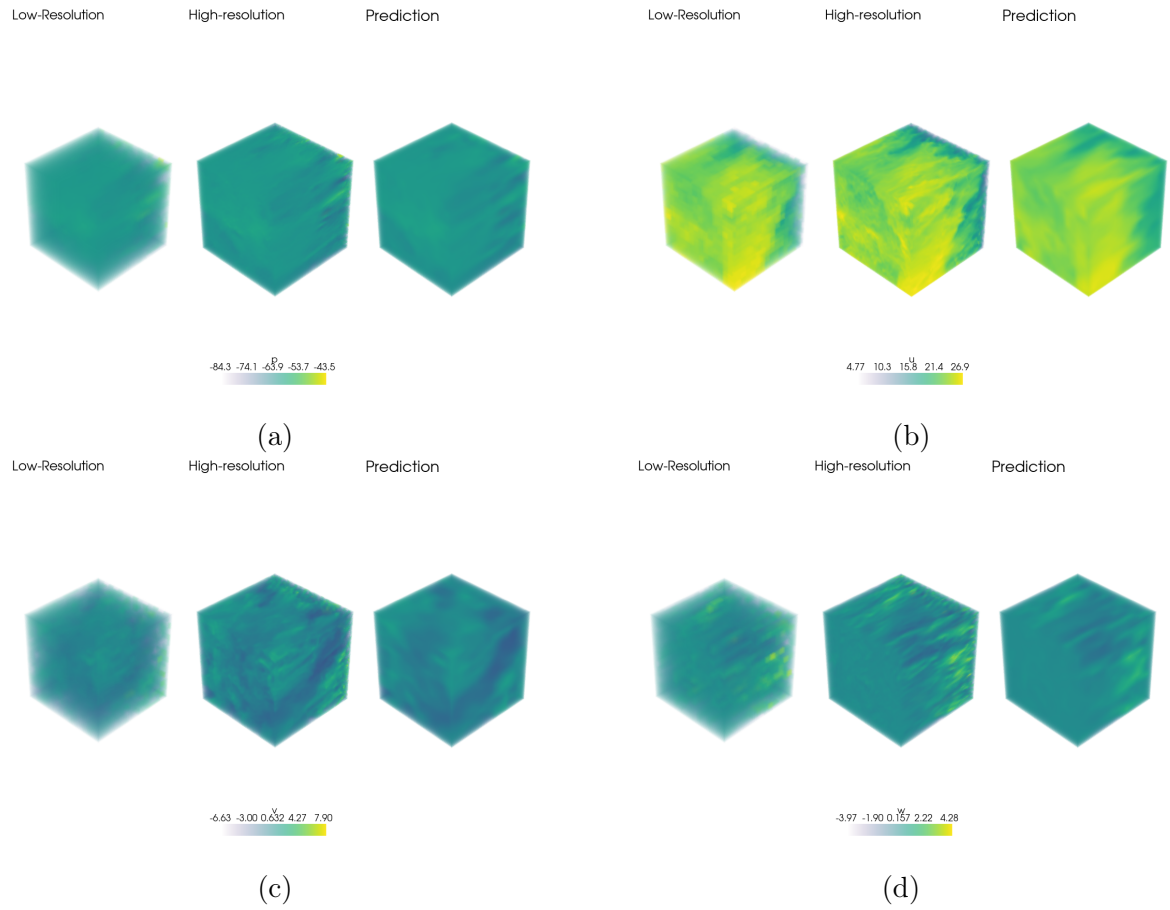


Figure 8: Example of one inference of the model trained on the ABL dataset. The four physical variables are p (a), u (b), v (c) and w (d). Downscaling factor is $c_d = 4$.

4.3 ABL Dataset

Finally a dataset of atmospheric boundary layer (ABL) simulations is used to try the model. This dataset has time resolution of 1000 snapshots (800 training and 200 validation) and spatial resolution (128, 128, 64), which covers a range of (6000m, 6000m, 1000m). The model is trained on 64^3 cubes, which corresponds to multiple sub-domains of the original simulations. The sub-domains are picked randomly in the full domain by the data loader,

which increases drastically the number of samples. Let's observe the evaluation results for a model with downscaling factor $c_d = 4$. An example of inference on one sample is shown in Fig.8 with the four physical channels (p, u, v and w).

The evaluation metrics and comparison to the interpolation baseline are shown in Tab.5. The

Table 5: Comparison of Model Predictions and Interpolation Baseline (ABL Dataset)

Method	MAE P	MAE U	MAE V	MAE W	Err K (%)	Err RMSV (%)	$\overline{\nabla \cdot \vec{u}}$
Prediction	0.6847	0.9246	0.4934	0.3980	1.64	0.81	0.1881
Interpolation	0.7756	0.6839	0.4994	0.4187	1.98	0.99	0.5074

results are a bit deceiving. The model represents better the overall energy of the simulation and the divergence but does way worse than interpolation on the u field. The reason for such poor performances is assumed to be the same as for the isotropic dataset: the data is not showing turbulence patterns as the resolution is too coarse, so the model can't learn general patterns.

5 Discussion

The results demonstrate that the inital U-net architecture can be successfully extended to three spatial dimensions and achieves good perfomances, depending on the dataset. The main factor for good performances is the presence of physical patterns in the dataset that can be learned by the model. The isotropic and ABL datasets, which have rather coarse resolutions compared to the size of eddies, exhibit performances similar with tri-linear interpolation. However the MHD dataset with higher resolution showed an improvement of around 20% on all physical fields compared to interpolation and a way better estimation of the kinetic energy available, demonstrating the effectiveness of this approach.
Examples of over-fitting or under-parametrized models were presented, together with possible mitigation strategies, to highlight the importance of selecting appropriate training data and hyperparameters.
Short-term improvements could focus on the use of more physically relevant evaluation metrics. In particular, metrics related to the representation of turbulent eddies, such as the Kolmogorov length scale, could provide a more informative assessment of model performance. Better synthetic data generation could also help training. Additionally, a workflow that applies the model sequentially to each snapshot of a three-dimensional simulation to reconstruct a higher-resolution time series could be developed. A longer-term improvement would be to apply a model directly on complete three-dimensional simulations as input. This would require to use 4D U-net, which is not easy to implement, or incorporate the time dimension in another way.

6 Conclusion

This work demonstrates that deep learning–based downscaling using a 3D U-Net is feasible for turbulent flow reconstruction, provided that the training data contain sufficiently rich and physically meaningful structures. While the approach shows limited benefits on coarse-resolution datasets, it significantly outperforms trilinear interpolation on high-resolution turbulent data, particularly in terms of energy-related metrics. Several directions for further improvement are identified to better exploit the full potential of the approach, with a key next step being the application of the model to complete three-dimensional simulations in order to capture both spatial coherence and temporal consistency.

References

[1] Jiang, Chiyu Max, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, et al. 2020. «Mesh-freeFlowNet: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework». arXiv:2005.01463.

[2] 'Isotropic Dataset - JHTDB'. n.d. Accessed 6 December 2025. https://turbulence.idies.jhu.edu/datasets/homogeneousTurbulence/isotropic.

[3] The Well. (n.d.). Retrieved 12 December 2025, https://polymathic-ai.org/the_well/