INTERNATIONAL JOURNAL of RENEWABLE ENERGY RESEARCH
N. J. Ward, Vol.16, No.1, March, 2026

Unsupervised Deep Learning of Turbulent Heat Transfer via Generative Adversarial Networks

Nicholas J. Ward*‡

*Department of Mechanical Engineering, Texas Tech University, Lubbock, TX, USA 79409

‡ Corresponding Author; Department of Mechanical Engineering, Texas Tech University, Lubbock, TX, USA 79409. Email: nicholas.ward@ttu.edu

Received: 12.02.2024 Accepted: 22.03.2024

Abstract- Machine learning and deep learning can be useful in providing insight to near-wall turbulence. The research question that this study seeks to answer is how to effectively predict the turbulent heat transfer of a 2D channel flow using a generative adversarial network (GAN). The first objective of this study is to predict the heat transfer using the wall variables p, $\partial u/\partial y$ and $\partial w/\partial y$ only. The second objective is to extrapolate the trained GAN model to temperature gradients at higher Reynolds numbers. Direct numerical simulations (DNS) were performed to determine the training data, and machine learning algorithm was developed to effectively capture the flow physics in a turbulent channel flow. The predictions from the proposed GAN correlated well with the DNS data at Re_τ = 180. The predictions were effectively captured within 2% of the DNS training data. The temperature was not required to measure the temperature gradients using the GAN model proposed in this study. Additionally, the GAN was also able to capture the flow physics for the test data at higher Reynolds numbers. To accomplish this, an appropriate normalization technique was required for the predictions. This work will be useful in future work to be applied to other problems in fluid mechanics, such as drag reduction, capturing near-wall flow physics, and super-resolution.

Keywords Turbulent channel flow, turbulent heat transfer, artificial intelligence, deep learning, generative adversarial network (GAN).

1. Introduction

Machine learning (ML) and deep learning (DL) have had extensive and increased applications to multiple problems. Some of these applications include speech recognition [1, 2], pattern recognition [3, 4], data augmentation [5, 6], and image processing [7, 8]. However, over the last couple decades, there has been significant interest in using ML and DL for fluid mechanics [9]. By doing so, ML and DL can provide additional insight into the physics of turbulence, as well as reduce the computational cost associated with conventional methods to capture the physics of canonical fluid flows.

Kim et al. [10] provided the foundation for the direct numerical simulation (DNS) of turbulent channel flows for the incompressible Navier-Stokes equations. Jimenez and Moin [11] defined the minimum flow unit that captured the physics in the DNS of low to moderate viscous Reynolds numbers (Re_τ). To date, these DNS results have been applied up to Re_τ = 5200, as noted in Lee and Moser [12].

Antonia et al. [13] determined that there was a high correlation between the streamwise velocity fluctuations and the temperature fluctuations near the channel wall. Studies by Jeong and Hussain [14] and Jeong et al. [15] determined how to identify and educe the most impactful coherent structures present in near-wall turbulence. Agostini and Leschziner [16] applied a spectral analysis to determine the impact of turbulent coherent structures of different scales on near-wall turbulence. Additionally, studies by Schoppa and Hussain [17], Hutchins and Marusic [18], and Lozano-Duran et al. [19] provided the´ foundational basis for determining how near-wall turbulence is sustained in turbulent channel flows and boundary layers. It is also worth noting some limitations for using DNS. Because of its high computational cost, DNS is not always practical for large-scale systems. This high computational cost is associated with resolving all physical process at all scales. These limitations demonstrate a need for complementary approaches, such as large eddy simulations, Reynolds-averaged Navier-Stokes, or ML/DL based techniques.

There have also been significant advancements in ML and DL for near-wall physics in turbulent channel flows and heat transfer. A few examples of these problems have included drag reduction, turbulent inflow generation, near-wall turbulence, and turbulent heat transfer. Choi et al. [20] established the foundation for active control in drag reduction

techniques in DNS, and Babcock et al. [21] extended this study using neural networks to learn and analytically derive the relations for active drag control. Lee et al. [22] applied a neural-network-based controller that was able to reduce skin friction by as much as 20% in a turbulent channel flow. Bae and Koumoutsakos [23] applied a reinforcement-based learning method to optimize the drag reduction in a channel. For synthetic inlet generation, Fukami et al. [24] developed a convolutional-neural-network-based encoder-decoder algorithm to reconstruct the turbulent inlet conditions for a low Reynolds number channel flow. Yousif et al. [25] then applied a generative adversarial network to reconstruct turbulent inlet flows using lower resolution input data. ML and DL has then been extended to studies in near-wall turbulence modeling. Wang et al. [26] applied a neural network architecture to predict the near-wall velocity from PIV data to reconstruct wall-bounded turbulence. Ezhilsabareesh et al. [27] analyzed the effects of limited near-wall data on reconstructing the turbulence statistics and energy spectra in channel flows. Additionally, there have been extensive applications for ML and DL for turbulent heat transfer. Following the work by Antonia et al. [13], Kim and Lee [28] was able to predict the temperature gradients from wall variables using a convolutional-neural-network-based ML algorithm. Kim et al. [29] was then able to apply a DL-based algorithm for Prandtl number effects in turbulent heat transfer. Cai et al. [30] was able to apply physics-informed neural networks to different heat transfer problems, including forced and mixed convection, two-phase flows, and power electronics. Unlike DNS and other computational fluid dynamics approaches, DL models are able to learn complex patterns and relationships between input data, and be trained on large datasets to learn the underlying physics, make predictions based on those patterns, and generalize to unseen data if properly trained. Therefore, by choosing the right inputs, the DL is able to determine accurate solutions without having to solve for the governing equations directly, which can be computationally expensive.

The goal of this work is to investigate the underlying physics of near-wall turbulence via deep learning. This study seeks to address the following question: can the turbulent heat transfer of a 2D channel flow be effectively predicted via generative adversarial networks (GANs)? The first objective of this study is to predict the heat transfer using wall variables only. The second objective is to extrapolate the trained GAN model to temperature gradients at higher Reynolds numbers.

This paper is organized in the following structure. Section 2 describes the numerical set-up of the DNS data utilized for training the GAN. Section 3 explains the methodology of the proposed GAN. This section describes in detail the deep learning algorithm implemented, such as the generator and discriminator architecture, as well as the chosen values of the hyperparameters. Additionally, it defines the objective functions that the proposed deep learning algorithm optimizes with respect to the architectures and hyperparameters implemented. Section 4.1 demonstrates the turbulent heat transfer and flow statistics generated from the GAN with respect to the DNS, and validates the performance of the GAN compared to other models. Section 4.2 reconstructs untrained 2D turbulent heat transfer data at higher Reynolds numbers using the trained model at lower Reynolds numbers. The conclusions and future works from this study are stated in section 5.

2. Computational Setup

Direct numerical simulations (DNS) were performed to determine the training data utilized in this study, which is explained in the next section. DNS were conducted for turbulent channel flows using the POONPACK code, which was developed by Lee and Moser [12]. For this study, the streamwise and spanwise coordinates are denoted by x, and z, respectively. Additionally, the streamwise, wall-normal and spanwise velocities, as well as the temperature are denoted by u, v, w, and T, respectively. The incompressible continuity, momentum, and energy equations, which are determined as

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

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{Re_\tau} \nabla^2 \boldsymbol{u}, \tag{2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \frac{1}{Re_\tau Pr} \nabla^2 T, \tag{3}$$

are solved in Kim and Moin [31] by time advancement of the wall-normal vorticity and the Laplacian of the wall-normal velocity equations. Equations 1-3 are non-dimensionalized using the friction velocity u_τ, channel half-height δ, kinematic viscosity ν, and the temperature difference ΔT between the wall and the channel center, respectively. The Fourier-Galerkin method was applied in the streamwise and spanwise directions, and a seventh-order B-spline collocation method was applied in the wall-normal direction. The time advancement method is accomplished using an implicit-explicit scheme that applied a third-order Runge-Kutta for non-linear terms, and the Crank-Nicolson for the viscous terms. The turbulent channel flow is driven via pressure gradient that varies in time. For more details about the code, refer to Kim and Moin [31].

For the turbulent channel flows used in this study, the numerical discretization details are shown in Table 1. In this study, three DNS simulations were conducted at three different bulk Reynolds numbers (Re_b). The three Re_b values were used to determine two different viscous Reynolds numbers ($Re_\tau = u_\tau \delta / \nu$). Additionally, the Prandtl number Pr was set to be 0.71 for each DNS considered in this study.

Table 1. Details of the numerical discretization employed for the simulations of the turbulent heat transfer. The computational grid spacing is L_x and L_z, and N_x and N_z is the grid spacing in each direction

Re_τ	Re_b	Pr	$L_x \times L_z$	$N_x \times N_z$	Δx^+	Δz^+	Nu
180	2857	0.71	$4\pi\delta \times 2\pi\delta$	192×192	11.78	5.89	6.112
360	6250	0.71	$2\pi\delta \times \pi\delta$	384×384	5.89	2.95	11.22
540	9804	0.71	$2\pi\delta \times \pi\delta$	576×576	5.89	2.95	16.34

3. Deep Learning Algorithm

For the model used in this study, a DL algorithm was developed to effectively capture the flow physics in a turbulent channel flow without directly solving the Navier-Stokes equations.

To effectively capture low-Reynolds-number turbulent channel flows, a GAN [32] was constructed. The GAN is an unsupervised deep learning model [9, 28] comprised of two architectures: a generator, and a discriminator. The generator is utilized to construct flow fields at each time step using either a random noise vector or other images as an input during training. For this study, the inputs are the following wall variables: wall pressure p, as well as the wall-normal velocity gradients $\partial u/\partial y$ and $\partial w/\partial y$. After training is completed, the generator then predicted the new dataset, which is the wall-normal temperature gradient $\partial \tilde{T}/\partial y$. The discriminator, which was utilized during the training process only, provided feedback to the generator to either accept or reject the outputted flow fields with respect to the real (DNS) data.

To ensure that it worked effectively, the GAN model required a significant number of samples because increasing the number of samples enabled reduced errors between the generated samples and the DNS data. For the simulations at Re_τ = 180, 5,000 samples of data were respectively used to reconstruct the turbulent channel flow with heat transfer. By utilizing these DNS samples as inputs into the discriminator, the GAN model was able to generate the new data for each respective time step. The GAN model accomplished this by training these samples for 100 epochs.

In addition to the number of samples and training iterations, the GAN model required loss functions for both the generator and the discriminator. These loss functions provided the GAN model with two functions to optimize during each training iteration. The loss functions were determined from the pixel-to-pixel GAN (Pix2Pix GAN). This GAN model was chosen because it has proven to prevent mode collapse during the training, as well as to enable image-to-image translation between the real and predicted samples [33, 34]. For the model to be trained properly, the loss functions needed to be defined. The combined loss function $\mathcal{L}_{tot}$ for the GAN training as a function of the generator and discriminator outputs (G and D, respectively) is defined as

$$\mathcal{L}_{tot} = \min_G \max_D [\mathcal{L}_{CGAN}(G, D) + \lambda \mathcal{L}_{L2}(G)], \quad (4)$$

where $\mathcal{L}_{CGAN}$ is the loss function of the conditional GAN model, λ is the L2 loss function weight, and $\mathcal{L}_{L2}$ is the L2 loss function. The L2 loss function weight λ is set to 100 for the purposes of this study. The loss functions for $\mathcal{L}_{CGAN}$ and $\mathcal{L}_{L2}$ are respectively defined as

$$\mathcal{L}_{CGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x)))] \quad (5)$$

$$\mathcal{L}_{L2}(G) = \mathbb{E}_{x,y}[\|y - G(x)\|_2] \quad (6)$$

where $\mathbb{E}[\cdot]$ is the expected value, x is the input data from the wall variables, y is the real temperature gradient data, $G(x)$ is the data outputted from the generator, and $\|\cdot\|_2 = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|\cdot|^2}$ is the L2 matrix norm. The L2 matrix norm is related to the mean squared error (MSE) loss as $MSE = L_2^2/N = (1/N)\|\cdot\|_2^2$, which is useful in minimizing the distance between datasets with N total samples in regression. Using the L2 or MSE as an additional loss term to the generator has proven to be an effective constraint in previous studies [28, 35].

After determining the appropriate loss functions, the GAN model then required objective functions utilized to provide optimized parameters to the loss functions during each training iteration. This indicated that an optimizer needed to be defined for the GAN model. Numerous optimizers have been proposed in previous works, such as stochastic gradient descent (SGD) [36], RMSProp [37], etc. However, for this study, the adaptive moment estimator (Adam) optimizer was utilized for adaptive learning rate optimization [38, 39]. This optimizer has been proven to enable faster convergence in less training iterations for multiple GAN architectures, particularly the Pix2Pix GAN utilized in this study. By inputting the initial learning rate l, as well as the decay rates in calculating the first and second moments (i.e. the mean and variance) β_1 and β_2, the adaptive learning rate functions were determined at each timestep. For the purposes of this study, the input learning rates for the generator and discriminator were 0.0002, and the decay rate β_1 was 0.5. Additionally, a discriminator weight of 0.5 was applied to the loss functions to prevent the GAN from overfitting.

The architecture that was utilized for the generator in this study is shown in c. Additionally, the description of each layer presented in Figure 1 is indicated in Table 2. The random noise vector (RN) that was used as an input in the generator

represented a Normal distribution. This RN layer was used as an input layer in each of the encoder and decoder blocks in the generator. The encoder block stepped down the size of the input images by a factor of two in each instance. Each encoder block that was defined in Figure 1a included the RN layer, as well as a 2D convolutional (Conv2D) layer, a batch normalization (BN) layer, and a leaky rectified linear unit (LR) layer. The BN layer [40] is defined as

$$\mu = \frac{1}{n}\sum_{i} Z^{(i)} \tag{7}$$

$$\sigma = \frac{1}{n}\sum_{i}\left(Z^{(i)} - \mu\right) \tag{8}$$

$$Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}} \tag{9}$$

$$\bar{Z} = \gamma Z_{norm}^{(i)} + \beta \tag{10}$$

where μ is the mean, $Z^{(i)}$ is the input sample at each instance i, σ is the standard deviation, $Z_{norm}^{(i)}$ is the normalized input, ϵ is a constant (10^{-12}) to ensure the denominator does not go to 0, $\bar{Z}$ is the modified input based on a constant multiplier γ and a bias term β, and averaged in each training batch. After each Conv2D layer, the LR layer is defined as

$$f(x) = \begin{cases} \alpha x, & if\ x \le 0 \\ x, & otherwise \end{cases} \tag{11}$$

with a slope coefficient α, which was set to 0.2 for this study. The LR layer has been proven to perform well in both generators and discriminators for GAN models. A Conv2D layer and a rectified linear unit (Relu) activation layer, which is defined in Equation 11 where α = 0, was then utilized a bottleneck between the final encoder block and the first decoder block.

The decoder block was utilized to step up the size of the input images by a factor of two, and was defined using the layers in Figure 1. Starting with a RN layer, the output was sent to a 2D transpose convolutional (Conv2DT) layer [41], which served as the inverse operation of the Conv2D layer. This Conv2DT layer is used to convert the input into higher-dimensional output images in the decoder block. Then, the next two layers were used for normalization purposes: BN, and the dropout layer [42], which is a normalization technique used to prevent overfitting in the GAN. The output of the dropout layer was then sent to a concatenation (Concat) layer, which added that output with the encoder block output of the same image size. After the Concat layer was utilized, a rectified linear unit (Relu) layer was used as the final normalization in each of the decoder blocks. A Conv2DT and a hyperbolic tangent (tanh) activation function, defined as

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{12}$$

was then utilized to output the temperature gradient. By combining each of these layers as shown in Figure 1, this constructed the generator as a U-net architecture [43, 44].

The architecture that was utilized for the discriminator is then displayed in Figure 1b. The description of each layer in the discriminator is indicated in Table 2. By taking the generated data for the temperature gradient and concatenating with the pressure and the wall-normal shear stresses, the discriminator was able to effectively predict the temperature gradient without measuring the temperature directly. After the Concat layer, the first convolutional block contained a Conv2D layer with a LR activation layer. The next five convolutional blocks also included a BN layer after the LR layer was implemented to ensure that the outputs reflected a Normal distribution. Then, the next two layers were a Conv2D and a sigmoid activation layer, which is defined as

$$sig(x) = \frac{1}{1 + e^{-x}}. \tag{13}$$

The sigmoid activation layer was implemented to ensure the classification-based losses in Equation 5 were implemented appropriately. The final output layer shown in Figure 1b reflects a partial section of the generated temperature gradient that is conditionally trained to correctly output the final result one 12×12 patch at a time. Implementing a patch-based discriminator in the GAN has been proven to be effective during both classification- and regression-based losses during training (i.e. PatchGAN [33]).

The instantaneous losses for the generator and discriminator at each iteration are displayed in Figure 2. For the purposes of this study, the number of training epochs was set to 100, and the number of samples used for training was 5,000. This indicated that the number of training iterations for this study was 5×10^5. The losses for both the generator and the discriminator are initialized to zero, but are then at their respective maxima within the first 10,000 iterations. This indicated that the minimum number of iterations required for the GAN model to converge well-enough with the DNS results is 40,000.

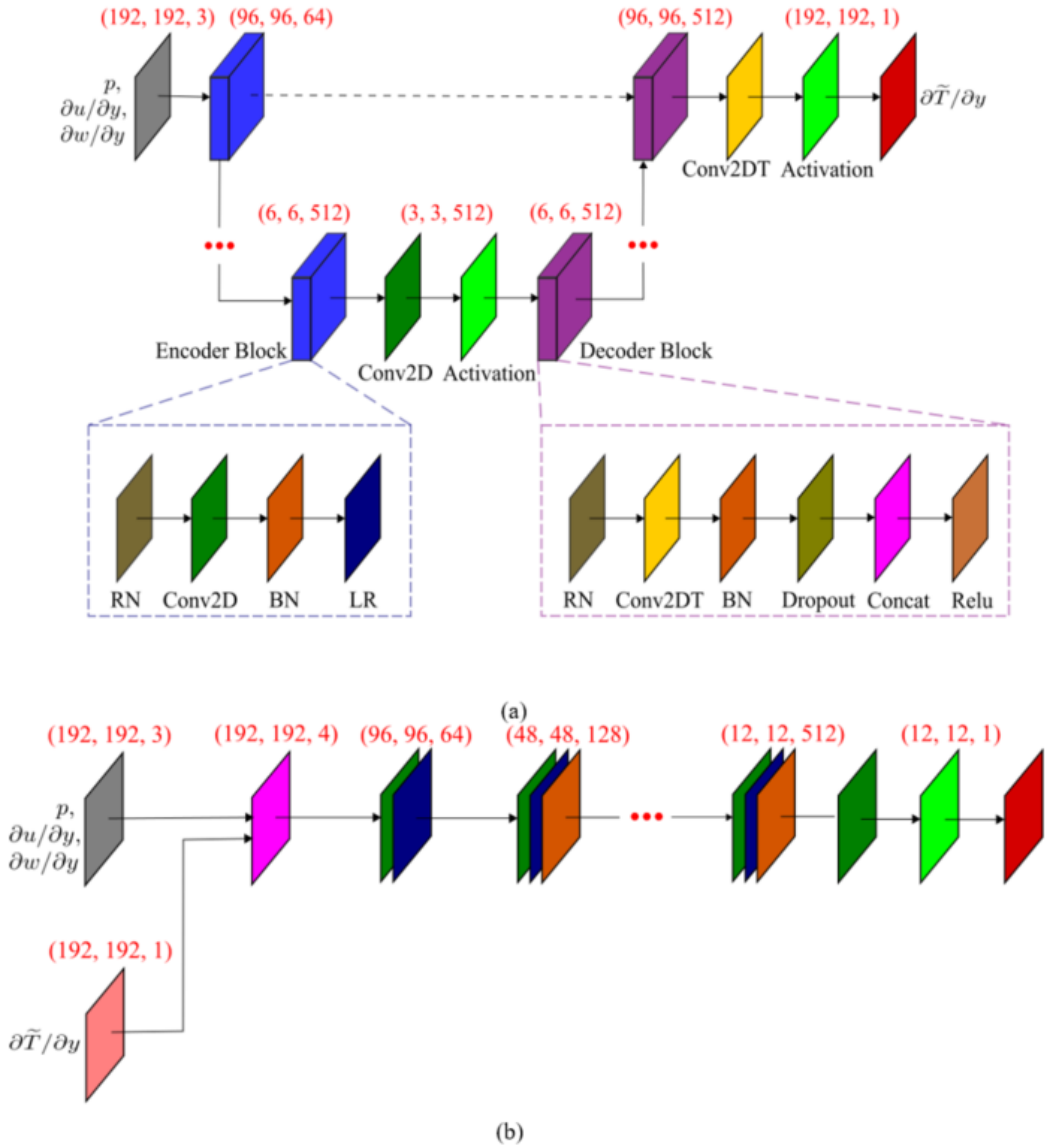

Fig. 1. Generator and discriminator architecture utilized in the GAN for this study. Each red circle corresponds to a repeat of the convolutional block denoted in red.

Table 2. Description of each layer used in the generator of the proposed GAN model

Generator	*Description*
RN	Random normal initialization layer (μ =0, σ =0.02)
Conv2D	2D convolutional layer
BN	Batch normalization
LR	Leaky rectified linear unit (α =0.2)
Activation	Activation layer (*tanh*)
Conv2DT	2D transpose convolutional layer
Dropout	Dropout (r_{drop} =0.5)
Concat	Concatenate layer
Relu	Rectified linear unit
Discriminator	*Description*
Concat	Concatenate layer
Conv2D	2D convolutional layer
LR	Leaky rectified linear unit (α =0.2)
BN	Batch normalization
Activation	Activation function (sigmoid)

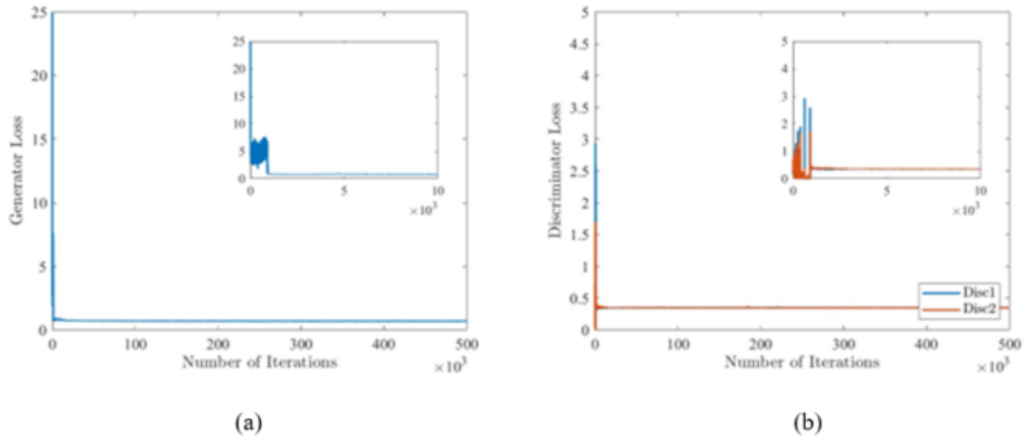

Fig. 2. Loss functions as a function of the number of training iterations for the GAN proposed in this study.

4. Results and Discussions

In order to establish the proposed GAN as a feasible alternative to other methods, the correlation coefficients were compared for each of the models. The models that were compared to the GAN were supervised DL methods using a multilayered CNN-based architecture, including the one proposed by Kim and Lee [28]. Table 3 describes the proposed GAN performance with each of the different inputs utilized in this study to determine the temperature gradient. Based on the number of samples, it is clear that the GAN performance surpasses other supervised learning techniques. In addition, because it utilizes a combination of classification and regression loss terms, as well as trains and optimizes itself using the proposed architectures and optimizer, the proposed GAN is able to improve the reconstruction results and accomplish this more efficiently. Although the CNN completed the training faster than the proposed GAN, the overall inaccuracies do not justify the usage of these methods, especially with less training data. It is also important to note that the DL algorithm based on Kim and Lee [28] does obtain good results, but trains very slowly. For the results shown in Table 3, the computational time was over 100 hours to train 50 samples. For reference, the computational time was 14 hours to train 5,000 samples using the proposed GAN. This indicates that the GAN used in this study is more computationally efficient and provides more accurate results during training.

4.1. Reconstruction Statistics

This section discusses the reconstruction statistics of the proposed GAN versus the DNS data. Figure 3 shows instantaneous snapshots of the output temperature gradient and the input wall variables. An important observation shown here is that there is a high correlation between the temperature gradient in Figure 3a and the streamwise velocity gradient in Figure 3b. This observation was also confirmed in Abe et al. [45]. Additionally, by setting the two other wall variables to be the spanwise velocity gradient and the pressure, it can be shown that the correlation of the temperature gradient with respect to all three inputs will increase. This has also been confirmed in Kim and Lee [28] and Kawamura et al. [46, 47].

Table 3. Description of the correlations for the real versus generated data used in the proposed GAN model as a function of the inputs at Re_τ=180

Inputs	*GAN*	*CNN*	*DL [28]*
$p, \partial u/\partial y, \partial w/\partial y$	0.999	0.983	0.980
$\partial u/\partial y, \partial w/\partial y$	0.998	0.974	0.973
$p, \partial u/\partial y$	0.998	0.980	0.972
$p, \partial w/\partial y$	0.995	0.848	0.950
$\partial u/\partial y$	0.997	0.960	0.968
$\partial w/\partial y$	0.990	0.772	0.923
p	0.984	0.589	0.902

In order to reconstruct the turbulent heat transfer appropriately, the data was first standardized to have a mean of 0 and a standard deviation of 1. This was done by calculating the new normalized dataset $\bar{X}_{train}$ via

$$\bar{X}_{train} = \frac{X_{train} - \mu_{train}}{\sigma_{train}} \tag{14}$$

where X_{train} is the training data, μ_{train} is the mean of the training data, and σ_{train} is the standard deviation of the training data. Additionally, the GAN inputs needed to be within −1 and 1 because of the hyperbolic tangent activation function. Therefore, the data needed to be normalized a second time to fit within the bounds of the GAN. This was then accomplished via

$$\bar{X}'_{train} = 2 * \frac{\bar{X}_{train} - \bar{X}_{train,min}}{\bar{X}_{train,max} - \bar{X}_{train,min}} - 1, \tag{15}$$

where $\bar{X}'_{train}$ is the normalized input suitable for the GAN training.

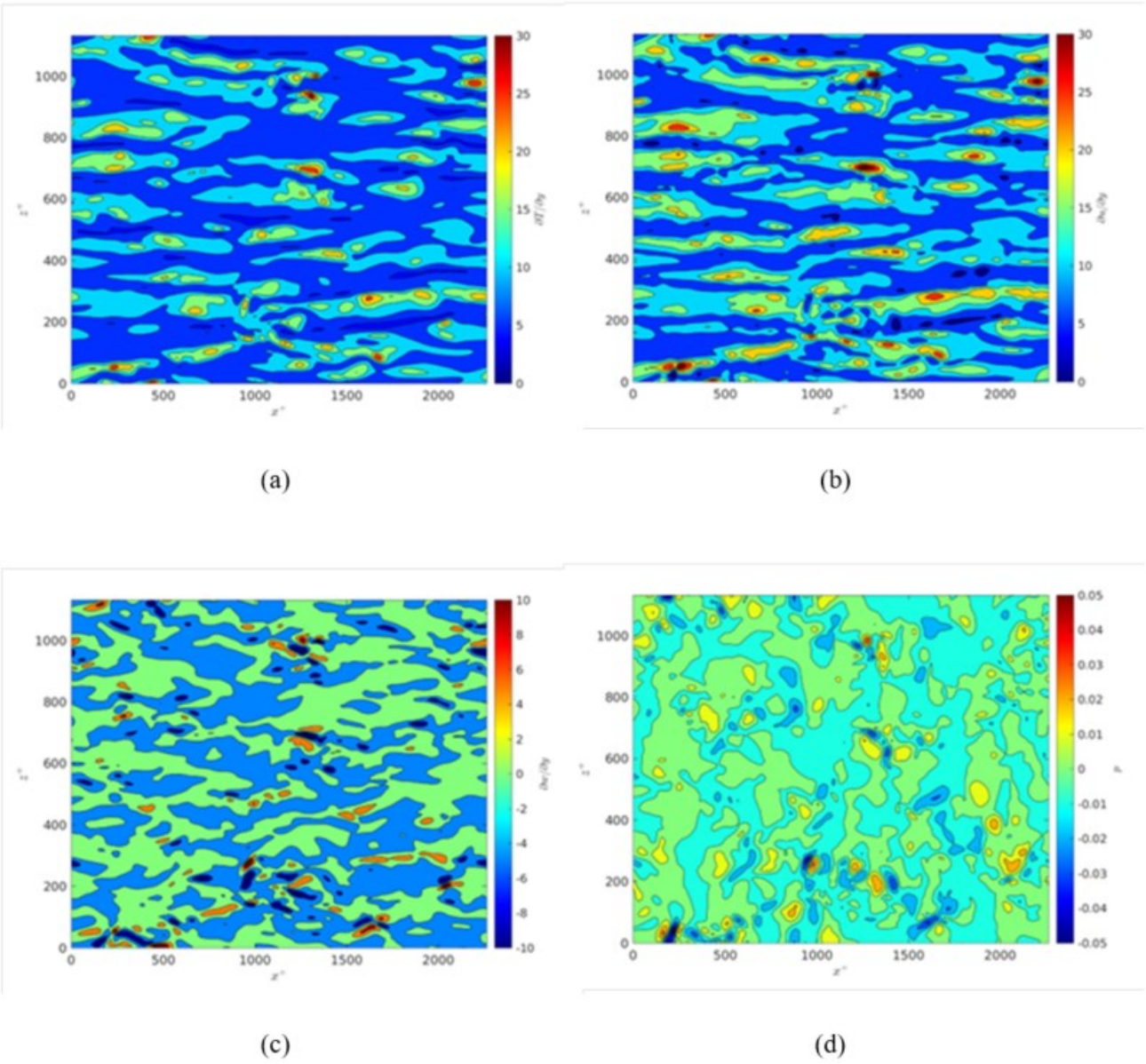

Fig. 3. Instantaneous snapshots of the DNS data at the same time for (a) $\partial T/\partial y$, (b) $\partial u/\partial y$, (c) $\partial w/\partial y$ and (d) p.

The instantaneous snapshots for the predicted, actual, and error are displayed in Figure 4 for Re_τ = 180. Based on the error plot in Figure 4c, it can be demonstrated that the majority of the predicted figure has negligible error from the DNS data. Although there are regions where the error is higher, this is because of the errors associated with the GAN being able predict temperature gradient with a high rate of change. However, the GAN was still able to perform well in these regions by limiting the error in those regions to 2%.

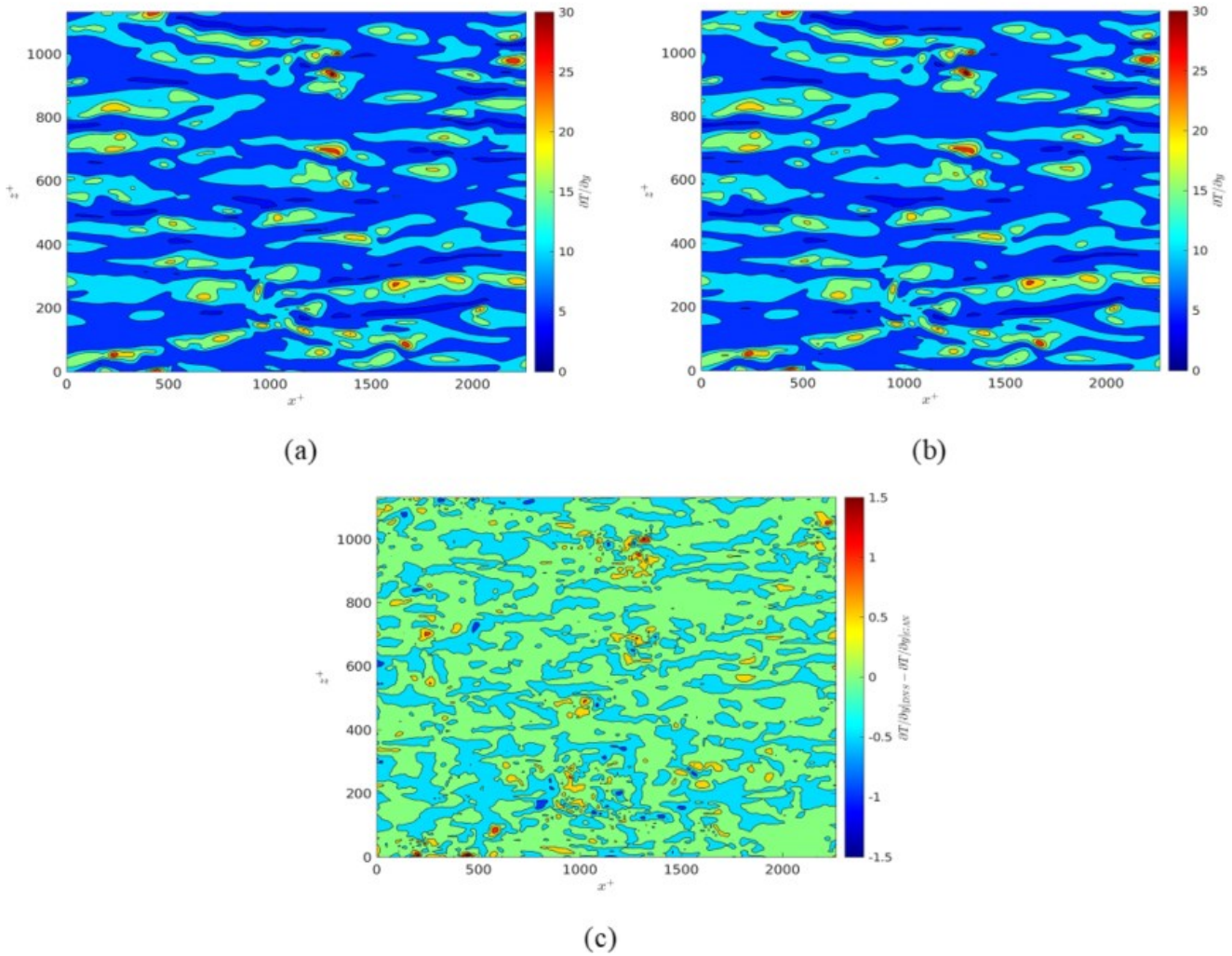

Fig. 4. Instantaneous snapshots of the (a) predicted temperature gradient $\partial\tilde{T}/\partial y$ compared to the (b) actual temperature gradient $\partial T/\partial y$, as well as the (c) error plot between the predicted and actual values.

4.1.1. GAN performance with two input variables

For the supervised learning techniques for multiple inputs, the model was compared using a combination of two of the three proposed inputs. Figure 5 demonstrates the instantaneous snapshots of the trained and tested temperature gradients given two inputs. Given these inputs, it can be determined that there is noticeable improvement between the single and multiple inputs. This verifies that incorporating more pressure and velocity gradient information improved the predictions for the temperature gradients at the trained Re_τ.

Likewise for the supervised learning techniques, the GAN demonstrated significant improvement in performance using multiple inputs. Additionally, the GAN was able to generate accurate predictions at every time step, which the supervised learning technique was also not able to accomplish with increased number of inputs. The predictions with multiple inputs for the GAN in comparison to the DNS data are shown in Figure 6. Likewise for the predictions shown in Figure 5, the predictions shown in Figure 6 are shown every 50 time steps. Based on the shown predictions, it is demonstrated that the predictions are the most accurate using $\partial u/\partial y$ and $\partial w/\partial y$ as the input to the GAN model. Like the model in Kim and Lee [28], the GAN was able to determine that the two wall-normal velocity gradients as inputs demonstrated the most accurate predictions.

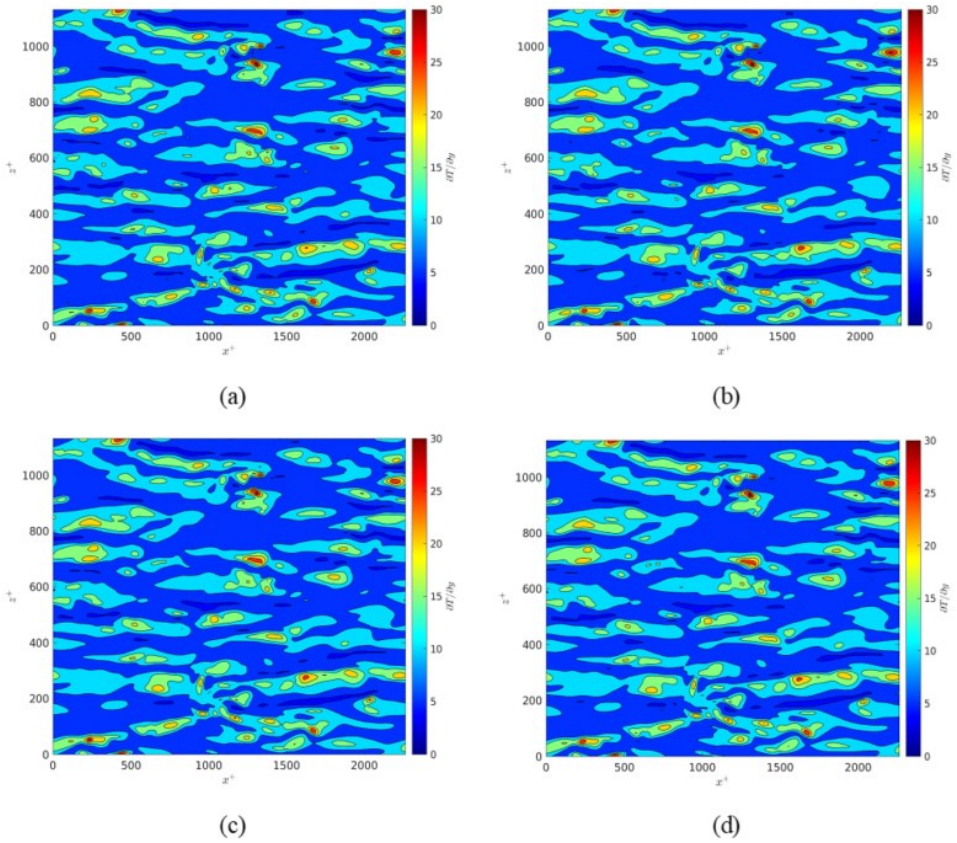

Fig. 5. Instantaneous snapshots of the predicted temperature gradients at the trained Re_τ with inputs being (a) $\partial u/\partial y, \partial w/\partial y$, (b) $p, \partial w/\partial y$ and (c) $p, \partial u/\partial y$, as well as (d) the actual temperature gradients.

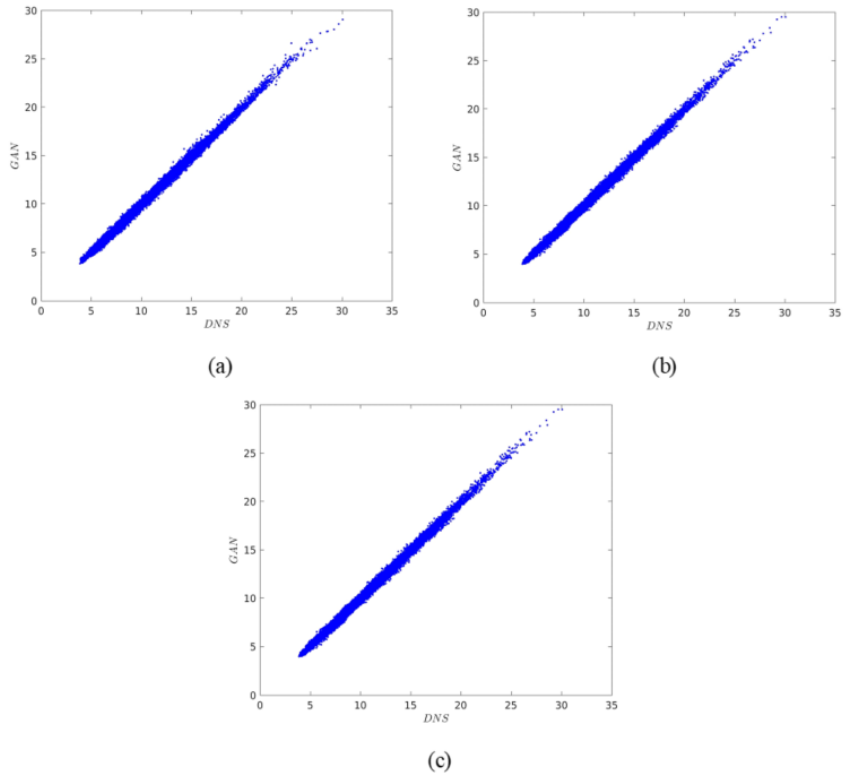

Fig. 6. Scatterplots comparing the temperature gradients generated from the GAN trained with (a) $\partial u/\partial y, \partial w/\partial y$, (b) $p, \partial w/\partial y$ and (c) $p, \partial u/\partial y$ versus the DNS data.

4.1.2. GAN performance with one input variable

In addition to the GAN training proposed in Figure 1, the GAN was also trained using only select inputs and compared to other supervised and unsupervised learning techniques. The compared learning techniques that were used for comparison were summarized in Table 2 using a single input or multiple inputs. For consistency, the same number of samples were trained to evaluate each model at 5,000 samples. Figure 7 demonstrates the instantaneous snapshots of the trained and tested temperature gradients. Given these inputs, it can be determined that using only one of the selected inputs are able to generate the predicted well enough. Like the model in Kim and Lee [28], the GAN was able to determine that the wall variable $\partial u/\partial y$ was the most important variable to determining the temperature gradients.

However, despite using the same number of samples, the supervised learning techniques still have issues in generating the appropriate predictions at every time step, which the proposed GAN is able to do. The predictions for the GAN in comparison to the DNS data are shown in Figure 8. The predictions shown in Figure 8 are shown every 50 time steps. Based on the shown predictions, it is demonstrated that the predictions are the most accurate using $\partial u/\partial y$ as the input to the GAN model, which is consistent with Kim and Lee [28].

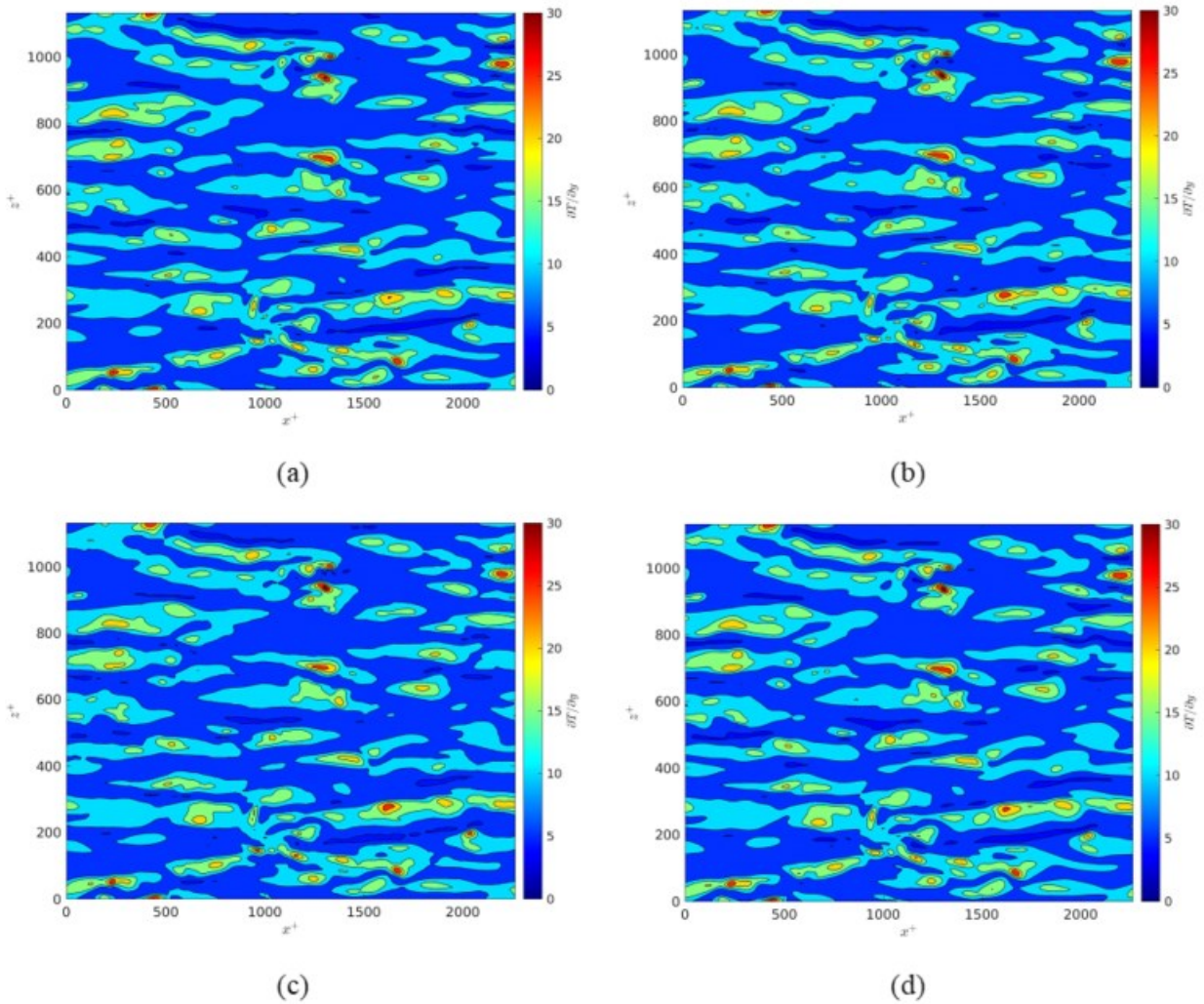

Fig. 7. Instantaneous snapshots of the predicted temperature gradients at the trained Re_τ with inputs being (a) $\partial u/\partial y$, (b) $\partial w/\partial y$ and (c) p, as well as (d) the actual temperature gradients.

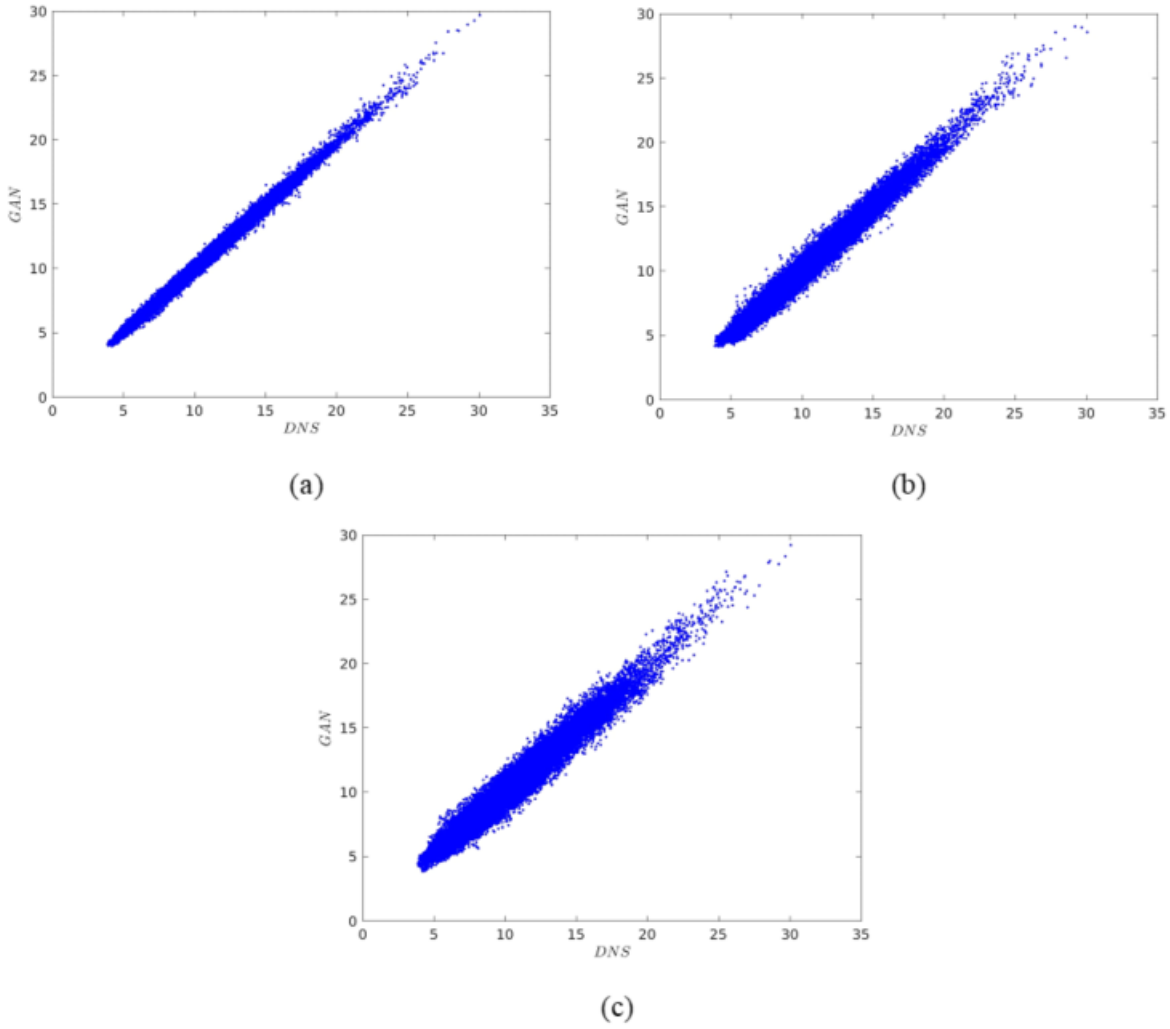

Fig 8. Scatterplots comparing the temperature gradients generated from the GAN trained with (a) $\partial u/\partial y$, (b) $\partial w/\partial y$ and (c) p versus the DNS data.

4.2. Reynolds Number Effect

This section discusses the Reynolds number effect to reconstruct the data from the trained model in Section 4.1. The mean and standard deviation for each variable was saved from the model training and was used to normalize the higher Reynolds number data. The input test data, except for the pressure, was normalized with respect to the Reynolds number via

$$\bar{X}'_{test} = \frac{X_{test}(Re_{\tau,train}/Re_{\tau,test}) - \mu_{train}}{\sigma_{train}}, \qquad (16)$$

where X_{test} is the test data. The output test data was normalized with respect to the Nusselt number

$$\left.\frac{\partial T'}{\partial y}\right|_{test} = \left.\frac{\partial T'}{\partial y}\right|_{train} \left(\frac{Nu_{train}}{Nu_{test}}\right). \qquad (17)$$

This is done because the non-dimensionalized wall-normal temperature gradient at the wall corresponds to the Nusselt number Nu for this channel flow.

The instantaneous snapshots for the predicted, and actual data are displayed in Figure 9 for Re_τ = 360 and Re_τ = 540. The results shown in Figures 9a and 9c for the Re_τ = 360 simulation, as well as Figures 9b and 9d for Re_τ = 540 were utilizing only 50 samples for the extrapolation. Additionally, the error plots at each Re_τ were displayed in Figures 9e and 9f. Because the original model was trained using 5,000 training samples at Re_τ= 180, using a proper normalization was able to obtain reasonable accuracy. While it is noted that increasing the size of the testing dataset to be at least 20% of the size of the total dataset (both training and testing combined) as a general recommendation, the proposed GAN model was still able to make reasonable predictions even with the testing dataset being only 1% of the size of the training dataset. Although accuracies do increase with increasing testing dataset sizes, this is still an encouraging result because

no additional data augmentation techniques were required [6]. Applying additional data augmentation could be a good tool to use for other testing datasets like the Re_τ = 360 and Re_τ = 540 datasets used in this analysis because the model could be prone to overfitting of the actual prediction, or increasing variance of the testing dataset based on the sample sizes. Despite that, the proposed GAN was able make the predictions within 5-10% of the actual dataset, even though the testing sample size was small in comparison to the training sample size.

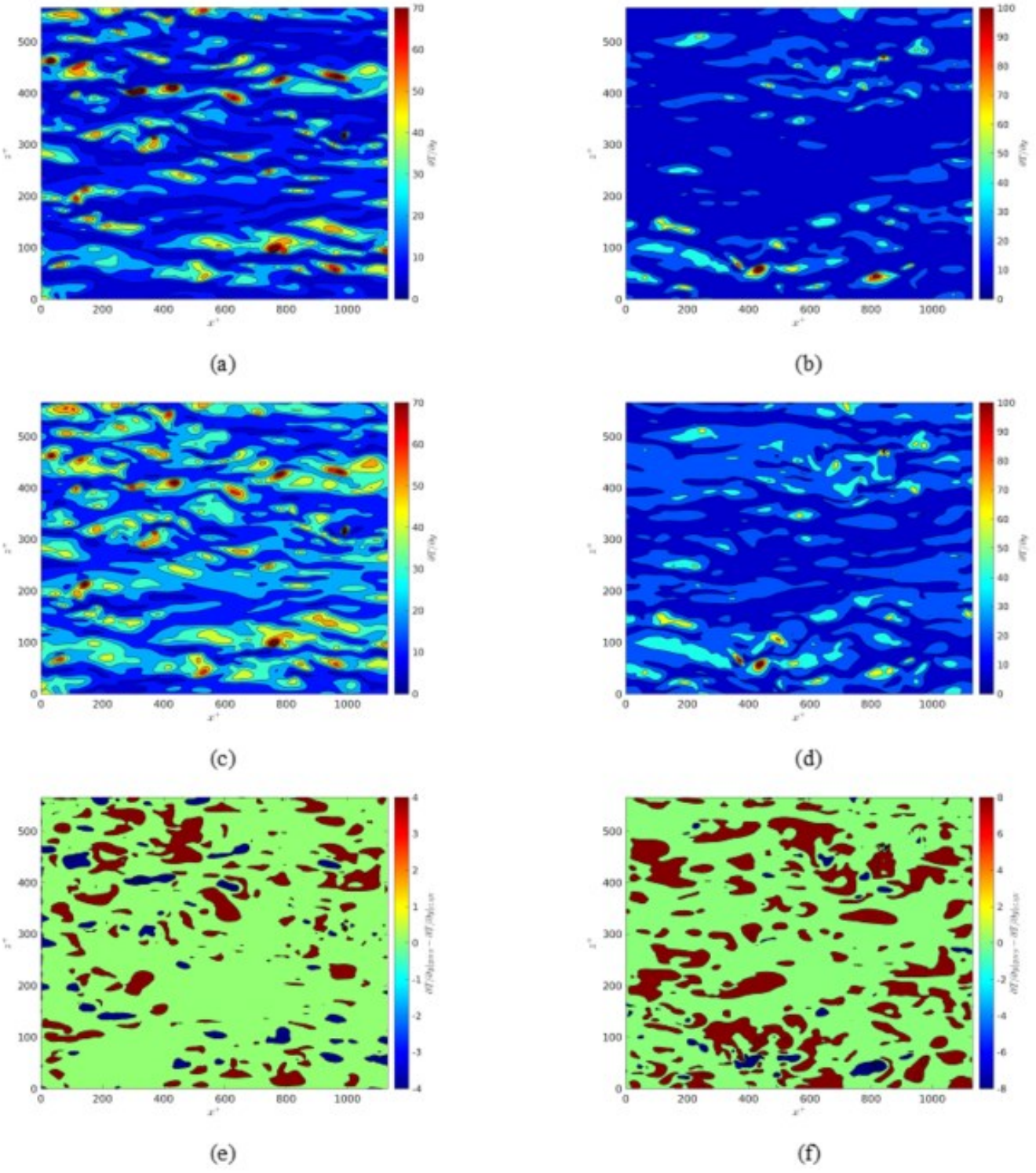

Fig. 9. Instantaneous snapshots of the predicted and actual temperature gradient at (a,b) Re_τ = 360 and (c,d) Re_τ = 540, as well as the (e,f) error plots for each Re_τ.

5. Conclusion

This study seeked to determine the turbulent heat transfer using a 2D Pix2Pix GAN as the framework for the unsupervised DL. The first objective of this study predicted the heat transfer using wall variables p, $\partial u/\partial y$ and $\partial w/\partial y$. The proposed GAN was able to predict the temperature gradient without having to measure the temperature directly. The GAN model performed well for training at the $Re_\tau = 180$ simulations, with the absolute errors being $\leq$2% even in regions of significant gradients in the values for $\partial T/\partial y$. Additionally, the proposed GAN model was able to perform significantly better than the supervised learning techniques that were shown in previous studies. Likewise, the GAN model showed improved results even with limited inputs, such as single inputs as well as multiple inputs. The second objective seeked to extrapolate the trained GAN model to temperature gradients at higher Reynolds numbers. The GAN also effectively captured Reynolds number data up to three times that of the training data. However, proper normalization was required for accurate predictions. The proposed GAN model was able to demonstrate the predictions for the higher Reynolds numbers within 5-10% error using a testing sample size that is only 1% of the size of the training dataset.

This model showed that it is able to make predictions for near-wall turbulent flow physics with heat transfer. For future work, this model could be used for other near-wall flow physics problems, such as physics-informed neural networks for flow and heat transfer predictions, as well as drag reduction predictions. By incorporating the governing equations into the DL algorithm, it may improve the predictions of the proposed GAN model for the temperature gradient predictions. This could be particularly helpful for making predictions in three-dimensional problems, where the computational expense is significantly increased in comparison to the 2D model proposed here. Also, this model can be useful for drag reduction techniques. This DL model could be applied to active flow control strategies, such as blowing and suction at the wall. The proposed model could be used to optimize the control strategies depending on the inputs required for reducing the drag coefficients in certain flow configurations. These are each very promising applications for the model, which will help to promote ML and DL models for flow physics and near-wall turbulence.

Acknowledgements

The author appreciates Dr. Stephen Ekwaro-Osire and Dr. Fazle Hussain for helpful discussion related to this study. Computational resources provided by the Texas Tech University High-Performance Computing Center are acknowledged.

References

[1] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," IEEE Transactions on Audio, Speech and Language Processing, vol. 21, no. 5, pp. 1060–1089, 2013.

[2] J. Padmanabhan, M. Jose and J. Premkumar, "Machine learning in automatic speech recognition: A survey," IETE Technical Review, vol. 32, no. 4, pp. 240–251, 2015.

[3] C. M. Bishop and N. M. Nasrabadi, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.

[4] D. Melati, Y. Grinberg, M. K. Dezfouli, S. Janz, P. Cheben, J. H. Schmid, A. Sanchez-Postigo and D. X. Xu, "Mapping the global design space of nanophotonic components using machine learning pattern recognition," Nature Communications, vol. 10, no. 1, pp. 1–9, Oct. 2019.

[5] T. Dao, A. Gu, A. J. Ratner, V. Smith, C. De Sa and C. Re, "A kernel theory of modern data augmentation," in Proc. International Conference on Machine Learning, 2019.

[6] S. Dabetwar, S. Ekwaro-Osire and J. P. Dias, "Fatigue damage diagnostics of composites using data fusion and data augmentation with deep neural networks," Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems, vol. 5, no. 2, 2022.

[7] B. S. Kusumo, A. Heryana, O. Mahendra and H. F. Pardede, "Machine learning-based automatic detection of corn-plant diseases using image processing," in Proc. 2018 International Conference on Computer, Control, Informatics and its Applications (IC3INA), pp. 93–97, 2019.

[8] H. Zerouaoui and A. Idri, "Reviewing machine learning and image processing-based decision-making systems for breast cancer imaging," Journal of Medical Systems, vol. 45, no. 1, pp. 1–20, Jan. 2021.

[9] S. L. Brunton, B. R. Noack and P. Koumoutsakos, "Machine learning for fluid mechanics," Annual Review of Fluid Mechanics, vol. 52, pp. 477–508, 2020.

[10] J. Kim, P. Moin and R. Moser, "Turbulence statistics in fully developed channel flow at low Reynolds number," Journal of Fluid Mechanics, vol. 177, pp. 133–166, 1987.

[11] J. Jimenez and P. Moin, "The minimal flow unit in near-wall turbulence," Journal of Fluid Mechanics, vol. 225, pp. 213–240, 1991.

[12] M. Lee and R. D. Moser, "Direct numerical simulation of turbulent channel flow up to Reτ ≈ 5200," Journal of Fluid Mechanics, vol. 774, pp. 395–415, 2015.

[13] R. A. Antonia, L. V. Krishnamoorthy and L. Fulachier, "Correlation between the longitudinal velocity fluctuation and temperature fluctuation in the near-wall region of a turbulent boundary layer," International Journal of Heat and Mass Transfer, vol. 31, no. 4, pp. 723–730, Apr. 1988.

[14] J. Jeong and F. Hussain, "On the identification of a vortex," Journal of Fluid Mechanics, vol. 285, pp. 69–94, 1995.

[15] J. Jeong, F. Hussain, W. Schoppa and J. Kim, "Coherent structures near the wall in a turbulent channel flow," Journal of Fluid Mechanics, vol. 332, pp. 185–214, 1997.

[16] L. Agostini and M. Leschziner, "Spectral analysis of near-wall turbulence in channel flow at Reτ = 4200 with emphasis on the attached-eddy hypothesis," 2017.

[17] W. Schoppa and F. Hussain, "Coherent structure generation in near-wall turbulence," Journal of Fluid Mechanics, vol. 453, pp. 57–108, 2002.

[18] N. Hutchins and I. Marusic, "Large-scale influences in near-wall turbulence," Philosophical Transactions of the Royal Society A, vol. 365, no. 1852, pp. 647–664, Mar. 2007.

[19] A. Lozano-Duran, N. C. Constantinou, M.-A. Nikolaidis and M. Karp, "Cause-and-effect of linear mechanisms sustaining wall turbulence," Journal of Fluid Mechanics, 2020.

[20] H. Choi, P. Moin and J. Kim, "Active turbulence control for drag reduction in wall-bounded flows," Journal of Fluid Mechanics, vol. 262, pp. 75–110, 1994.

[21] D. Babcock, C. Lee, B. Gupta, J. Kim and R. Goodman, "Active drag reduction using neural networks," in Proc. International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing, 1996, pp. 279–287.

[22] C. Lee, J. Kim, D. Babcock and R. Goodman, "Application of neural networks to turbulence control for drag reduction," Physics of Fluids, vol. 9, no. 6, pp. 1740–1747, 1997.

[23] H. J. Bae and P. Koumoutsakos, "Scientific multiagent reinforcement learning for wall-models of turbulent flows," arXiv preprint arXiv:2106.11144, 2021.

[24] K. Fukami, Y. Nabae, K. Kawai and K. Fukagata, "Synthetic turbulent inflow generator using machine learning," Physical Review Fluids, vol. 4, p. 064603, 2019.

[25] M. Z. Yousif, L. Yu and H.-C. Lim, "High-fidelity reconstruction of turbulent flow from spatially limited data using enhanced super-resolution generative adversarial network," arXiv preprint arXiv:2109.04250, 2021.

[26] H. Wang, Z. Yang, B. Li and S. Wang, "Predicting the near-wall velocity of wall turbulence using a neural network for particle image velocimetry," Physics of Fluids, vol. 32, no. 11, p. 115105, 2020.

[27] K. Ezhilsabareesh, C. Atkinson, A. Lozano-Duran, P. J. Schmid, J. Jimenez and J. Soria, "Effect of limited near-wall inlet data on the direct numerical simulation of turbulent channel flow," Journal of Physics: Conference Series, vol. 1522, p. 012019, 2020.

[28] J. Kim and C. Lee, "Prediction of turbulent heat transfer using convolutional neural networks," Journal of Fluid Mechanics, vol. 882, p. A18, 2020.

[29] H. Kim, J. Kim and C. Lee, "Interpretable deep learning for prediction of Prandtl number effect in turbulent heat transfer," Journal of Fluid Mechanics, vol. 955, 2023.

[30] S. Cai, Z. Wang, S. Wang, P. Perdikaris and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," Journal of Heat Transfer, vol. 143, no. 6, Jun. 2021.

[31] J. Kim and P. Moin, "Transport of passive scalars in a turbulent channel flow," Turbulent Shear Flows, vol. 6, pp. 85–96, 1989.

[32] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2016.

[33] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks," arXiv preprint arXiv:1611.07004, 2016.

[34] J. Henry, T. Natalie and D. Madsen, "Pix2Pix GAN for image-to-image translation," 2021.

[35] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila and H. Xiao, "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," Journal of Computational Physics, vol. 406, p. 109209, 2020.

[36] V. Nagarajan and J. Z. Kolter, "Gradient descent GAN optimization is locally stable," in Proc. Neural Information Processing Systems (NeurIPS), 2017.

[37] T. Tieleman and G. Hinton, "RMSProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. International Conference on Learning Representations, 2015.

[39] S. J. Reddi, S. Kale and S. Kumar, "On the convergence of Adam and beyond," in Proc. International Conference on Learning Representations, 2018.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. International Conference on Machine Learning, 2015.

[41] H. Noh, S. Hong and B. Han, "Learning deconvolution network for semantic segmentation," in Proc. IEEE International Conference on Computer Vision, 2015.

[42] N. Srivastava, G. Hinton, A. Krizhevsky and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

[43] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Proc. MICCAI, 2015.

[44] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," in Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 2018.

[45] H. Abe, H. Kawamura and Y. Matsuo, "Surface heat-flux fluctuations in a turbulent channel flow up to Reτ = 1020 with Pr = 0.025 and 0.71," International Journal of Heat and Fluid Flow, vol. 25, no. 3, pp. 404–419, Jun. 2004.

[46] H. Kawamura, K. Ohsaka, H. Abe and K. Yamamoto, "DNS of turbulent heat transfer in channel flow with low to medium-high Prandtl number fluid," International Journal of Heat and Fluid Flow, vol. 19, no. 5, pp. 482–491, Oct. 1998.

[47] H. Kawamura, H. Abe and Y. Matsuo, "DNS of turbulent heat transfer in channel flow with respect to Reynolds and Prandtl number effects," International Journal of Heat and Fluid Flow, vol. 20, no. 3, pp. 196–207, Jun. 1999.