Noname manuscript No.
(will be inserted by the editor)

Particle streak velocimetry using Ensemble Convolutional Neural Networks

Alexander V. Grayver · Jerome Noir

Received: - / Accepted: -

Abstract Insert your abstract here.

Keywords Streak analysis · Neural Networks · Turbulent Flow

1 Introduction

Particle Image Velocimetry (PIV) is arguably the most widely used technique to quantitatively study experimental flows [9]. A common work-flow would consist of seeding flow with luminescent particles and taking pairs of pictures with a known short time separation to capture an instantaneous flow state. By splitting a pair of images into (possibly overlapping) windows and cross-correlating between them allows one to infer the direction and magnitude of the flow. With modern computers and digital cameras, PIV has experienced wide adoption in the scientific community **add refs**. For cross-correlation to work properly, one has to ensure that the exposure time is short enough such that particles do not move more than a few pixels and the two images have clearly identifiable correlations, thus the mean motion has to remain predominantly linear. Violating this condition, for instance because of insufficient laser intensity or too fast flow, results in so called streaks - traces of particles.

Streaks in the PIV images are commonly considered an experimental failure since they render cross-correlation techniques very inefficient or even inapplicable.

A. V. Grayver
Institute of Geophysics, ETH Zurich
Sonneggstrasse 5
8092 Zurich, Switzerland
Tel.: +41-44-6333154
E-mail: agrayver@erdw.ethz.ch

J. Noir
Institute of Geophysics, ETH Zurich
Sonneggstrasse 5
8092 Zurich, Switzerland
Tel.: +41-44-6337593
E-mail: jerome.noir@erdw.ethz.ch

When flow velocity imposes constraints for which the camera and light source at hand cannot capture steady particles, one either has to use a higher speed camera or/and a stronger light source. Both of these solutions have financial and safety constraints, which often cannot be fulfilled. Although not suitable for conventional cross-correlation methods, streak images do contain information about the flow. Although one does not know the direction, information on the mean velocity magnitude and azimuth can be inferred from streak images. There appears to be few attempts to do this, which motivated us to design a method for extracting information about the flow from streaks.

To this end, we applied an ensemble of Convolutional Neural Networks (CNN) trained on streak images to draw a statistical prediction of the displacement magnitude and corresponding azimuth. The potential of CNNs has long been recognized for applications related to the face and handwriting recognition [7, 10], although their power has been fully discovered only recently when deeper (that is, with more layers) networks became feasible to train within reasonable amount of time, mostly due to emerge of Graphic Processor Units (GPUs) [6, 5]. An interested reader is referred to a recent overview of the deep learning with important development milestones listed [8].

Before we proceed to describing methodology and results, it is worth to mention that a crucial ingredient to a success of CNNs is a suitably large training set, which a CNN is supposed to learn from without facing an over-fitting. From this perspective, streak analysis appears almost an ideal problem. Similar to PIV, we make a reasonable assumption that within a sufficiently small interrogation window velocity exhibits simple functional form (for instance, constant or linear function). This allows us to quickly generate an unlimited number of training images. Adding variability in number of streaks per window, their thickness and intensity will well mimic a realistic experimental scenario. Therefore, we can exhaustively sample the parameter space and feed the network with as many sample as needed to attain an acceptable accuracy. The specific numbers and parameters will be discussed in the corresponding sections below.

2 Methods

2.1 Problem setup

We aim at inferring the displacement and azimuth from a gray scale image of size $n \times n$ pixels depicting streaks (Figure 1). In real settings, these images are excerpts (obtained, for instance, after applying a sliding window) of a larger $N \times N$ pixels image taken by a camera. For purposes of our experiments, we chose $n = 48$, but the approach presented below is general. We assume that within $n \times n$ window, the velocity is a constant, thus streaks have the same length and direction with small variations, treated as noise. In contrast, the number of streaks ($[2, 10]$), their color intensity ($[200, 255]$, where 0 is black and 255 is white) and thickness ($[2, 4]$ pixels) vary. Additionally, streaks are allowed to go outside the image, which is often the case in real settings, thus a CNN is supposed to be resilient to such cases. The maximum displacement, $r_{\max}$, of a particle within an image is assumed to be a half of the chosen window size, that is $r_{\max} = 24$ pixels in our case. Each streak

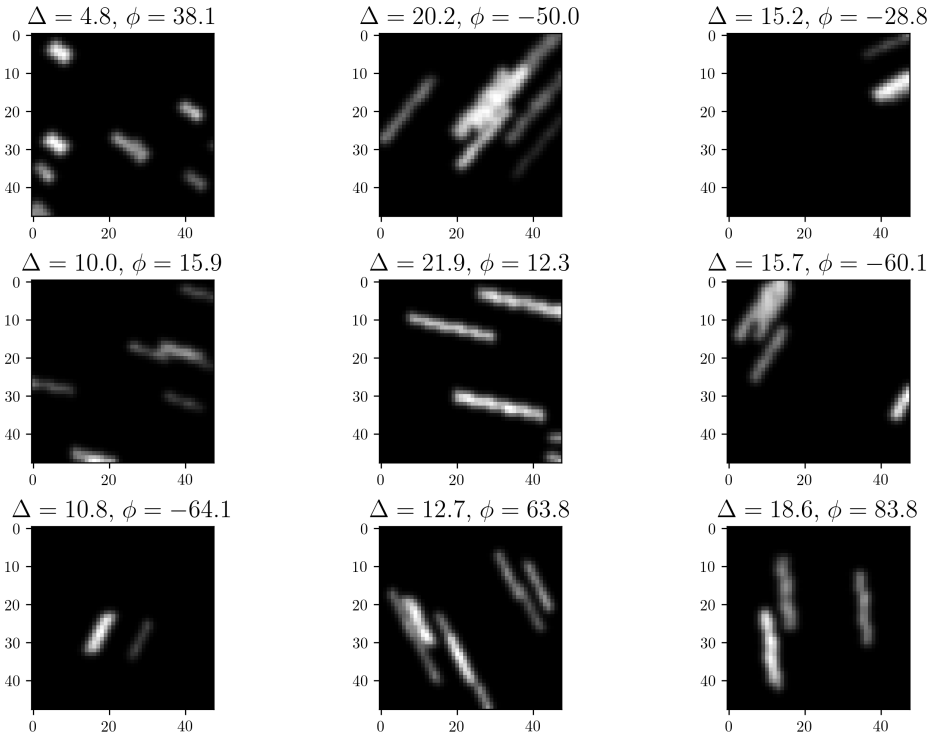

Fig. 1 Subset of generated streak images with corresponding displacements and azimuth angles given in titles used for training and validation.

is thus characterized by two numbers Δ, ϕ, denoting displacement and azimuth, respectively.

To generate train images, we sampled Δ and ϕ from the distributions

$$\Delta \sim \sqrt{U(0,1)} \cdot r_{\max} \tag{1}$$

and

$$\phi \sim U(-\pi/2, \pi/2), \tag{2}$$

respectively. This choice ensures that we sample a half-circle of radius $r_{\max}$ uniformly. $\phi = 0$ corresponds to the positive x-axis and increases clockwise. Note that limits for ϕ cover only half of the circle since we have inherent ambiguity with respect to the direction. Figure 1 shows a selection of streak images generated using the stated procedure. The outlined procedure enables generation of millions of images in just a few minutes on a regular PC.

2.2 Network architecture and implementation

To prevent overfitting and improve representative power of the network, we added batch normalization at the end of each convolution unit [4] and dropout with the 30% drop fraction [11].

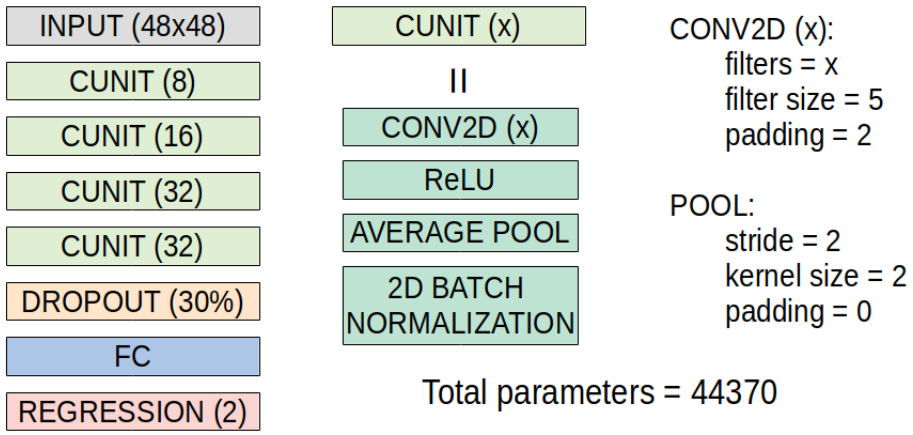

Fig. 2 Architecture of the network. Left: sequence of layers and corresponding parameters in the brackets. Each CUNIT layer consists of four actual layers listed in the middle column. Relevant layer parameters are given on the right. Total number of learnable parameters in the network is 44370.

To increase accuracy and robustness of the prediction [3], we trained an ensemble of networks, which consists of 10 networks. All networks have identical architecture and parameters, but trained using different randomly chosen initial weights [1, 2] and shuffled batched, thus forming a unique sequence of batches supplied to ADAM optimizer. The outputs of all models are averaged to get a final prediction.

Finally, the network was implemented using the PyTorch library. The complete implementation can be found using the link below.

3 Results

3.1 Network accuracy

3.2 Validation with DNS

3.3 Validation with experimental images

4 Conclusions

Extension: non-constant velocity (polynomials -¿ more output variables and possibly deeper network). Streak localization and tracking: more complex architectures.

Acknowledgements We thank authors of PyTorch, mathplotlib and h5py python packages for making them available, Meredith Plumley for sharing her DNS data, Adrian Tasistro-Hart for a number of insightful discussions on CNNs. The Juoyter Python notebooks of all programs used in this study can be found here `https://github.com/agrayver/streakcnn`.

References

1. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256 (2010)
2. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034 (2015)
3. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
4. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
5. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1725–1732 (2014)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
7. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks **8**(1), 98–113 (1997)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
9. Raffel, M., Willert, C.E., Scarano, F., Kähler, C.J., Wereley, S.T., Kompenhans, J.: Particle image velocimetry: a practical guide. Springer (2018)
10. Simard, P.Y., Steinkraus, D., Platt, J.C., et al.: Best practices for convolutional neural networks applied to visual document analysis. In: Icdar, vol. 3 (2003)
11. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)