SANDIA REPORT
SAND2020-10569
Printed September 2020

Sandia National Laboratories

Conditional Generative Adversarial Networks for Solving Heat Transfer Problems

Olivia Heiner, Matthew Martinez

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

Generative Adversarial Networks (GANs) have been used as a deep learning approach to solving physics and engineering problems. Using deep learning for these problems is attractive in that reasonably accurate models can be inferred from only raw data, eliminating the need to define the exact physical equations governing a problem. We expand on previous work [Farimani et al. 2017] using GANs to generate steady-state solutions to the two-dimensional heat equation. Using a basic conditional GAN (cGAN), we generate accurate solutions for rectangular domains conditioned on four edge boundary conditions (MAE<0.5%). For finding steady-state solutions over arbitrary two-dimensional domains (not constrained to rectangles), we use a cGAN designed for image-to-image translation. We train this GAN on various types of geometric domains (circles, squares, triangles, shapes with one circular or rectangular hole), achieving accurate results on test data made up of geometries similar to those in training (MAE<1%). For both of these GANs, we experiment with different loss function terms, showing that a term using the gradients of solution images significantly improves the basic cGAN but not the image-to-image GAN. Lastly, we show that the image-to-image GAN performs poorly when applied to two-dimensional geometries that vary in structure from training data (MAE<8% for shapes with multiple holes or different shaped holes). This demonstrates the cGAN's lack of generalizability. While the cGAN is an accurate and computationally efficient method when trained and tested on similarly structured data, it is a much less reliable method when applied to data that is slightly different in structure from the training data.

CONTENTS

1. INTRODUCTION

Advances in deep learning have led to data-driven approaches for solving problems within the fields of physics and engineering [1] [2]. Deep neural networks are capable of recognizing input features and learning mappings to output representations based solely on raw data. This makes data-driven approaches advantageous in that they have the potential to infer approximate solutions to physical problems that may have complex or unknown causal models [3]. In addition, aspects of these deep learning models — such as their computational efficiency and ability to generalize to different problem domains — make them useful for physical problems in which the causal model is already well known.

A useful application of deep learning models to physical problems is for modeling transport phenomena. [3] shows that deep learning models can be used as surrogate models for solving two transport problems that are conventionally approached with numerical methods. While these numerical methods are well known and fairly simple, [3] shows an approach using conditional generative adversarial networks (cGANs) that has advantages in computational efficiency and adaptability between different physical problems, while still maintaining good accuracy. The cGAN architecture used to generate solutions in [3] works for both steady-state heat transfer and fluid flow problems. This shows that even if the underlying partial differential equations differ between problems, the cGAN architecture is able to successfully infer solutions for whichever problem it is trained on. This adaptability makes it highly attractive as a potential model for other physical problems.

Although the cGAN architecture can be trained to solve various types of problems, we show that it has difficulty generalizing to data that has a slightly different structure from the training data. We experiment with a cGAN trained on example solutions of the steady-state heat transfer problem and show that while the cGAN performs well on new data with similar structure to training data (i.e. similar types of shapes), it is significantly less reliable for inferring solutions to geometries that are slightly more complex than those in the training data. If not resolved, this presents a major disadvantage of applying cGANs to scientific problems, where the accuracy of solutions is critical.

We also experiment with different loss functions for the generator network. All GANs are characterized by a similar binary cross entropy (BCE) loss function [4], but adding more terms to the loss function can further aid the GAN in solving specific problems. [2] showed that when using GANs for image generation, using the BCE loss in addition to the L1 loss between generated images and ground-truth solution images gives improved results. We experiment with this loss function term, as well as other additional terms that could improve training for the heat transfer problem specifically. In doing so, we attempt to increase the smoothness and decrease the noise of generated steady-state solutions.

2. METHOD

2.1. Generative Adversarial Networks

A generative adversarial network (GAN) is a learning model made up of two different networks: a generator and a discriminator. The generator G learns a mapping from a random input x in latent space to an output y that matches some data distribution of interest: $G : x \rightarrow y$ [4]. The discriminator D is trained to distinguish between real and generated samples. In this way, the discriminator and generator compete against each other: the generator's goal is to get better at "faking" the discriminator, while the discriminator's goal is to identify fake generated samples.

For our experiments we use a conditional GAN (cGAN), where the output y is conditioned on some input z, resulting in a mapping $G : (x,z) \rightarrow y$ [5]. The objective loss function for the cGAN can be expressed as:

$$\mathcal{L}_{BCE}(G,D) \;=\; \mathbb{E}[\log(D(z,y))] \;+\; \mathbb{E}[\log(1-D(z,G(x,z))] \tag{1}$$

Where $D(z,y)$ is the probability that sample y came from the data distribution (rather than the generator) given condition z, and $G(x,z)$ is a fake generated sample conditioned on z. The discriminator is trained to maximize this function, while the generator is trained to minimize it. Training reaches a theoretical equilibrium when the discriminator gives both real and fake samples a fifty percent probability of being real.

2.2. Additional Loss Terms

Previous work has shown that using an L1 or L2 loss term in addition to the binary cross entropy (BCE) loss term improves results and decreases blurring [6] [2]. In training the generator, we calculate the L1 norm of the mean absolute error (MAE), which is given by:

$$\mathcal{L}_{MAE} = ||y_z - G(x,z)||_1 \tag{2}$$

where y_z is a real data sample conditioned on z, and $G(x,z)$ is a generated sample conditioned on z.

In previous work applying GANs to the heat transfer problem, only the BCE and MAE loss terms were used [3]. We experiment with two additional terms. The first additional term we use calculates loss based on the gradients of the generated solution and the ground-truth solution. By calculating the L1 loss between these gradients, we attempt to generate smooth results with less sharp changes in the generated temperature field. We calculate the gradients with a sobel operator [7], which runs two 3x3 kernels convolutionally across the image; S_x for calculating the gradient in the x direction, and S_y for calculating the gradient in the y direction. The gradient magnitude S is calculated as[1]:

[1]The sobel operator typically takes the square root of this magnitude. We leave out the square root, as its derivative is undefined at zero and can cause problems in backpropogation.

$$S = S_x^2 + S_y^2 \tag{3}$$

The L1 distance between the sobel gradient of real data samples S_{real} and generated data samples S_{gen} is then calculated as:

$$\mathcal{L}_{GRAD} = ||S_{real} - S_{gen}||_1 \tag{4}$$

The last term we experiment with is a total variation term for denoising [8]. For total variation, we want to minimize the sum of the absolute gradients over the entire generated image. We average the total variation in both the x and y direction, resulting in the following loss term:

$$\mathcal{L}_{TV} = \frac{1}{2n} \sum_i \sum_j [(G(x,z)_{i+1} - G(x,z)_i) + (G(x,z)_{j+1} - G(x,z)_j)] \tag{5}$$

The final joint loss function is:

$$\mathcal{L} = L_{BCE} + \lambda_1 L_{MAE} + \lambda_2 L_{GRAD} + \lambda_3 L_{TV} \tag{6}$$

We vary each lambda to see how each loss term affects the performance of the GAN.

3. GAN ARCHITECTURE

We experiment with two different GAN architectures for solving the heat transfer problem: a basic conditional deep convolutional gan (cDCGAN) and a more specialized cDCGAN used for image-to-image translation, known as the pix2pix GAN.

3.1. Basic cDCGAN Architecture

We adapt the basic cDCGAN from [9]. The generator G for this GAN is given two inputs: a vector x of random noise and a condition vector z. The generator then passes x through a series of convolutional layers, each paired with a batch normalization (BN) layer [10] and a rectified linear unit (ReLU) activation function [11]. The final output is put through a tanh function, resulting in an output image. The discriminator D is also given two inputs: a condition vector and the image that the discriminator classifies. The discriminator is a binary classification network that processes its inputs through a series of deep convolutional layers with BN and leaky ReLU (LReLU) activations [12], and the final output is put through a sigmoid activation function. This output is the scalar probability that the input image is a real solution, as opposed to a generated one.

3.2. Pix2Pix GAN Architecture

The second architecture we use is the pix2pix GAN for image-to-image translation, adapted from [2][2]. This GAN architecture can be applied to various image-to-image applications without the need for extensive parameter tuning. Previous work demonstrates the success of this GAN in diverse applications such as stylistic transfer, image colorization, background removal, and more [2]. This architecture has been used in previous work with the heat transfer problem [3], and provides an advantage over the basic cDCGAN in that the boundary conditions are input as an image, as opposed to just four edge values. This allows us to define boundary conditions for arbitrary two-dimensional geometries, whereas the basic cDCGAN only allowed us to define a rectangular geometry. Furthermore, the basic cDCGAN maps both a noise vector and a condition vector to a solution image, whereas the pix2pix GAN directly maps the image containing boundary conditions to a solution image (no input noise vector).

3.2.1. U-Net Generator

The generator network for the pix2pix GAN uses an encoder-decoder architecture. The input image goes through a series of downsampling convolutional layers, until it reaches a bottleneck and is upsampled to its original size. Any information from the input image that is needed to construct the output representation must pass through the bottle neck. In image-to-image translation problems, it is useful to maintain low-level information, such as the structure of edges. Because of this, previous work proposes using skip connections that pass information between equivalently sized layers of the encoder-decoder architecture, allowing low-level information to pass by the bottleneck [2]. This is known as a U-Net encoder-decoder [13].

3.2.2. Patch-GAN Discriminator

The discriminator for the pix2pix GAN uses a patch-GAN architecture [2], which only penalizes structure on a local scale. The discriminator runs an NxN patch convolutionally across the image array, classifying each patch as real or fake. The classification outputs for each patch are then averaged, giving the final output of the discriminator D.

4. EXPERIMENTS

4.1. Data

Two different datasets were generated, one for each of the two GAN architectures. Both datasets contain input/solution pairs, where the input is an encoding of boundary conditions and the

[2]The pix2pix GAN (including both the U-Net and Patch-GAN) parameters and architecture are the same as those given by [2].

solution is the corresponding temperature of the steady-state solution to the two-dimensional heat equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 u}{\partial t^2} \tag{7}$$

The input vector for the basic cDCGAN encodes four boundary conditions, where each boundary condition is generated randomly from the continuous uniform distribution on the range 0 °C-100 °C. Each boundary condition corresponds to one edge of a 32x32 rectangular surface and is generated independently of the other edges. The corresponding steady-state solution in each pair is represented as a 1-channel 32x32 image, where the entire image is the domain over which we solve for the solution. 10,000 samples are generated in this way and are split 90/10 into train/test sets. A representative data sample is plotted in figure 4-1.

The data for the pix2pix GAN differs from the data for the basic cDCGAN in that it represents the heat transfer problem for various two-dimensional geometries (including some partially-filled geometric domains), rather than just a 32x32 rectangular surface. The input is encoded as a 2-channel 256x256 image: the first channel represents the boundary conditions on the border of the geometric domain of interest, and the second channel represents the interior region of that geometric domain with a boolean mask. The boundary conditions are generated randomly from the continous uniform distribution on the range 0 °C-100 °C, similar to the cDCGAN. The area of the input image outside of the domain of interest is set to 0. The training set consists of squares, rectangles, triangles, and circles. Variants of these geometries with fully filled domains, domains with a single circular hole, and slotted domains (single rectangular hole) make up the 11 geometries in the training set. 1,000 samples of each of these 11 geometries are generated, with a 90/10 train/test split. 4 additional geometries — squares with 4 holes, squares with one hexagonal hole, right triangles with one hole, and general triangles with one hole — are generated only for the test set, with 100 samples each. In total, there are 9,900 training samples and 1,500 test samples. A representative data sample is plotted in figure 4-2.

Solutions for all sample pairs are generated by the finite element method (FEM) with MATLAB's PDE Toolbox [14].

4.2. Experiments with Basic cDCGAN

We run experiments training the GAN from section 3.1 in PyTorch [15]. The GAN is trained on data described in section 4.1. We run trials using four different combinations of the loss functions from section 2:

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} \tag{8}$$

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} + \lambda_1 \mathcal{L}_{MAE} \tag{9}$$

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} + \lambda_1 \mathcal{L}_{MAE} + \lambda_2 \mathcal{L}_{GRAD} \tag{10}$$

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} + \lambda_1 \mathcal{L}_{MAE} + \lambda_2 \mathcal{L}_{GRAD} + \lambda_3 \mathcal{L}_{TV} \tag{11}$$

We run each experiment for 200 epochs, using a training batch size of 256. The model from the epoch with the lowest mean absolute error (MAE) between the GAN-generated solution and the ground-truth solution is used to evaluate performance for each of the loss functions.

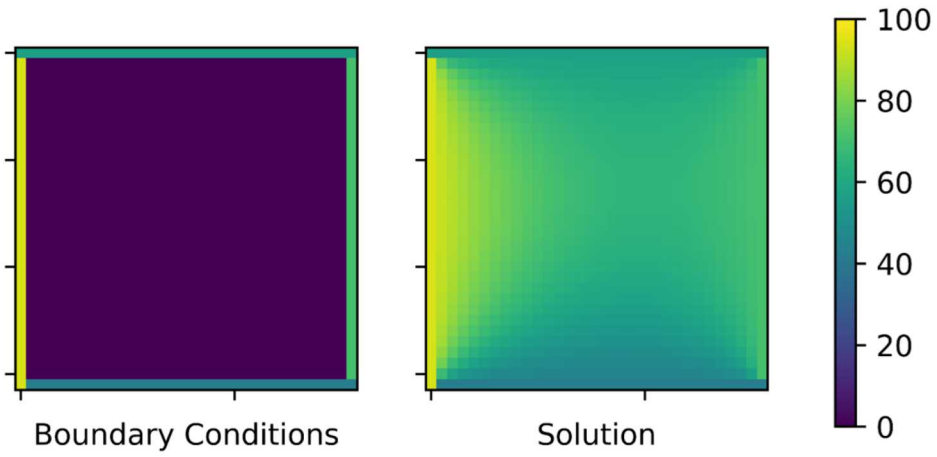

Figure 4-1 Data Sample for Basic cDCGAN: Sample boundary conditions/solution pair generated for training basic cDCGAN. Solution generated by FEM. Units in °C.

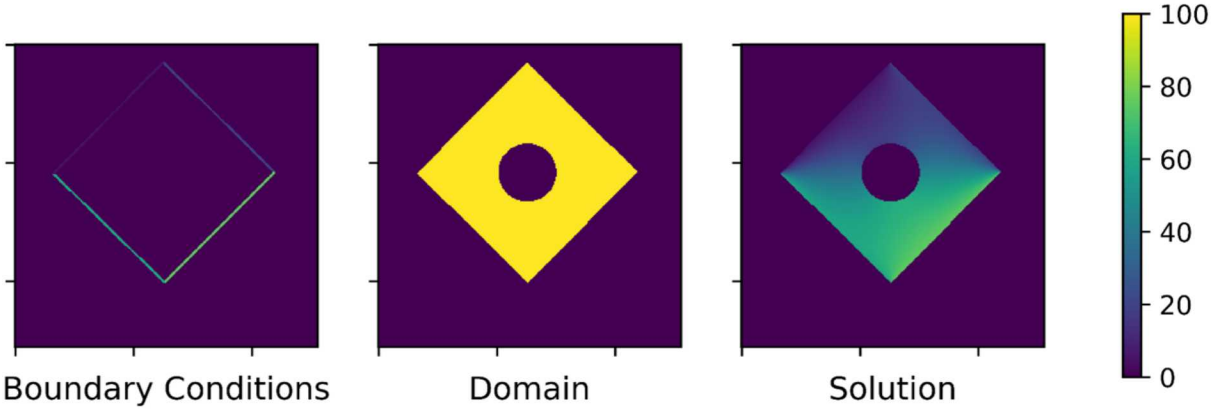

Figure 4-2 Data Sample for Pix2pix GAN: Sample input/solution pair generated for training pix2pix GAN. Boundary conditions are the first channel of the input, the geometric domain mask is the second channel of the input. Third image is solution generated by FEM. Units in °C.

4.3. Experiments with Pix2Pix GAN

We use the pix2pix GAN from section 3.2 as an image-to-image approach for solving the heat transfer problem for various geometries. The GAN is trained on data described in section 4.1. We experiment with different loss functions, using equations 9,10, and the following equation:

$$\mathcal{L}_{total} = \mathcal{L}_{BCE} + \lambda_1 \mathcal{L}_{MAE} + \lambda_3 \mathcal{L}_{TV} \tag{12}$$

We run each experiment for 200 epochs in PyTorch, using a training batch size of 32. The model from the epoch with the lowest MAE between the GAN-generated solution and the ground-truth solution is used to evaluate performance for each of the loss functions. In addition to evaluating the different loss functions, we evaluate the performance of the pix2pix GAN on some geometries that are not used in training. These geometries are described in section 4.1. This is to analyze how the GAN generalizes to data with a slightly different structure from training data.

5. RESULTS

We use a test set as described in section 4.1 to evaluate the performance of the basic cDCGAN on 32x32 rectangular surfaces. The basic cDCGAN achieves realistic results for solving the heat equation on this data. A representative generated solution from the test set is shown in figure 5-2. For data with a range of 100 °C, equations 8, 9, 10, and 11 achieve MAEs of 1.182 °C, 0.941 °C, 0.386 °C, and 0.603 °C, respectively. This gives the loss function expressed in equation 10 the best performance, with an MAE of 0.386 °C. This shows an improvement from the loss function (equation 9) used in previous works [3] [2] by adding the L1 loss between the gradients of the GAN-generated solutions and ground-truth solutions. Using the total variation of generated solutions, however, does not improve the loss function.

We evaluate results of the pix2pix GAN on a test set of unseen data consisting of various two-dimensional geometries, as described in section 4.1. To evaluate the generalizability of the pix2pix GAN, the test set includes four different geometries that aren't present in the training data (squares with 4 holes, squares with one hexagonal hole, right triangles with one hole, and general triangles with one hole). For data with a range of 100 °C, equations 9, 10, and 12 achieve mean absolute errors (MAEs) of 1.228 °C, 1.502 °C, and 1.244 °C, respectively. This gives the loss function expressed in equation 9 the best performance, with an MAE of 1.228 °C. This loss function is the loss function used in previous works for pix2pix problems [2] and the heat transfer problem [3]. Although adding a term involving the generated solutions' gradients helps the basic cDCGAN, it does not show any improvement for the pix2pix GAN. Like the basic cDCGAN, using total variation (equation 12) also does not provide improvement.

The pix2pix GAN is able to generalize well to unseen data, but only if the unseen data is made up of geometric structures similar to those in the training data. It doesn't not generalize well to geometries that are slightly more complex than those in the training data. The GAN achieves an MAE of 7.150 °C on squares with a hexagonal hole and an MAE of 2.255 °C on squares with four holes, which are both significantly higher than the MAE over the whole test dataset. Without

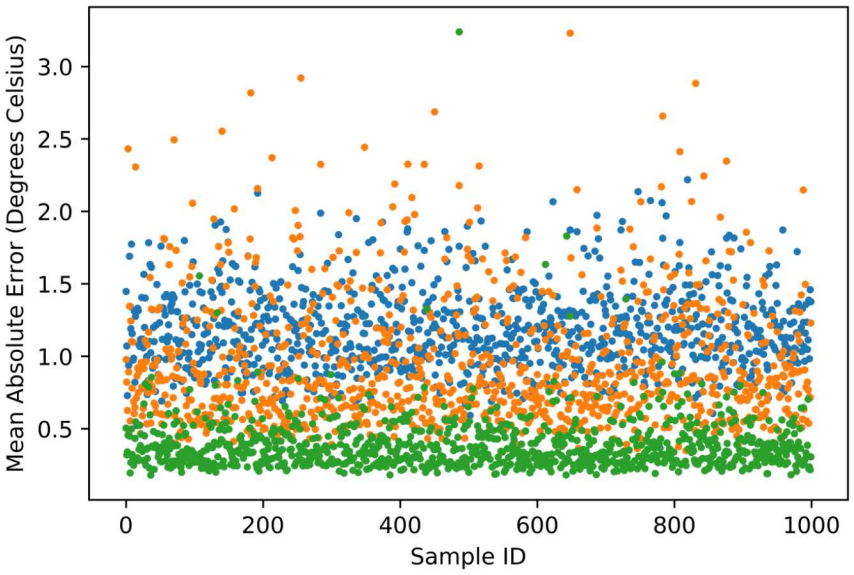

Figure 5-1 Evaluation of Different Loss Functions For Basic cDCGAN: Performance of the GAN on test data. Three different loss functions (equations • 8, • 9, and • 10) are shown. Loss function with total variation term (equation 11) ommitted from visualization, as the total variation term did not have much effect on results.

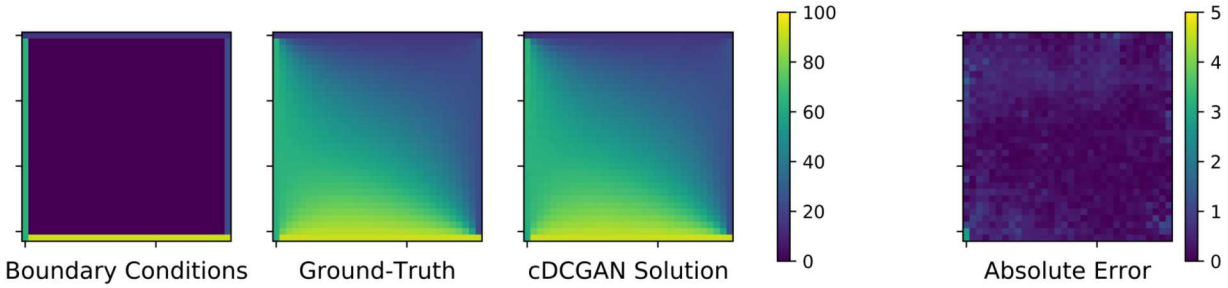

Figure 5-2 Solutions Generated with Basic cDCGAN: Solution generated using loss function 10. Units in °C.

those two shapes, the GAN achieves an MAE of 0.693 °C, which is similar to previous work [3]. The other geometry included only in the test data is a triangle with a hole, which had an MAE of 0.645 °C, making it better than the average MAE over the entire test set. Although triangles with a hole were not in the training data, the training data included both triangles and other shapes that had a circular hole. Their higher similarity to training geometries could explain why the GAN performed better on triangles with a hole than on squares with a hexagonal hole and squares with four holes, even though they were all excluded from training data.

Overall, the pix2pix experiments show successful results on training geometries, but difficulty generalizing to data that is even just slightly more complex than the training data. This is highlighted in figure 5-5, where the MAEs of different test samples are plotted. Some representative solutions generated from test data are shown in figures 5-3 and 5-4.

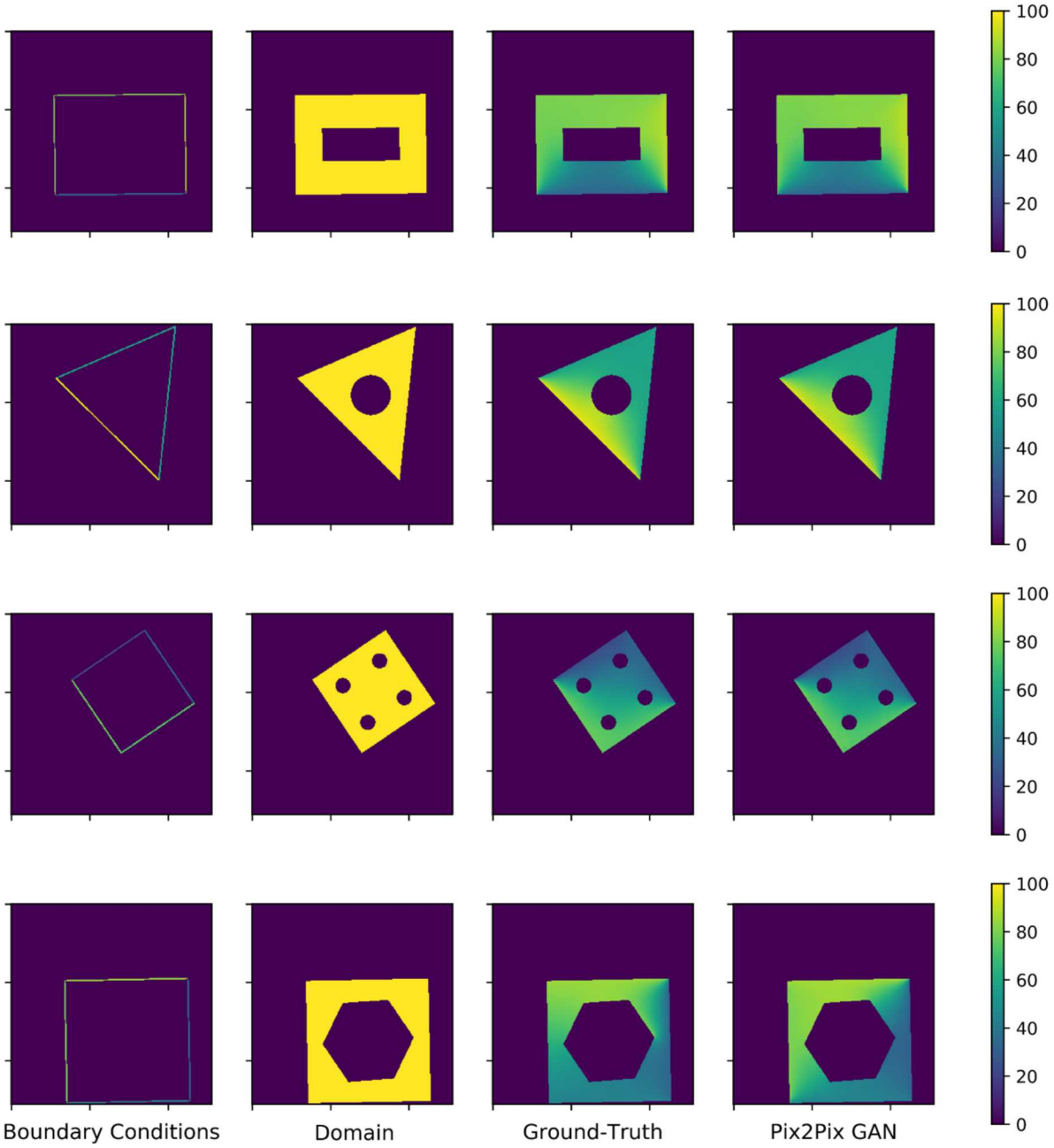

Figure 5-3 Solutions Generated with Pix2Pix GAN: Geometries from top to bottom row: slotted rectangle (in train and test data), triangle with hole (only in test data), square with four holes (only in test data), and square with a hexagonal hole (only in test data). Units in °C.

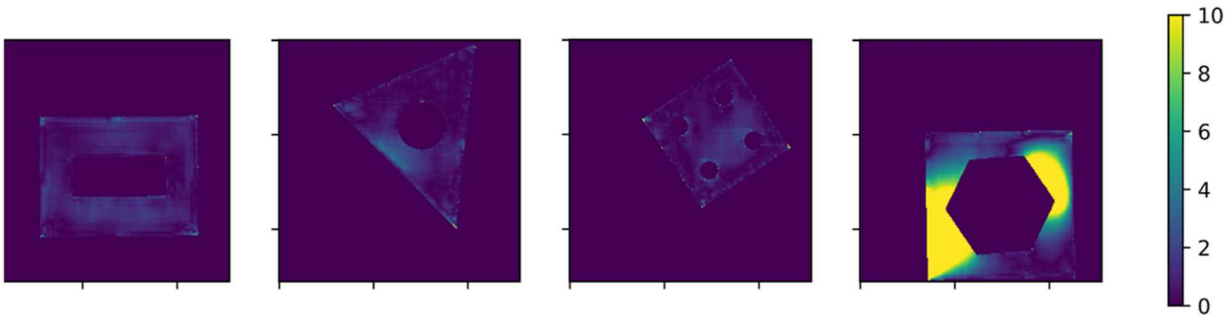

Figure 5-4 Absolute Error (Per Pixel) of Pix2Pix Solutions: Absolute error between pix2pix generated solutions and ground-truth solutions for the samples plotted in figure 5-3. Units in °C. Last plot (square with hexagonal hole) is an example of where the GAN failed to generalize to unseen geometries, resulting in high error.

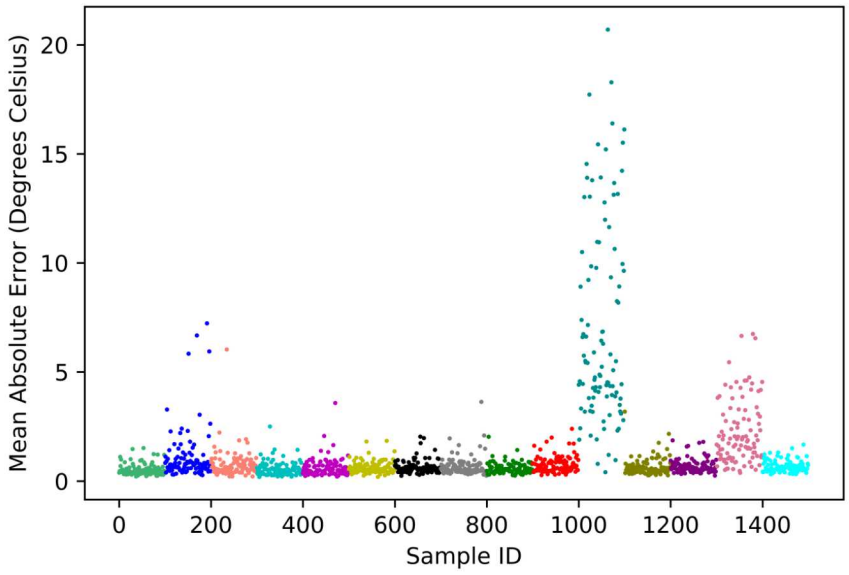

Figure 5-5 Pix2Pix Gan Performance on Test Set Geometries: Mean Absolute Error for 1500 test set samples. Legend (in order from left to right): • triangle, • slotted circle, • rectangle, • circle, • square, • circle with hole, • square with hole, • rectangle with hole, • triangle with hole, • slotted rectangle, • square with hexagonal hole, • right triangle, • right triangle with hole, • square with four holes, • slotted square.

6. CONCLUSION

In this report we have used two different GANs (a basic cDCGAN and a pix2pix GAN) as surrogate models for finding solutions to the heat equation over two-dimensional surfaces. We have evaluated the performance of these GANs when trained with different loss functions. For the basic CDCGAN, we have shown that using the L1 loss between the gradients of generated solutions and the gradients of ground-truth solutions improves training. For the pix2pix GAN, the additional loss function terms we tested didn't offer any improvement on the typical pix2pix loss function. We also showed that both GANs were able to generalize well to test data, as long as the test data was similar in structure to the training dataset. When evaluated on geometric surfaces that are slightly more complex than those in the training data, however, the pix2pix GAN suffered from high error. This exposes a weakness in using GANs as surrogate physics and engineering models. While the GAN is appealing in its ability to learn a reasonably accurate solution mapping from only raw data, this mapping is unreliable and has limited use if it can't generalize well to new data. Until generalizability is improved, the GAN will remain a less reliable method than others for accurately solving physics and engineering problems.

REFERENCES

[1] R. Sharma, A. B. Farimani, J. Gomes, P. Eastman, and V. Pande, "Weakly-supervised deep learning of heat transport via physics informed loss," 2018.

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

[3] A. B. Farimani, J. Gomes, and V. S. Pande, "Deep learning the physics of transport phenomena," 2017.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[5] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[6] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CVPR 2016*, 2016.

[7] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project 1968*, 02 2014.

[8] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.

[9] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[11] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv*, p. arXiv:1803.08375, Mar. 2018.

[12] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[14] MATLAB. Natick, Massachusetts: The MathWorks Inc.

[15] 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," *NIPS 2017 Workshop*, 10 2017.

DISTRIBUTION

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Tu-Thach Quach	09364	tong@sandia.gov
Justin Newcomer	01462	jtnewco@sandia.gov
Michael Glinsky	01684	meglins@sandia.gov
Technical Library	01911	sanddocs@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.