Binary Maps for Image Separation in Iterative Neuronal Network Applications

Roman Lehmann, Stanislav Arnaudov, Markus Hoffmann, and Wolfgang Karl

Karlsruhe Institute of Technology, Institute of Computer Engineering, Kaiserstraße 12, 76131 Karlsruhe

Abstract Generating a series of images is an important task in various fields of scientific research, e. g. Computational fluid dynamics (CFD). In the past years, solutions based on deep neural networks gained importance. In these tasks, it's often necessary to declare regions of interest in the image. Furthermore, the NN should only perform on these regions and the rest should be ignored. With this paper, we propose an innovative and easy method for implementing this behavior in the field of CFD.

Keywords U-net, binary maps, generator, flow simulation

1 Introduction

Generating a series of images is an important task in various fields of scientific research, e. g. Computational fluid dynamics (CFD). In the past years, solutions based on deep neural networks gained importance [1], for example in applications where the results don't need to be fully accurate. For these tasks, it's often necessary to declare regions of interest (ROI) in the image to preserve constant regions and concentrate the influence of the neuronal network on a specific area. This becomes even more important in cases where results of a neuronal network are used as an input again. We call these cases iterative applications.

For CFD applications this issue is related to the sharp separation of the simulation area and its boundary. It is essential that the fluid simulation does not ignore boundaries, like obstacles within the stream,

and that these boundaries do not introduce false interferences into the simulated stream. Using an image-to-image approach [2] to create a sequence of simulation steps, an obvious idea to define a sharp separation is the usage of binary maps. Such maps define a region of interest within a picture with a true value for the corresponding pixel and false otherwise. In combination with a neuronal network, these maps can be used as an additional parameter track for the network and as a filter for a post image processing step. We will show that both applications are needed in order to get good predictions for the simulation results with a sharp separation of the simulation area and its boundary.

2 State of the art

The main task in our approach is an image-to-image translation. By now the image-to-image translation through CNNs is well established and has found numerous applications [3–6]. [2] has specifically stated how the "community-driven research" has popularized their work by applying it in different ways [7–9]. We see our work as another demonstration of [2]. This time in the context of CFD.

The field of CNNs provides various approaches of handling with ROIs. [10, 11] use different NNs for generating and applying the ROIs. This leads to results with a probability, which is desired in the given tasks, but not in ours. Other works like [12, 13] use binary mask to define ROIs. But they use these mask as an pre image processing step only. Our application of the binary map goes further with respect to the combined application.

3 Methodology

The task is to build a network that can predict the next frame of a 2D flow simulation based on the previous one. Our focus of this work is on the boundaries of the simulation area, obstacles for example, and their stability in iterative evaluations of the network. Each frame represents a time step of the simulation and consists of a three-channel image. Two of the channels encode the velocity fields in x- and y-direction and the third channel is the pressure field of the fluid. For

this paper we did not construct a single holistic model that can handle all the simulation's parameters. Our effort is concentrated on taking a relative simple model and investigate the influence of the application of binary maps. We call this simple model the *constant model* because we do not vary simulation parameters like the inflow speed.

3.1 Simulation setup and data generation

The training data was generated by performing simulations of in-compressible fluid flow around a rectangular object in a channel. The simulations are modelled according to the Navier-Stokes equations for in-compressible flow. Because we are interested in the image representations of the simulations, we are dealing only with the 2D case. Several boundary conditions describe the simulation setup:

- Inflow condition on the left side of the channel
- Outflow condition on the right side of the channel
- No-slip condition on the bottom and top side of the channel as well as the sides of the object.

The simulation setup has three separate adjustable parameters: inflow speed g, fluid density ρ and fluid kinematic viscosity v. For the constant model we took the simulation with $\rho = 0.2$, $v = 0.0009$ and $g = 1.5$. The choice of the parameter is deliberate. The values are chosen so that the Reynolds number [14] of the simulations in the range of $[90, 450]$. We were interested whether the build models can predict the emerging *Kármán vortex street* [15]. Thus, the Reynolds numbers were chosen so that the effect can occur.

The simulations were performed numerically by solving the differential equation describing the flow – the Navier-Stokes equations. This was done with a numerical solver library – *HiFlow*^{3} [16] – that works on the base of the finite element method [17]. The time step for the solver was set to 0.035 seconds. This means a single time step of the simulation corresponds to 0.035 seconds of physical time.

The numerical solver library on itself cannot be used to render the simulation results to images. For this reason, we used *ParaView* [18]

to load the simulation data and exported it as a sequence of images in *PNG* format. We used the default "Grayscale" color preset of ParaView to visualize the results. Each frame of the simulation was exported as three separate grayscale images. Finally, the images were cropped to select a subset of the space that contains the object and space behind it. For training the neuronal network, we rendered 1904 frames of the simulation (66 seconds of the simulated physical time).

After all images were generated, a test-train split was created. The split was done by random and resulted in 80% of the data was used for training and the rest for testing.

The binary map was created be locating the obstacle and set the size to the same length and width like the other images.

3.2 Training approach and network details

We based our generative models almost entirely on [2]. We use the conditional GAN approach to train a generator network that can perform image-to-image translation. As explained in [2], the traditional GAN method uses a random vector z as an input to the generator network G to generate output y, $G : z \to y$. Conditional GANs also feed an input image x to the generator, $G : x, z \to y$. [2] and [19] suggest that in certain cases the usage of z can be usefully, but we decided not to include for our generator as we want a deterministic network. The discriminator network is modelled with the function $D : x, y \to v$ that evaluates the likelihood of y being a real image. To note is that the discriminator network has access to the real image x and tries to guess, if y is the real or generated output.

We adopt the objective function of the discriminator network and we modify it slightly by leaving out the random vector z.

$$\begin{aligned} \mathcal{L}_{cGAN}(G, D) &= (\mathbb{E}[\log D(x, y)] + \mathbb{E}[\log D(x, G(x))])/2 \\ &= (\log D(x, y) + \log D(x, G(x)))/2 \end{aligned} \tag{3.1}$$

where x is the input image and y is the target image. We leave out the expected value calculation as we do not use the random vector z in our loss function. In contrast to unconditional GANs, both the generator and the discriminator network have access to the input

image. The objective is divided by two to slow down the training of the discriminator relative to the generator as suggested by [2].

The objective for the generator network is composed of two parts — the value of the discriminator as well as a $L1$ distance loss between the target and the predicted images. According to [2] the $L1$ loss promotes less blurring and captures the low frequency details of the images. The $L1$ loss is given by:

$$\mathcal{L}_{L1}(G) = \mathbb{E}[\|y - G(x)\|_1] \tag{3.2}$$

The final object for the generator is thus:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN} + \lambda \mathcal{L}_{L1}(G) \tag{3.3}$$

For all models we used $\lambda = 100$ as done in [2].

3.3 Network architecture

For our generator we use the U-Net [20] variant proposed in [2]. It is a standard encoder-decoder [21] model with skip connections between parts of the encoder and the decoder. Our network uses blocks of layers of the form convolution-normalization-ReLu [22]. The encoder-decoder first downsamples the input till a bottleneck layer is reached and what follows is an upsampling to the original size of the input image.

For the discriminator, we follow the method of [2] and we use their PatchGAN discriminator network. This is a convolutional network that classifies patches of the input as real or predicted. To note is that the whole image is given as an input. The majority of the results in [2] show that patches of size 70×70 yield the best results but in our case, the experiments showed otherwise. We, therefore, we opted out for using patches of size 286×286 pixels.

3.4 Training details

We trained the model with the generated dataset. When loading the images in memory, we first resize them to an appropriated for a network size of 1024×256 (width $\times$ height). Then we apply random

crops as well as add random noise to each channel of the images. We do this to force the generator to learn the actual features of the simulation and make over-fitting harder. To investigate the effect of using a binary map to determine the obstacle. We developed four different training:

- **no-mask:** no binary mask is used at all
- **no-mask-after:** the binary mask is multiplied to the input image. The binary mask itself is also fed as additional input into the generator network but not multiplied with the predicted image.
- **no-mask-before:** only the predicted image is multiplied with the binary image. No binary mask is fed into the network or is multiplied to the input image.
- **mask:** the binary mask is multiplied to the input image. The binary mask itself is also fed as additional input to the generator network. The predicted image is multiplied with the binary image, too.

The binary map as additional input gives the network the information where the obstacle is. The zeroed values can't provide this information due the grayscaled image.

For the training procedure, we follow the standard approach in [23]. With each mini-batch, we first optimize the discriminator and then the generator with the discussed objectives. We use Stochastic Gradient Descent [24] with the Adam optimizer [25] with a learning rate of 0.0002 and standard momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The used batch size for the constant model was set to 3. Those are relatively small numbers for batch sizes but [2] suggests that the U-Net architecture benefits from small batches in image-to-image translation problems.

The constant model was trained for 45 epochs and evaluated on a single Nvidia GTX 980Ti GPU. The Implementation of the models was done in *PyTorch* [26] python library for machine learning.

4 Results

At this point we want to mention why the following results are showing the beginning of the vortex street and not a fully distinctive turbulent flow. The reason can be found in the training data and the very short amount of time the vortex street needs to establish within the stream. Therefore, the neuronal network is well-trained to predict the continuation of the distinctive turbulent flow but less highly trained for the first simulation steps where the vortex street is establishing. That is why prediction problems have a higher impact on the first steps and are therefore more visible in these images, although the same problems can be observed in all simulation steps as shown later on.

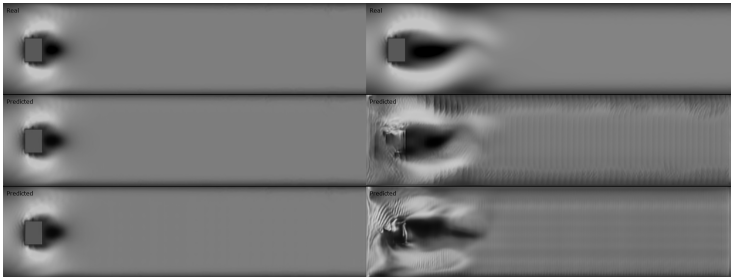

Figure 4.1: 1. Line: Step 1 and step 20 of a finite element simulation, 2. Line: Predicted step 1 and step 20 without the usage of the pressure field, 3. Line: Predicted step 1 and step 20 with usage of the pressure field; No binary mask used, x-velocity shown

We start with the mask-free prediction. Figure 4.1 shows what happens: The obstacle vanishes within the stream and this has in return a bad impact on the stream itself. Even adding more information by using the pressure field of the stream in addition to the velocity field for prediction isn't a solution.

As the first step prediction seams to be useful, the intuitive next development is to multiply every prediction with the binary mask before using it iteratively as the new input data. Results for that are shown in figure 4.2. One can see, that this idea also leads to insuffi-

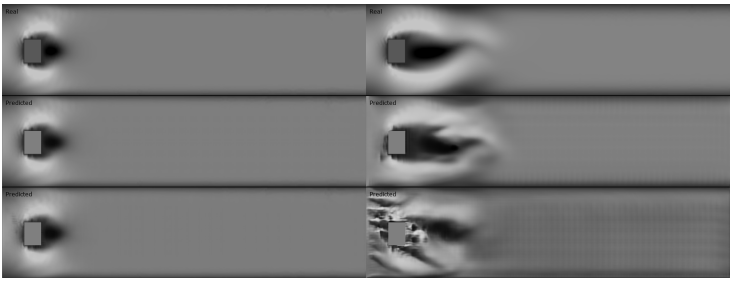

Figure 4.2: 1. Line: Step 1 and step 20 of a finite element simulation, 2. Line: Predicted step 1 and step 20 without the usage of the pressure field, 3. Line: Predicted step 1 and step 20 with usage of the pressure field; Binary mask used after prediction, x-velocity shown

cient results. Adding more information with the pressure field even produces worse results with respect to the accuracy of the stream.

After observing that the simple post image processing step isn't the solution, we turned it the other way round and set the binary map as an additional data stream for the neuronal network. The idea here is that the network is able to learn the sharp separation with the help of this map. In figure 4.3 it is obvious that this isn't the right way either.

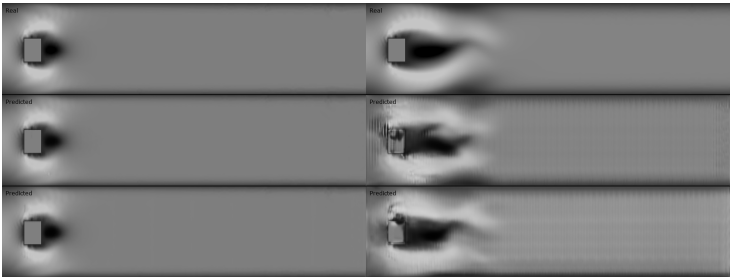

Figure 4.3: 1. Line: Step 1 and step 20 of a finite element simulation, 2. Line: Predicted step 1 and step 20 without the usage of the pressure field, 3. Line: Predicted step 1 and step 20 with usage of the pressure field; Binary mask used within neuronal network, x-velocity shown

Combining both approaches, adding the binary map to the neuronal network and using it for post-processing the result, is the next logical step at this point. Figure 4.4 shows, that this approach preserves the obstacle perfectly and results in good predictions. There are relics on the image, but they are very homogeneous and can be filtered with common image processing steps like opening and closing. The stream itself is in both predictions very close to the numerical simulation.

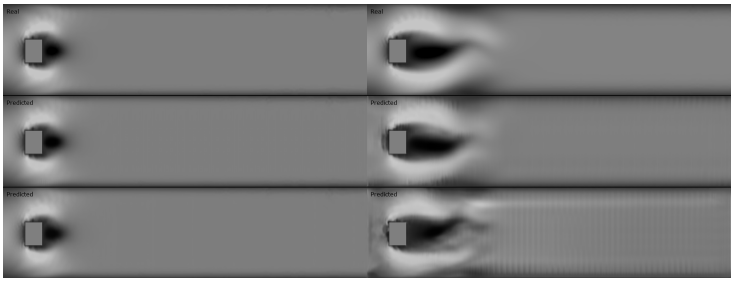

Figure 4.4: 1. Line: Step 1 and step 20 of a finite element simulation, 2. Line: Predicted step 1 and step 20 without the usage of the pressure field, 3. Line: Predicted step 1 and step 20 with usage of the pressure field; Binary mask used combined, x-velocity shown

For a real quality quantification we used a measurement to compare different images with the focus on the human observer. Implying that the result doesn't have to be fully accurate, we used the Peak Signal Noise Ratio (PSNR) as the metric. It is connected to the mean square error (MSE) in the following way:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{255}{\mathrm{MSE}}\right) \quad [\text{decibel}]. \tag{4.1}$$

Higher values are connected to less observable differences, in general a PSNR over 30 means that the human eye cannot detect any difference [27, 28]. We started the PSNR evaluation at simulation step 90 to show that even in the well-trained time steps of the simulation where the vortex street is completely visible a relevant difference is measurable. Figure 4.5 not only shows that the combined approach results in the best predictions but also that a bad application of the

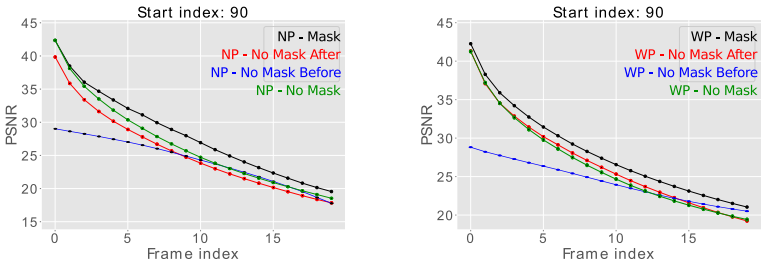

Figure 4.5: Left: PSNR values for 20 iterative steps, starting with step 90, no pressure field used; Right: Added pressure field.

binary map can result in even worse predictions than applying no binary map.

5 Summary

Defining regions of interest with the help of binary masks for iterative neuronal network applications like predicting CFD results is an important issue for such predictions. As seen in figure 4.1 to 4.4 applying no binary mask leads to wrong results very quickly. Applying only one approach — train the mask or using it as a post-processing step — can preserve the obstacle but cannot avoid interferences on the stream. Only applying both strategies results in appropriate predictions even when more information, like the pressure field, is used. The PSNR values in figure 4.5 are showing that this is even true for very well-trained parts of the simulation. This figure also demonstrates that a wrong application of a binary mask can lead to worse predictions than applying no mask at all. Therefore, we suggest a combined application of a binary mask for iterative network applications with sharp separations of regions of interest.

References

1. O. Hennigh, “Lat-net: Compressing lattice boltzmann flow simulations using deep neural networks,” 2017.

2. P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016. [Online]. Available: http://arxiv.org/abs/1611.07004

3. B. Zhao, W. Yin, L. Meng, and L. Sigal, "Layout2image: Image generation from layout," *International Journal of Computer Vision*, pp. 1 – 18, 2020.

4. Y. Liu, Z. Qin, Z. Luo, and H. Wang, "Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks," *CoRR*, vol. abs/1705.01908, 2017. [Online]. Available: http://arxiv.org/abs/1705.01908

5. M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2018–2025.

6. T. Park, M. Liu, T. Wang, and J. Zhu, "Semantic image synthesis with spatially-adaptive normalization," *CoRR*, vol. abs/1903.07291, 2019. [Online]. Available: http://arxiv.org/abs/1903.07291

7. S. Moschoglou, S. Ploumpis, M. Nicolaou, A. Papaioannou, and S. Zafeiriou, "3dfacegan: Adversarial nets for 3d face representation, generation, and translation," *ArXiv*, vol. abs/1905.00307, 2019.

8. B.-K. Kim, G. Kim, and S.-Y. Lee, "Style-controlled synthesis of clothing segments for fashion image manipulation," *IEEE Transactions on Multimedia*, vol. 22, pp. 298–310, 2020.

9. S. S.-C. Chen, H. Cui, M. Du, T. Fu, X. S. Sun, Y. J. Ji, and H. Duh, "Cantonese porcelain classification and image synthesis by ensemble learning and generative adversarial network," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, pp. 1632 – 1643, 2019.

10. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

11. L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667.

12. J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," *CoRR*, vol. abs/1412.1283, 2014. [Online]. Available: http://arxiv.org/abs/1412.1283

13. S. Eppel, "Setting an attention region for convolutional neural networks using region selective features, for recognition of materials within glass vessels," *arXiv preprint arXiv:1708.08711*, 2017.

14. K. T. Trinh, "On the critical reynolds number for transition from laminar to turbulent flow," 2010.

15. T. v. Kármán, "Ueber den mechanismus des widerstandes, den ein bewegter körper in einer flüssigkeit erfährt," *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, vol. 1911, pp. 509–517, 1911. [Online]. Available: http://eudml.org/doc/58812

16. S. Gawlok, P. Gerstner, S. Haupt, V. Heuveline, J. Kratzke, P. Lösel, K. Mang, M. Schmidtobreick, N. Schoch, N. Schween, J. Schwegler, C. Song, and M. Wlotzka, "Hiflow3 – technical report on release 2.0," *Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)*, vol. 0, no. 06, 2017. [Online]. Available: https://journals.ub.uni-heidelberg.de/index.php/emcl-pp/article/view/42879

17. G. Strang and G. Fix, *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, 2008. [Online]. Available: https://books.google.de/books?id=K5MAOwAACAAJ

18. J. Ahrens, B. Geveci, and C. Law, "Paraview : An end-user tool for large data visualization," *Energy*, vol. 836, p. 717–732, 2005. [Online]. Available: http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:ParaView:+An+end-user+tool+for+large+data+visualization$\#$0

19. X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," 2016.

20. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

21. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. [Online]. Available: https://science.sciencemag.org/content/313/5786/504

22. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

23. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

24. J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952. [Online]. Available: http://www.jstor.org/stable/2236690

25. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

26. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

27. D. Mehra, "Estimation of the image quality under different distortions," *International Journal Of Engineering And Computer Science 8*, 2016.

28. Y. Shiao, T. Chen, K. Chuang, C. Lin, and C. Chuang, "Quality of compressed medical images," *Journal of digital imaging : the official journal of the Society for Computer Applications in Radiology 20*, 2007.