Received: 8 October 2022 | Accepted: 14 December 2022

DOI: 10.1002/pamm.202200241

Reduced order fluid modeling with generative adversarial networks

Mirko Kemna[1], **Alexander Heinlein**[1,*], and **Cornelis Vuik**[1]

[1] Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

Surrogate models based on convolutional neural networks (CNNs) for computational fluid dynamics (CFD) simulations are investigated. In particular, the flow field inside two-dimensional channels with a sudden expansion and an obstacle is predicted using an image representation of the geometry as the input. Generative adversarial neural networks (GANs) have been shown to excel at such image-to-image translation tasks. This motivates the focus of this work on investigating the specific effect of adversarial training on model performance. Numerical results show that the overall accuracy of the GANs is generally lower compared to an identical generator model trained directly on the ground truth using an L1 data loss. On the other hand, GAN predictions are often visually more convincing and exhibit a lower continuity residual.

1 Introduction

The recent rapid development in the field of machine learning (ML) holds a great potential for scientific machine learning (SciML), an emerging field dealing with the combination of scientific computing and machine learning. Both fields can be combined in various ways, for instance, by improving existing computational models using machine learning techniques or by enhancing machine learning algorithms using numerical methods. For an overview of the various aspects of SciML; see, e.g., [1]. Here, we are concerned with training neural networks (NNs) as reduced order surrogate models to replace high fidelity computational fluid dynamics (CFD) simulations with potentially very high computational cost. Compared to other popular approaches, such as physics-informed neural networks (PINNs) [2], in which the neural network is used to discretize one given boundary value problem (BVP), we train a single model to be able to predict the solutions of a given partial differential equation (PDE) on a whole range of geometries. This is a difficult task for classical model order reduction techniques (see, e.g., [3]), since the parameterization of our range of geometries would be challenging and would lead to a very large parametrization space.

Specifically, we are considering the task of mapping from rasterized two-dimensional (2D) geometries to the corresponding solutions of the (stationary) incompressible Navier–Stokes equations using convolutional neural networks (CNNs); see [4, 5]. This can be seen as an image-to-image translation task, at which so-called generative adversarial networks (GANs) have proven particularly successful; cf. [6]. GANs offer an indirect way of learning from data, i.e., by training the generating model in an *adversarial game* in which it competes against another neural network acting as a *critic*. This adversarial training imparts unique properties onto the generator and makes GANs an interesting object of study.

While previous studies found GANs to be successful as reduced order models for PDE problems, the specific impact of the adversarial training on the model performance has not been investigated in detail. In particular, to the best of our knowledge, a detailed comparison of a GAN model and a model directly trained on the data with the same network architecture cannot be found in the literature. Results for GAN models were either not compared to a baseline neural network at all [7, 8] or the baseline model had a different architecture from the GAN generator network [9], which makes it difficult to draw conclusions. This work tries to investigate the specific effect of the GAN training by investigating how the results obtained from GANs differ from those produced by identical generator networks trained directly on ground truth. Due to space limitations, we focus on our most important findings. The full numerical study and corresponding discussion can be found in the master's thesis [10].

The paper is structured as follows: First, the data generation pipeline and the CNN models employed in our study are described. Then, we present and discuss numerical results of our comparison of GAN and directly trained CNN models. Finally, we give a short conclusion of our findings.

* Corresponding author: e-mail a.heinlein@tudelft.nl

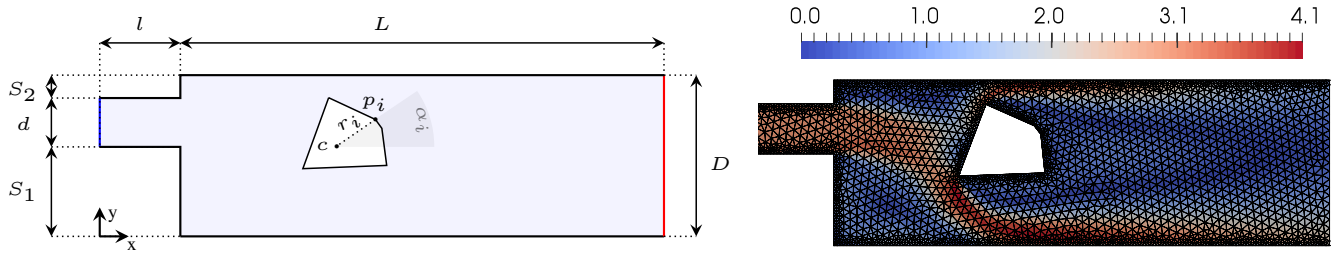

Fig. 1: Illustration of an exemplary case from the data set. **Left:** Channel geometry determined by the fixed lengths $L = 3\,\mathrm{m}, l = 0.5\,\mathrm{m}, D = 1\,\mathrm{m}$ and the variable lengths d, S_1, and S_2. The different parts of the boundary are color coded as inlet (blue), outlet (red) and wall (black). The obstacle is generated as a star-shaped polygon with a random number of vertices placed around a center point, with a vertical size not exceeding $1 - d$ to prevent overly large flow velocities. **Right:** Magnitude of the velocity field of the corresponding solution; the simulation mesh is locally refined close to the boundary.

	inlet	outlet	walls
U	$\mathbf{u} = U\hat{\mathbf{e}}_x$	$\partial_x \mathbf{u} = \mathbf{0}$	$\mathbf{u} = \mathbf{0}$
p	$\partial_x p = \mathbf{0}$	$p = \mathbf{0}$	$\partial_x p = \mathbf{0}$

Table 1: Boundary conditions imposed during the CFD simulations. Inflow speed is $U = 3\,\mathrm{m\,s^{-1}}$

2 Methodology

Data generation To train our reduced order NN model, we created reference data based on numerical simulations. For this purpose, we first algorithmically generated a set of 10 000 two-dimensional geometries; see Fig. 1 (left) for a sketch of an exemplary problem domain. Each domain consists of an inlet channel, opening into a larger channel section which contains a star-shaped polygonal obstacle. The obstacle as well as the position and width of the inlet are chosen randomly. Then, we spatially discretize the domain into an unstructured quasi-2D mesh of triangular prisms using the open source software Gmsh [11]. The size parameter was set to $2\,\mathrm{cm}$ at walls and $5\,\mathrm{cm}$ elsewhere, resulting mesh that is refined at the walls; see Fig. 1 (right).

Complemented by the boundary conditions listed in Table 1, each of the 10 000 cases defines a problem of stationary, incompressible flow through a sudden expansion. This scenario was chosen due to the interesting bifurcation behavior documented experimentally in [12] and reproduced successfully in our simulations; see Fig. 5. As the incoming flow exits the inlet, it will attach to the upper or lower wall of the large channel, depending on whichever is closer to the inlet. Thus, the center position of the inlet is a bifurcation point, which is useful for highlighting the qualitative differences between GAN models and directly trained ones; cf. in Section 3.

The stationary flow of an incompressible Newtonian fluid with density ρ and kinematic viscosity ν is described by the Navier-Stokes equations

$$\nu\,\Delta\mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{u} = \frac{1}{\rho}\nabla p, \tag{1a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{1b}$$

where $\mathbf{u}$ is the flow velocity and p is the pressure. We set $\rho = 1000\,\mathrm{kg\,m^{-3}}$ and $\nu = 10^{-5}\ \mathrm{m^2 s^{-1}}$, corresponding roughly to the properties of water at room temperature. If we take the inflow speed of $3\,\mathrm{m\,s^{-1}}$ and the inlet width of $0.5\,\mathrm{m}$ as the characteristic quantities, we obtain a Reynolds number of $\mathrm{Re} = 1.5\cdot 10^5$ and can therefore safely assume turbulent flow [13]. Hence, we additionally employ a turbulence and a boundary layer model, which greatly reduces the required resolution of the computational mesh (down to approx. 10 000 cells) compared to a direct numeral simulation (DNS). In particular, we use a Reynolds-averaging model with k-ε closure; see [14, p. 72].

We solve the Navier–Stokes equations Eq. (1) using the finite volume method (FVM) via the open source CFD toolbox OpenFOAM [15]. As the numerical scheme, we employ the Semi-Implicit Method For Pressure-Linked Equations (SIMPLE) iterative method first proposed in [16], which is implemented in the *simpleFoam* solver in OpenFOAM. The settings for the solvers and schemes are mostly kept the same as in the OpenFOAM tutorial case *backwardFacingStep2D*. As convergence criterion, the initial residual of the field equations is used with thresholds 2×10^{-4},1×10^{-2} and 2×10^{-3} for $\mathbf{u}$, p and others respectively. Convergence typically occurred after between 500 and 1500 iterations. Given the relatively high Reynolds range and the random geometry generation, the problems were not guaranteed to have a stationary solution. This would manifest as oscillation rather than convergence during solving. If convergence did not occur after 3000 steps, we omitted the corresponding case. Subject to the this, convergence typically took between 4 and 12 seconds on an Intel Core i7-6700K CPU at 4.00 GHz.

After a solution had been obtained on the computational mesh, we interpolated it onto a regular image grid with a resolution of 256×85 using the *cellPoint* method from the sampling utility of OpenFOAM. In order to focus on the most relevant fluid

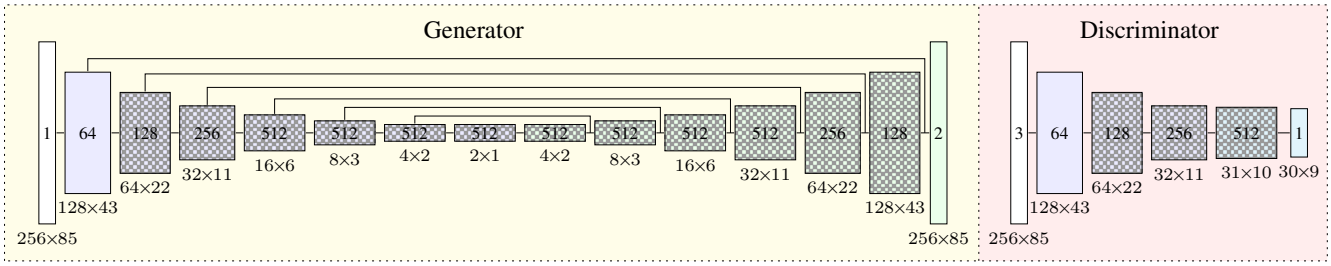

Fig. 2: Visualization of the GAN model architecture. Boxes represent layers of artificial neurons, lines represent connectivity. Information flows from left to right. Labels inside layers represent number of channels, labels beneath represent layer dimension. Blue, green and cyan represent down-, up- and unstrided convolutions, respectively. Hatched layers employ batch normalization.

dynamics and avoid an overly large aspect ratio, we restricted sampling to the first $3\,\mathrm{m}$ in x-direction. For an example, see Fig. 3.

Machine learning implementation We implemented the NN models using the open source machine learning library TensorFlow [17] in version 2.9. As a template for building a GAN fluid model, we used the *pix2pix* architecture, first presented in [6]. This GAN model was developed as a general purpose image-to-image translation tool, and has been successfully applied across a large variety of tasks.

The architecture of the generator and discriminator networks is visualized in Fig. 2. Both models are fully convolutional with 4×4 kernel size and no densely connected layers. Up- and down-sampling operations are 2-stride convolutions. Batch normalization is used in most hidden layers (hatched layers in Fig. 2). All hidden layers use parametric rectified linear units without bias as activation functions, with parameter 0 for up- and 0.3 for down-convolutions. The generator maps from a 256×256 image encoding the problem geometry to two output images of the same size, one for each velocity component. It has a *U-Net* type bottleneck architecture [18], such that each layer in the decoder operates on the previous layer as well as the equivalent layer from the encoder (through concatenation along the channel dimension). This should provide a useful prior given that a significant amount of low-level structure is shared between the input and output images. The output layer of the generator is linear. The discriminator maps from a $256 \times 256 \times 3$ tensor encoding geometry and the two velocity components to a 30×9 heat map of belief with range $(0, 1)$ (achieved through using a logistic activation at the output layer). The convolutional structure is chosen such that not every value in the input tensor affects every value in the output. Instead, each output pixel has a receptive field consisting of a $70 \times 70 \times 3$ batch of the input (or smaller close to the boundary). The authors of [6] termed this discriminator architecture *PatchGAN*, meaning that the discriminator cannot identify patterns larger than this patch size. Moreover, since the operation performed is the exact same for all patches, the discriminator output can be understood as a type of texture or style loss. All kernels are randomly initialized to a normal distribution with zero mean and a standard deviation of $\sigma = 0.01$.

The $10\,000$ training samples were split into training and test sets in a $9:1$ ratio. Since no major hyperparameter optimization was done, no validation set was used. Training was done in mini-batches of size 32 using the Adam (adaptive moment estimation) optimizer [19] for both generator and discriminator, with a learning rate of 10^{-4} and decay rate parameters of 0.5 and 0.999 for 1st and 2nd moment estimates respectively. As mentioned above, we compare performance outcomes between adversarial training and training on ground truth. The adversarial loss is based on cross-entropy of the discriminator output, whereas the ground truth loss is measured in the L1 metric. The generator and discriminator loss functions are given by

$$\mathcal{L}_{\text{gen}} = \lambda_{\text{GAN}} \log D(\mathbf{x}, \mathbf{c}) + \lambda_{\text{L1}} \overline{\|G(\mathbf{c}) - \mathbf{x}\|_1} \tag{2a}$$

$$\mathcal{L}_{\text{disc}} = \lambda_{\text{GAN}} \left(\log D(\mathbf{x}, \mathbf{c}) + \log \left(1 - D\left(G(\mathbf{c}), \mathbf{c} \right) \right) \right), \tag{2b}$$

with functions G and D correspond to the generator and discriminator, respectively, $\mathbf{c}$ and $\mathbf{x}$ representing the geometry input (conditional) and the ground truth velocity fields respectively, as well as weight parameters $\lambda_{\text{GAN}} > 0, \lambda_{\text{L1}} > 0$. Using the L2 norm instead of L1 could be an alternative, perhaps more natural choice for this problem, however, we observed numerical instabilities in the gradient calculation when using an L2 loss function.

Note that modifications to the architectures and training methods described above were also tested, including but not limited to, increasing the number of layers in the discriminator (and thereby also patch size), increasing kernel size, and changing the relative training rates between generator and discriminator. We will not discuss these results for the sake of brevity, but they are overall similar and support the statements and conclusions made in the remainder of the paper.

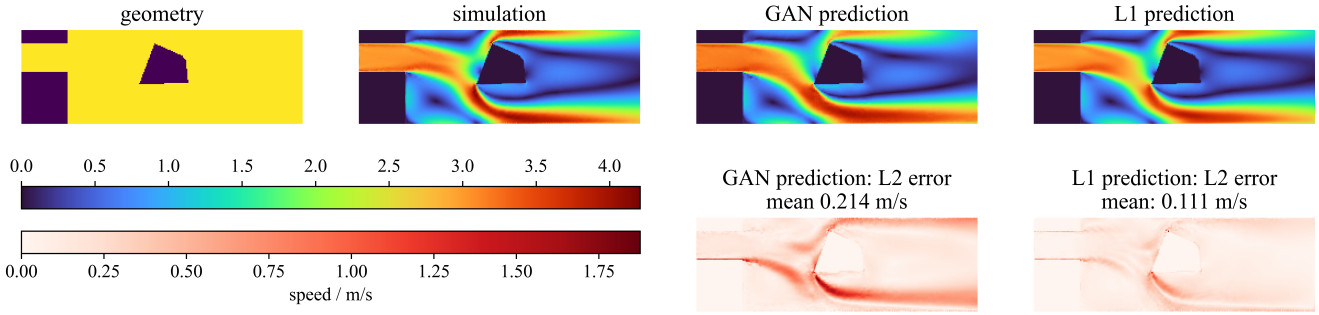

Fig. 3: Comparison between the velocity predictions of the two model types for the example geometry from Fig. 1 shown alongside the ground truth and geometry input. Lower row shows L2 error fields. Both models are trained for 200 000 steps.

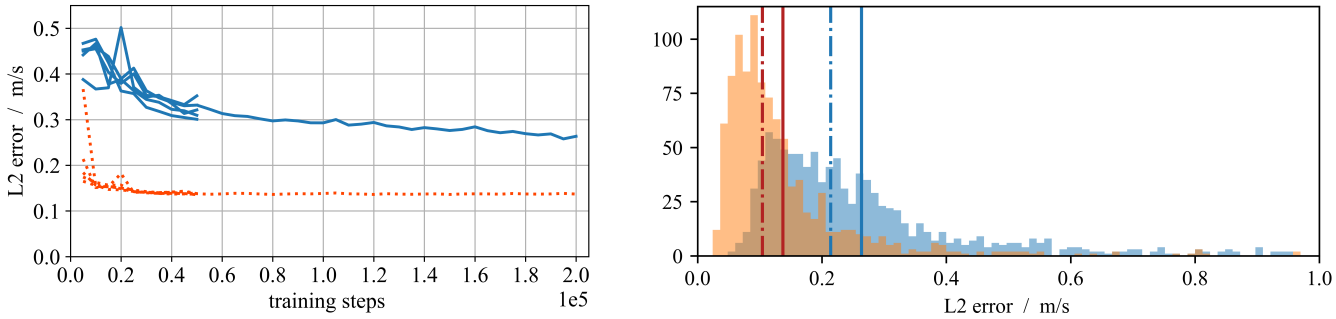

Fig. 4: Prediction accuracy on test set in terms of L2 error for the GAN (blue) and L1-trained (orange/red) models. **Left:** Model performance as a function of training progression. **Right:** Error distributions after 200 000 steps visualized in a histogram. Dashed and solid lines show median and mean respectively.

3 Results

To investigate the effect of adversarial training, we compare generators trained purely adversarially to those trained purely on ground truth. For estimating the sensitivity of the results to the random initialization, we trained five models with each approach for 50 000 steps. We then continued the training up until 200 000 steps with one model of each type. Training for 1000 steps took on average 181 s for the pure L1 model and 274 s for the GAN model on a NVIDIA GeForce GTX 1080 Ti graphics chip; hence, one iteration took roughly 51 % longer when adding the discriminator network for our model architectures. After this expensive offline phase, the evaluation of the generator network for predicting the flow field took only about 0.17 s on the CPU named above (excluding TensorFlow overhead), which is between one and two orders of magnitude faster than the simulation on the hardware.

Model accuracy Arguably the most important metric for assessing how well a trained neural network performs as a reduced order fluid model is the error to ground truth on the unseen test samples. As error measure we use the L2 metric, i.e., we compute the mean magnitude of the pointwise error between the velocity fields. We also call this measure the L2 error. Figure 3 shows the accuracy of the model predictions for the example geometry from Fig. 1 by visualizing the magnitude of the velocity error field. Also when looking at the whole test set, we generally observe a better performance for the directly trained models; see Fig. 4. After 50 000 training steps the average L2 error of the five directly trained models on the test set is 57 % smaller than that of the GAN models, and between the two models trained for 200 000 steps, the difference is still 48 %. This can partly also be attributed to the more complicated training process of GANs subject to the evolving dynamic between generator and discriminator.

For a more detailed picture, we also compare the distribution of L2 errors on the test set, as visualized in the right plot of the figure. The GAN model is outperformed in terms of the L2 error in 91.6 % of test set cases. Moreover, for testing the models' generalization to types of geometries different from the training distribution, we created two additional test sets with differing obstacles. One of them contained geometries with two star-shaped obstacles instead of one, and in the other data set, the obstacles were not polygonal but curved (isolines of the Rosenbrock function). As expected, the L2 errors increased for both models compared with the original data set; for instance, for the GAN model, the mean L2 error was $60 - 90$ % larger. Moreover, we observed that, again, the L1 model was more accurate compared with the GAN model, with a 26 % lower mean L2 error for the data set with two obstacles and a 9 % lower mean L2 error on the curved obstacles.

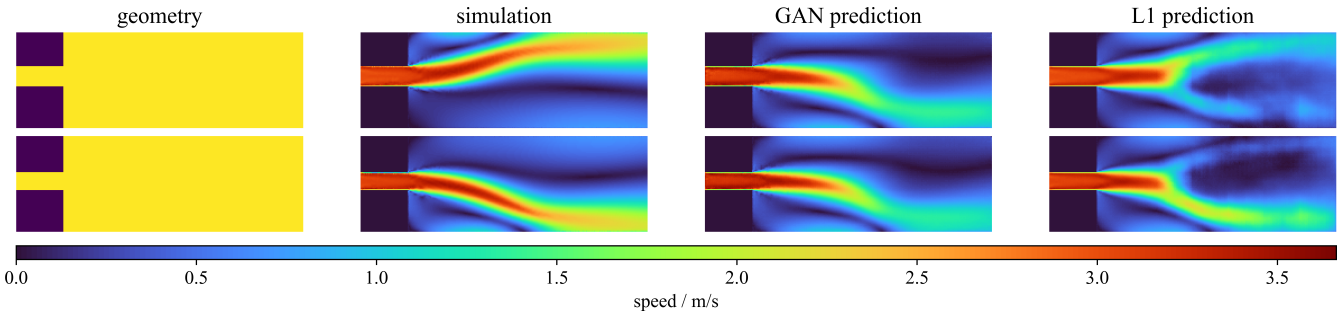

Fig. 5: Flow speed simulation and prediction for two geometries close to the bifurcation point (no obstacle, slightly different inlet positions). Models were trained for 50 000 steps.

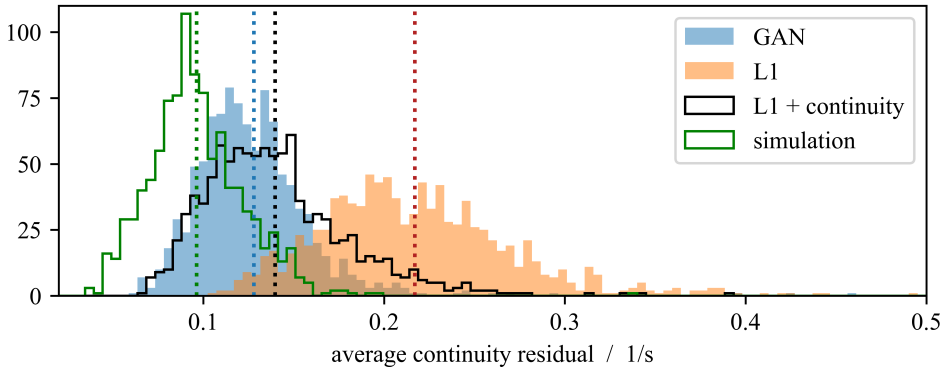

Fig. 6: Histogram visualizing the distribution of the average smoothed continuity residual on the test set predictions of both models after 50 000 steps. The dotted lines show the mean values of the corresponding distributions.

Qualitative differences In order to better understand the differences between the two model types, it is illustrative to investigate how the models handle the bifurcation phenomenon of the sudden expansion geometry. Figure 5 compares predictions for two similar geometries close to the bifurcation point. In both cases, the GAN prediction is almost exactly the same. In fact, further tests showed that whenever the outlet is approximately centered, the GAN will predict flow attachment to the same side. The L1 model on the other hand produces predictions in both cases that look akin to a superposition of the two possible bifurcation outcomes. This behavior shows the qualitative difference in how these models handle uncertainty, resulting from the different cost functions used for training: If there is ambiguity in the training data such that for a particular input there are two equally plausible outputs, any convex combination of these two will yield the same expected L1 loss. The generator may thus learn to produce such superpositions as seen in Fig. 5. In contrast, adversarial training disincentives such outcomes, and instead forces clear-cut decisions in the face of uncertainty. This is because such a superposition is nonphysical flow condition that never occurs in the training data and could therefore easily be detected as artificial by a well-trained discriminator.

Continuity Residuals Beyond looking at the error to ground truth, another relevant performance metric in the context of solving differential equations is the residual. Here, we restrict ourselves to an investigation of the residual of the continuity equation since evaluating the Reynolds-averaged momentum equation is much more involved. To compute the divergence of the velocity field, we use a standard three point central difference scheme on the image grid. In order to avoid boundary effects, we only consider grid points that do not directly neighbor a point outside of the domain. After evaluating the discretized continuity equation, we apply a smoothing operation by filtering five times with a 3×3 *box blur* kernel. While the smoothing operation does not change the results qualitatively, it reduces high frequency residual artifacts in areas of large gradient introduced by interpolating to the image grid.

The distribution resulting from applying this procedure to the test set predictions of both model types is visualized in Fig. 6. On this metric, the GAN model performs significantly better than the model trained directly on ground truth using an L1 loss, which produces a mean residual approximately 70 % larger on average. Indeed, qualitatively speaking the GAN outputs often have a smoother, more convincing appearance, which is difficult to measure but clearly correlates with a low continuity residual. Again, this result can be understood as a consequence of the properties of the different training methods. As noted in Section 2, the discriminator feedback can be understood as a kind of learned texture loss. For the discriminator, learning the exact mapping between a geometry and a flow field is hard, but learning the typical *texture* of training images (which, in particular, will show a small continuity residual) is much easier. This may be caused by the PatchGAN discriminator architecture employed in our test; however, we might observe a similar behavior for other types of discriminators.

λ_{GAN}	1	1	1	1	0
λ_{L1}	0	1	10	100	1
L2 error	0.33	0.22	0.16	0.14	**0.14**
continuity residual	0.13	**0.12**	0.12	0.13	0.22

Table 2: Performance statistics on the test set for hybrid models with different weighting in the cost functions (see Eq. (2). All models trained for 50 000 steps. Boldface values correspond to best performance in each metric (row).

A different approach to achieve a low continuity residual is to simply add the mean residual directly as a loss term to the L1 cost function; this is similar to the physics-aware approach described in [20]. A model trained this way showed almost the same residual performance as the GAN, see also Fig. 6. The effect on model accuracy was negligible.

Hybrid training We also tested the effect of training on a combination of adversarial and L1 loss. Table 2 shows the performance of three of these hybrid models alongside the previous models. The weighting was informed by observing the magnitudes of adversarial and L1 loss during training. The generator loss while training a pure GAN model was typically about one order of magnitude larger compared to the L1 loss in direct training. The data show that the combined training approach can, to some degree, combine the advantages of both model types. In particular the models trained with $\lambda_{\mathrm{L1}} = 10$ and $\lambda_{\mathrm{L1}} = 100$ produce L2 errors similar to the pure L1 model while retaining the low continuity residual of the pure GAN model.

4 Conclusions

We have investigated the impact of adversarial training on CNN-based surrogate models for CFD simulations. The numerical results show that GAN training lays the focus on producing outputs that are difficult to distinguish from training examples, rather than the best possible accuracy to ground truth. This leads to low residuals of the continuity equation and visually convincing outcomes. However, training a generator directly using the ground truth data yielded reduced order models with higher accuracy in our experiments. Moreover, the training cost for GAN models was higher due to slower convergence as well as the additional cost of training the discriminator model.

If the additional training cost is not prohibitive, a hybrid model that combines adversarial training with a direct data loss can combine the strengths of both approaches.

References

[1] P. Benner, A. Klawonn, and M. Stoll (eds.), Special Issue: Scientific Machine Learning, GAMM-Mitteilungen, Vol. 44, 2021.
[2] M. Raissi, P. Perdikaris, and G. Karniadakis, Journal of Computational Physics **378**(feb), 686–707 (2019).
[3] A. Quarteroni, A. Manzoni, and F. Negri, Reduced Basis Methods for Partial Differential Equations (Springer, 2016).
[4] X. Guo, W. Li, and F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (ACM, August 2016).
[5] M. Eichinger, A. Heinlein, and A. Klawonn, Electronic Transactions on Numerical Analysis **56**, 235–255 (2022).
[6] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, Image-to-image translation with conditional adversarial networks (July 2017), pp. 5967–5976.
[7] A. B. Farimani, J. Gomes, and V. S. Pande, Deep learning the physics of transport phenomena (September 2017).
[8] C. Jiang and A. B. Farimani, Deep learning convective flow using conditional generative adversarial networks (2020).
[9] H. Jiang, Z. Nie, R. Yeo, A. B. Farimani, and L. B. Kara, Journal of Applied Mechanics **88**(5) (2021).
[10] M. Kemna, Reduced order models for fluid flow with generative adversarial networks, Master's thesis, Technische Universiteit Delft, 2022.
[11] C. Geuzaine and J. F. Remacle, Int. Journal for Numerical Methods in Engineering **79(11)**, pp. 1309–1331 (2009).
[12] T. Mullin, J. R. T. Seddon, M. D. Mantle, and A. J. Sederman, Physics of Fluids **21**(1) (2009).
[13] S. A. Orszag, Journal of Fluid Mechanics **50**(4), 689–703 (1971).
[14] W. M. H. Versteeg, An Introduction to Computational Fluid Dynamics, 2nd edition (Pearson Education (US), 2007).
[15] H. Jasak, International Journal of Naval Architecture and Ocean Engineering **1**(2), 89–94 (2009).
[16] S. Patankar and D. Spalding, International Journal of Heat and Mass Transfer **15**(10), 1787–1806 (1972).
[17] M. Abadi, A. Agarwal, P. Barham et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.
[18] O. Ronneberger, P. Fischer, and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Lecture Notes in Computer Science, (Springer International Publishing, 2015), pp. 234–241.
[19] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (December 2014).
[20] V. Grimm, A. Heinlein, and A. Klawonn, Physics-aware convolutional neural networks for two-dimensional flow predictions, in preparation.

