Vortex Boundary Identification using Convolutional Neural Network

Marzieh Berenjkoub* University of Houston and Nvidia Inc.
Guoning Chen† University of Houston
Tobias Günther‡ FAU Erlangen-Nürnberg

ABSTRACT

Feature extraction is an integral component of scientific visualization, and specifically in situations in which features are difficult to formalize, deep learning has great potential to aid in data analysis. In this paper, we develop a deep neural network that is capable of finding vortex boundaries. For training data generation, we employ a parametric flow model that generates thousands of vector field patches with known ground truth. Compared to previous methods, our approach does not require the manual setting of a threshold in order to generate the training data or to extract the vortices. After supervised learning, we apply the method to numerical fluid flow simulations, demonstrating its applicability in practice. Our results show that the vortices extracted using the proposed method can capture more accurate behavior of the vortices in the flow.

Keywords: Vortex boundary, convolutional neural network

1 INTRODUCTION

Vortex dynamics plays a crucial role in determining the behaviour of fluid flows across a wide range of Reynolds numbers ranging from laminar to highly turbulent regimes. They are easy to observe around us, for instance in the wake behind cars or on the wingtip of aeroplanes. However, as noted by Haller [15], "few pause to examine what *vortex* strictly means." In laymen terms, we can envision them as fluid parcels that rotate coherently, which is, however, not a concise definition lending itself to the implementation of an extraction algorithm. In fluid dynamics, vortices are nowadays referred to as elliptic Lagrangian coherent structures (LCS). They are either characterized through scalar measures such as the instantaneous or Lagrangian-averaged vorticity deviation [16], or as regions enclosed by curves that maximize an energy functional [15]. The threshold-based extraction from region-based measures is rarely practical, because vortices dissipate and transfer energy within the inertial subrange, resulting in various sizes and angular momenta. Overcoming the numerical challenges of accurately locating the boundary of a vortex is rewarded by valuable insight about the vortex behavior. For instance, a wiggly vortex boundary is a sign of an imminent vortex breakdown – an important physical event often studied in the energy cascades of turbulence flow. In addition, the extent and area of a vortex can be more accurately computed with the vortex boundary detected, which allows for quantification, and thereby the creation of summaries to compare time steps or ensemble members. Further, such information can be important in the study of eddies in the oceans – large ocean current structures that transport nutrition and heat. Recently, deep learning methods have been proposed to identify vortices in ocean currents, for instance from sea surface height [29] or from the Okubo-Weiss criterion [9]. Similarly, the training on IVD for a binary segmentation of a vortex required a threshold to be set on the training data [8, 41]. In this work, we identify vortices directly from the velocity field and utilize an experimentally acquired vortex model [40] to characterize the location of the vortex boundary. In contrast to previous work, our method does not require the setting of thresholds and is therefore able to identify weak and strong vortices alike.

To train the neural network, a large training set with example flows that contain vortices with different configurations that may be seen in the real-world flow is needed. To generate such a large training set, we employ a parametric flow model that allows us to precisely specify the configuration of a vortex, including its center, shape, orientation, and size. With this model, we generate thousands of synthetic flow patches with ground truth vortex labels for the neural network to learn. We used this set of synthetic flows to train and compare multiple models with different architectures including a CNN, a Resnet, and a Unet, which take the velocity information as input. After supervised learning, we apply these trained networks to real-world flow to demonstrate their effectiveness. Our results show that among all three network architectures tested, the Unet extracts the most accurate vortex boundaries from synthetic flow. When applied to real-world flow, all three networks extract vortex boundaries of strong and weak vortices alike. In contrast, existing thresholding methods (e.g., IVD), require different manual thresholds.

2 RELATED WORK

Vortex Extraction and Boundary Identification. Vortex extraction remained an active field of research in fluid dynamics and scientific visualization despite decades of work [12, 22, 34, 36]. Extraction methods are generally characterized into region-based techniques that calculate scalar fields, such as λ_2 [24] or the Q-criterion [23], and line-based techniques that compute the vortex coreline around which the particles rotate [4, 32, 39, 42]. Recently, objective methods [14, 15] shifted back into focus, aiming for a steady reference frame [2, 11, 13].

In this paper, we extract vortex boundaries. In their seminal work, Banks and Singer [4] proposed a predictor-corrector method that constructs a vortex coreline. Starting from this coreline, they performed a region-growing until a vorticity or pressure threshold was reached. Bauer et al. [5] terminated the region-growing once the swirling strength, i.e., the imaginary part of the Jacobian's eigenvalues, falls below a threshold. Inspired by the velocity profile of the Rankine vortex model, Garth et al. [10] identified the vortex boundary as surface with largest tangential velocity magnitude. For steady flows, Lagerstrom [28] extracted isolated closed streamlines, which was extended by Petz et al. [33] to vortex hierarchies formed from nested closed streamlines. Recently, Haller [15] characterized vortex boundaries objectively as the largest nested elliptic LCS, which is a material line that preserves arc length in incompressible flow.

Deep Learning for Vortex Extraction. In recent years, several deep learning approaches appeared for vortex extraction. The classification approach of Bin and Li [6] categorized flow patches into rotating (cw/ccw), saddles and others. More specific features have been searched with the classification network by Ströfer et al. [38], looking for recirculation, boundary layers and a horse shoe vortex. Similar to the methods above, Deng et al. [8, 41] applied supervised training, for which they derived a vortex groundtruth from the velocity field by applying a user-defined threshold to the instantatenous vorticity deviation (IVD) [16] in order to produce a binary mask, identifying vortices and non-vortices. This threshold is chosen at training time and will generally depend on the angular momentum of the vortices. If vortices decay, there will not be a unique threshold that works in the entire data set. Given the sea surface height, Lguensat et al. [29] extracted ocean eddies. Franz et al. [9] detected ocean

*e-mail: mberenjkoub@cs.uh.edu
†e-mail: chengu@cs.uh.edu
‡e-mail: tobias.guenther@inf.ethz.ch

eddies by training a neural network that receives a vortex measure as input, namely the Okubo-Weiss criterion [31, 43]. To track the vortices over time, they applied a recurrent neural network (RNN) afterwards. Bai et al. [3] sent images of streamlines into a CNN to detect ocean eddies. Kim and Günther [26] developed a CNN that extracts a reference frame in which an unsteady flow becomes steady, enabling vortex coreline extraction. The network was trained with noisy synthetic data to improve the robustness. In another related approach, Liu et al. [30] extracted shock waves using CNNs.

Deep learning techniques have also been applied to other scientific visualization problems, such as viewpoint selection for volume rendering [21], streamline/surface selection [17], vector field reconstruction [18], and temporal information reconstruction [19].

3 Our Method

Our framework first trains a suitable neural network then applies the trained network to the real-world flow to extract vortex boundaries. In order to effectively train the selected neural network, a training flow data set that is sufficiently representative for the real-world flow is needed, that is, the flow configuration of each sample (with or without vortices) should be similar to what may be seen in the real-world flows. In addition, the ground truth vortex boundary should be known for each sample, which is not always possible in real-world flow as described earlier. To address this challenge, we adopt the synthetic flow generation framework introduced by Kim and Günther [26] with a few modifications so that the vortex boundary can be labeled precisely (Section 3.1). This synthetic flow generation framework relies on the setting of a number of parameters to generate flows with various vortex configurations. To ensure that the generated flows are representative, we fit these parameters so that they result in the synthetic flows that are as similar to the patches obtained from the real-world flow as possible (Section 3.2). After generating training sets, we apply them to train three representative network architectures to study their capability of learning vortex boundary characteristics (Section 3.4).

3.1 Synthetic Generation of Training Vector Fields

Since we aim to train the neural network for vortex boundary identification in a supervised manner, the ground truth distance to the vortex boundary is required during training. In order to obtain the exact boundary of a vortex in our synthetic model, we modify the parametric model of Kim and Günther. In our model, the velocity at any point $\mathbf{x} = (x, y)$ is given by the following formula:

$$\mathbf{v}(\mathbf{x}) = \mathbf{S}_i \cdot \mathbf{x} \cdot \frac{v_0(\|\mathbf{x}\|)}{\|\mathbf{x}\|}, \quad \text{with} \quad v_0(r) = \frac{r}{2\pi r_c^2 \left((\frac{r}{r_c})^{2n} + 1 \right)^{\frac{1}{n}}} \tag{1}$$

where $v_0(r)$ is Vatistas' experimentally measured velocity profile [40], r_c is the radius with maximum velocity and n controls the shape of the velocity profile. The parameter n controls the shape of the velocity profile. We refer to Figure 2 of the work [26] for more information about the effect of n. Matrix $\mathbf{S}_i$ with $i \in \{1, 2, 3\}$ defines one of the following three base shapes:

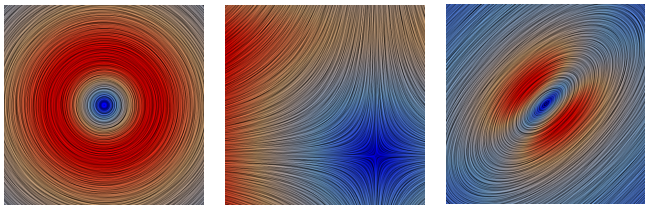
$$\underbrace{\mathbf{S}_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{\text{saddle}} \quad \underbrace{\mathbf{S}_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{\text{center (cw)}} \quad \underbrace{\mathbf{S}_3 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}}_{\text{center (ccw)}} \tag{2}$$

Among these three base shapes, only $\mathbf{S}_2$ and $\mathbf{S}_3$ contain vortices. Similar to [10], we formally characterize vortex boundaries as locations with maximal tangential velocity. Thus, the signed distance to the boundary of a vortex in each of these two cases is:

$$d(\mathbf{x}) = r_c - \|\mathbf{x}\| \tag{3}$$

The positive distance indicates locations inside the vortex, while negative distance means outside. Since $\mathbf{S}_1$ does not contain vortices, the signed distance in this example flow is set to an arbitrary and large enough negative value, e.g., -10 in our experiment.

To introduce variations to the location, orientation, and size of the vortex, we define a random linear transformation matrix $\mathbf{A}$ and a random translation vector $\mathbf{t}$ to transform the domain from $\mathbf{x}$ to $\mathbf{x}'$,



$s_x = s_y = 1,\ t_x = t_y = 0,$ $r_c = 0.5, n = 2, \theta = 0$ | $s_x = 2, s_y = 0.5, t_x = -t_y = 0.5, r_c = 2.5, n = 6, \theta = 0$ | $s_x = 1, s_y = 0.5, t_x = t_y = 0,$ $r_c = 0.5, n = 2, \theta = \pi/4$

Figure 1: Examples of synthetically generated vector field patches. The velocity magnitude is color-coded from blue (0) to red (0.23).

and the velocity from $\mathbf{v}(\mathbf{x})$ to $\mathbf{v}'(\mathbf{x}')$:

$$\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} + \mathbf{t} \tag{4}$$

$$\mathbf{v}'(\mathbf{x}') = \mathbf{A} \cdot \mathbf{v}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{v}(\mathbf{A}^{-1} \cdot (\mathbf{x}' - \mathbf{t})) \tag{5}$$

$$d'(\mathbf{x}') = d(\mathbf{A}^{-1} \cdot (\mathbf{x}' - \mathbf{t})) \tag{6}$$

To control the shape of the deformed vortices, we compose the linear transformation from a rotation θ and a non-uniform scaling (s_x, s_y):

$$\mathbf{A}(\theta, s_x, s_y) = \begin{pmatrix} s_x \cos(\theta) & -s_y \sin(\theta) \\ s_x \sin(\theta) & s_y \cos(\theta) \end{pmatrix} \tag{7}$$

In our experiments, θ and $\mathbf{t}$, s_x, s_y, r_c and n are chosen from a Gaussian distribution given their respective value ranges. See Fig. 1 for examples of patches. The range and distribution of the values of these parameters will be determined in Section 3.2. The sign of the distance field $d'(\mathbf{x}')$ identifies the interior and exterior of a vortex.

Note that the above synthesis framework poses a strong constraint to the generated flow, i.e., the synthetic flow can contain at most one vortex. This is because if more than one vortex exists, they may interact with each other (e.g., two vortex regions overlap and merge), making their boundary difficult to prescribe precisely. Although such a multi-vortex configuration often arises in real-world flows, we demonstrate later that for certain flows where vortices are sufficiently far away from each other, we can subdivide the flow domain into small regions (i.e., patches), each of which contains at most one vortex. This way our subsequent fitting processing (see next section) will be able to compute the ideal combination of the above parameters to generate a synthetic flow that is as close to a subdivided patch as possible.

3.2 Parameter Space Fitting

The above synthesis framework requires to set the values of some parameters (i.e., θ, t_x, t_y, s_x, s_y, r_c and n) to generate an analytic flow. To ensure that the generated flows are physically authentic and representative for the real-world flows, we fit these parameters (especially their ranges) to the real-world flows. Given a real-world flow, we sub-divide the flow into small patches so that each patch contains at most one vortex (or part of a vortex). Then, we utilize the simulated annealing process as described in [26] to obtain a combination of the values of the above parameters so that the generated flow has minimal distance from the reference patch. We fit vortices and saddles separately. Based on the fitted parameter values for all patches, we compute the distribution of the values of individual parameters. We model the distributions with Gaussians, which can later be sampled to generate more training data patches, which will have characteristics similar to the real data. For angle θ, we apply a uniform distribution to not prefer any orientation. The distribution of the fitted parameter spaces and the error plots of the fitting for vortices and saddles are provided in the supplemental document.

3.3 Training Data Generation

A key ingredient to a successful training is the adequate preparation of the input data. Aside from the raw size of the (possibly augmented) training data, the shape of the data requires equally careful consideration. Many of the more recent network architectures, such as CNNs, ResNets and Unets utilize the spatial relationships of the input pixels, i.e., they operate on image data that is laid out on

regular grids. For this reason, we similarly preserve the spatial embedding of the inputs by sampling the synthetically generated vector field patches onto regular grids. Each vector field patch is thereby represented as a 3D array, where the third dimension contains the velocity vector components. The training data is generated by using the parametric vector field synthesis (Section 3.1). As described earlier, we applied a linear transformation including scaling and rotation to generate enough training data for the network to generalize from, in total 25,000 flows, with some examples shown in Fig. 1. To evaluate how well the network generalizes, we split the data into a training and testing set by a ratio of 9:1.

Normalization of Patches The training data was generated on a physical domain of size $\mathscr{X} \times \mathscr{Y} = [-2,2]^2$ with velocity components in the range $[-2,2]$. Since unseen data might be given on a different domain size with velocity magnitudes in very different orders of magnitude, we follow [26] and normalize the input data to the same domain that was used during training by applying an appropriate scaling and shifting.

3.4 Deep Learning for Vortex Boundary Identification

We aim to compare three common convolutional neural network architectures, i.e., a conventional convolutional neural network (CNN), a Resnet [20], and a Unet [37], that are trained to generate a binary segmentation from a given velocity field. This section describes the three networks and their respective hyper-parameters used in our study. We assume that the velocity field is given in a near-steady reference frame, which can be achieved by estimating a feature flow field [42] (Galilean invariant), by linear optimization [2, 11, 13] (objective) or by deep learning [26].

CNN Our first network is a conventional convolutional neural network. Through the course over multiple convolutional layers (kernel size 3×3 and stride 2×2), the patch resolution is reduced while the number of feature maps increases from 64 to 128. As activation, we chose a rectified linear unit (ReLU) and applied batch normalization. Once the feature maps are calculated, we combine the feature maps using fully-connected layers. As before, batch normalization and ReLU activations are applied. Finally, to avoid overfitting, a dropout layer with 0.5% chance is added to the network. The number of neurons of the last layer is set to match the target resolution, here, the initial 64×64 target patch.

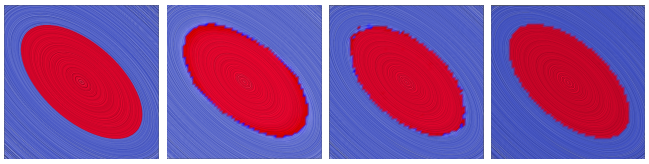
Resnet Our Resnet model is based on the Resnet20 architecture by He et al. [20]. The core idea of ResNet is to introduce an identity shortcut connection or skip connection that skips one or more layers. In our implementation, we use a learning rate 0.001 and place 6 residual blocks. Each block includes three convolutional layers.

Unet Our Unet model is based on the original model proposed by Ronneberger et al. [37]. In addition to skip connection, it also includes a concatenation with the correspondingly cropped feature map from the contracting path. In the end, a 1×1 convolutional layer is used to make the number of feature maps equal to the number of segments which are desired in the output. We use a depth of three and a dropout of 1%. The activation function is set to ReLu and we use max pooling between the layers.

Implementation We implemented all of our models using Keras [7] with Tensorflow [1] as backend. We applied the Adam optimizer [27] and trained for 100 epochs with a learning rate of 0.001. As loss function, we applied a binary cross entropy loss. For Resnet, the learning rate is varied based of the epoch number. We set the learning rate equal to 0.001 until epoch number 80 and after that it decreased to 1e-4. Throughout the training, we set the batch size to 256. During training, the loss may temporarily increase. Thus, we eventually select the model with the smallest testing error. We used ParaView to create the visualizations shown in this paper.

4 Results

We evaluate the method on both synthetically generated and numerically simulated data that has not been seen during training at the task of vortex boundary extraction. Subsequently, we analyze the performance.



(a) Ground truth (b) CNN (c) Resnet (d) Unet

Figure 2: Vortex boundary extraction results on two test split samples (rows), i.e., unseen patches of our synthetic vector field. The L2 errors are 0.12 (CNN), 0.06 (Resnet) and 0.02 (Unet) for all three architectures, respectively. The result of the Unet architecture matches the ground truth best.

	CNN	Resnet	Unet
TP	0.24015 (0.04508)	0.23388 (0.04395)	**0.24236** (0.04571)
TN	0.75435 (0.04657)	0.71564 (0.04063)	**0.75582** (0.04611)
FP	0.00288 (0.00003)	0.04160 (0.00317)	**0.00142** (0.00002)
FN	0.00237 (0.00023)	0.00864 (0.00016)	**0.00016** (0.00000)

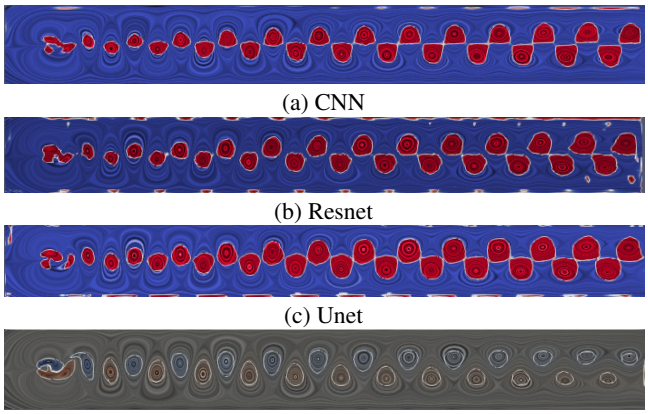
Table 1: Average true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for 200 random samples of the test set. The standard deviation is reported in brackets. The scores of the best network are highlighted **bold**.

4.1 Testing on Synthetic Data

We used our synthetic vector field patches to train a CNN, a Resnet, and a Unet, as discussed in Section 3.4, using the velocity vectors as the input. To evaluate their accuracy, we applied these three trained networks to the extraction of the vortex boundaries from a number of unseen synthetic flows, generated with our model. Figure 2 compares the results obtained from these three networks. From these results, we see that the Unet obtains a boundary closer to the ground truth (i.e., the white ellipses) than the other two networks. This is also supported by the quantitative evaluation where the L_2 errors for CNN, Resnet, and Unet are 0.12, 0.06 and 0.02, respectively, and their F-scores are 0.990, 0.924, and 0.997, respectively. The average true positive (TP), true negative (TN), false positive (FP), and false negative (FN) vortex detections for 200 random samples of the test set are reported in Table 1. Unet achieved the highest TP rate, whereas Resnet achieved the highest TN rate. Differences were minimal. In contrast to the neural network approaches, region-based approaches such as the IVD method [16] require a threshold to be set. The threshold that leads to an isoline that matches the ground truth, i.e., the tangential velocity extremal line, is different for each synthetic patch, since the angular momentum of the vortices varies. Thus, there is no single threshold that works for all vortices.

4.2 Testing on Numerical Data

Since the network was trained on synthetic data, we will now test our method on unseen numerical data. For this, we report our results on the 2D flow behind a cylinder, referred to as the Cylinder flow. The simulation was carried out with Gerris [35] and contains a viscous fluid that was injected from the left into a domain bounded by solid walls with slip boundary condition. The flow has a Reynolds number of $Re = 160$, which causes the flow to form the characteristic von-Kármán vortex street. Since vortices move with almost constant speed, we could subtract a constant ambient motion to obtain a near-steady reference frame. In our analysis, we took a time step where the vortex shedding is fully formed. The core region of a vortex in this flow has a motion close to that of a rigid body rotation, which helps to preserve the shape of the vortex. However, the concentrated vorticity in the vortex cores will diffuse due to viscosity (i.e., friction) and the absence of external forces to maintain the rotation. Standard threshold-based vortex measures are unable to faithfully capture



(a) CNN

(b) Resnet

(c) Unet

(d) IVD

Figure 3: Comparison of boundary extraction with cnn (a), Resnet (b), and Unet(c) using the CYLINDER flow. The input of networks are velocity patches. Our method shows Unet outperforms the other networks. (d) shows the IVD result with threshold value of ± 0.03.

the size of the vortices throughout the entire domain, when using a single threshold.

For this flow, we generate $100k$ synthetic flow patches using the fitted parameter ranges and distributions to train the three neural networks (Section 3.4), including the velocity field and the binary segmentation for each patch. We use a binary cross entropy as loss function. The training error plots of the three networks are provided in the supplemental document. We terminated the training after 100 epochs, since further improvements were negligible.

Next, we apply the three trained neural networks to the CYLINDER flow. Fig. 3 shows the results obtained using the three networks. The identified vortex boundaries are represented by the white curves, while the color plots show the classification of the individual points in the domain – red stands for inside the vortices while blue for outside. From these results, we can see that all three networks can highlight flow regions overlapping with places with possible vortical flow as suggested by the patterns of the LIC texture but with varying shapes. Visually, the vortex boundaries identified by the trained Unet are smoother than the other two results, and their shapes are more aligned with the expected shapes of the vortices for this flow based on our knowledge. More importantly, the small vortices right behind the cylinder are well separated in the Unet result, while in the other two results, they are not separable and form rather unnatural shapes. Further, the Resnet model generated small false positives at the right end of the domain.

Comparison with the thresholding method. Thresholding of certain attribute fields is a popular approach to identify vortices. Regions with values above (or below) certain threshold values are considered within vortices. The attributes that are often used for vortex extraction include vorticity, Q-criterion [23], and λ_2 [24]. In what follows, we compare our results with an objective vortex measure, named instantaneous vorticity deviation (IVD) [16].

Fig. 3 (d) shows the iso-contours computed based on the IVD of this flow. For IVD, we used a local neighborhood of 5×5 voxels. As mentioned earlier, our method not only captures all vortices behind the cylinder, it also reveals more accurate physics of the vortex shedding of this flow. In particular, the vortices extracted using our trained Unet exhibit increasing sizes when moving to the right of the domain. This accurately represents the diffusion of the vorticity concentration (also shown in [25]). In contrast, the vortices identified using the iso-contours of the vorticity field are shrinking when moving to the right end due to the decrease of the vorticity concentration caused by the diffusion. In addition, the vortices obtained using the iso-contouring of the IVD field exhibit shearing behavior around the cylinder, indicated by their shapes with small tails. In contrast, the vortices extracted using our method have more elliptical shape behind the cylinder.

Model	Training time [hrs]	Inference time [secs]
CNN	0.9	62
Resnet	9.6	42
Unet	17.6	42

Table 2: Training time of the networks (left column) and inference time of 500 patches (64×64) (right column). The Unet trains almost twice as long as Resnet, but achieves equal inference time.

We also wish to point out that Deng et al. [8] trained a CNN for binary vortex classification by predicting a thresholded IVD field using a threshold that was arbitrarily selected at training time. If the network is trained well, they inherit the properties and thereby also the limitations of IVD. Our approach, on the other hand, detects increasing vortex sizes the longer the vortices were diffusing their angular momentum.

Performance. We trained the neural networks on an NVIDIA Tesla K40m GPU. The training time and the inference time are reported for all three networks in Table 2. While CNN was with only 1 hour training time the fastest to train, we could see that it performed worst. The Resnet trained for about 10 hours and Unet for about 18 hours. The inference time of a 64×64 patch was for all networks similar, ranging from 42 seconds (Resnet, Unet) to 62 seconds (CNN). The differences in the training time stem mainly from the deeper networks for Resnet and Unet. Deep convolutional networks without skip connections have difficulties to learn, since back-propagation suffers from vanishing gradients in the early layers, which leads to slow learning.

5 Conclusion and Future Work

Automatic extraction of vortex boundary is challenging. To address that, we took a machine learning (ML) approach. Different from other ML methods, we use a synthetic model to generate a large training set with various vortices whose boundaries can be labeled automatically. We applied the generated large training set to train three different neural networks, including a CNN, a Resnet, and a Unet. We evaluated the trained networks on unseen synthetic data and numerically simulated data. We found that Unet outperforms the other two networks in the accuracy of the vortex boundary extraction. Compared to threshold-based methods that rely on vorticity, we can detect forming vortices directly in the wake of the cylinder, a regime that exhibits shear flow, which creates false positives in vorticity. Further, our vortex detection shows that the size of vortices increases down the flow, while their angular momentum decreases, resulting in smaller vorticity values.

In our work, we determined the distribution of model parameters by fitting the parametric model to numerical data. Numerical simulations with other data characteristics will therefore not be part of the parameter space seen by the network. In order to increase the parameter space, we sampled the Gaussian of each parameter independently. While this allows for more combinations that other flows might assume, it also creates configurations that are unlikely to happen in practice. To improve the method, we would like to train on a wider range of fluid flows for the fitting process, while also explicitly modeling the correlation between the individual Gaussian distributions. We will also include multiple vortices in our synthetic model to better capture vortex interactions. Further, we would like to investigate the parameterization of the synthetic model more, based on larger collections of laminar and turbulent flows. Lastly, it would be interesting to turn the binary segmentation problem into a multi-class segmentation problem, accounting for different vortex scales. For this, it will be difficult to obtain a ground truth.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This research was partially supported by NSF IIS 1553329 and by the Swiss National Science Foundation (SNSF) Ambizione grant no. PZ00P2_180114.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[2] I. Baeza Rojo and T. Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization)*, 26(1):280–290, 2020.

[3] X. Bai, C. Wang, and C. Li. A streampath-based RCNN approach to ocean eddy detection. *IEEE Access*, 7:106336–106345, 2019. doi: 10.1109/ACCESS.2019.2931781

[4] D. C. Banks and B. A. Singer. Vortex tubes in turbulent flows: Identification representation, reconstruction. Technical report, 1994.

[5] D. Bauer, R. Peikert, M. Sato, and M. Sick. A case study in selective visualization of unsteady 3d flow. In *Proceedings of the conference on Visualization'02*, pp. 525–528. IEEE Computer Society, 2002.

[6] T. Bin and L. Yi. CNN-based flow field feature visualization method. *International Journal of Performability Engineering*, 14(3):434, 2018.

[7] F. Chollet et al. Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k*, 7(8):T1, 2015.

[8] L. Deng, Y. Wang, Y. Liu, F. Wang, S. Li, and J. Liu. A CNN-based vortex identification method. *Journal of Visualization*, 22(1):65–78, 2019.

[9] K. Franz, R. Roscher, A. Milioto, S. Wenzel, and J. Kusche. Ocean eddy identification and tracking using neural networks. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 6887–6890. IEEE, 2018.

[10] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface techniques for vortex visualization. In *VisSym*, vol. 4, pp. 155–164, 2004.

[11] T. Günther, M. Gross, and H. Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics (TOG)*, 36(4):141, 2017.

[12] T. Günther and H. Theisel. The state of the art in vortex extraction. In *Computer Graphics Forum*, vol. 37, pp. 149–173. Wiley Online Library, 2018.

[13] M. Hadwiger, M. Mlejnek, T. Theußl, and P. Rautek. Time-dependent flow seen through approximate observer killing fields. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1257–1266, 2019.

[14] G. Haller. An objective definition of a vortex. *Journal of Fluid Mechanics*, 525:1–26, 2005.

[15] G. Haller. Lagrangian coherent structures. *Annual Review of Fluid Mechanics*, 47:137–162, 2015.

[16] G. Haller, A. Hadjighasem, M. Farazmand, and F. Huhn. Defining coherent vortices objectively from the vorticity. *Journal of Fluid Mechanics*, 795:136–173, 2016.

[17] J. Han, J. Tao, and C. Wang. FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2880207

[18] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang. Flow field reduction via reconstructing vector data from 3-D streamlines using deep learning. *IEEE computer graphics and applications*, 39(4):54–67, 2019.

[19] J. Han and C. Wang. TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2019)*, 2020.

[22] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. In *Computer Graphics Forum*, vol. 35, pp. 643–667. Wiley Online Library, 2016.

[23] J. Hunt. Vorticity and vortex dynamics in complex turbulent flows. *Transactions of the Canadian Society for Mechanical Engineering*, 11(1):21–35, 1987.

[24] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

[25] J. Kasten, J. Reininghaus, I. Hotz, and H.-C. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *Transactions on Visualization and Computer Graphics (Vis'11)*, 17(12):2080–2087, 2011.

[26] B. Kim and T. Günther. Robust reference frame extraction from unsteady 2d vector fields with convolutional neural networks. *Computer Graphics Forum (Proc. EuroVis)*, 38(3), 2019.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] P. A. Lagerstrom. Solutions of the navier–stokes equation at large reynolds number. *SIAM Journal on Applied mathematics*, 28(1):202–214, 1975.

[29] R. Lguensat, M. Sun, R. Fablet, P. Tandeo, E. Mason, and G. Chen. Eddynet: A deep neural network for pixel-wise classification of oceanic eddies. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 1764–1767. IEEE, 2018.

[30] Y. Liu, Y. Lu, Y. Wang, D. Sun, L. Deng, F. Wang, and Y. Lei. A cnn-based shock detection method in flow visualization. *Computers & Fluids*, 2019. doi: 10.1016/j.compfluid.2019.03.022

[31] A. Okubo. Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences. In *Deep sea research and oceanographic abstracts*, vol. 17, pp. 445–454. Elsevier, 1970.

[32] R. Peikert and M. Roth. The "parallel vectors" operator: a vector field visualization primitive. In *VIS '99: Proceedings of the conference on Visualization '99*, pp. 263–270. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[33] C. Petz, J. Kasten, S. Prohaska, and H.-C. Hege. Hierarchical vortex regions in swirling flow. In *Computer Graphics Forum*, vol. 28, pp. 863–870. Wiley Online Library, 2009.

[34] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matkovic, and H. Hauser. The state of the art in topology-based visualization of unsteady flow. *Computer Graphics Forum*, 30(6):1789–1811, September 2011.

[35] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004.

[36] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualization: feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, Dec. 2003.

[37] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

[38] C. M. Ströfer, J. Wu, H. Xiao, and E. Paterson. Data-driven, physics-based feature extraction from fluid flow fields. *arXiv preprint arXiv:1802.00775*, 2018.

[39] D. Sujudi and R. Haimes. Identification of swirling flow in 3-d vector fields. In *12th Computational Fluid Dynamics Conference*, p. 1715, 1995.

[40] G. H. Vatistas, V. Kozel, and W. Mih. A simpler model for concentrated vortices. *Experiments in Fluids*, 11(1):73–76, 1991.

[41] Y. Wang, L. Deng, Z. Yang, D. Zhao, and F. Wang. A rapid vortex identification method using fully convolutional segmentation network. *The Visual Computer*, pp. 1–13, 2020.

[42] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege. Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2007)*, 13(6):1759–1766, November – December 2007.

[43] J. Weiss. The dynamics of enstrophy transfer in two-dimensional hydrodynamics. *Physica D: Nonlinear Phenomena*, 48(2-3):273–294, 1991.