CS230

Turbulence Enrichment using Generative Adversarial Networks

Akshay Subramaniam, Man-Long Wong, Raunak Borker and Sravya Nimmagadda*
Department of Aeronautics & Astronautics
Stanford University
{akshays,wongml,rborker,sravya}@stanford.edu

Abstract

Generative Adversarial Networks (GANs) have been widely used for generating photo-realistic images [2, 5, 3]. A variant of GANs called SRGAN [4] has also been used successfully for image super-resolution where low resolution images can be upsampled to a $4\times$ larger image that is perceptually more realistic. However, when such generative models are used for physical data, the governing equations may not be obeyed by the generated data. In this work, we develop a method for generative modeling of turbulence. We incorporate a physics informed learning methodology by a modification to the loss function that tries to minimize the residuals of the governing equations for the generated data. We train two models: a supervised model and a GAN, and show that they both outperform bicubic interpolation. Using the physics informed learning is also shown to significantly improve the model's ability to capture the physical governing equations.

1 Introduction

Modeling turbulence accurately is extremely challenging especially in capturing high order statistics due to its intermittent nature. GANs have been shown to perform better than other data driven approaches like PCA in capturing high order moments [1]. In addition, generating physically realistic realizations are important for physical data; a constraint not present when using generative models for images. Incorporating this constraint into the GAN framework would be crucial to its performance in this context.

Here, we propose to enrich a low resolution turbulent field with high frequency content using a generative adversarial network. The input to our model is a low resolution flow field that consists of four 3D fields each of size $16 \times 16 \times 16$. We then use a GAN to upsample each of these four fields to $64 \times 64 \times 64$. Such an enrichment algorithm may be used in many engineering settings. It can be used to upscale low resolution simulation data, generate physically realistic high frequency to compute structural loads on a wind turbine or upsample experimental data that are limited by sensor resolution.

2 Related work

To our knowledge physics informed GANs have not been studied previously. In the recent works [6], [7] physics based neural networks were developed to infer solutions to a partial differential equation (PDE). To achieve this they used a mean squared error loss function with contributions from not only

*Code for our models is available at https://github.com/akshaysubr/TEGAN

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

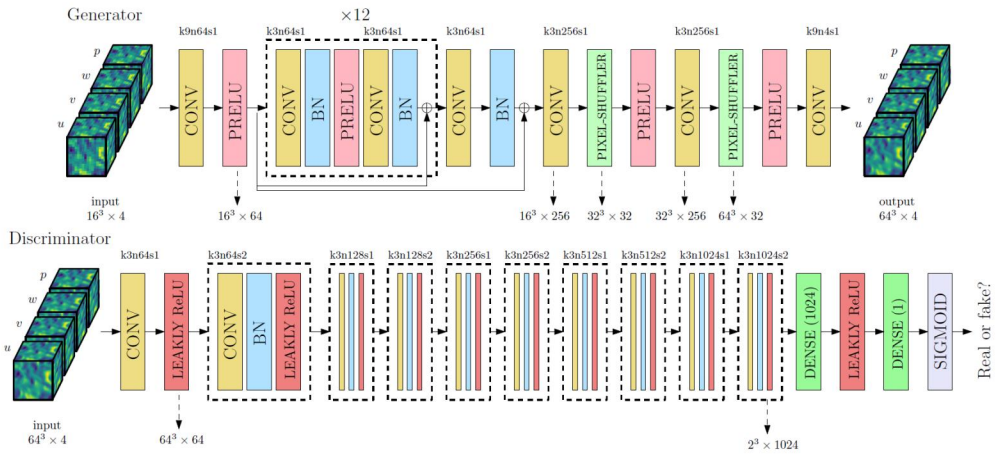


Figure 1: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of channels (n) and stride (s) indicated for each convolutional layer.

the error in the solution, but also from the residual of the governing PDE. This was implemented and tested on a model 1D problem, but would not be a feasible approach for solving large 3D chaotic systems like turbulent flows. In [4], GANs were used for photo-realistic super-resolution of images. The results shown were encouraging, but the exact same approach cannot be applied to our problem of interest as being photo-realistic does not guarantee that the generated solution (image) is physically-realistic. Finally, in [1] GANs were used an alternative to the typically used Principal Component Analysis (PCA) to generate complex geological models. GANs were shown to better capture distributions of multiple quantities of interest. All the referenced articles have been a source of inspiration for our work.

3 Dataset and Features

Data is generated using an in-house fluid flow simulation code (PadeOps) run on a compute cluster using 256 processors for multiple days. We perform a time-evolving direct numerical simulation on a $64 \times 64 \times 64$ uniform grid and collect snapshots separated in time by more than one integral time scale. This ensures that each example is statistically decorrelated. Each snapshot is comprised of four fields - three components of the velocity vector (u, v, w) and the kinematic pressure (p) each of size $64 \times 64 \times 64$. Low-resolution data is then generated by filtering the high resolution data down to $16 \times 16 \times 16$ using a compact support explicit filter that is derived as a best approximation to the sharp spectral filter. The velocity components of the high resolution data are normalized (and non-dimensionalized) by the root mean square of the velocity and the pressure by the mean square of the velocity. The dataset is divided as Train/Dev/Test split: 920 (79.3%)/120 (10.3%)/120 (10.3%). Sample images of the high and low resolution data are presented in Section 5.

4 Methods

4.1 Model

For the task of upsampling the low resolution data in a physically consistent manner, we use a GAN[2] in a fashion similar to super-resolution applications for image data [4].

The generator has a deep residual network architecture with each residual block having convolutional layers with batch normalization. The discriminator has a deep convolutional architecture with fully connected layers in the end for binary classification. The architectures of the generator and discriminator are depicted pictorially in Figure 1.

4.2 Loss Functions

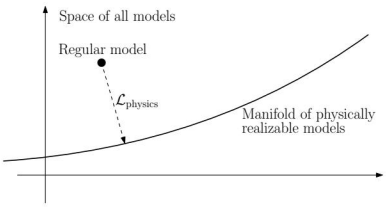
The flow field is constrained by the continuity and pressure Poisson equations:

$$\nabla \cdot \boldsymbol{u} = 0,$$
$$-\nabla^2 p = \nabla \boldsymbol{u} : \nabla \boldsymbol{u}^T$$

where u,v, w and p represent the three velocity components and pressure respectively. The above equations might not be satisfied exactly by the model's generated output. To counter this, the residual of the above equations can be used as a regularizer for the model through a physics loss.

The loss function minimized for the generator network during training is a combination of a content loss $\mathcal{L}_{\text{content}}$ and a physics loss $\mathcal{L}_{\text{physics}}$.

$$\mathcal{L}_{\text{GAN}} = (1 - \lambda_{\text{A}})\, \mathcal{L}_{\text{resnet}} + \lambda_{\text{A}} \mathcal{L}_{\text{adversarial}}$$
$$\mathcal{L}_{\text{resnet}} = (1 - \lambda_{\text{P}})\, \mathcal{L}_{\text{content}} + \lambda_{\text{P}} \mathcal{L}_{\text{physics}}$$
$$\mathcal{L}_{\text{content}} = (1 - \lambda_{\text{E}})\, \mathcal{L}_{\text{MSE}} + \lambda_{\text{E}} \mathcal{L}_{\text{enstrophy}}$$
$$\mathcal{L}_{\text{physics}} = (1 - \lambda_{\text{C}})\, \mathcal{L}_{\text{pressure}} + \lambda_{\text{C}} \mathcal{L}_{\text{continuity}}$$



- **Content loss**: $\mathcal{L}_{\text{content}}$
  $\mathcal{L}_{\text{MSE}}$: Mean squared error between the high resolution and generated fields.
  $\mathcal{L}_{\text{enstrophy}}$: Mean squared error in the derived enstrophy field Ω ($\Omega = \boldsymbol{\omega} \cdot \boldsymbol{\omega}$, where $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$) to sensitize the generator to high frequency content.
- **Physics loss**: $\mathcal{L}_{\text{physics}}$ Residuals of the continuity ($\mathcal{L}_{\text{continuity}}$) and pressure Poisson ($\mathcal{L}_{\text{pressure}}$) equations given above similar to [6]. As depicted in figure above, the inclusion of physics loss forces the our NN to generate on physically realizable solutions.
- **Adversarial loss**: a logistic loss function is used similar to that defined in [4].

To train the discriminator, we use the logistic loss based on predicted labels for real and generated data.

4.3 Training

4.3.1 TEResNet

A model with just the residual generator network without the adversarial component is termed TEResNet. We first train TEResNet to convergence and tune hyperparameters like the number of residual blocks and the physics loss parameters.

4.3.2 TEGAN

The model with both the residual network generator and the discriminator depicted above is termed TEGAN. The generator in TEGAN is first initialized using the weights from the trained TEResNet while the discriminator is initialized using the Xavier-He initialization.

For the first few iterations in the training process (~ 300), the discriminator alone is trained to negate the advantage that the generator has because of its pre-trained weights. Then, both the generator and discriminator are trained together with the active adversarial loss until the losses saturate and the discriminator's output saturates at 0.5. The Adam optimizer is used for updating the weights and training both the networks.

5 Experiments/Results/Discussion

A batch size of 5 was chosen for training of both TEResNet and TEGAN because of memory constraints of the GPU used for training. We choose $\alpha = 1.0e-4$ as the learning rate for training from the hyper-parameter search. We also examine the effect of the physics loss weight λ_{P} through experiments on TEResNet. Figure 2 shows the content and physics losses during training of TEResNet

with different values for λ_{P}.We see that adding a non-zero weight to the physics loss improves the physics residual by almost an order of magnitude. We choose $\lambda_{\mathrm{P}} = 0.125$ as this gives good compromise between the two losses. Another interesting observation is that for higher weightage to the physics loss, the trivial solution of zero fields becomes a local minimum.

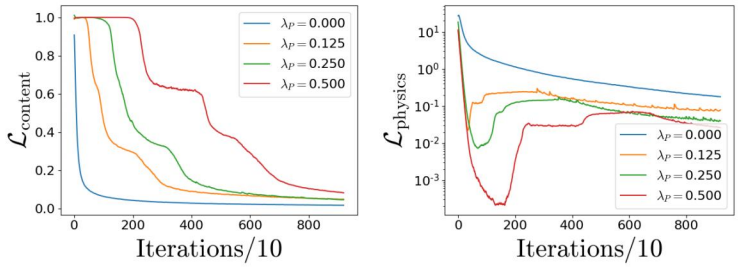


Figure 2: Losses against iterations. Left: content loss; right: physics loss.

To train the TEGAN model, we initialize its weights using the trained TEResNet for the generator. We then train only the discriminator until the discriminator is able to distinguish between real and generated data. Then we train the discriminator and generator together training the discriminator twice as often as the generator. To improve the stability for training TEGAN, we add a staircase decay for the learning rate. We set the decay rate to 0.5 and chose a decay step of 400 by running a case without learning rate decay and estimating the number of steps required to go close to the minimum. Figure 3 shows the convergence of TEGAN during training. It can be seen that the discriminator output for generated data saturates at 0.5 and the physics loss converges to a smaller value compared to the initial value from TEResNet.

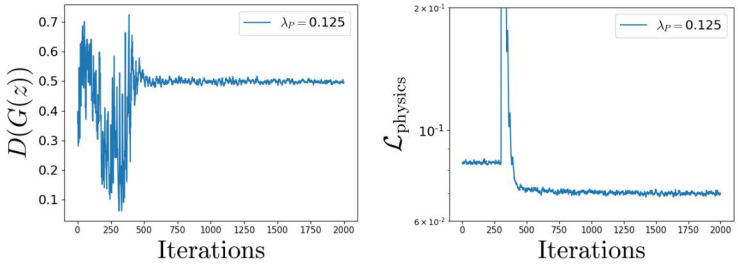


Figure 3: Left: discriminator output for generated data; right: physics loss of TEGAN.

Figure 5 compares the qualities of upsampled data from bicubic interpolation, TEResNet and TEGAN. Both TEResNet and TEGAN outperform the bicubic interpolation in reconstructing small-scale features. This is also evident from the plots of the velocity energy spectra in Figure 4. The output from TEResNet and TEGAN are indistinguishable visually but table 1 shows that there is more than 10% improvement of TEGAN over TEResNet in minimizing the physics loss while the content losses of both models are similar.

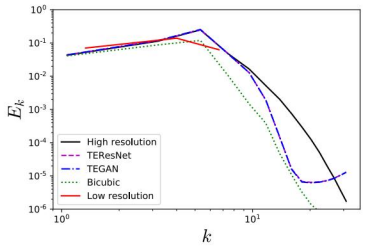


Figure 4: Comparison of the velocity energy spectra for the different upsampling methods.

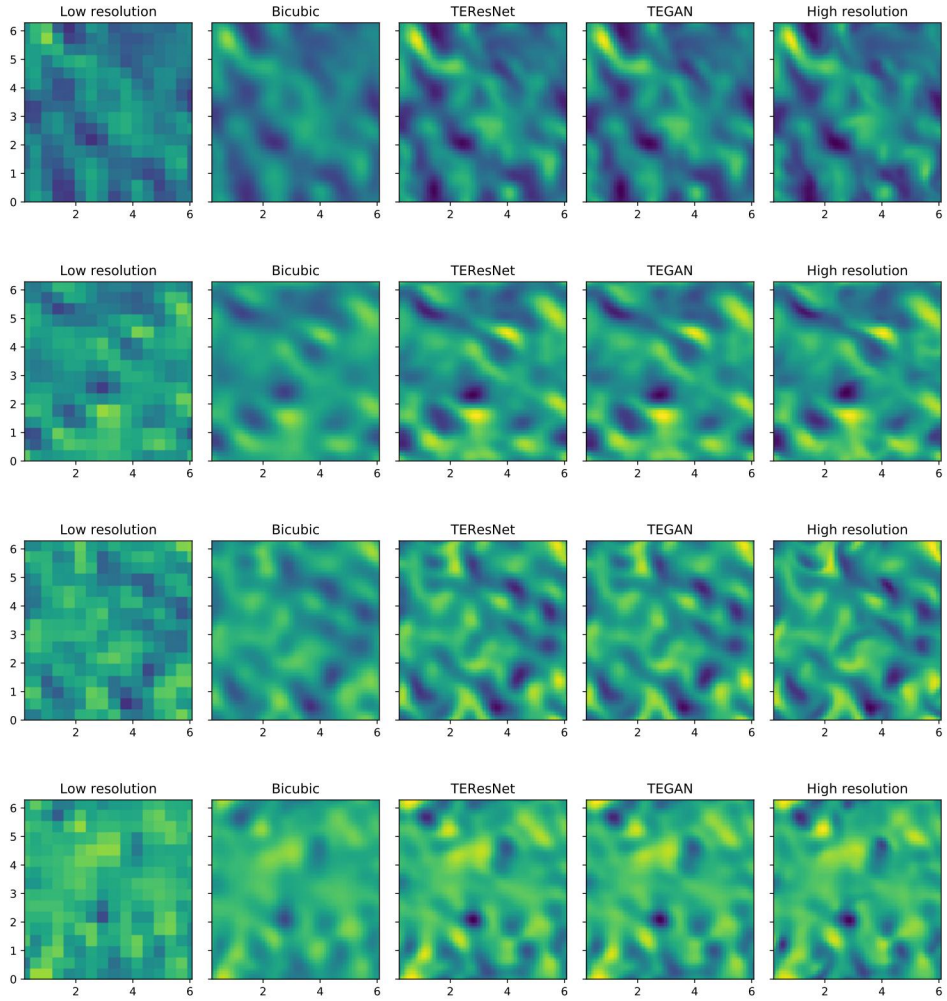


Figure 5: Comparisons of the (from left to right) low resolution, bicubic interpolation, TEResNet TEGAN and high resolution fields. Plots are of the u, v, w and p (from top to bottom) on a slice of the 3D field.

	$\mathcal{L}_{\text{content}}$		$\mathcal{L}_{\text{physics}}$	
	Dev	Test	Dev	Test
TEResNet	0.049	0.050	0.078	0.085
TEGAN	0.047	0.047	0.070	0.072
% Difference	4.1	6.0	10.3	15.2

Table 1: Comparison of losses between TEResNet and TEGAN at the end of training.

6 Conclusion/Future Work

In this work, we present two models, TEGAN and TEResNet, for turbulence enrichment. We show the effect of different losses and the impact of physics based losses for improved performance of the networks. While both TEResNet and TEGAN outperform traditional bicubic interpolation, TEGAN captures the physics better than TEResNet.

The current TEGAN architecture and training can be further improved in the future by the implementation of gradient penalty based Wasserstein GAN (WGAN-GP) [3] for overcoming few of the shortcomings of the traditional GANs mainly related to stabilized learning. Also, using wider distributions of data and tasking the discriminator with physics based classification along with discrimination can be explored for better performance of the TEGAN in the future.

References

[1] S. Chan and A. H. Elsheikh. Parametrization and generation of geological models with generative adversarial networks. *arXiv preprint arXiv:1708.01810*, 2017.

[2] 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*, pages 2672–2680, 2014.

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[4] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint*, 2016.

[5] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.