Deconvolution of turbulent flow using Convolutional Neural Networks

Olga Doronina
Mechanical Engineering

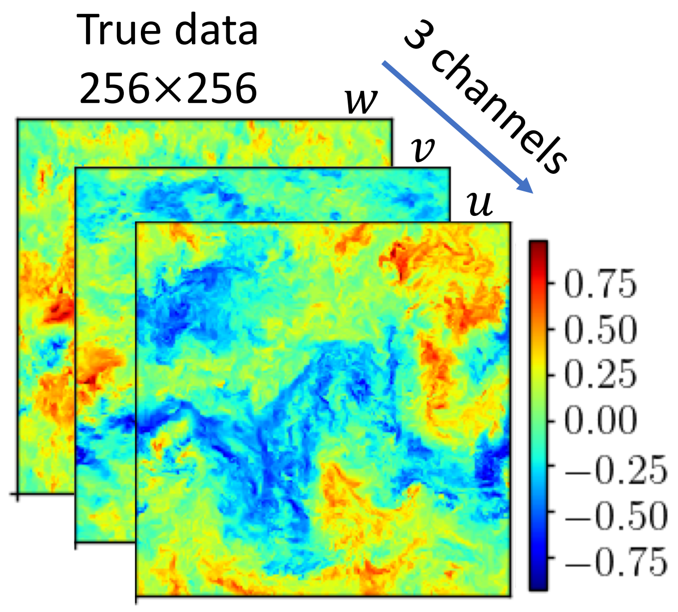
Problem

It is computationally expensive to resolve all scales in a turbulent flow simulation (Direct Numerical Simulation (DNS)) thus it is common practice to use a cheaper, coarse-grained simulation such as Large Eddy Simulation (LES). LES uses coarse grids and simulates only coarse-grained (resolved) variables while modeling the subgrid effects. The simulated data output by the LES models can be thought of as a filtered velocity field, i.e. the simulation 'smooths' out the velocity values and erases small structures.

The aim of this project is to recover the true structure of a velocity field from its coarse-grained computation using an convolutional neural network architecture.

Data

True data
256×256



DNS simulation [4]:

- 3D domain: $[0, 2\pi]$
- From $t = 0$ to 10.056 (5028 time samples)
- Large eddy turnover time: $T_L = 1.99$
- Integral scale: $L = 1.364$

Data construction for 2D slices:

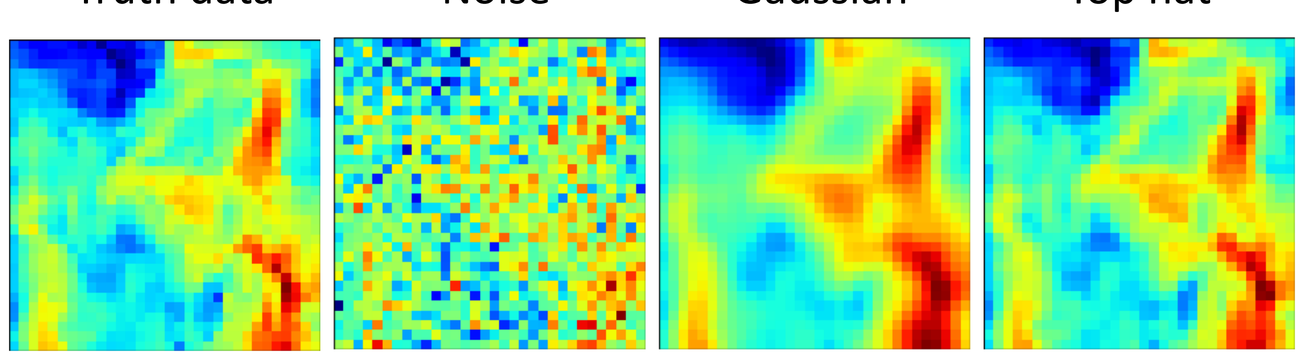
- Rotation (90°, 180°, 270°)
- Reflection
- Shifting (by 10 points)

Difficulties:

- Size = 256×256×3×(float size)
- For 32 bit: 1 Slice = 1.6MB

12,120 examples for training and validation

Zoomed 32×32 Truth data	Zoomed 32×32 Noise	Zoomed 32×32 Gaussian	Zoomed 32×32 Top hat



y_train — x_train (filtered with one of the filters)

y_predicted = Restored truth data

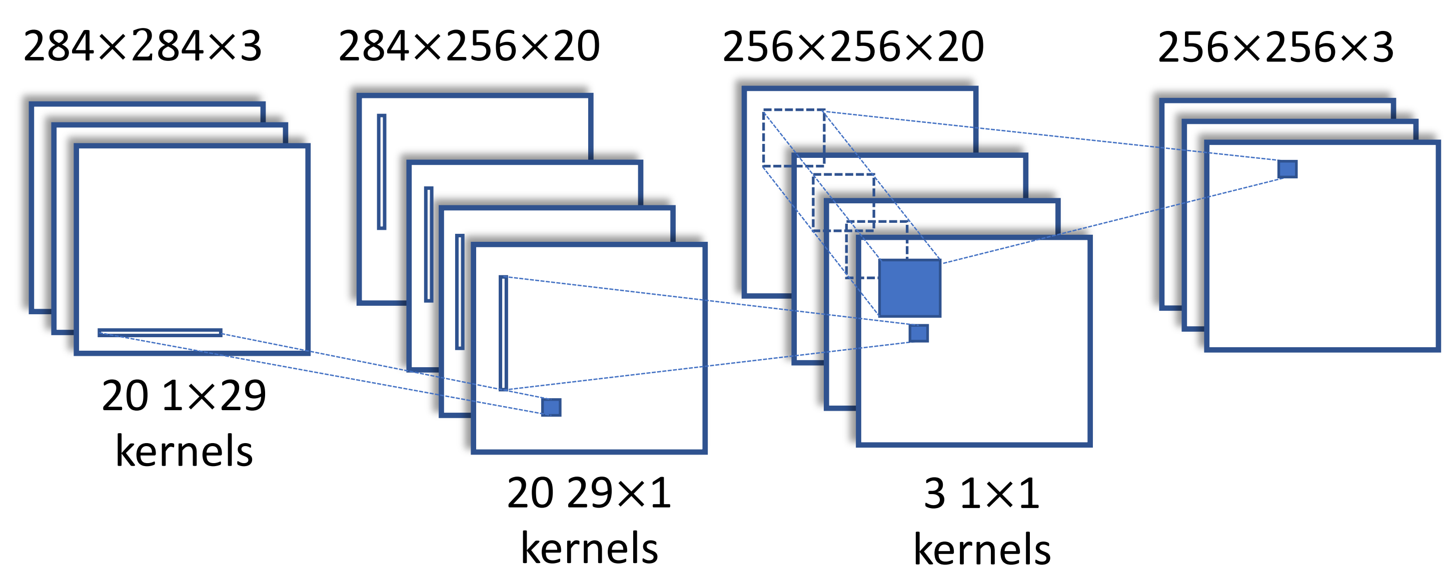
Deconvolution

LES velocity filtering can be modeled as a translation-invariant convolution $x = k * y$, where y is the original field, k is the convolutional kernel and x is the resulting filtered velocity field.

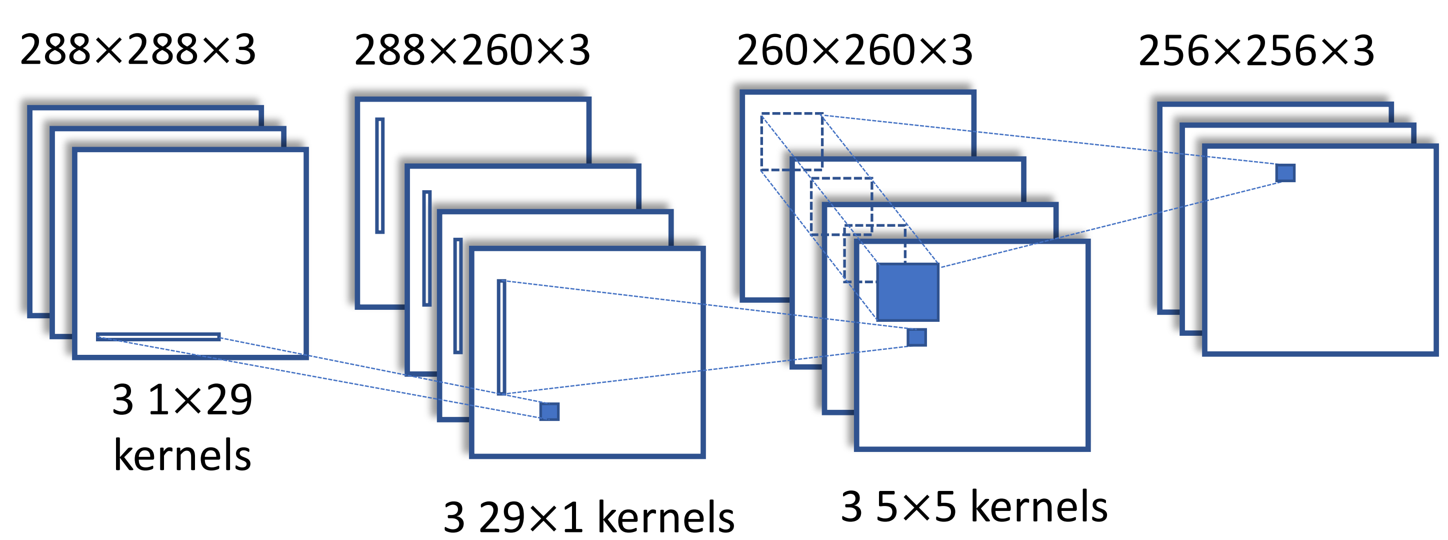
Thus restoration of the original velocity field is an inverse process called deconvolution and can be expressed as $y = k^{\dagger} * x$, where $k^{\dagger}$ denotes the pseudo-inverse kernel.

Deconvolutional Convolutional Neural Network

The network architecture is adopted from [2], where the authors use kernel separability achieved by singular value decomposition (SVD) of the pseudo-inverse kernel $k^{\dagger} = USV^T$. This allows them to express a 2D convolution as a weighted sum of separable 1D kernels and avoid rapid weight-size expansion.



Denoising Convolutional Neural Network



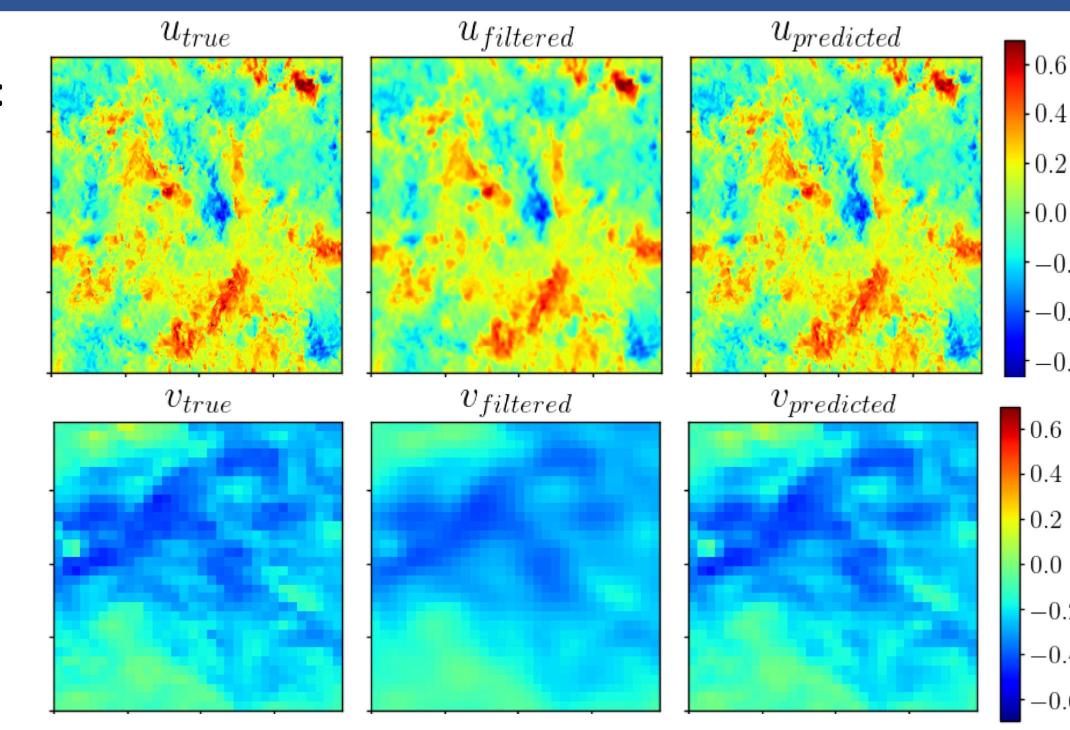
For both networks:

- Activation function: $tanh$
- Loss function = mean squared error
- Optimizer = 'adam'

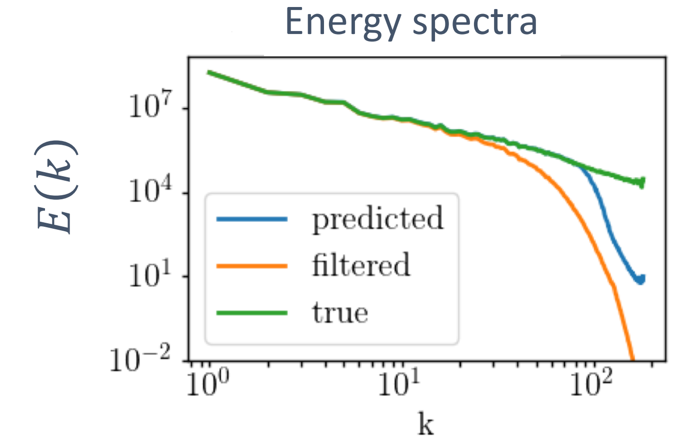
Results: Gaussian filter

Full domain: 256×256

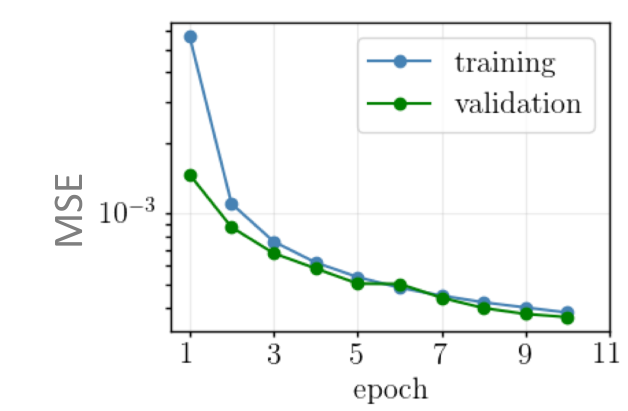
Zoomed: 32×32



Energy spectra



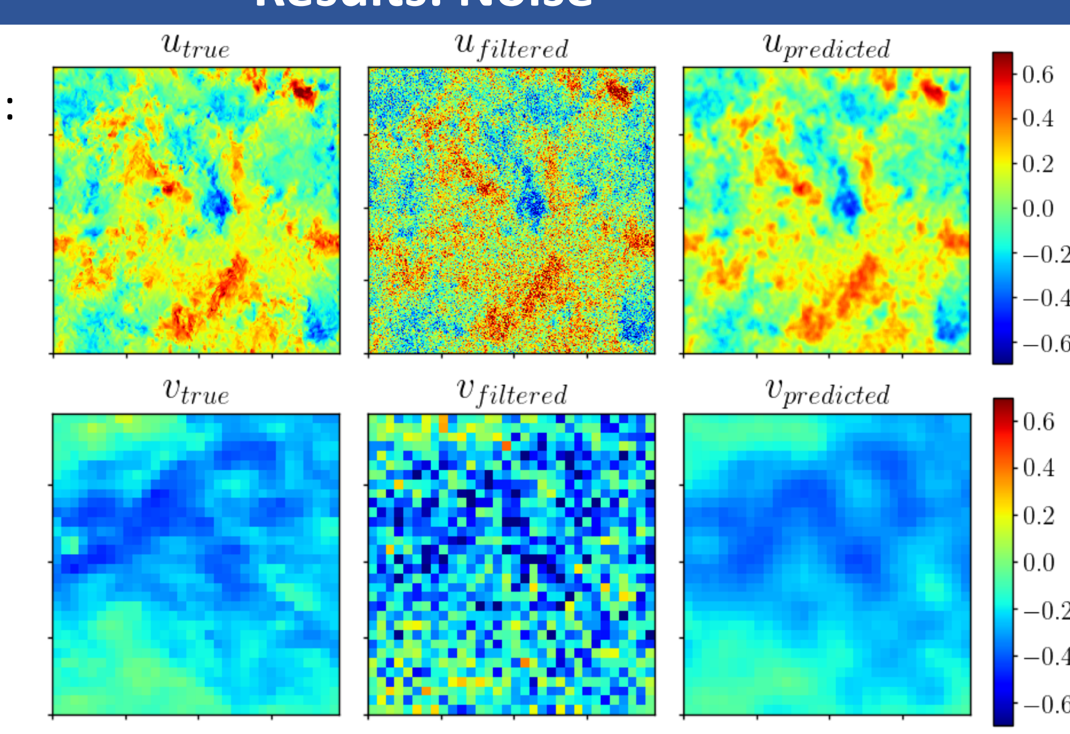
MSE loss



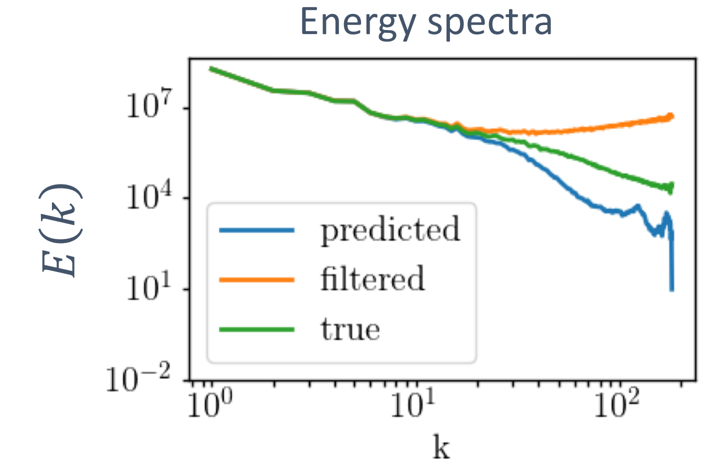
Results: Noise

Full domain: 256×256

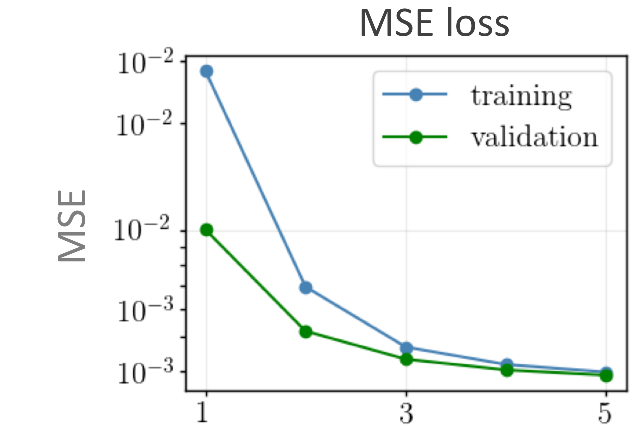
Zoomed: 32×32



Energy spectra



MSE loss



References

[1] Maulik, Romit, and Omer San. "A neural network approach for the blind deconvolution of turbulent flows." *Journal of Fluid Mechanics* 831 (2017): 151-181.

[2] Xu, Li, et al. "Deep convolutional neural network for image deconvolution." *Advances in Neural Information Processing Systems*. 2014.

[3] Guo, Xiaoxiao, Wei Li, and Francesco Iorio. "Convolutional neural networks for steady flow approximation." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[4] Johns Hopkins Turbulence Database (http://turbulence.pha.jhu.edu/)