Using a convolutional neural network to super-resolve airflow around a building

Vargiemezis, T.
Stanford
tvarg@stanford.edu

Bachand, N.
Stanford
nbachand@stanford.edu

1. Abstract

While small-scale structures can have a large effect on the full flow field and important engineering quantities of interest, they are difficult to resolve in fluid simulations. Increasing spatial resolution makes numerically solving the underlying Navier-Stokes equation increasingly expensive. This, coupled with the success of recent super-resolution techniques for natural-images, has sparked interest in applying deep neural network to super-resolve flow fields from lower resolution data. However, many techniques balance the expressivity of state-of-the-art super-resolution networks with limitations to enforce physical constraints or curtail unexplainable results. To investigate this trade-off, we compare the enhanced deep super-resolution network (EDSR) [13], developed for natural-image super-resolution, with the hybrid downsampled skip-connection/multi-scale (DSC/MS) model, built specifically for super-resolving flow fields. We also explore the effect of pretraining the EDSR model on natural images. The pre-trained model gives the best performance, while the EDSR model in general performs better than the base DSC/MS. While the EDSR model is able to reconstruct test data omitted from the training time-series, all models perform much worse with test data taken as the simulation advances in time away from the time steps used for training

2. Introduction

Designers often rely on computational fluid dynamics (CFD) simulations to understand airflow through the urban canopy. CFD simulations can predict pedestrian comfort and safety, urban heat island effects, and wind loading on buildings. However, deploying CFD simulations at a large scale and early in the design phase has challenges. To correctly resolve turbulence, CFD simulations must model all three spatial dimensions and the temporal dimension. Because most CFD schemes are first order accurate in space and time, halving the error requires approximately sixteen times more computational work [6]. This scaling makes it increasingly expensive to resolve peak quantities (e.g., wind speed and pressure), which are often important design constraints.

Engineering applications require interpretable and predictable models [2] because poor predictions can have severe ramifications. Super-resolution methods partially address these concerns by leveraging physics-based solvers to calculate coarse quantities before refining with data-driven methods. While some work has leveraged the underlying physics constraints to perform super-resolutution [9], it remains difficult to enforce physics constraints within deep learning techniques [2]. Even if physics constraints could be enforced, the subgrid physics cannot be fully represented. To capture subgrid phenomena, most CFD methods rely on subgrid models to capture physics occurring below the temporal and spatial resolution of the numerical stencil. Because filtering out subgrid processes from the underlying Navier-Stokes equations gives no closed-form solution, these subgrid models are inherently approximate. The main idea behind the subgrid models is that the smallest scales are more universal while larger scales depend on geometry and the boundary conditions. Thus, the behavior of the smallest scales can be described by simple models without explicitely resolving all the scales in the flow. These approximations in CFD solvers motivate the idea that data-driven super resolution could perform well at super-resolving fluid flows. Past studies have found that combining physics-based and data-driven methods can outperform either method alone [2]. Recently, Kochkova et al. showed that purely data-driven super-resolution methods outperformed methods that enforced existing turbulence models as the underlying basis.

CFD simulation presents both computational and accuracy difficulties when resolving small-scale fluid phenomena. Speeding up the turnover time of fluid simulations for urban planning and building design could help introduce these tools earlier in the design process. Additionally, ambiguity in subgrid fluid physics indicates that borrowing data-driven super-resolution methods from image processing may be a good way to super-resolve flow fields.

In this project, we will attempt to super resolve the Reynolds normal stresses from low resolution images. All images will be 2D snapshots of the flow field at the build-

ing's plane of symmetry, with the mean flow moving from left to right. The deep learning model will apply to a single simulation, learning from high fidelity data from the first 2000 time steps and super-resolving low fidelity simulation data for unseen timesteps. We perform two experiments:

1. From the first 1800 steps we select randomly 200 intermediate steps as the test set, and the rest 1440+160 as training and validation sets

2. We keep the trained model from experiment 1 but we change the test set with the last 200 steps we didn't use.

By following these two approaches we can test how the model performs when it predicts intermediate or future unseen steps.

While we would like to apply this method to upscale images directly from low-fidelity simulations, synchronizing low fidelity and high fidelity simulations is challenging due to the chaotic nature of turbulence. To focus on the vision component of the work, we will create low-resolution input images from 4x down-sampled high-resolution data. Our methods will then upscale the low-resolution images 4x (to the original resolution), attempting to reproduce the original high fidelity data. Considering the universality of smaller scales in a flow, we hope that "filtering out" the smaller scales by downsampling and then super-resolving to predict those again, will have the same effect as a combination of a coarse and fine simulations.

The remainder of the report is structures as follows: Section 3 mentioned related work, section 4 describes the selected method, section 5 shows the preprocessing of the dataset, section 6 presents the results, and finally, section 7 concludes with the future work.

3. Related Work

Successful super resolution methods for natural image processing have inspired a growing body of work applying deep learning methods to super resolve fluid flows [3]. Most of this literature focuses on testing the merit of data-driven super resolution techniques for fluid dynamics. Therefore, most methods attempt to recreate high fidelity simulation data from synthetically down-sampled input images. However, there is not a consensus approach to super resolution in fluid dynamics. Different approaches often stem from conflicting assumptions about how known physics should integrate with data-driven approaches.

Erichson et al. apply a neural network to recreate a high fidelity flow field from limited sensor inputs [1]. They replicate sensor measurements by sampling the high fidelity fluid data at a limited number of points. Next, a two-layer fully connected network attempts to recreate the high dimensional flow field. They apply the model to multiple fluid dynamics simulations, but specifically re-train on each simulation. This method recreates the high dimensional flow field well, and significantly outperforms PCA. The authors favor a shallow network because of its greater interpretability, ease of use, and success. The fully connected layers eliminate the need for gridded sensor measurements and do not apply a cartesian basis. While this work does not explicitly enforce known physics, it curtails uninterpretable inference methods by limiting the model's expressivity . However, these potential benefits are only realized by foregoing most components of state-of-the-art super-resolution networks.

Most other work uses convolutional neural networks because of their popularity in natural images super-resolution [3]. Fukami et al. design a convolutional neural network meant to super-resolve turbulent flows by taking advantage of the similarities between larger and smaller scales [8]. A series of downsampling and upsampling, separated by two convolutional layers each and using skip connections, worked to capture these different scales in a setup called Down. The model combines the output from these layers with that from a series of parallel convolutional layers with different kernel sizes. Figure 2 gives a schematic of this architecture, referred to as the hybrid downsampled skip-connection/multi-scale (DSC/MS) model. We chose the DSC/MS model as a reference for comparing our results because it is a fully convolutional network, without imposed physics, and specifically designed to super-resolve turbulence. However, it is not based on state-of-the-art networks for natural image super-resolution, and instead relies heavily on assumptions about the structure of turbulent flow. It also uses large kernel sizes, which leads to more trainable parameters than multiples smaller-kernel convolutional layers in series. These decision to break from state-of-the-art super resolution network design in favor of an application-specific architecture make this networks as a good point of comparison for convolutional networks designed only for natural-image super-resolution.

Other work, inspired by the correlation between the flow field at different times, combines spatial and temporal information for super resolution. Recently, Fukami et al. combined the DSC/MS model with a temporal network to interpolate the flow between images of the flow from two different times [**?**]. Liu et al. take a different approach and obtain increased performance by including neighboring low resolution frames as inputs for their network. [14]. While this can draw on important information from other frames, the approach does not leverage state-of-the-art methods in video processing and video super-resolution like 3D convolution. The networks also attempts to replicate forward, backward, and central differencing by feeding two input frames (corresponding to the inputs to the respective numerical scheme), into three parallel convolutional blocks. This

setup is therefore redundant and presumes the network will mimic known numerical methods. Because most attractive applications for super-resolution in fluid dynamics involve limited high resolution data (and therefore merit super-resolution of low resolution data), unsupervised learning techniques could be very useful. Deng et.al apply a generative adversarial network (GAN) model, originally designed for super-resolving natural images, to fluid flow fields [4]. Gao et al. take another approach and train a convolutional super-resolution network with a physics-based loss function instead of high-resolution training labels, alleviating the need for any high resolution data [9]. Other recent work on physics-informed neural networks (PINNs) has used a loss function that includes both a training loss with respect to the data and the quality of the solution to a differential equation. To evaluate the differential equation, the network backpropogates with respect to the input data to compute first derivatives [17]. Recently, Eivazi et al. extended this technique to super-resolve flow fields [5]. In a recently submitted paper, Obiolis-Sales extend the PINN framework to multiscale super-resolution, where a seperate network learns which portions of the flow field should be super-resolved to better represent the important flow features [15]. This technique derives inspiration from adaptive mesh refinement, where a numerical PDE solution may refine the mesh at selective locations to better resolve gradients. While these methods are promising and concerns such as interpretability, overfitting, and learning unwanted numerical artifacts, Kochkov et al. found that enforcing a known solution basis can detract from the performance of super-resolution networks [10].

While many of the networks are creative, they often offer solutions to known problems in natural image processing without considering the state-of-the-art in that field. In building the models around known physics, the models sometimes forgo better expressively or different, and potentially better, training paths. The vast majority of the networks listed above are much shallower than the natural-image super resolution network, the enhanced deep super-resolution network (EDSR) [13], that won the NTIRE2017 Super-Resolution Challenge. The lack of deeper models seems to partially result from only including layers who's intended function cannot be easily interpreted in terms of known characteristics of fluid flows.

4. Methods

We are primarily interested in the performance of the EDSR super resolution models, developed for natural-image super-resolution, on super resolving fluid flow fields around a building. We investigate the performance of this model with and without training on natural images. We compare the full EDSR against a base model, the DSC/MS architecture, designed specifically for super-resolving turbulent flows. We also include two smaller versions of the EDSR model for a more complete comparison against the DSC/MS model. Table 1 summarizes the differences between these five models.

Model	Parameters	ResBlocks	Features
full EDSR	43 mil	32	256
small EDSR	212k	5	35
tiny EDSR	124k	1	35
DSC/MS	216k	NA	$\leq$35

Table 1. Network size for the three EDSR models and the DSC/MS model. The number of features refers to the number of hidden layers per convolutional layer. For the DSC/MS model, the number of hidden layers varies across the network

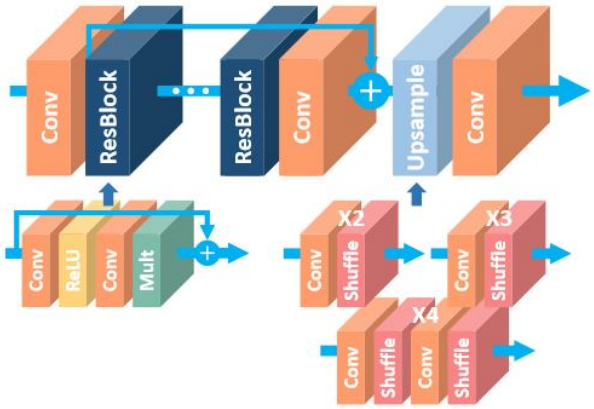


Figure 1. EDSR model architecture. Figure replicated from Lim et al. [13]

Figure 1 shows the EDSR model architecture. The ResBlocks are based on the original ResNet architecture, but without the batch normalization layers. Lim et al. found that this change significantly improved the performance of the network because batch normalization limits the range of features the network can express. This is likely because the batches are fairly small, and there are numerous small-scale features in a dataset that a super-resolution network needs to learn to resolve. Therefore, the batches are likely not representative of the entire training set, therefore limiting performance. After the series of Resblocks, we apply the 4x upscaling layer consisting of two convolutional layers and two shuffle layers. The number of hidden layers is the same across all convolutional layers, all of which use a 3x3 kernel. Table 1 gives the number of hidden layers and ResBlocks for each of our three EDSR model configurations. To implement this model, we started with the PyTorch [16] implementation provided by Lim et al [12]. We also used the provided pre-training weights in our pre-trained model. Our full model used their exact implementation with speci-

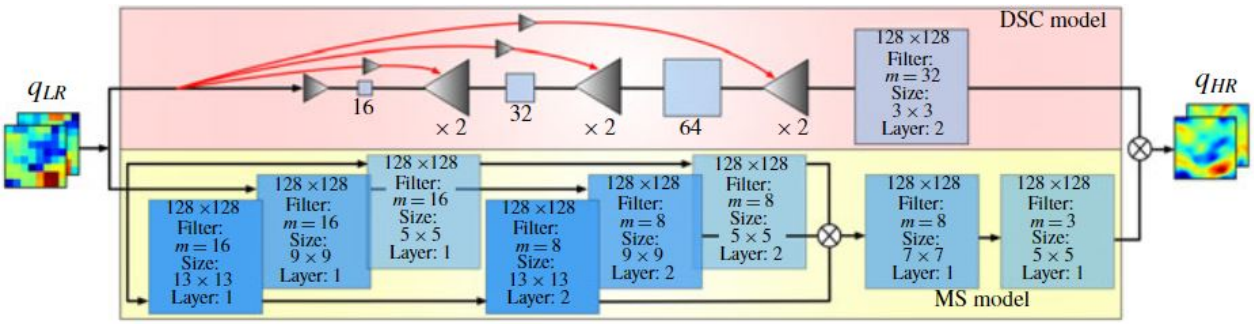


Figure 2. DSC/MS model architecture. Figure replicated from Fukami et al. [8]

fications for our dataset. We then modified the hyperparameters to create the small and tiny variations.

Figure 2 shows the architecture of the base, DSC/MS, model. As described in section 3, this model combines two methods. First, it dowsamples the initial image by 8, applies three convolutions, and then upscales by 2. At this step, the output is combined with a 4x down sampled version of the original image, and the same pattern repeats until the output is the same resolution as the image. These skipped connections and periodic downsampling attempts to provide spatial equivariance and combine information from different turbulent scales. The second method uses a series of parallel convolutional layers with different kernel sizes. The architecture then combines the output of both methods. We implemented this model into the same PyTorch framework used for the EDSR model. Fukami et al. also provided a Keras implementation [7] that we referenced for our PyTorch implementation. However, the different structure of PyTorch made this implementation non-trivial. Lastly, the original DSC/MS model functions as an autoencoder with the input low resolution images upscaled (using nearest neighbor) beforehand. To limit redundant information, we instead inputted low resolution images to the network and added the EDSR upscaling layers to the end of the network.

For training all models, we used L1 loss between the high resolution data and the super-resolved model outputs. To evaluate the test data, we instead use a more physically meaningful metric: turbulent kinetic energy (TKE). We can define the average TKE as:

$$k = \frac{1}{2}\left(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}\right) \tag{1}$$

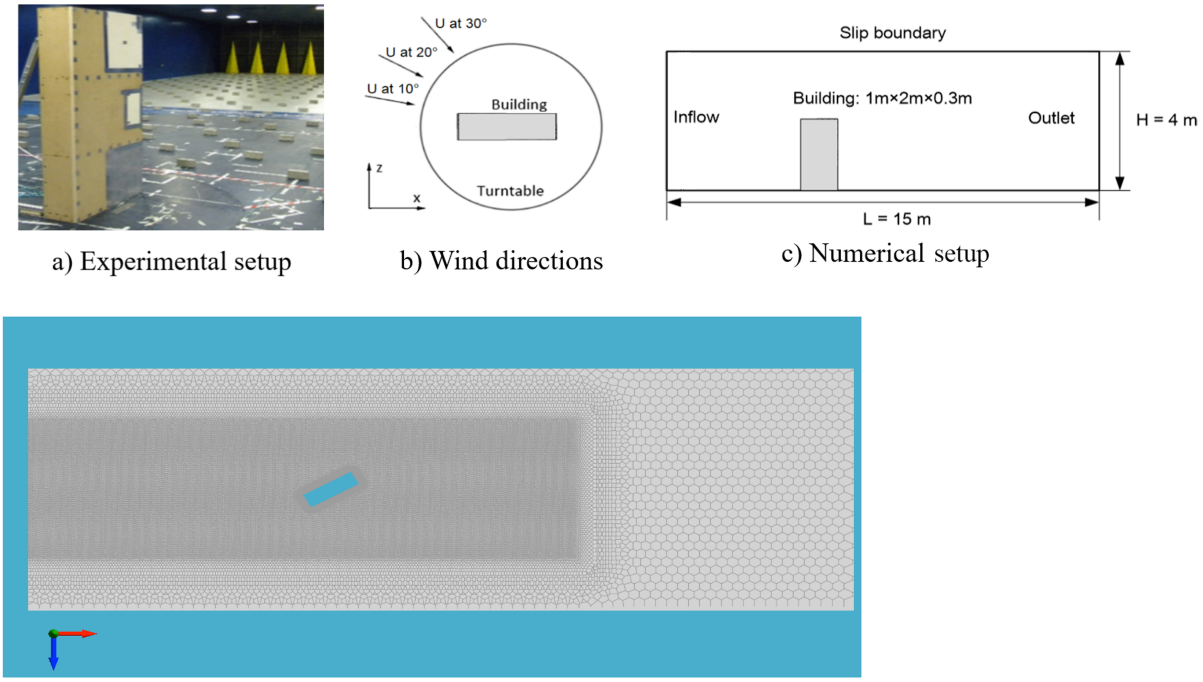
where $\overline{(\cdot)}$ represents the time average, which can be taken by averaging over all the saved timesteps. This can also be interpreted as the average error over all the reconstructed images, instead of just comparing the HR and reconstructed images for every timestep. Correct reconstruction of TKE indicates that we conserve the energy when we pass through the networks.

5. Dataset

The computational set-up is designed to reproduce the experimental set-up in the ABL wind tunnel of the Politecnico di Milano [11], shown in Figure 3(a). The convention for the definition of the wind direction, and details of the computational domain and mesh are shown in Figure 3(b)-(d). The building geometry measures 1 m long × 2 m high × 0.3 m wide and is placed in a rectangular box shaped domain. We are interested in the velocity field around the high-rise building. For this project, we focus on the mid-plane, and we capture the instantaneous Reynolds normal stresses, namely $u'u'$, $v'v'$, and $w'w'$, that correspond to the streamwise, vertical and spanwise directions, respectively.

Our dataset consists of snapshots of the three Reynolds stresses every 4 ms, and each image is in grayscale. The pre-processing of the data includes: 1) cropping and 2) concatenate the 3 components into one RGB image.

The image cropping helps in reducing the size of the image and focus only in the area of interest, which is the flow around the building. This is where we expect to observe most of the smaller flow scales that are difficult to retrieve from low-resolution simulations/images. The concatenation step helps in combining all the information into a single RGB image. The grayscale of the original image ensures that we can capture the magnitude of the Reynolds stresses for each component with only one RGB value, since it's the same number in all 3 channels. This means that we can uniquely define each component with only one channel, and we concatenate them so that we include all the information into one RGB image. Figure 4 shows the cropped images of the instantaneous fields of the three Reynolds stresses, and the concatenation of the 3 components into one RGB image. The procedure of cropping and concatenating is repeated for every saved time step, which eventually creates our final data set.



a) Experimental setup b) Wind directions c) Numerical setup

d) Mesh overview

Figure 3. Experimental and numerical setups.

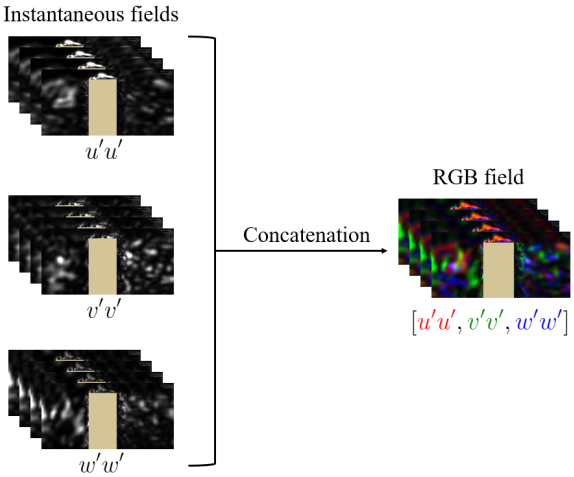


Figure 4. Data pre-processing

6. Results & Discussion

6.1. Experiment

For both all EDSR models and the DSM/MS model, we used the same hyperparameters used in the originally published EDSR model [13]. Because these seemed to work well, and the model is expensive and time-consuming to train, we did not search for better parameters.

We used the ADAM optimizer with beta parameters of 0.9 and 0.999, respectively. For the numerical stability parameter epsilon, we used 1e-8. Our mini batch size was 16. While the DSC/MS model originally used a minibath size of 100, we did not have enough memory to implement this. Therefore, we used 16 to be consistent with the EDSR model. We did not use training folds because we had numerous training and validation data, particularly since the training only occurs on a 48 by 48 pixel patch (which is approximately 2% of the total low-resolution image).

6.2. Results

For all five models trained, figure 5 shows the training loss and figure 6 shows the PSNR on the validation images. All the models show decreasing loss and increasing performance on the validation set throughout the 300 epochs, although most of them plateau. Initially this indicates that the models did not overfit the data, although some of our later results do suggest overfitting.

Comparing the EDSR models with and without pretrained weights, the loss function of the pretrained model begins much lower, and remains lower throughout all 300 epochs. There is a similar pattern for the PSNR on the validation dataset, where the pretrained model continues to outperform the EDSR model trained from scratch across all

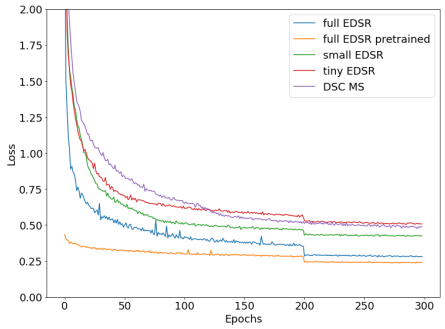


Figure 5. L1 training loss

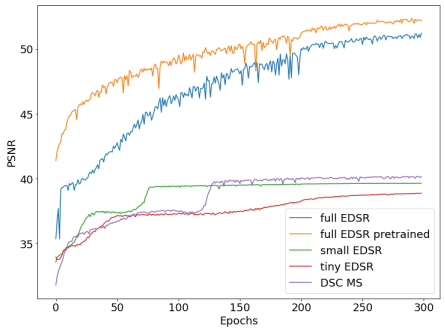


Figure 6. PSNR for validation data

300 epochs. Although we expected the pretrained EDSR model and non-pretrained EDSR models to eventually converge, the pretraining continues to give better performance throughout the training. Finding that pretraining on natural images can help super-resolution models resolve fluid flow fields is particularly striking, given the hesitancy of other work to rely on architectures designed for natural images (see 3).

Comparing the training behavior of the small EDSR model and the DSC/MS model shows that the DSC/MS model is slower to learn, but does reach a slightly better PSNR on the validation data. While this indicates that a model architecture built for fluid flows can compete with a successful natural image architecture, further tuning hyperparameters for the small EDSR model could make up this difference. Lastly, while the small EDSR and DSC/MS models have a similar number of parameters, the small EDSR model trained in less than 20% of the time. This is likely because the complex connectivity and multiple downsampling and upsampling layers slow down the forward and backward pass. Therefore, based on training time, the small EDSR model is a much more efficient.

Training the tiny, small and full EDSR models indicates that a deeper model is able to better learn important information for super resolving fluid flow fields. This indicates that, contrary to some assumption in other work, deeper networks have the potential to significantly improve performance.

Figure 7 shows the average TKE comparison between the HR images and the model reconstruction for experiment 1. The assessment is done in terms of contour plots and pixel-to-pixel comparison in scatter plots. We focus on the roof of the building, which is where high average TKE is observed. Figure 8 shows the same plots for experiment 2. Finally, table 2 shows the root-mean-square error (RMSE) between the HR TKE and the model TKE.

6.3. Discussion

The bicubic method completely fails to reconstruct correctly the flow field around the edges of the building. It merges the building with the flow field, and smooths out the contour lines away from the building. Regarding the scatter plot, ideally, all the points should follow the red $x = y$ line. Many points deviate from that straight line, which agrees with the large RMSE. The straight lines of black spots in the scatter plot correspond to the edges of the building that are not captured correctly.

The base DSC/MS method does a better job, qualitatively, at the edges of the building. However, there are still regions where the flow field "penetrates" the building. Even though the RMSE of this model is higher than the bicubic interpolation, we can see that it is trying to reconstruct the discontinuities, rather than smoothing them out. This is an indication that it is going towards the right path.

The tiny EDSR model also shows penetration of the fluid in the building surface, especially at the top edge. In addition, the smoothing effect is still visible at the contour lines, but RMSE is much lower and the scatter plot is less spread out compared to the previous cases. This is a good indication that making the model bigger might help.

The small EDSR model shows the same behavior with the tiny EDSR, but less pronounced. Some penetration is observed at the top edge, but the left edge is well reconstructed. This results in slightly better RMSE and less spread in the scatter plot.

The scratch EDSR model is a significant improvement compared to all the previous cases. The edges of the building are reconstructed correctly, and no penetration is observed. We can finally see that the light blue line/color of $k \sim 4$ touches the upper surface of the building, and the discontinuities are almost identical to the HR case. The RMSE is an order of magnitude smaller compared to the other cases and this can be seen from the scatter plot, where most of the points follow the straight line.

The pretrained EDSR shows the same behavior as the scratch EDSR. The edges and the TKE structure is identical to the HR case. The model is further, which is indicated by the lower RMSE.

Experiment 2 does not give as good results as experi-

	Bicubic	DSC/MS	Tiny	Small	Scratch	Pretrained
Exp.1	0.3228	0.3646	0.1764	0.1624	0.0228	0.0151
Exp.2	0.3108	0.3829	0.4346	0.3461	0.3403	0.3321

Table 2. RMSE based on TKE between HR and all the models.

ment 1. Both Bicubic and DSC/MS method have the same behavior with experiment 1; penetrating fluid and smooth contours. Tiny and small EDSR models fail to reconstruct correctly all the sides of the building. In experiment 1 we could see that only the top side shows problems. This is the main reason of the higher RMSE by 2-3 times, compared to the previous experiment. Finally, the scratch and pretrained EDSR models also fail to reconstruct the edges of the building, something that was not observed in experiment 1. However, the contour plots of the average TKE is reconstructed correctly. The discrepancies at the edges of the building results in RMSE of an order of magnitude higher than experiment 1.

Based on these two experiments, the first performs better because we are reconstructing images in intermediate steps. Considering that the timestep of the simulation is only 4ms, subsequent timesteps are strongly correlated, and thus reconstructing them can be an relatively easy task since the images are pretty similar as $dt \rightarrow 0$. On the other hand, when the tail of the data is used, the model is tested to completely unseen images or flow patterns around the building. Thus, it's difficult to distinguish between the "new" flow structures, when those hit the building, and the building surfaces.

To better understand why the experiment 2 fails, we can also see the pixel-wise difference of average and instantaneous TKE for both experiments in figure 9, for the pretrained model.

On average, we can see that experiment 2 is as good as experiment 1 at the top of the building, but fails away from it. The structures close to the building are determined by the building geometry and thus, doesn't depend (on average) on the selected time series. In contrast, away from it, the flow structures depend on the incoming flow, and in future timesteps they can be different from what the model has seen. Regarding the instantaneous difference, we can see that, overall, experiment 1 gives lower errors, as we already saw from the previous results.

7. Conclusion & Future Work

To investigate the potential of deep neural networks to super-resolve flow fields from low resolution data, we implemented the EDSR and DSC/MS models. The EDSR model was originally created for natural images while the DSC/MS model was specifically developed for turbulent flows. We found that the EDSR model outperformed the DSC/MS model, and was even preferable when scaled down to less trainable parameters. Our best network used pretrained weights from training on natural images and a much deeper network than the majority of other models used for super-resolving fluid flows. These results further highlight the need for fluid dynamics super-resolution efforts to draw from state-of-the-art work on natural image super-resolution.

While our models did not show overfitting within the time series where training data was interspersed with testing data, they performed poorly on subsequent images taken as the model advanced in time away from the training data. This underscores the limitations of this super-resolution approach, and inspires future work that could help bring this method into engineering design practice. This work could include combined super-resolution and LSTM models, where the model could leverage temporal patterns to predict further out from the training dataset. Future work should also investigate transformer models, where attention to different parts of the image could learn to attend to similarities across images.

8. Contributions

Vargiemezis ran the fluid dynamics simulation and preprocessesed simulation data. Vargiemezis post processed test results, creating all TKE plots.

Bachand modified the EDSR code base for the given dataset and adjusted hyperparameters to create the small and tiny EDSR models. Bachand also implemented the DSC/MS in PyTorch and included it in the EDSR model framework. Bachand created L1 loss and PSNR validation plots.

Both wrote the report, conducted the literature review, thought through the project, and trained models.

Links to the code basis are in the references for PyTorch [16], EDSR [12], and DSC/MS [7].

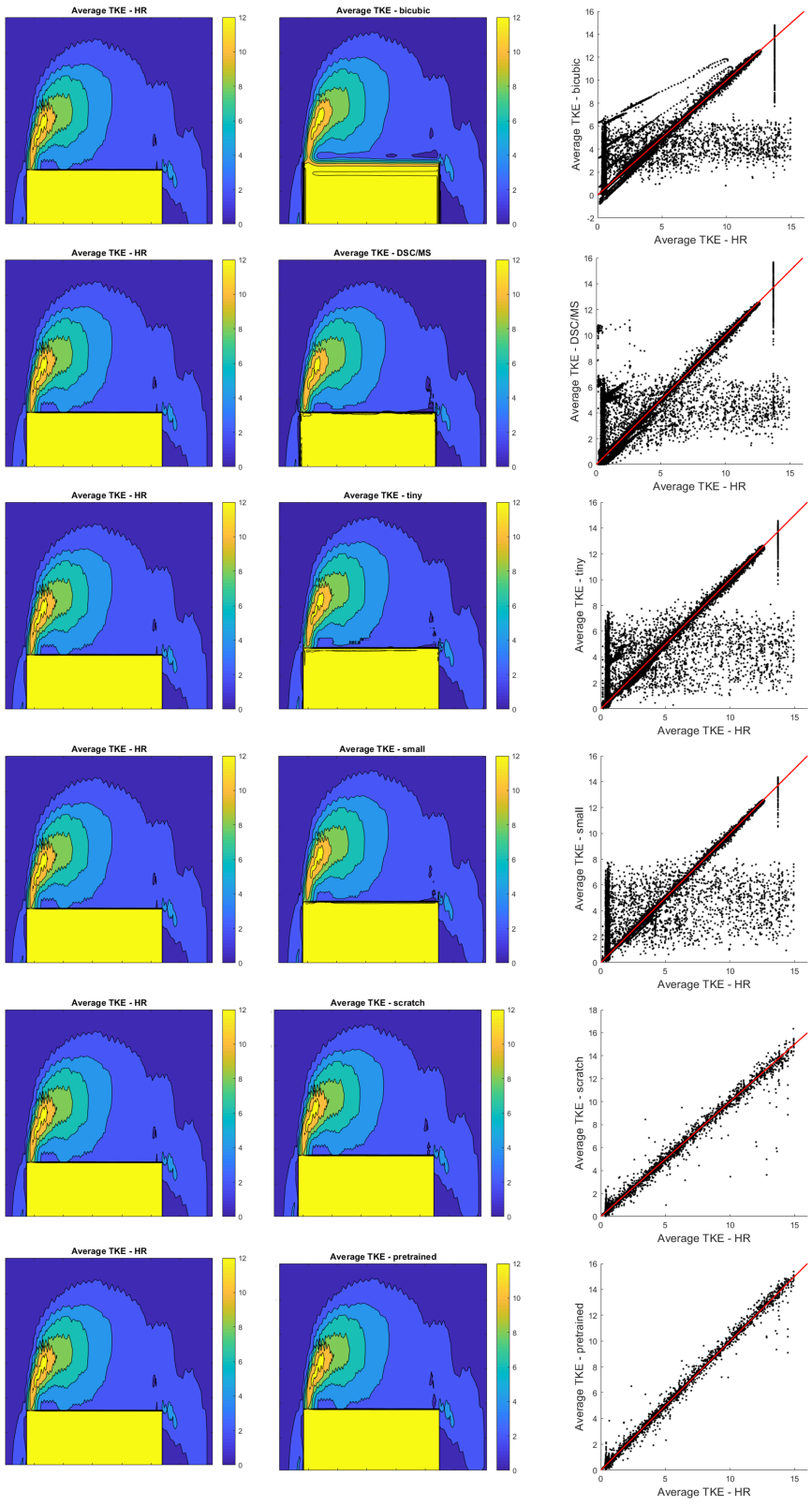


Figure 7. Contour plots and scatter plots comparison for intermediate random samples

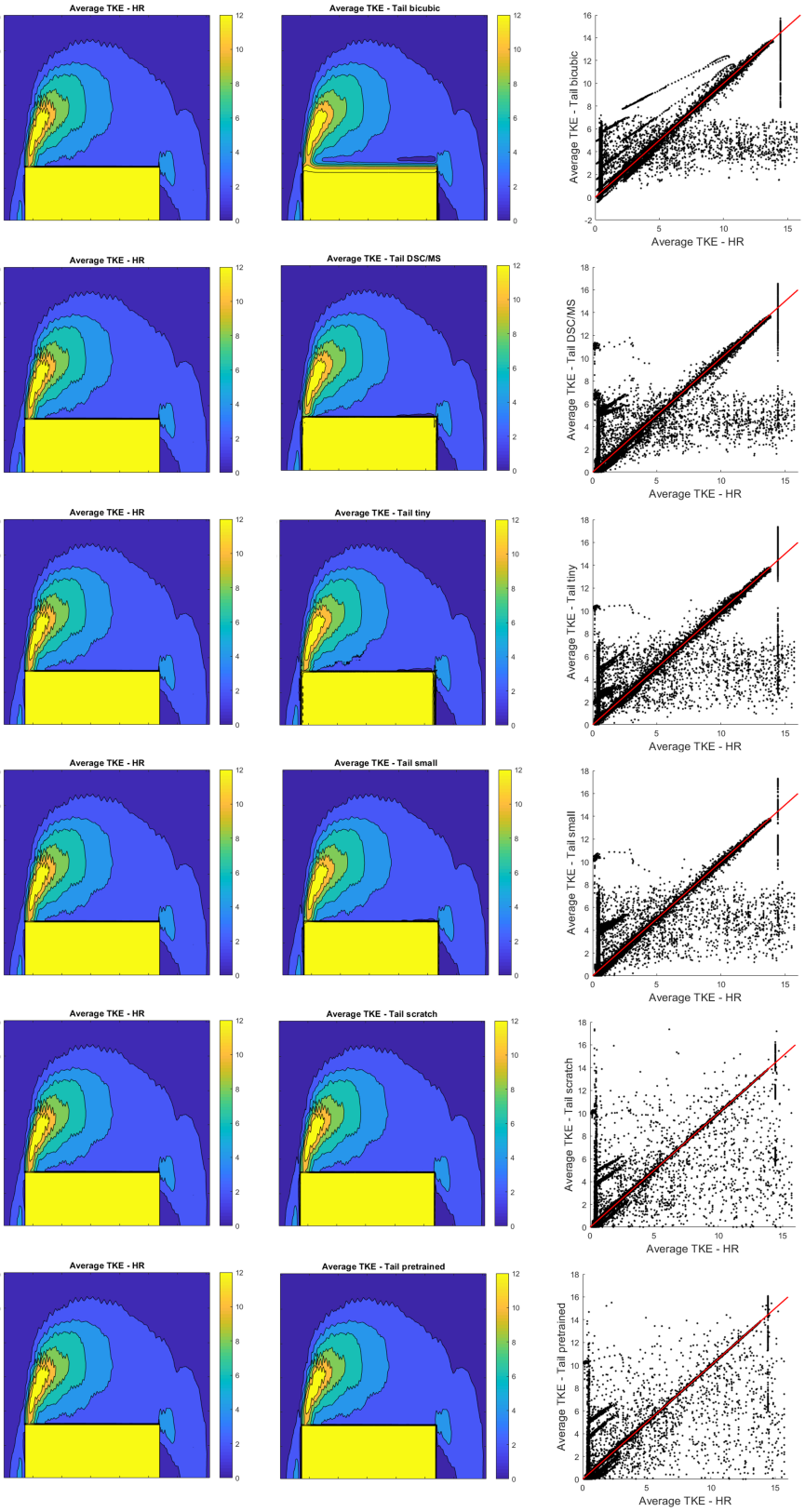


Figure 8. Contour plots and scatter plots comparison for tail samples

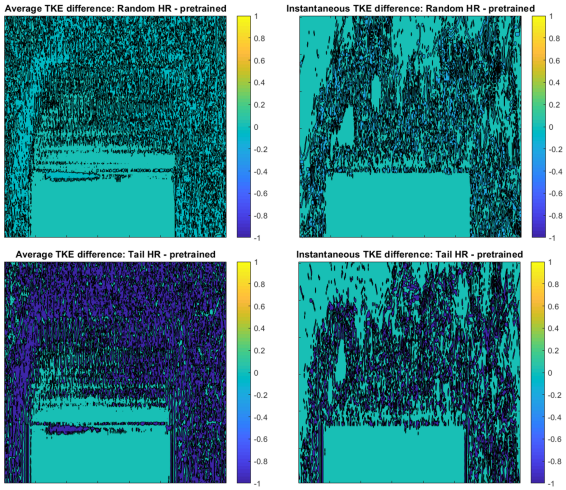


Figure 9. Pixel-wise difference of average and instantaneous TKE for experiments 1 and 2 using the pretrained model

References

[1] Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476, 2020. 2

[2] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech. 2020*, 52:477–508, 2019. 1

[3] Giovanni Calzolari and Wei Liu. Deep learning to replace, improve, or aid cfd analysis in built environment applications: A review. *Building and Environment*, 206, 12 2021. 2

[4] Zhiwen Deng, Chuangxin He, and Yingzheng Liu. Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. *Phys. Fluids*, 2019. 3

[5] Hamidreza Eivazi and Ricardo Vinuesa. Physics-informed deep-learning applications to experimental fluid mechanics. 2022. 3

[6] Ron Fedkiw. Lecture notes in continuous mathematical methods with an emphasis on machine learning, January 2022. 1

[7] Kai Fukami;. $\texttt{hdsc}_m s$. 2018. 4, 7

[8] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 7 2019. 2, 4

[9] Han Gao, Luning Sun, and Jian-Xun Wang. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, 33(7):073603, jul 2021. 1, 3

[10] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, Stephan Hoyer, Andrea L Bertozzi, S H Designed, S H Performed, and S H Analyzed Data;. Machine learning-accelerated computational fluid dynamics. 2022. 3

[11] G. Lamberti, L. Amerio, G. Pomaranzi, A. Zasso, and C. Gorlé. Comparison of high resolution pressure measurements on a high-rise building in a closed and open-section wind tunnel. *Journal of Wind Engineering and Industrial Aerodynamics*, 204:104247, 2020. 4

[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Edsr-pytorch. https://github.com/sanghyun-son/EDSR-PyTorch, 2017. 3, 7

[13] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. pages 1132–1140, 07 2017. 1, 3, 5

[14] Bo Liu, Jiupeng Tang, Haibo Huang, and Xi Yun Lu. Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, 32, 2 2020. 2

[15] Octavi Obiols-Sales, Abhinav Vishnu, Nicholas Malaya, and Aparna Chandramowlishwaran. Nunet: Deep learning for non-uniform super-resolution of turbulent flows. 2022. 3

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 3, 7

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2 2019. 3