Particle streak velocimetry using ensemble convolutional neural networks

Alexander V. Grayver1 · Jerome Noir1

Received: 19 July 2019 / Revised: 27 December 2019 / Accepted: 30 December 2019 / Published online: 21 January 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract
This study reports an approach and presents its open-source implementation for quantitative analysis of experimental flows using streak images and convolutional neural networks (CNN). The latter are applied to retrieve a length and an angle from streaks, which can be used to deduce kinetic energy and directionality (up to an 180° ambiguity) of an imaged flow. We developed a quick method for generating essentially unlimited number of training and validation images, which enabled efficient training. Additionally, we show how to apply an ensemble of CNNs to derive a formal uncertainty on the estimated quantities. The approach is validated on the numerical simulation of a convective turbulent flow and applied to a longitudinal libration flow experiment.

Graphic abstract

1 Introduction

Particle image velocimetry (PIV) is arguably the most widely used technique to quantitatively study experimental flows (Westerweel 1997; Raffel et al. 2018). A common workflow would consist of seeding a 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 academic and industrial domains (Tropea and Yarin 2007). However, 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 (Westerweel 1997). Violating this condition, for instance because of an insufficient laser intensity or a 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 less efficient or even inapplicable. When flow velocity imposes constraints for which the camera and light source at hand cannot capture instantaneous particle positions, one either has to use a higher speed camera or/and a stronger light source. Both of these solutions quickly hit financial and safety constraints, which often cannot be overcome in academia. Therefore, there exists a need for alternative processing techniques. Although not suitable for conventional cross-correlation methods, streak images do carry information about the flow. Indeed, streaks provide quantitative information on the mean velocity magnitude and azimuth of the displacement of the particles over the exposure time of the frame. The only information lost in a streak image, when compared to a pair of PIV images, is the direction of the flow, leading to an ambiguity of 180°, no matter which algorithm will be used to analyze the images. Several methods can be used to extract quantitative information from a streak image. Among them is the probabilistic Hough transform (Galamhos et al. 1999), a widely used algorithm in computer vision to detect linear features in images. However, this approach requires fine-tuning and often fails at coping with complex situations, such as overlapping streaks.

To the best of authors knowledge, there are no other reliable algorithms to recover quantitative information from streak images, which motivated us to design a new method using recent developments in the area of machine learning.

In particular, 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 (Lawrence et al. 1997; Simard et al. 2003), 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 the emergence of affordable Graphic Processor Units (GPUs) (Krizhevsky et al. 2017; Karpathy et al. 2014). An interested reader is referred to a recent overview of the deep learning with important development milestones listed (LeCun et al. 2015).

Before we proceed to describe 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, building a training set from numerical simulations appears disadvantageous for two reasons: (1) producing a set of numerical simulations, which exhaustively cover the anticipated range of parameters is extremely resource demanding if not infeasible and (2) since our main goal is to study fast turbulent flows, any numerical simulation will be band-limited, thereby lacking information on some wavelengths. In contrast, and similar to the PIV, we make a reasonable assumption that within a sufficiently small interrogation window velocity exhibits a simple functional form (for instance, constant or linear function). This allows us to quickly generate an unlimited number of window-based training images. Adding variability in number of streaks per window, their thickness and intensity mimics a realistic experimental scenario sufficiently well. Therefore, we can exhaustively sample the parameter space and feed the network with as many samples 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 streak image of $N \times N$ pixels. A typical example is illustrated in Fig. 1.

Similar to how PIV is applied to a pair of images, we split an image into $(n \times n)$ sub-windows. Assuming the velocity is sufficiently uniform over the $(n \times n)$, we aim at determining the mean length, that is the displacement over the exposure time, and azimuth for each sub-window.

A closer look at the image reveals that the number of streaks, their intensity and their width, vary within each sub-window. Additionally, in many cases, the streaks leave the $(n \times n)$ interrogation area. To make our CNN resilient to this variability, we generate train and validation images of $(48 \times 48)$ with a random number of streaks varying between 2 and 10, a random color intensity between 90 and 255 and a random thickness between 2 and 4 pixels. The center of each streak is chosen randomly over the entire sub-window, such that streaks are allowed to go outside the image.

To generate train and validation images, we sampled displacement, $\Delta$, and azimuth, $\phi$, from uniform distributions

$$\Delta \sim U(1, \Delta_{\text{max}})$$

and
respectively. Hence, in a single training or validation image all streaks have the same displacement and azimuth. Here, \( \phi = 0 \) corresponds to the positive \( x \)-axis and increases clockwise.

Note that limits for \( \phi \) cover only half of the circle minus \( \delta = 5^\circ \) on one end to mitigate ambiguity with respect to the direction. Once streaks are generated, the intensity within each streak is perturbed and the image is blurred with a Gaussian kernel of \( 3 \times 3 \) pixels. In this study, the maximum displacement, \( \Delta_{\text{max}} \), of a particle within an image is assumed to be half of the chosen window size, that is, \( r_{\text{max}} = 24 \) pixels.

Figure 2 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

We aim at building a regression CNN that, given a single window image, outputs two real numbers, corresponding to the displacement and azimuth. To this end, we have created a network with the architecture shown in Fig. 3. As it can be seen, four convolutional units form the core of the network, each consisting of a convolutional layer, activation layer (in this case, rectified linear unit), average pooling layer used to downsample the input followed by a batch normalization layer. An increasing depth of the convolutional layers is a common choice aimed at giving a network ability to learn more complex features from the input. A fully connected (FC) layer at the end flattens the output of the convolution layers and, unlike previous layers, uses the trivial identity activation to transforms
input parameters linearly into two output parameters, corresponding to predicted variables.

To prevent overfitting and improve training speed, we used batch normalization at the end of each convolution unit (Ioffe and Szegedy 2015) and a dropout layer with the 30\% drop fraction (Srivastava et al. 2014). Additionally, following common practice, the outputs are standardized and centered. The input grayscale images have a single eight-bit channel and are normalized by 255 to remain in the [0, 1] range.

Whereas there may exist more efficient architectures, we found that for our problem increasing the depth of the network did not improve the overall performance significantly, nor did increasing and/or decreasing filter sizes. Additionally, we found that, in contrast to majority of the reported CNN architectures, using the average pooling instead of the max pooling results in a slightly higher accuracy.

To increase accuracy and robustness of the prediction, we built an ensemble consisting of ten networks (Hinton et al. 2015). All networks have the same architecture and parameters, but were trained using different randomly chosen initial weights (Glorot and Bengio 2010; He et al. 2015) and random shuffling to form a unique sequence of batches supplied to an optimizer. The outputs of all ensemble members are averaged to obtain the final prediction and its variance. The latter may seem like a small rotations, especially when streaks are short. For instance, in Fig. 5 (bottom row, middle), two streaks coincidentally joined and formed what appears to be a longer streak, but CNN prediction is not confused since other streaks provide enough information for an accurate prediction. Additionally, Fig. 6 shows nine images which produced the worst predictions of the displacement. Clearly, most of these cases are associated with situations where originally longer streaks have all been displaced outside the window. Note that such situations do not represent a failure of the CNN, but rather limitation of the window-based approach. Similar situations are likely to occur in experimental images and unfortunately they leave little chance for any sliding window approach to

Each network in the ensemble has been trained for 100 epochs. The ADAM optimizer with an initial learning rate value of 1e–3 was used. The learning rate was halved if no sufficient decrease in the validation loss was observed for the past 10 epochs. To make the training more robust to potential outliers (for instance, consider a situation when all streaks leave the window), we chose to minimize the Huber loss (Huber et al. 1973) rather than conventional least-squares loss. Huber loss represent a hybrid \( L_2 - L_1 \) distribution with a high tolerance to the presence of long tails in the original distribution. No pathologies between training and validation losses were observed during the training, specifically the validation loss remained a bit higher than the training loss, indicating that the network learns some generic features of the dataset. Training of each network took \( \approx 90 \) min on a single CPU–GPU system. The modest training time makes it feasible to train networks for window sizes larger that the window of 48 \( \times \) 48 pixels adopted here, although larger interrogation windows were not necessary for our problems.

Figure 4 shows RMSE values for all networks separately as well as the RMSE of the ensemble. RMSE was calculated as

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \tilde{x}_i)^2},
\]

where \( \tilde{x} \) and \( x \) are the true and predicted values of \( \Delta \) or \( \phi \), respectively.

It is evident that the ensemble performs systematically better than any single network. CNN ensemble achieves the accuracy of 1.05 pixels for the displacement and 8.5° for the angle predictions, respectively. The latter may seem like a large error; however, one should realize that having a 48 \( \times \) 48 pixels window inevitably leads to limitations in recognizing small rotations, especially when streaks are short.

Figure 5 shows a random selection of validation images with the corresponding ensemble CNN predictions and true values. We see that the ensemble CNN has learned to make accurate prediction even for complex situations when some streaks overlap, join or leave the window. For instance, in Fig. 5 (bottom row, middle), two streaks coincidentally joined and formed what appears to be a longer streak, but CNN prediction is not confused since other streaks provide enough information for an accurate prediction. Additionally, Fig. 6 shows nine images which produced the worst predictions of the displacement. Clearly, most of these cases are associated with situations where originally longer streaks have all been displaced outside the window. Note that such situations do not represent a failure of the CNN, but rather limitation of the window-based approach. Similar situations are likely to occur in experimental images and unfortunately they leave little chance for any sliding window approach to

\[
\text{Fig. 3} \quad \text{Architecture of the used network. Left: sequence of layers and corresponding parameters in the brackets. Each CUNIT layer consists of four actual layers listed in the middle column. Relevant layer parameters in the network is 44,370}
\]

\[
\text{Table} \quad \text{Architecture of the used network. Left: sequence of layers and corresponding parameters in the brackets. Each CUNIT layer consists of four actual layers listed in the middle column. Relevant layer parameters in the network is 44,370}
\]
Fig. 4 Top: RMSE values of individual networks (markers) and ensemble (line) for displacement $\Delta$ and angle $\phi$ calculated on the validation set. Bottom: histograms of the residuals for the displacement and angle. Histograms for both absolute and relative (i.e., normalized by the true displacement) residuals are shown.

Fig. 5 Random images with the predicted displacement and angle depicted as magenta lines. The true values are given in titles and predicted values in brackets.
make an accurate prediction. We will discuss possible ways to mitigate this in the Conclusions section. It is worth to note that despite an inaccurate displacement prediction, the angle is predicted accurately in most of these cases.

### 3.2 Validation using synthetic flows from numerical simulations

After confirming the excellent performance of the network on the validation set, we turn to a direct numerical simulation (3D-DNS) of a complex thermal convection turbulent flow published recently (Plumley et al. 2016). In their paper, the authors simulate rapidly rotating thermal convection in a square box, heated from below and cooled from the top with a vertical gravitational field pointing downward and a vertical axis of rotation. To validate our CNN algorithm, we used a simulation obtained for $E = 10^{-7}$, where $E$ is the Ekman number representing the ratio of viscous to rotational effects, and a Raleigh number that is 90 times the supercritical. Each velocity field is calculated on a uniform grid of size $192^3$. For the purpose of this study, we extracted the horizontal component of the velocity in a horizontal plane at mid-depth. Because of the limited resolution of the simulation, each velocity component has been interpolate on a $1024 \times 1024$ pixels grid. Figure 7 left shows the displacement map obtained from the full DNS for an arbitrary exposure time. Since the CNN will act on $(48 \times 48)$ interrogation windows with the 50% overlap, we apply a sliding window averaging with $(48 \times 48)$ and 50% overlap (Fig. 7, right). The window averaged displacement is the best reconstruction we can possibly achieve with a $48 \times 48$ pixels window and a 50%
overlap used in this study; thus, all later results will be compared against it.

To reduce subjectivity, we generated 30 randomly seeded streak images using the original DNS velocity (Fig. 7, left). We first split the image into 48 × 48 pixels subwindows, inside which we calculate the mean linear displacement \( d = \bar{u} dr \), where \( \bar{u} \) represents the horizontal velocity average over 48 × 48 pixels. In each subwindow, we generate a random seeding, varying the density (from 1 to 10 seeds per subwindow) and radius (from 2 to 4 pixels) of the seeds and reconstruct the associated streaks. Finally, we apply a Gaussian filter with a kernel of 3 × 3 pixels to mimic the natural variability of intensity observed in experimental images. We repeat the entire process to generate 30 independent streak images from the same velocity field. The trained CNN ensemble was applied to all images and the resulting averaged reconstruction of both displacement and angle are shown in Fig. 8. We see that the reconstructed images recover the general structure of the flow very well. The only undesired effect is that the ensemble

![Fig. 8](image-url)
CNN struggles to correctly predict large displacements. When the displacement reaches half of the window size, CNN systematically predicts smaller displacements. The reason for this is not a deficiency of the CNN, but rather the fact that for displacements as large as half the window size it becomes very likely that most of the streaks go outside the window, thus a smaller displacement is a rational prediction. To solve this problem, one could apply a larger window (for instance, $64 \times 64$ pixels), although this will reduce the overall resolution and may render constant velocity assumption less accurate. In real experimental settings, it may be advantageous to apply several window sizes and either choose one or combine obtained results, depending on a particular experimental setup.

Additionally, we compared the performance of the CNN ensemble with a more conventional technique based on the probabilistic Hough transform (Galamhos et al. 1999). The results of the Hough transform are shown in Fig. 8. Clearly, the approach based on the Hough transform shows a less robust behaviour, especially in predicting the displacement.

Finally, Fig. 9 shows one of the streak images generated from the DNS with the true and predicted displacement and angle shown as red and yellow lines, respectively. Note that while the whole velocity field is used to generate the background streak images, the overlying red streaks are reconstructed at the location of the CNN grid point using the averaged displacement and angle maps (Fig. 7). Generally, the ensemble CNN performs well. There is, however, accuracy reduction in regions of large displacements for reasons described above and across areas of vertical shearing, where angle predictions become locally less accurate. Interestingly, these vertical shear regions also lead to abnormal streak orientation (close to horizontal) when using the window averaged field values of the DNS itself (red streaks in Fig. 9).

### 3.3 Application to the flow driven by longitudinal libration in laboratory experiment

Finally, we apply our algorithm to a laboratory experiment depicted in Fig. 10. The apparatus consists of a straight cylinder filled with water and set in rotation on a turntable at 1 Hz. The cavity is 286 mm high with a radius of 140 mm. The bottom lead is covered with a topography made of blocks ($64 \times 64 \times 8$ mm). A second motor is used to oscillate the container on the turntable, resulting in the so-called longitudinal libration.
**Fig. 10** Experimental setup of the longitudinal libration experiment (Meister 2018)

**Fig. 11** Predicted displacement and angle (left) along with their standard deviations (right) inferred from the ensemble CNN for an experimental image shown in Fig. 13
In such a system, flows at low libration amplitudes are in the form of linear inertial waves and inertial modes. As the amplitude increases, the flows become unstable and eventually saturate with a significant geostrophic component, i.e. mainly two dimensional. For the purpose of this experimental validation, we have set the libration frequency to 1 Hz and the amplitude to $34^\circ$, which corresponds to the geostrophic saturation state aforementioned.

To image the flow, we use a 1 W continuous diode laser to illuminate a horizontal plane of the fluid seeded with fluorescent particles. The particles absorb in the green and emit in the orange. By mounting a narrow band pass filter on the lens of the camera, we reject the light coming from the reflections of the laser on the side walls. While greatly improving the contrast of the images, the use of a band-pass filter reduces the amount of light reaching the CCD sensor, as a consequence we must work with relatively large exposure time of 21 ms with a sampling frequency of 30 images per second. This limitation leads to streak images as shown in Fig. 1 rather than sharp images of individual particles that would be appropriate for PIV. Meanwhile, our CNN algorithm is well suited for extracting quantitative information from this type of images. We first apply our algorithm to a single frame prior to analysing image sequence covering 10 s of the libration in a statistically steady state.

A pre-processing step was necessary for real images to filter out the CCD matrix noise. We achieved this by a simple thresholding of the image intensity at the value of 50. After that, image was split into overlapping $48 \times 48$ pixels windows, resulting in a total of 2400 window images. The ensemble CNN was then applied to predict displacements and angles as well as their standard deviations with the latter serving to be a proxy for the output uncertainties.

Figures 11 and 12 show the predictions of the displacement and azimuth in the form of maps and histograms for one randomly chosen frame of the movie. Figure 13 shows the selected experimental image we used and window-centered predictions drawn as lines. First of all, we see that experimental flow exhibits rather complex structure, with regions where streaks show non-linear behaviour of the flow within a window, which violates our initial assumptions. Nonetheless, the ensemble CNN produces coherent and spatially correlated predictions (Fig. 11). It is also interesting to analyse the standard deviation maps.
Large absolute errors in the displacement map appear to be correlated with regions of fast flow. On the other hand, large uncertainties in the angle occur around regions with vertical velocity shear, as observed in DNS synthetic images, specifically where velocity seems to flip the direction.

After checking that the algorithm behaves well on a single frame, we applied the CNN ensemble to 300 frames taken at 30 fps to capture the temporal evolution of the flow. Subsequently, we calculated the mean kinetic energy from the displacement for each frame as

\[ E = \frac{1}{P} \sum_{i=1}^{P} \left( \Delta_i l_i \right)^2, \]  

where \( P \) is the number of grid points in a frame, \( \Delta_i \) is the \( i \)th displacement in pixels, \( l = 0.25 \cdot 10^{-3} \) m/pixel is the calibration factor from pixels to meters for our optical setup, \( \tau = 21 \) ms the exposure time. The timer-series of the mean kinetic energy and its Power Spectral Density are shown in Fig. 14. As anticipated, we can clearly identify the dominant forcing frequency of 1 Hz and the harmonics at 2 Hz and 3 Hz, resulting from non-linear effects.
In terms of performance, ensemble CNN attains a speed of seven full images (each totaling to 2400 windows) per second on an NVIDIA GoForce GTX 1070Ti powered PC. This level of performance allows us to comfortably process long sequences of images taken during experiment, which we use to study evolution of the kinetic energy. Using a more modern GPU will likely also enable real-time predictions at rates of a few dozens of images per second, which could be more relevant to industrial applications.

4 Conclusions

We presented an open-source ensemble CNN aimed at quantitative analysis of complex experimental flows using streak images. The presented approach can be applied in situations when classical PIV is inaccurate or not possible due to experimental setup or hardware limitations. The advantage of the presented approach is the ease of training/validation set generation as well as recovery of the prediction uncertainty through application of the ensemble method. We foresee that the approach may need an experiment-dependent tailoring to achieve the best performance. According to our experience, this only takes a single day on an average PC equipped with a modern GPU.

The potential extensions of the presented approach may include the use of multiple window sizes applied to the same image, more complex functional parameterizations of the displacement and angle fields within a window (e.g., using a polynomial). Finally, a completely different approach based on an image segmentation and object localization and/or tracking can be applied to extract information about individual streaks and their evolution in time. We hope to investigate these directions in details in our future studies.

Acknowledgements We thank authors of PyTorch, matplotlib and h5py libraries for making them available, Meredith Plumley for sharing her DNS data, Adrian Tasistro-Hart for a number of insightful discussions on CNNs. Comments by two anonymous reviewers helped improve the manuscript. The Jupyter Python notebooks of all programs used in this study can be found here https://github.com/agrayver/streakkcnn.

References

Galambos C, Matas J, Kittler J (1999) Progressive probabilistic Hough transform for line detection. In: Proceedings of 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149), vol 1. IEEE, pp 554–560

Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, vol 9, pp 249–256

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp 1026–1034

Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv preprint: arXiv:1503.02531

Huber PJ et al (1973) Robust regression: asymptotics, conjectures and monte carlo. Ann Stat 1(5):799–821

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint: arXiv:1502.03167

Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 1725–1732

Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386

Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: a convolutional neural-network approach. IEEE Trans Neural Netw 8(1):98–113

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436

Meister A (2018) Effects of boundary topography on the flow forced by libration in longitude in a rotating cylinder, master thesis, ETH Zurich

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S, Wallach H, Larochelle H, Beygelzimer A, d’ Alché-Buc F, Fox E, Garnett R (2019) PyTorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32:8024–8035

Plumley M, Julien K, Marti P, Stellmach S (2016) The effects of Ekman pumping on quasi-geostrophic Rayleigh–Bénard convection. J Fluid Mech 803:51–71

Raffel M, Willert CE, Scarano F, Kähler CJ, Wereley ST, Kompenhans J (2018) Particle image velocimetry: a practical guide. Springer, Berlin

Simard PY, Steinkraus D, Platt JC (2003) Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the seventh international conference on document analysis and recognition, Edinburgh, UK, 2003, pp 958–963. https://doi.org/10.1109/ICDAR.2003.1227801

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

Tropea C, Yarin AL (2007) Springer handbook of experimental fluid mechanics. Springer Science & Business Media, Berlin

Westerweel J (1997) Fundamentals of digital particle image velocimetry. Meas Sci Technol 8(12):1379

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.