Fast prediction of indoor airflow distribution inspired by synthetic image generation artificial intelligence

Article *in* Building Simulation · March 2023

DOI: 10.1007/s12273-023-0989-1

8 authors, including:

Cary Faulkner
University of Colorado Boulder
18 PUBLICATIONS **79** CITATIONS
SEE PROFILE

Dominik Jankowski
Graz University of Technology
2 PUBLICATIONS **1** CITATION
SEE PROFILE

John Castellini
University of Colorado Boulder
9 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Wangda Zuo
Pennsylvania State University
174 PUBLICATIONS **2,759** CITATIONS
SEE PROFILE

Fast Prediction of Indoor Airflow Distribution Inspired by Synthetic Image Generation Artificial Intelligence

Abstract

Prediction of indoor airflow distribution often relies on high-fidelity, computationally intensive Computational Fluid Dynamics (CFD) simulations. Artificial intelligence (AI) models trained by CFD data can be used for fast and accurate prediction of indoor airflow, but current methods have limitations, such as only predicting limited outputs rather than the entire flow field. Furthermore, conventional AI models are not always designed to predict different outputs based on a continuous input range, and instead make predictions for one or a few discrete inputs. This work addresses these gaps using a Conditional Generative Adversarial Network (CGAN) model approach, which is inspired by current state-of-the-art AI for synthetic image generation. We create a new Boundary Condition CGAN (BC-CGAN) model by extending the original CGAN model to generate 2D airflow distribution images based on a continuous input parameter, such as a boundary condition. Additionally, we design a novel feature-driven algorithm to strategically generate training data, with the goal

of minimizing the amount of computationally expensive data while ensuring training quality of the AI model. The BC-CGAN model is evaluated for two benchmark airflow cases: an isothermal lid-driven cavity flow and a non-isothermal mixed convection flow with a heated box. We also investigate the performance of the BC-CGAN models when training is stopped based on different levels of validation error criteria. The results show that the trained BC-CGAN model can predict the 2D distribution of velocity and temperature with less than 5% relative error and up to about 75,000 times faster when compared to reference CFD simulations. The proposed feature-driven algorithm shows potential for reducing the amount of data and epochs required to train the AI models while maintaining prediction accuracy, particularly when the flow changes non-linearly with respect to an input.
Keywords: artificial intelligence, indoor airflow, conditional generative adversarial network, computational fluid dynamics.

1. Introduction

Simulation of indoor airflow distribution can be used for understanding indoor air quality, thermal comfort, and building energy eficiency. Computational Fluid Dynamics (CFD) methods are a popular approach for indoor airflow simulation [1] that numerically solve the governing equations of fluid flow,

such as conservation of mass, momentum, and energy. For example, Han et al. [2] used a CFD approach to simulate data center cooling scenarios to provide guidance for reducing energy consumption while meeting thermal requirements. Researchers have also used CFD methods to study the impact of ventilation strategies on thermal comfort [3, 4, 5], evaluate dispersion of airborne pollutants [6, 7, 8], and more. While these methods have been useful in many applications related to indoor airflow, they are still limited in a number of ways. First, CFD simulations are computationally expensive [9] and may be too slow for applications such as long-term evaluations (e.g., annual simulations) or optimization problems requiring thousands of realizations. Additionally, it is often infeasible to perform real-time or faster simulations using CFD, which can be required for emergency management scenarios [10]. Therefore, there is a need for computationally eficient methods of indoor airflow distribution prediction.

Modifications to traditional CFD methods have been proposed to accelerate these numerical simulations, such as Fast Fluid Dynamics methods [11, 12]. While these methods can be used for real-time or faster flow simulations, they may sacrifice some accuracy compared to traditional CFD methods. They are also unable to perform real-time simulations for more complex

airflow scenarios. Data-driven regression methods, such as in situ adaptive tabulation [13, 14], can quickly predict key flow information, such as the occupied zone temperature. However, these methods are often used to predict a few key outputs rather than the entire flow distribution.

Artificial intelligence (AI) methods have emerged as a popular approach to address some of the limitations of indoor airflow prediction. Zhou and Ooka [15] analyzed using deep neural networks for isothermal airflow distribution prediction in an office room and found the trained networks could accurately predict the velocity distribution 1.9 million times faster than reference CFD simulations. Generative Adversarial Network (GAN) [16] models have become a popular AI approach, particularly for generating synthetic images [17, 18]. Variations of the original GAN model have been introduced, such as Conditional Generative Adversarial Network (CGAN) [19] models, which generate synthetic data based on categorical labels. In [19], the categorical labels include different single digit numbers and the CGAN model generates synthetic hand-drawn images for each specified digit. These models have been used for generating synthetic images of faces at different ages [20], producing power demand profiles for different types of buildings [21], and predicting wireless networking environments [22]. CGAN models trained by

CFD simulations have been used for flow prediction, for example Chen et al. [23] predicted airflow over different airfoil shapes and Wang et al. [24] predicted the temperature distribution of different vortex generator designs for film cooling.

While significant advances have been made to use AI for indoor airflow prediction, some limitations still exist. First, many AI models are designed to predict one or a few critical outputs in the airflow (e.g., average temperature in the occupied zone or temperature values at a few sensor locations) rather than the entire airflow distribution, which is necessary for many modeling applications. GAN and CGAN models have shown significant potential for image generation, including prediction of flow distribution, and are thus worthy of further research for indoor airflow prediction. However, they are often designed to make predictions based on discrete categorical inputs, for example different specific designs or geometries. However, in many scenarios it is useful to be able to make predictions based on continuous input parameters. For example, boundary conditions, such as supply airflow rate in indoor airflow simulations, may be optimized based on a continuous design space. This is because the supply airflow rate can be optimized to be any possible value within a defined range, rather than being limited to a few select values.

Finally, generating training data for AI models using CFD simulations can be time consuming, and it is not always clear how much data should be included to train the models.

To address the limitations discussed, we create a new Boundary Condition CGAN (BC-CGAN) model for predicting indoor airflow distribution based on a continuous input parameter, such as a boundary condition. Furthermore, we design a novel feature-driven algorithm for eficiently generating training data. The algorithm minimizes the amount of generated data while ensuring a diverse set of training data for AI models by selecting data points based on significant changes in the flow output. The feature-driven algorithm and BC-CGAN model are evaluated for two benchmark airflow cases: 1) an isothermal lid-driven cavity flow and 2) a non-isothermal mixed convection flow with a heated box. CFD simulations are used to generate the training, validation, and test data for both cases. Using the dimensionless Reynolds number as an input, the BC-CGAN model predicts the velocity magnitude distribution for the lid-driven cavity case. For the mixed convection flow with heated box case, the BC-CGAN model predicts the velocity magnitude and temperature distributions based on the heat flux of the box. The model is also designed to reproduce the existence of an obstruction in the flow, in this

case the box within the flow. Finally, noise is added to some of the boundary conditions in the CFD simulations for this case to mimic uncertainty in experimental conditions.

Our specific scientific contributions include: 1) creating a new BC-CGAN model by extending the traditional CGAN model to generate images of flow distribution using a continuous input parameter (e.g., a boundary condition); 2) designing a novel feature-driven algorithm to strategically reduce the amount of required training data while ensuring training quality for AI models; and 3) demonstrating the BC-CGAN model framework for two benchmark flow cases and showing the trained model can predict airflow distribution with less than 5% relative error and up to about 75,000 times faster when compared to reference CFD simulations.

The rest of this paper is organized as follows. The BC-CGAN model is introduced in Section 2. We then detail the feature-driven algorithm for generating training data in Section 3. Next, Section 4 outlines the entire workflow including generation of training data, model training, and model evaluation. After that, Section 5 presents the results for the isothermal lid-driven cavity flow case and Section 6 shows the results for the non-isothermal mixed convection flow with heated box case. Finally, conclusions are drawn

in Section 7.

2. New Boundary Condition Conditional Generative Adversarial Network (BC-CGAN) Model

In this section, we first provide an overview of the original GAN and CGAN models, then detail our new BC-CGAN model.

2.1. Original GAN and CGAN Models

The CGAN model used to develop BC-CGAN is based on the original GAN model [16], but modified to generate images based on different classes. GAN-based models are selected for this study because of their strength in image generation [25], which is useful for applying to prediction of indoor airflow distribution. Furthermore, they have been extended for 3D image generation applications [26, 27, 28], which can be beneficial for 3D airflow prediction. As shown in Figure 1 (left), the original GAN model consists of two competing neural networks: a generator (G) and discriminator (D). The generator receives a vector containing randomly generated noise (z) as an input and attempts to output an image accordingly (G(z)). The discriminator receives a mixture of real images (x) randomly selected from the training data and synthetic or “fake” images produced by the generator.

The discriminator then attempts to correctly classify each image as real or fake, and the output of the discriminator ($D(G(z)|x)$) is compared with the correct classification of real or fake. Based on this final outcome, the weights of the generator (θ_G) and discriminator (θ_D) networks are updated and the process repeats, starting with new batches of training images and new noise inputs. Early in the training process, the generator has not learned how to output realistic images, so the discriminator is able to easily classify the images and the generator performs poorly. As the generator is trained over many iterations, it learns how to produce more realistic images and is able to fool the discriminator. Eventually, the generator produces images that are so realistic that the discriminator can no longer distinguish between the real and fake images. Ideally, the training process would reach a quasi-equilibrium state where the discriminator has a 50-50 guess at whether images are real or fake, and the training process can stop. Once the model is trained, the discriminator is no longer needed and synthetic images can be generated by providing a noise input to the generator. Although GAN models can be dificult to train [29], in part because they involve two neural networks, they have demonstrated advantages in image prediction over other types of models [29].

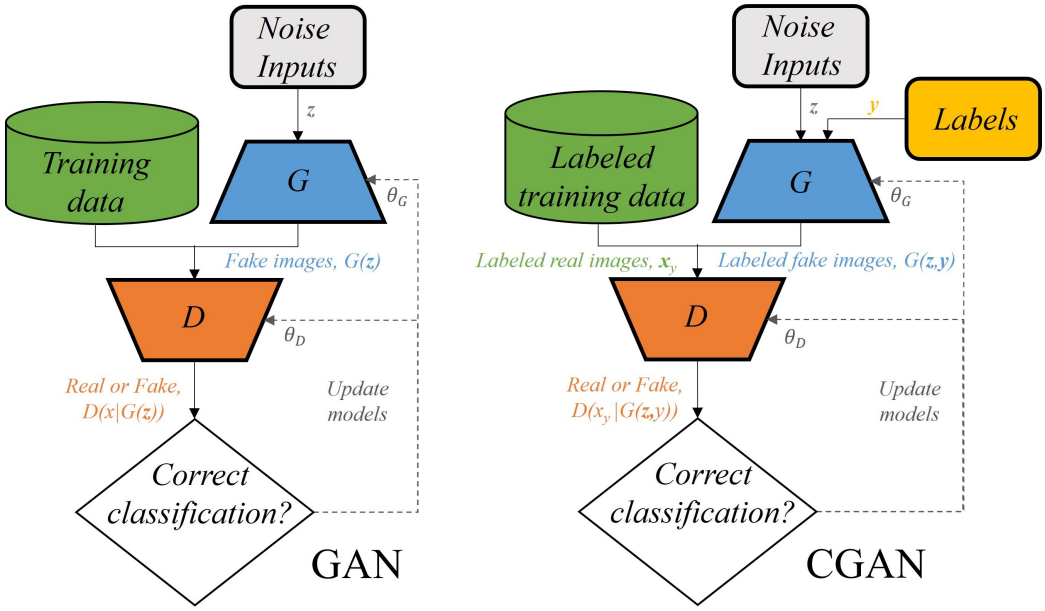


Figure 1: GAN training architecture (left) and CGAN training architecture (right).

What differentiates the CGAN model from the original GAN model is the addition of labels, as shown in Figure 1 (right). The generator receives a label (y) as an input in addition to the noise input and attempts to produce an image based on this label ($G(z, y)$), for example an image of a specific digit in [19]. The discriminator then instead receives labeled real images (x_y) mixed with labeled images produced by the generator, and determines whether an image is real or fake considering the received label. After training, synthetic images can be generated for a specific category by providing the generator with a label and noise input. The use of labels

is convenient for generation of images based on specific categories, which is accomplished by assigning unique labels to the training data from each category. Since the original GAN model generates images based on the aggregated, non-categorized training data, the CGAN model is adopted in this work to utilize labels for categorizing training data. This can mean predicting airflow patterns for different labeled building designs, as in [30]. The use of labels, however, also adds complexity to training the model, because the model needs to be trained to generate images for multiple categories.

2.2. BC-CGAN Model

We extend the previously described CGAN model to create a new BC-CGAN model in this work that generates images using continuous inputs, rather than a few discrete input classes. An example of using discrete input classes would be prediction of airflow distribution in different rooms, where each specific room configuration would be the discrete input class determining the output airflow. Indoor airflow simulations often involve input parameters that can be considered continuous rather than discrete, such as a boundary condition like the supply airflow rate. This can be considered as a continuous parameter because it can have any possible value within a defined range (e.g., any value between 1-5 kg/s), rather than only a few possible values (e.g., a

few potential room configurations). The change in output airflow distribution can then be studied by varying the input parameter continuously within its range.

Modifying the existing CGAN model to make predictions based on continuous input variables poses a challenge, since the original CGAN model is designed to make predictions based on discrete integer labels. Although a continuous input range can be discretized to assign labels for a CGAN model to many possible values within that range, this cannot cover every possible value, and a simpler method can be adopted. Thus, the structure of the CGAN models needs to be changed to address this challenge. First, the we want the trained CGAN model to receive an input that represents a specific continuous input value. Additionally, it is convenient during the training process to assign discrete labels to the training data, but a method is needed to convert these labels to the actual values of the continuous input for the model.

The new BC-CGAN model is shown in Figure 2. The difference in this model is the input to the generator, where the noise inputs are replaced with inputs defined specifically by labels, using a translator. A noise input is used in the previous CGAN models to generate a distribution of images within a category, for example different handwriting styles for a specific digit in [19].

In this work, we want to generate a specific output given a specific input, rather than some distribution of outputs for a given input. Thus, the noise aspect is removed, since a single output for each input is desired rather than a distribution of outputs for each input. It should be noted Zheng et al. [31] proposed an alternative CGAN approach considering continuous inputs with the noise aspect included to produce a distribution of outputs.

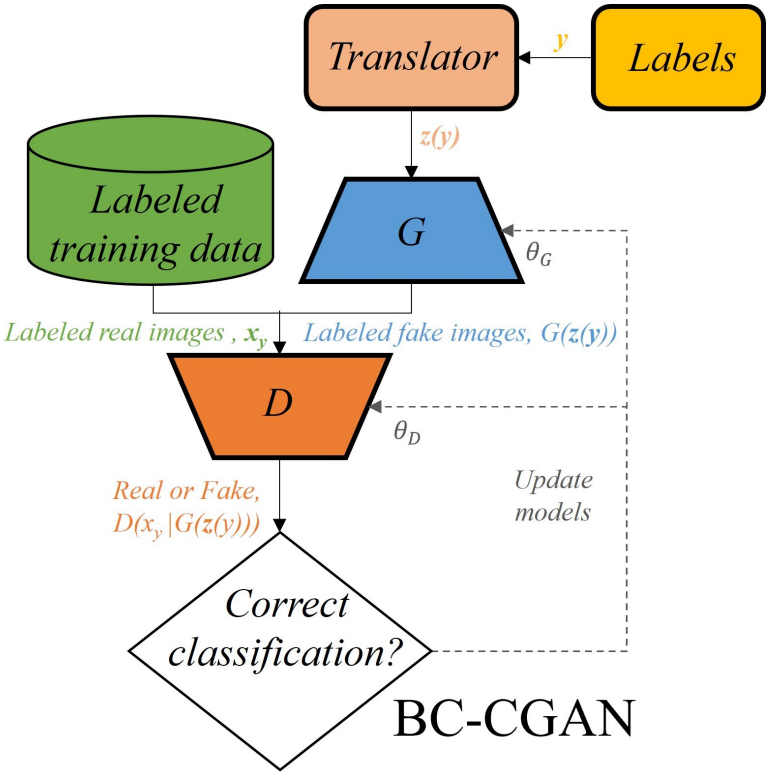


Figure 2: BC-CGAN training architecture.

The BC-CGAN model uses labels during the training process as a convenient

method for categorizing training data using an integer value. For generating images after the model is trained, we want to provide the model specific input parameter values directly instead of labels. The translator is thus needed during the training process to convert the received integer label to a specific input parameter value for the generator. This allows for: 1) the convenient use of labels during the training process and 2) the ability for the generator to learn to produce images based on an input parameter value, therefore removing the need for labels and a translator after the model is trained. Since the discriminator is only used during the training process, a simple integer value label can be provided to the discriminator and a translator is not needed to convert the label to its parameter value.

The goal of the translator is to map the received labels to the associated input parameter values. Consider a scenario where the input boundary condition for a given flow is a characteristic velocity U_0. For this case, a training label (y) is assigned to each training data image defined by their different U_0 values. The purpose of the translator is then to map y to its corresponding U_0 value. The function the translator uses to map these values, $U_0(y)$, is dependent on the training data distribution. Assume a uniformly distributed training dataset where each data point is chosen based

on a change in U_0 of ΔU_0, calculated as:

$$\Delta U_0 = (U_{0,max} - U_{0,min})/n_{train}, \quad (1)$$

where $U_{0,max}$ and $U_{0,min}$ are the minimum and maximum U_0 values in the training dataset, respectively, and n_{train} is the total number of training data points in the dataset. The function for the translator to map y to a corresponding U_0 would then be:

$$U_0(y) = \Delta U_0 \cdot y + U_{0,min}. \quad (2)$$

Finally, interpolation is used so the input to the generator, z(y), is a value between zero and one:

$$z(y) = [U_0(y) - U_{0,min}]/[U_{0,max} - U_{0,min}], \quad (3)$$

After training, images can be generated for a specific U_0 value by replacing $U_0(y)$ in Equation 3 with the desired U_0 value. allows for prediction of images using the continuous input value directly and without the need for a label or translator. Furthermore, the BC-CGAN model can be trained to make predictions considering multiple boundary conditions by using a vector containing multiple labels. The translator would then output a vector based

on the input label vector (i.e., $z(y)$). As a starting point, this paper considers one varying input parameter for each case.

3. Novel Feature-driven Algorithm for Generation of Training Data

Although trained AI models can produce results quickly, generation of training data for the AI models can be time consuming (e.g., by using CFD). Using uniformly distributed inputs for training data generation is a simple approach, but this may include more training data than an AI model actually needs. For an indoor airflow prediction AI model, training data may be generated by varying the supply airflow rate by a constant step of $\Delta\dot{m}$ in each CFD simulation. This may pose a problem when the outputs vary non-linearly with the inputs, for example if the airflow distribution varies non-linearly with the supply airflow rate. If the resolution of generated training data is not suficiently high, a uniformly distributed training dataset may exclude crucial points in regions where the gradient of the outputs with respect to the inputs ($\nabla_x f$) is high. This may exclude crucial data points that capture the non-linear trends between inputs and outputs. On the other hand, in regions where $\nabla_x f$ is low, a high resolution of generated training data may result in many redundant data points and excessive time required

to produce the training data. When generation of training data is costly, a non-uniformly distributed training dataset can be beneficial for training AI models, since redundant training data points can be avoided while still including suficient training data for the AI models.

To address this problem, we propose a novel feature-driven algorithm to create non-uniformly distributed training datasets that minimize the amount of generated training data for AI models. The algorithm strategically selects training data points based on significant changes in the outputs with respect to the inputs. In its strategic selection, the algorithm includes more data points in regions where $\nabla_x f$ is high and excludes redundant data points in regions where $\nabla_x f$ is low. The feature-driven algorithm can be used for multiple inputs (e.g., multiple varying boundary conditions). However, we focus on a single varying input parameter in this paper and explain the algorithm in detail assuming one varying input.

The feature-driven algorithm flowchart is detailed in Figure 3. First, initial grid points for the training dataset are included to provide a few baseline points, as well as to create a defined range for the training data. The grid points for the training data are defined by their different values of inputs, represented by x. The algorithm begins by computing the changes in critical

outputs between neighboring grid points, for example between x_1 and x_2. For each pair of neighboring grid points, if the change in critical outputs (e.g., $|f(x_2) - f(x_1)|$) is greater than the defined threshold, ε, a new grid point is added between those two points (e.g., at $(x_2+x_1)/2$). In this case, a new CFD simulation is performed based on the input parameter defined by the new grid point, and the simulation output is added to the training data. The results from comparing neighboring grid points are cached to avoid performing the same comparison redundantly during the process. If the change in critical outputs does not exceed ε, then no new grid point is added between those two points and a CFD simulation is not performed. After completing this for each pair of neighboring grid points, the algorithm checks if any new grid points were added in the most recent iteration. If no new points were added, then the process ends since further iterations will not add any new grid points. If there were new grid points added in the most recent iteration, then the algorithm checks if a maximum resolution has been reached. The initial resolution is the difference between values of the initial neighboring grid points, e.g., $x_2 - x_1$. A defined maximum resolution can be useful to prevent excessive training data from being generated with neighboring inputs very close together. The process ends if the maximum resolution has been

reached. If the maximum resolution has not been reached, then the resolution increases by a factor of two (e.g., $(x_2 - x_1)/2$). After increasing the resolution, the set of grid points are reset with the newly added grid points. The process repeats by computing the change in critical outputs between neighboring grid points, now with the newly added grid points. The process ends once either no new grid points are added in an iteration or once the maximum resolution is reached.

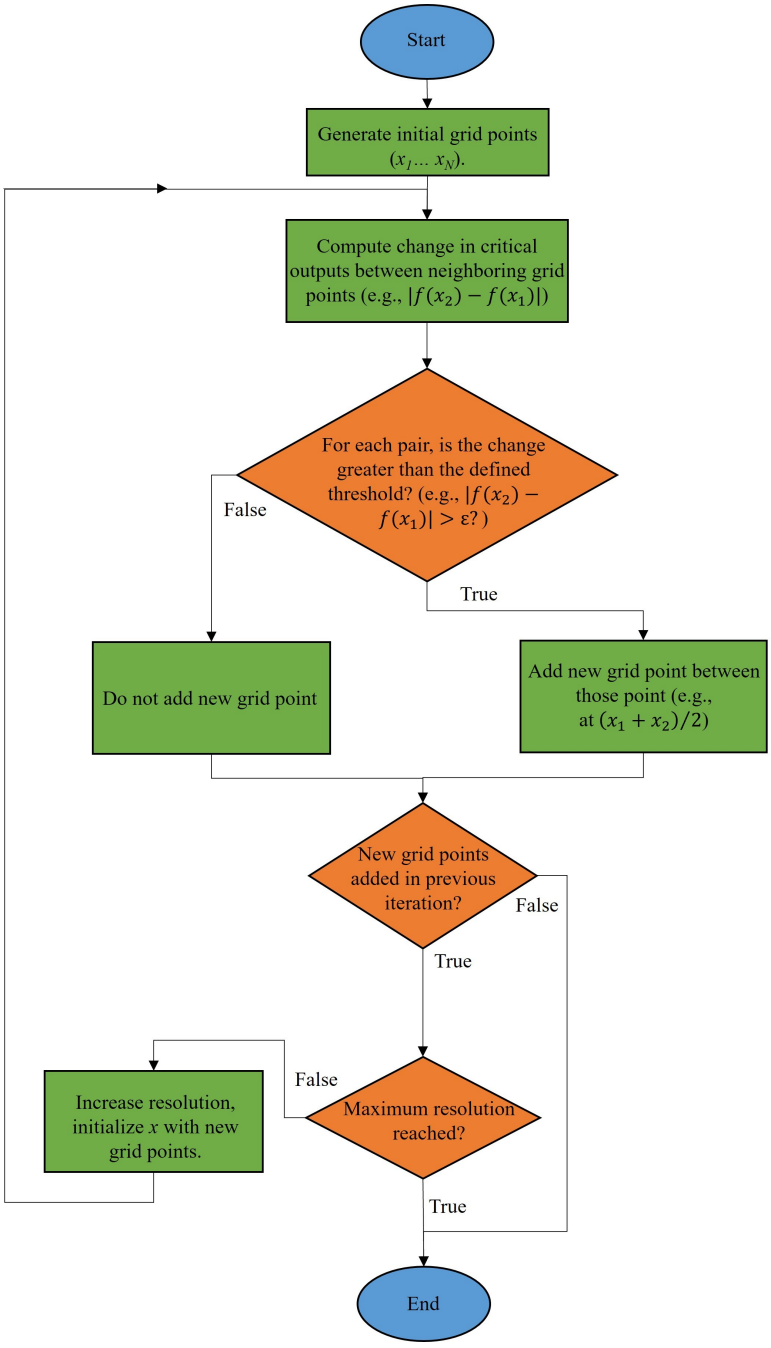


Figure 3: Flowchart of the feature-driven algorithm.

4. Description of Workflow

The description of the entire workflow including generation of training data, model training process, and model evaluation is shown in Figure 4 and described in this section.

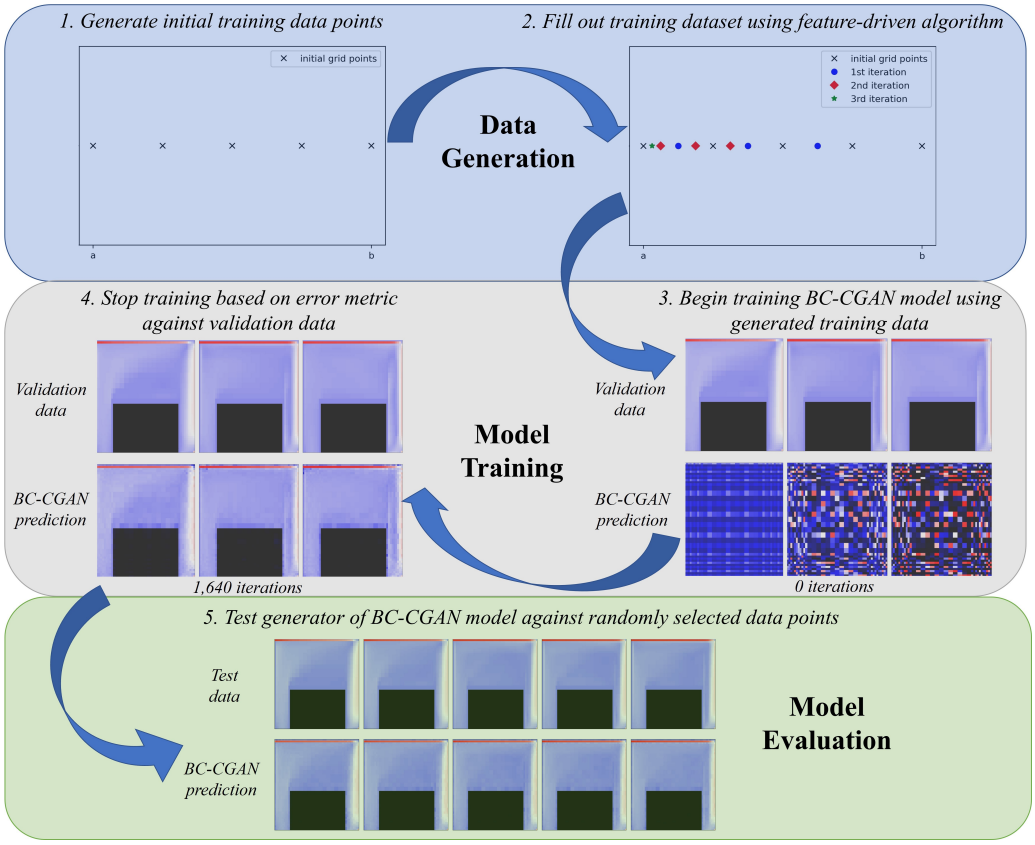


Figure 4: Description of entire workflow including generation of training data, model training, and model evaluation.

The workflow begins with the data generation process. Initial training

data points are first generated to provide baseline points for the feature-driven algorithm and to define a range for the training data. In this case, a few CFD simulations are performed to generate these initial points. Using too many initial points can potentially include redundant training data, so fewer initial points should be used to allow the algorithm to determine what data is necessary to be included. For the cases studied in this paper, we find that, for a given ε, the feature-driven algorithm only becomes sensitive to the initial data points when a large amount of initial data is included. This is because using a large amount of initial data may include redundant data points that the algorithm would not include. Using fewer initial data points often results in very similar training data sets, because the algorithm is designed to find the necessary data points within the input range, which are similar regardless of the initial data points. Thus, we use only about ten initial data points for the cases in this paper, which uniformly span the range of the studied input parameters.

Next, the feature-driven algorithm is used to select what input parameter values should be used for the additional CFD simulations to generate the rest of the training data. In this paper, a few training datasets are generated with different settings of the feature-driven algorithm for each case. Additionally,

311 uniformly distributed training datasets are generated for comparison against
312 the training datasets produced by the feature-driven algorithm. BC-CGAN
313 models are then trained using the different training datasets, where the
314 initial settings of each BC-CGAN model are identical. We then compare the
315 training speed and prediction performance for the BC-CGAN models trained
316 by the different datasets. Since the training process is inherently stochastic,
317 we train ten BC-CGAN models for each training dataset to understand their
318 training and prediction performance over several runs.

319 To train the BC-CGAN models, one unique label is assigned to each
320 training data image within its dataset. Additionally, a few reference data
321 points are generated using CFD simulations for validation of the BC-CGAN
322 models during training. Periodically during the training process, the BC-CGAN
323 model produces flow distribution outputs based on the validation data input
324 values. Its outputs are then compared against the validation data points
325 and a relative error metric is computed for each output. The error metric
326 is calculated based on error between the flow outputs (e.g., velocity and/or
327 temperature) at each point in the flow. If the error metric between the
328 BC-CGAN prediction and validation data is below a defined threshold for all
329 the validation data points, then the model is considered to be suficiently

trained and the training stops. Otherwise, training continues until this criteria is satisfied. In this paper, we save the trained BC-CGAN models at different error metric thresholds to compare the tradeoff between training speed and prediction performance for different error thresholds.

Finally, the trained BC-CGAN models are evaluated against test data points that were not selected from the training or validation data. The test data is selected by using a bin sampling technique to produce ten random input values that span the defined input range. CFD simulations are then performed to generate the test data based on the ten input values. The BC-CGAN models are evaluated by generating their flow distribution outputs based on the test input values and computing their relative error against the test data. The BC-CGAN models are trained to predict 2D airflow distribution in this paper, but can be extended for 3D airflow prediction in future research.

5. Isothermal Case: Lid-driven Cavity Flow

The first case studied in this paper is an isothermal lid-driven cavity flow. We begin with a description of the setup for this flow case. Next, we show the settings used for the BC-CGAN model and the generated training

datasets for the BC-CGAN models in this case. Finally, the training and evaluation results for the BC-CGAN models trained by the different datasets are detailed.

5.1. Case Description

The setup of the isothermal lid-driven cavity flow case is shown in Figure 5. This is a meaningful case because, despite having a simple configuration, the flow pattern changes significantly depending on the initial conditions and boundary conditions. Because of this, it is a benchmark flow case that is frequently studied in the literature using both physical experiments [32, 33] and numerical simulations [34, 35, 36, 37]. This flow is contained in a box of length L on all sides. The left, right, and bottom walls are stationary, while the lid moves at a constant velocity of U_0 to the right. The motion of the lid causes a circulation pattern of the flow within the box. The flow is modeled as 2D, steady, incompressible, and isothermal in this study.

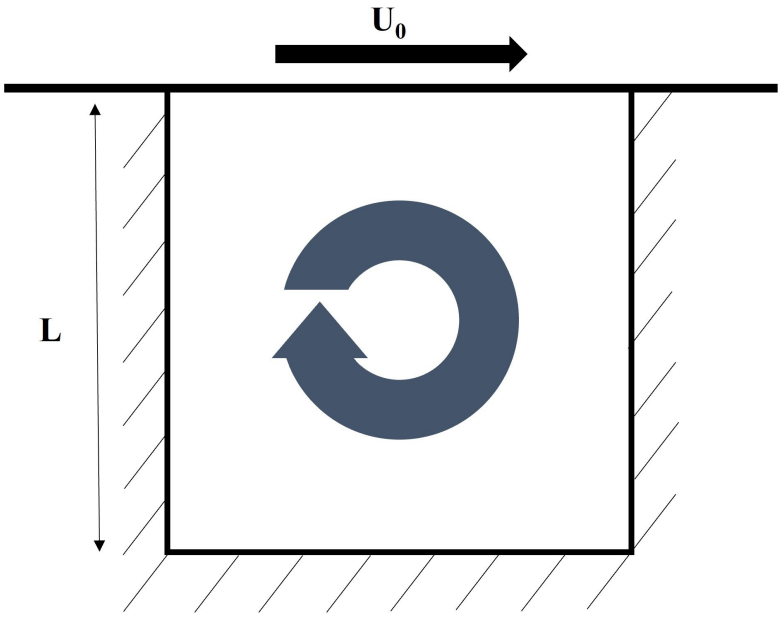


Figure 5: Diagram of the lid-driven cavity flow.

The lid-driven cavity flow is often defined by its Reynolds number (Re) in the literature [38, 39], which characterizes the ratio of inertial to viscous forces in the flow. Thus, we use Re as the input parameter for the flow in this study. For this case, Re is calculated as:

$$Re = U_0 L/v, \tag{4}$$

where U_0 is the constant velocity of the lid, L is the length of each side of the box, and v is the kinematic viscosity of the flow.

We define the range of Re values to be from 100 to 10,000. We hold U_0

and L constant while changing ν to vary the Reynolds number for the CFD simulations in this paper. All the other simulations settings and boundary conditions remain the same for each simulation, so the BC-CGAN models only receive Re as an input. The flow output we study for this case is the steady-state distribution of velocity magnitude. This means the training data produced by CFD simulations is comprised of velocity magnitude distribution data based on the input Re. The BC-CGAN models are then trained to output the velocity magnitude distribution based on the input Re. The CFD simulations used to generate the training, validation, and test data for this case are performed using the Fast Fluid Dynamics method [11, 12] on an AMD Radeon™ Pro WX 7100 GPU. The CFD simulations use a 64×64 non-uniform, structured grid, which is translated to a 36×36 uniform structured grid for the model training and evaluation to simplify the predictions while providing suficient resolution of the flow data. This prediction resolution was selected to balance the tradeoff of training time and resolution. The BC-CGAN model can be trained to predict a more resolved flow output, but this may increase the training time and require re-tuning of the model hyperparameters.

5.2. BC-CGAN Model Settings

The BC-CGAN model settings including neural network architectures and hyperparameters are described in this section. First, the architectures of the generator and discriminator are shown in Table 1 and Table 2, respectively. The generator uses a deconvolutional neural network [40] and the discriminator uses a convolutional neural network [41]. The generator receives an input defined by Re (which is the only varying parameter), as described in Section 2.2, and outputs a 36×36 image representing the velocity magnitude distribution. The discriminator receives a 36×36 image input as well as a label input corresponding to a Re value. The label input uses one-hot encoding to give the label of the image within the training dataset, with total number of training data points of n_{train}. The discriminator produces an output of zero or one, where zero corresponds to a classification of a “fake” image produced by the generator and one corresponds to a classification of a “real” image from the training dataset. The number of convolutional/deconvolutional layers and filter sizes for these layers impact the training performance of the generator and discriminator. For example, reducing the number of layers in these neural networks can speed up each training step, but can also negatively affect the training performance and result in more overall

iterations to successfully train the models. We selected these architectures for the generator and discriminator to have reasonable training speed and consistent convergence of the models.

Table 1: Summary of generator architecture.

Layer	Shape	Activation function
Input	200	N/A
Reshape	9 × 9 × 128	N/A
Deconvolution	18 × 18 × 128	ReLU
Deconvolution	36 × 36 × 64	ReLU
Deconvolution	36 × 36 × 32	ReLU
Deconvolution (output)	36 × 36 × 1	Sigmoid

Table 2: Summary of discriminator architecture.

Layer	Shape	Activation function
Label input	n_{train}	N/A
Image input	36 × 36 × 1	N/A
Reshape	36 × 36 × 2	N/A
Convolution	18 × 18 × 32	LeakyReLU
Convolution	9 × 9 × 64	LeakyReLU
Convolution	5 × 5 × 128	LeakyReLU
Convolution	5 × 5× 256	LeakyReLU
Flatten	6400	N/A
Output	1	Sigmoid

The hyperparameters for the BC-CGAN models were tuned carefully for this case. Certain hyperparameters, such as the learning rate, can significantly impact model training time and convergence or lead to overfitting [42]. For this case we used the Adam optimizer [43] with learning rate of 0.0002 and decay rate of 0.5. A batch size of 32 was used for training the models as well.

5.3. Training Datasets

Four training datasets are generated using the novel feature-driven algorithm and are compared against four uniformly generated datasets. A summary of all the training datasets is included in Table 3. The uniformly distributed datasets are defined by their uniform step (ΔRe), which describes the difference in Re for which training data is generated. These datasets include training data points from $Re = 100$ up to $Re = 10,000$ by step of ΔRe. Thus, a higher ΔRe results in fewer training data points that are further apart within the defined Re range, while a lower ΔRe results in more training data. The selected values of ΔRe are chosen to be factors of the input Re range, which is 9,900 in this case, as well as to include similar n_{train} values as the non-uniform datasets.

The non-uniform datasets are defined by their ε threshold, which is the threshold to decide whether a training data point is needed based on the change in outputs between neighboring data points, as described in Section 3. In this case, ε is dimensionless and is described by the change in velocity magnitude ($\Delta|U|$) at any of the critical output locations between neighboring Re data points, normalized by the velocity of the moving lid (U_0). A lower ε results in higher n_{train}, since it defines a smaller change in outputs to

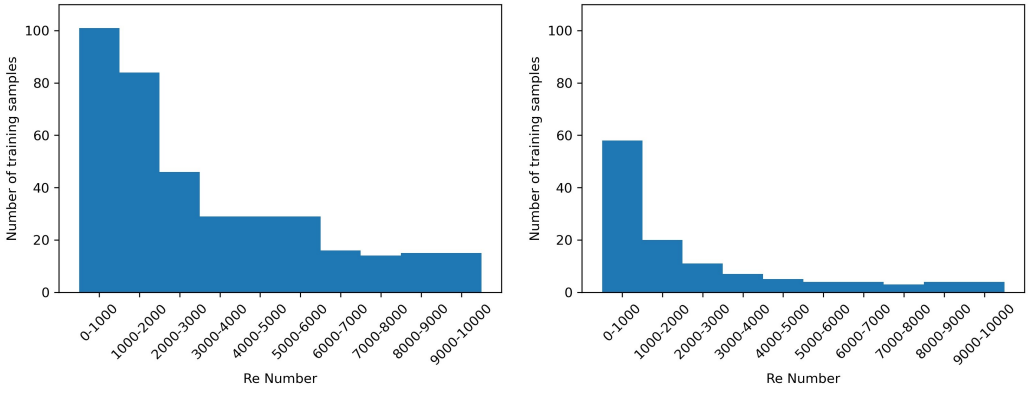
433 determine that additional training data points are needed. The ε values in
434 this case were chosen to provide datasets with a wide range of data points and
435 study the impact of training data size on the model training and evaluation.
436 The locations of the critical outputs used for the feature-driven algorithm in
437 this case are the center points of each cell when dividing the flow domain into
438 a 4×4 grid, resulting in 16 locations. These locations were chosen to include
439 a range of possible locations where the flow can change, but the critical
440 locations can be narrowed based on the studied flow in future research.

Table 3: Summary of training datasets.

Uniform/ Non-uniform	Uniform step (Δ Re)	ε threshold ($\Delta\|U\|/U_0$)	n_{train}
Uniform	25	N/A	397
Uniform	50	N/A	199
Uniform	100	N/A	100
Uniform	275	N/A	37
Non-uniform	N/A	0.001	378
Non-uniform	N/A	0.005	120
Non-uniform	N/A	0.01	67
Non-uniform	N/A	0.05	19

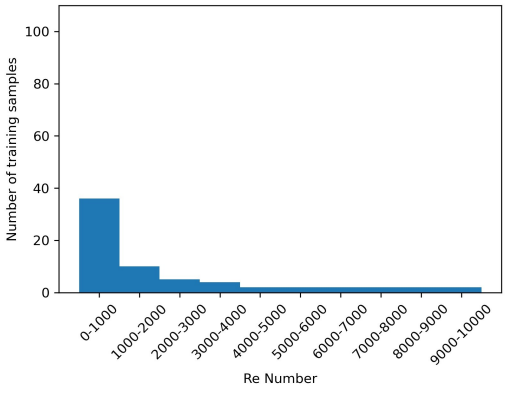
Histograms of the non-uniform training datasets are shown in Figure 6. The results show a clear trend: the lid-driven cavity flow changes more significantly at lower Re values, especially between 100 and 1,000. The $\varepsilon = 0.05$ dataset includes less than 5% of the training data points compared to the $\varepsilon = 0.001$ dataset, but still includes more of its training data between Re values of 100 and 1,000. This is unsurprising as the lid-driven cavity flow

447 transitions from laminar to turbulent in this region of Re. In particular,
448 the boundary layer near the lid and circulation pattern along the right wall
449 change more significantly in this region of Re.

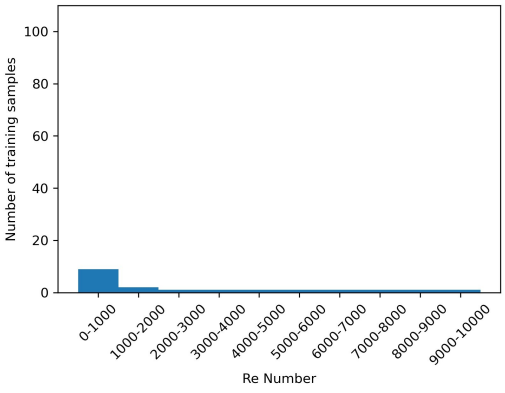


(a) ε = 0.001.

(b) ε = 0.005.



(c) ε = 0.01.



(d) ε = 0.05.

Figure 6: Histograms of training datasets generated using the feature-driven algorithm with different ε thresholds.

5.4. Training Results

We select five data points as validation data for this case using Re values of 100, 500, 1,000, 5,000, and 10,000. These points span the range of Re for this study, as well as include more points in the lower Re regime, since we found this is the regime where the flow changes more significantly. Sample validation results of the BC-CGAN model are shown in Figure 7. The top row shows the validation data from the CFD simulations, while the remaining rows show the BC-CGAN predictions at different error thresholds. For example, the BC-CGAN prediction results at the 40% error threshold are the saved validation results from when the BC-CGAN predictions were first below the error metric of 40% for all five validation data points. The error metric in this case is computed by weighting two error calculations: 55% root mean squared error (RMSE) and 45% maximum error. Rather than simply using RMSE, we include maximum error in this case to prevent points of large error in the predictions. The errors are also both normalized by U_0. The RMSE is calculated by considering the mean squared normalized error in velocity magnitudes at each location in the 36×36 2D flow domain. The maximum error is then the highest normalized error within the 36×36 domain. Initially during the training process, we saved the validation results

for different error metric thresholds of 5%, 10%, 15%, 25%, and 40%. Based on the qualitative results, we can see the BC-CGAN prediction at an error metric of 40% does not match well with the validation data. However, the BC-CGAN prediction seems to capture the trends of the validation data for error metrics of 15% and below. Consequently, for the remainder of this paper we save the trained BC-CGAN model and their validation results for error metric values of 5%, 10%, and 15%.

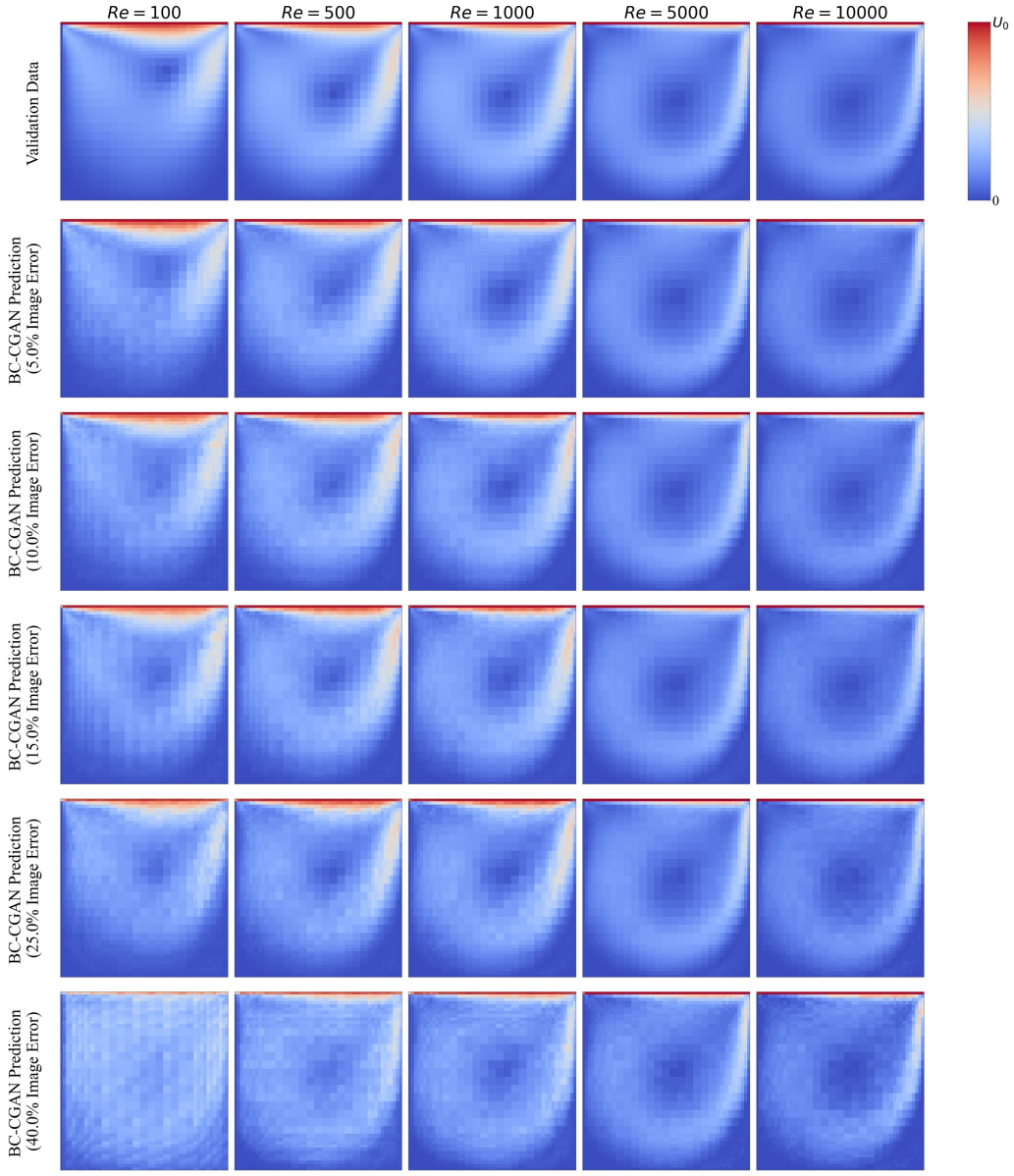
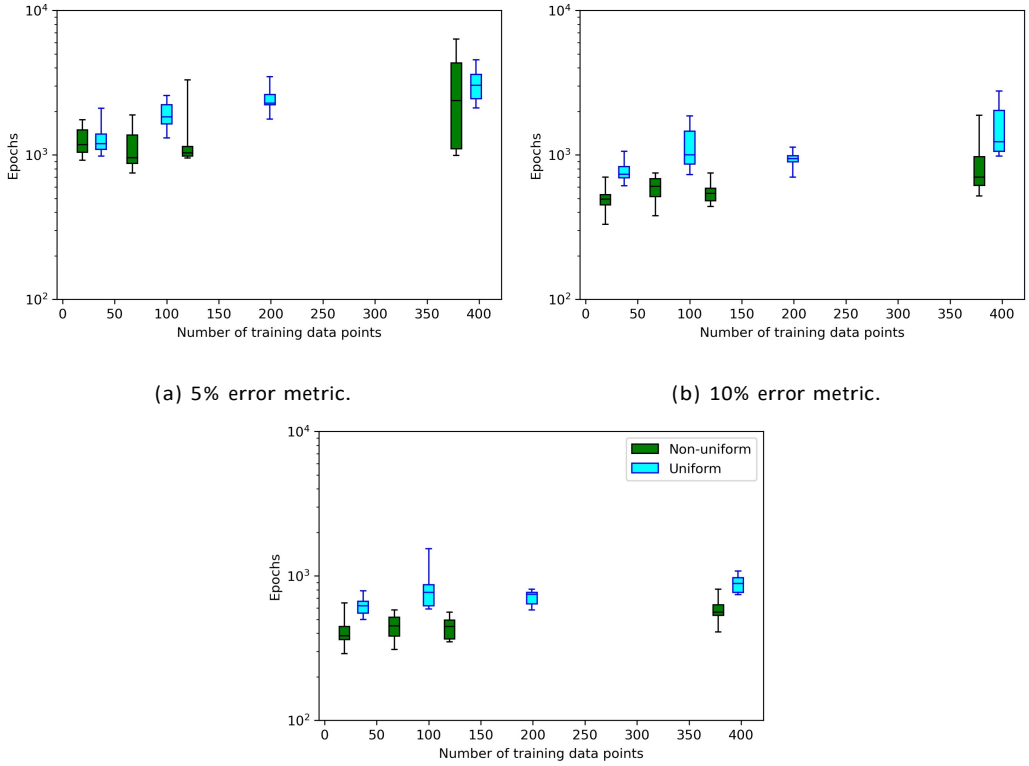


Figure 7: Sample validation results of the BC-CGAN prediction for different error metric thresholds.

476 The training results of the BC-CGAN models with different training

datasets and at different error metrics are shown in Figure 8. The training process is performed ten times for each training dataset since the training process is inherently stochastic. During each training process, the model is saved once the validation results satisfy error metric thresholds of 5%, 10%, and 15%. The number of training iterations or "epochs" required to achieve these error thresholds is also recorded and plotted (on a log-scale) in Figure 8. Each box plot shows the range of the number of epochs required to satisfy the different error metric thresholds over the ten runs for each training dataset. The training datasets are differentiated by the number of training data points included in each dataset, as well as whether they are non-uniformly or uniformly generated sets.



(a) 5% error metric.

(b) 10% error metric.

(c) 15% error metric.

Figure 8: Box plots of the number of epochs required to train the BC-CGAN model for different training datasets and error metrics.

First, we see less epochs are required to train the models when the error metric is higher. This is because less training is required for a looser error threshold compared to a stricter one. We also see that the non-uniform training datasets often require less epochs, especially for the error thresholds of 10% and 15%. This can occur because the non-uniform datasets are more

capable of capturing the non-linear trends between the input Re and output velocity distribution. For the 5% error metric, the non-uniform training dataset with 378 training data points required a wide range of epochs to converge. A wider range of epochs for model convergence often occurs for the lower error metrics, because there is more variability of when the model is able satisfy the convergence criteria for stricter error metrics. This seems more apparent for the non-uniform training datasets, especially the dataset with 378 data points. We found that the model often converged after around 1,000 epochs using this dataset, but when it took more epochs to converge it was usually because of higher error for the validation data point of $Re = 10,000$. The non-uniform datasets included less training data around this input value since the flow changes less significantly in this range. This seems to have a negative impact for a few training runs, especially when overfitting occurs based on more data included in Re regions where the flow outputs change more significantly. Finally, it seems that using less training data points often leads to fewer epochs required to train the models, especially for the uniform datasets. This can be a result of overfitting of the models when too much training data is included. For the non-uniform datasets, decreasing the amount of training data when there is less than about 150 data points

does not always reduce the number of epochs required to train the models. There may be an optimal amount of training data points for the non-uniform datasets around 50-150 data points.

5.5. Evaluation Results

Example evaluation results of the BC-CGAN model for the randomly selected test data points are shown in Figure 9. The BC-CGAN predictions match well with the test data based on a qualitative comparison, even at higher error metrics. The BC-CGAN predictions also capture the change in the lid-driven cavity flow pattern over the wide range of Re.

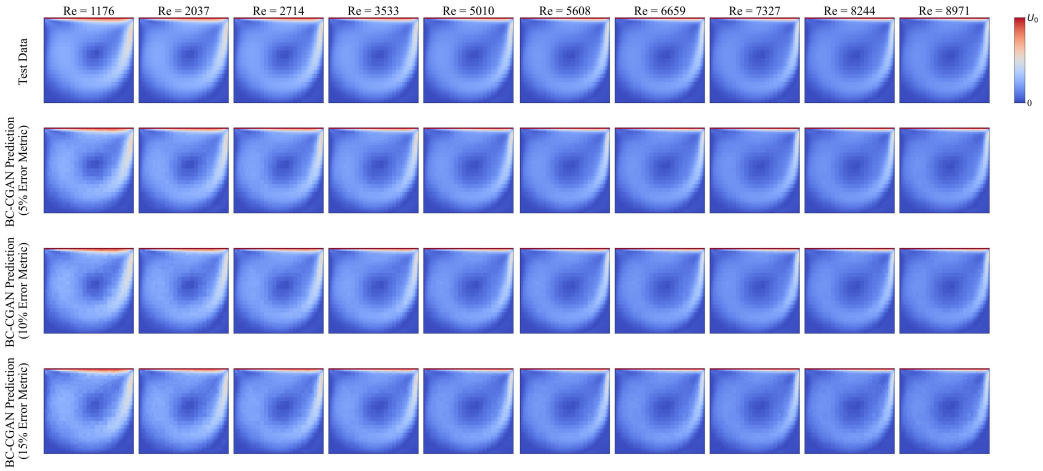
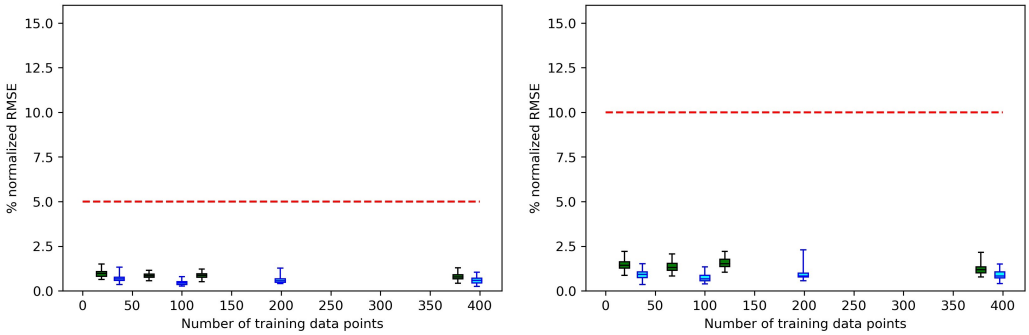


Figure 9: Example evaluation results for the trained BC-CGAN model with different error metrics.

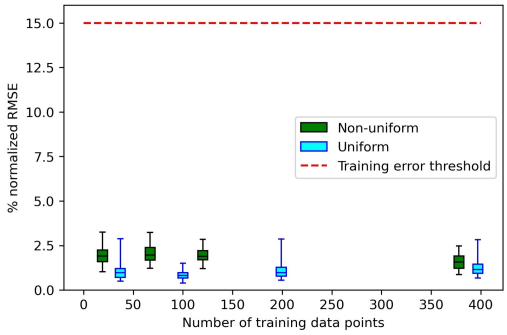
A comprehensive quantitative evaluation for the BC-CGAN models trained

with different datasets and at different error metric thresholds is shown in Figure 10. The box plots show the range of calculated RMSE (normalized by U_0) for the trained BC-CGAN models against the test data. Similar to Figure 8, the BC-CGAN models trained by the different datasets are differentiated by the number of training data points and whether they are non-uniform or uniform datasets. The defined training error metric threshold is also included in each plot for comparison.



(a) 5% error metric.

(b) 10% error metric.



(c) 15% error metric.

Figure 10: Box plots of the % normalized RMSE of the test predictions for the BC-CGAN models trained by diffrent datasets and error metrics.

First, the prediction error is higher when the training error metric threshold is higher. However, even for the error metric of 15%, the prediction error is almost always lower than 5%. This is because the error metric in the training process is only satisfied once all the validation data points are below the metric, so it is possible one validation data point had much higher error

compared to others. Furthermore, the error metric used to train the models included a combination of max error and RMSE to ensure predictions without points of very high error. Since the evaluation results are shown in terms of just RMSE for an easier understanding of the results, the prediction error is much lower than the training error metric. Finally, we see the BC-CGAN models trained by the uniform dataset often had lower prediction error compared to those trained by the non-uniform datasets. This may occur because the BC-CGAN models trained by the non-uniform datasets required less training epochs to satisfy the error metric criteria. Thus, while their predictions for the validation data may be similar to those of the models trained by the uniform datasets, they seem to perform slightly worse against the test data, perhaps due to less overall training. We also find that the test error for the models trained by the non-uniform datasets tended to be higher for the larger Re test values, which is the region of Re where the non-uniform datasets included less data. Conversely, the models trained by the uniform datasets often had higher error for the smaller Re test values, which is the region of Re where the flow changes more non-linearly. However, the prediction error is still well below the threshold for all the BC-CGAN models. Additionally, the difference in error for the models trained by the

different datasets is almost negligible, especially for the stricter training error metric thresholds. This is reasonable because the validation process to stop the training of the models is the same regardless of the dataset. So while the models trained by the different datasets may take different paths in their training processes, the final trained models should be similar in performance.

Once trained, the BC-CGAN models can generate predictions with an average speed of 7 ms per prediction. For this case, a CFD simulation took an average of 56 s, which means the trained BC-CGAN prediction was on average about 7,900 times faster than a CFD simulation. However, the time to generate data and train the BC-CGAN models must be considered as well. For all the training datasets considered in this case, it took an average of 11.4 minutes to train the models to satisfy the 5% error metric. We found that using less training data could often improve the training speed of the models while maintaining suficient accuracy, and the smallest tested training dataset required about 17.7 minutes to generate the training data. If only a few predictions are needed, then it would be faster to use CFD simulations because of the time required to generate data and train the BC-CGAN model. On the other hand, the time savings when using the BC-CGAN model increases as the number of required predictions increases. After accounting

for the time to generate data and train the BC-CGAN model, it becomes beneficial to use the BC-CGAN model over CFD simulations when more than 31 predictions are required for this case.

6. Non-isothermal Case: Mixed Convection Flow with Heated Box

The next case we study is a mixed convection flow with heated box. This case is chosen as a more complex flow compared to the lid-driven cavity flow, since it is non-isothermal, 3D, and includes an obstacle in the flow. It has also been used for indoor airflow simulation studies in the literature [44, 45]. We first describe the case setup, then summarize the BC-CGAN model settings and generated training datasets. Finally, the training and evaluation results are detailed.

6.1. Case Description

The setup of this case is shown in Figure 11. The flow is contained in a room with length of L in all dimensions. A heated box is in the center of the room, with dimensions of $L/2$ in all dimensions. The box generates heat with a uniform flux of Q_{box}. This is meant to represent an internal heat load within a room, for example occupants. Cold air is supplied to the room through the inlet along the top of the left wall with a velocity of U_{in} and

temperature of T_{in}. An outlet is located along the bottom of the right wall. The ceiling, floor, and remaining walls have temperatures of T_{cei}, T_{flo}, and T_{oth}, respectively. The flow is modeled as steady and incompressible in this study.

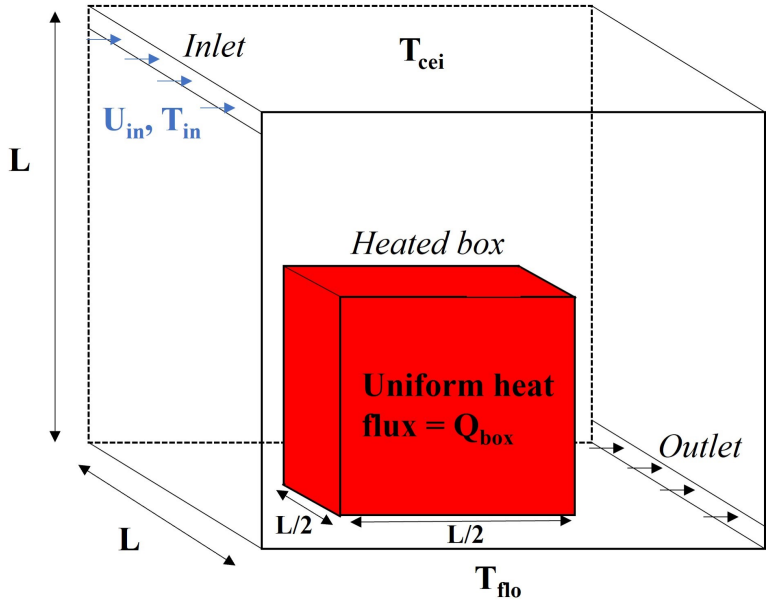


Figure 11: Diagram of the mixed convection flow with heated box case.

We select Q_{box} to be the input parameter for this case and vary this value from 0 W/m^2 to 50 W/m^2. Furthermore, we add noise to the boundary conditions of U_{in}, T_{in}, T_{cei}, T_{flo}, and T_{oth} for all the CFD simulations used to generate training, validation, and test data. Noise is added to U_{in} by randomly increasing or decreasing this value by up to ± 5% of its default

value of 1.36 m/s for each simulation. Similarly, the temperature boundary conditions are randomly increased or decreased by up to ± 0.5 °C of their default values. The default values for T_{in}, T_{cei}, T_{flo}, and T_{oth} are 22.2 °C, 25.8 °C, 26.9 °C. and 27.4 °C. This added noise can represent uncertainty in experimental conditions, for example. It also adds a potential challenge for the BC-CGAN model, since the model is only given the value of Q_{box} as an input. Since the BC-CGAN models assume the other boundary conditions are unchanged, their values (including the added noise) are not given as an input. Similarly, since the geometry (including locations of the box, inlet, and outlet), is unchanged, these are not given as an input. Future research can extend the BC-CGAN model to consider multiple varying boundary conditions for this case. The CFD simulations use a 44×44×44 non-uniform grid, similar to in [44]. For this case, the BC-CGAN model outputs the 2D airflow distribution at the mid-plane of the flow. It outputs both the velocity and temperature distribution, since it is a non-isothermal flow. Thus, the CFD data is translated to provide a 36 × 36 uniform grid of velocity and temperature data at the mid-plane of the flow. The model also must generate the box within the surrounding flow. This can be useful for when AI models are needed to detect obstacles in the flow.

6.2. BC-CGAN Model Settings

The architectures for the generator and discriminator are shown in Table 4 and Table 5, respectively. While they are mostly similar to the architectures used in the previous case described in Section 5.2, there are a few key differences. First, the generator produces a 72×36 output and the discriminator receives 72 × 36 image inputs. This is because the flow outputs in this study are the 36 × 36 distribution of both velocity and temperature. The other key difference is the output layer of the generator uses the Tanh activation function rather than Sigmoid. The activation function in the output layer outputs the value of velocity or temperature in the flow based on the information received at that node within the layer. In the training data for this case, a value of -1 is assigned to the points where the box is located to differentiate it from the fluid flow (represented by normalized velocity/temperature values from 0 to 1). Thus, the Tanh activation function is chosen for the generator in this case, because it can output values from -1 to 1 while the Sigmoid activation function only outputs values from 0 to 1. For this case, we used the Adam optimizer with learning rate of 0.0001 and decay rate of 0.25, as well as a batch size of 32 for training the models.

Table 4: Summary of generator architecture.

Layer	Shape	Activation function
Input	200	N/A
Reshape	18 × 9 × 128	N/A
Deconvolution	36 × 18 × 128	ReLU
Deconvolution	72 × 36 × 64	ReLU
Deconvolution	72 × 36 × 32	ReLU
Deconvolution (output)	72 × 36 × 1	Tanh

Table 5: Summary of discriminator architecture.

Layer	Shape	Activation function
Label input	n_{train}	N/A
Image input	72 × 36 × 1	N/A
Reshape	72 × 36 × 2	N/A
Convolution	36 × 18 × 32	LeakyReLU
Convolution	18 × 9 × 64	LeakyReLU
Convolution	9 × 5 × 128	LeakyReLU
Convolution	9 × 5× 256	LeakyReLU
Flatten	11520	N/A
Output	1	Sigmoid

6.3. Training Datasets

Two training datasets are generated using the feature-driven algorithm and are compared against three uniformly generated datasets, summarized in Table 6. The thresholds for the non-uniform datasets are dimensionless values of 0.05 and 0.10 and are chosen to provide two different sizes for the non-uniform datasets. These thresholds correspond to a relative change in

either velocity magnitude or temperature. If this threshold is exceeded for significant changes in either velocity or temperature at any of the critical output locations, a new point is added. The change in velocity magnitude is normalized by the maximum velocity for the case, which is the inlet velocity (U_{in}). The change in temperature is normalized by the difference between the maximum and minimum temperatures for the entire case ($T_{max} - T_{min}$), where T_{max} comes from the highest Q_{box} scenario and T_{min} is the cold inlet air temperature. The locations of the critical outputs are the center points of the cells when dividing the flow into a 5×5 grid, resulting in 25 locations.

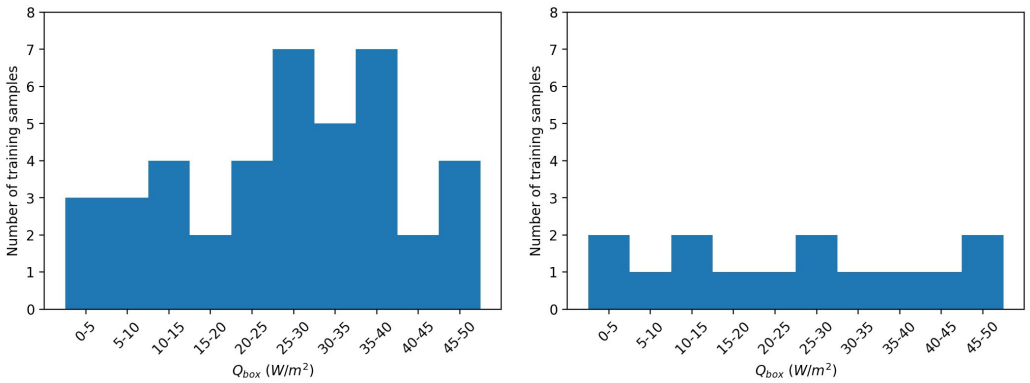
Uniform steps of 0.5, 2, and 25 W/m^2 are chosen to produce three uniform datasets with very different amounts of training data. In the previous case in Section 5, we found that using less training data often reduced the number of epochs required to train the models. Thus, we include the training dataset with uniform step of 25 W/m^2 to observe the impact of using very few training data points on the training and prediction performance of the BC-CGAN models. Furthermore, the training data points in this dataset are identical to the validation data points for this case. This was done intentionally to observe the impact on the performance of the BC-CGAN models when the training data is more biased towards the validation data.

Table 6: Summary of training datasets.

Uniform/ Non-uniform	Uniform step (Δ W/m^2)	ε threshold	Total number of training data points
Uniform	0.5	N/A	101
Uniform	2	N/A	26
Uniform	25	N/A	3
Non-uniform	N/A	0.05	40
Non-uniform	N/A	0.10	14

Histograms of the two non-uniform training datasets generated by the feature-driven algorithm are shown in Figure 12. Unlike the lid-driven cavity case, there is not a clear trend in the non-linearity between the inputs and outputs. It seems that there are more changes in the flow between Q_{box} values of 25-40, as shown in Figure 12a. In this region, the flow pattern in the room transitions from being dominated by the cold supply airflow to being significantly impacted by the thermal plume from the heated box. We see more noticeable changes in the boundary layers around the box because of this effect in this region. However, these changes do not seem to be very

large, as shown by the more uniform training dataset in Figure 12b with the looser ε threshold.



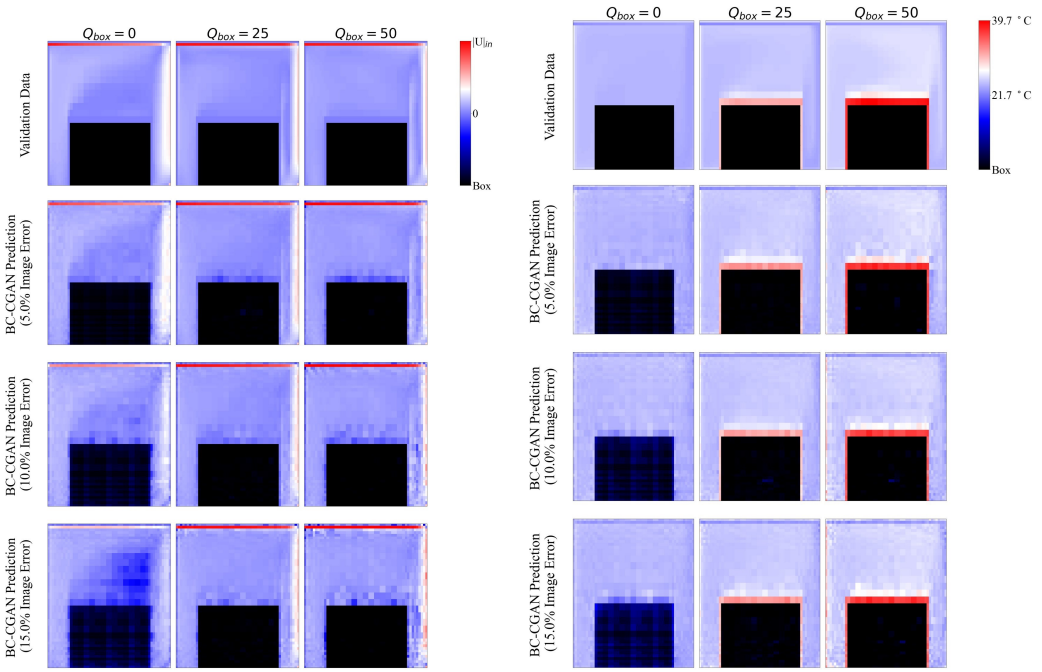
(a) ε = 0.05. (b) ε = 0.10.

Figure 12: Histograms of training datasets generated using the feature-driven algorithm with different ε thresholds.

6.4. Training Results

Sample validation results of the BC-CGAN model are shown in Figure 13. The validation data points selected for this case are Q_{box} values of 0, 25, and 50 W/m^2. The error metric is computed only using RMSE in this case, instead of a combination of RMSE and max error as in the previous case. This change was made because it was dificult for the model training to converge using a stricter error metric for this more complex flow case. Additionally, the error metric combines the error for both velocity and temperature by

scaling the ranges of both these values from 0 to 1. The RMSE is then calculated by considering the mean squared error using the scaled errors for both velocity and temperature at each location in the 36 × 36 2D flow domain. Because of the change in error metric calculation for this case, the BC-CGAN predictions are qualitatively more different than the reference CFD simulations, especially for error metric thresholds of 10% and 15%. The change in velocity magnitude is more subtle for this case, since the input parameter is a heat flux rather than Re. The most noticeable difference in velocity magnitude is between the right side of the box and the right wall. The boundary layer along the right wall thins as Q_{box} increases. There is also a more noticeable boundary layer along the right side of the box as Q_{box} increases, because the heat of the box causes the surrounding air to heat up and rise. The velocity magnitude just above the box also slightly increases with Q_{box} because of this buoyant flow. The change in temperature distribution for the different Q_{box} values is more apparent, since the increase in Q_{box} creates a significant thermal boundary layer surrounding the box.



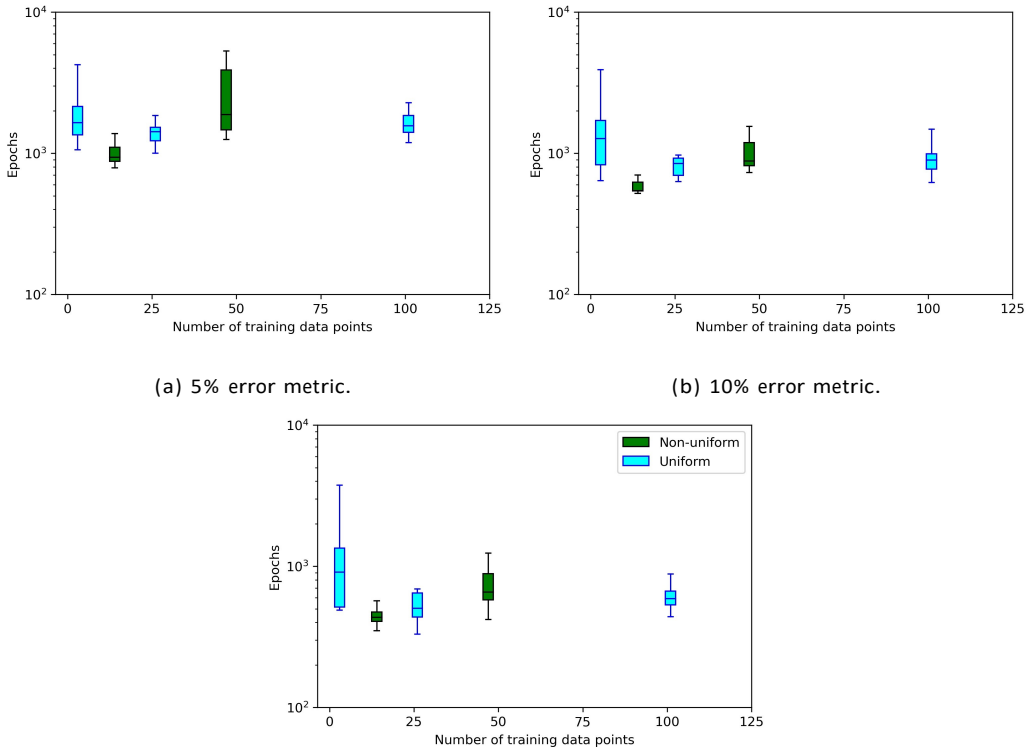
(a) Velocity prediction.

(b) Temperature prediction.

Figure 13: Validation results for velocity and temperature prediction with different error metrics.

The quantitative training results of the BC-CGAN model with different training datasets is shown in Figure 14. Similar to before, the higher error metric results in less epochs to train the models. Reducing the amount of training data seems to decrease the number of required epochs to train the models, until the uniform training dataset with only three data points. It seems that the training epochs can increase when the amount of training data is drastically reduced, especially for the looser error metric thresholds. The

non-uniform dataset with the least amount of training data typically requires the least number of epochs to train the models, while the non-uniform dataset with the most amount of training data often requires the most epochs to train the models. This discrepancy is likely because the outputs did not vary as non-linearly with the inputs for this case compared to the previous lid-driven cavity case.



(a) 5% error metric.

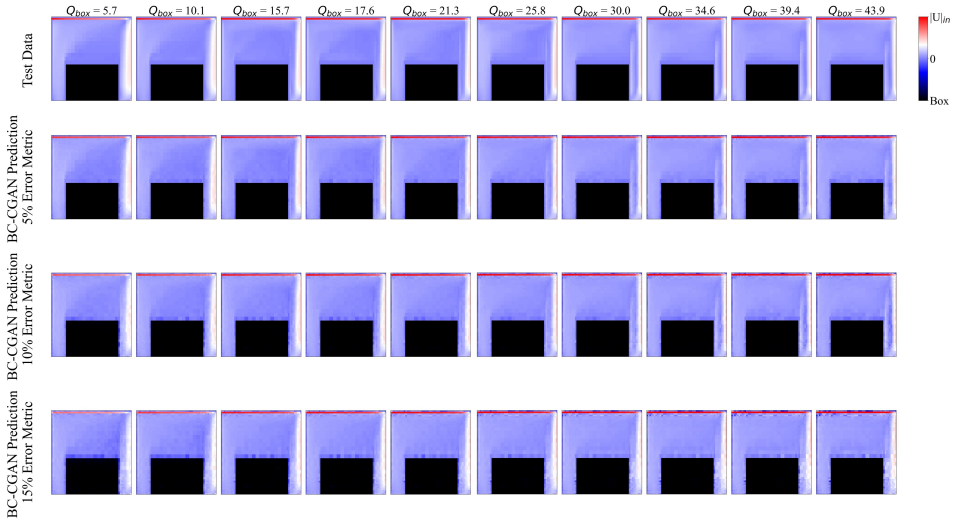
(b) 10% error metric.

(c) 15% error metric.

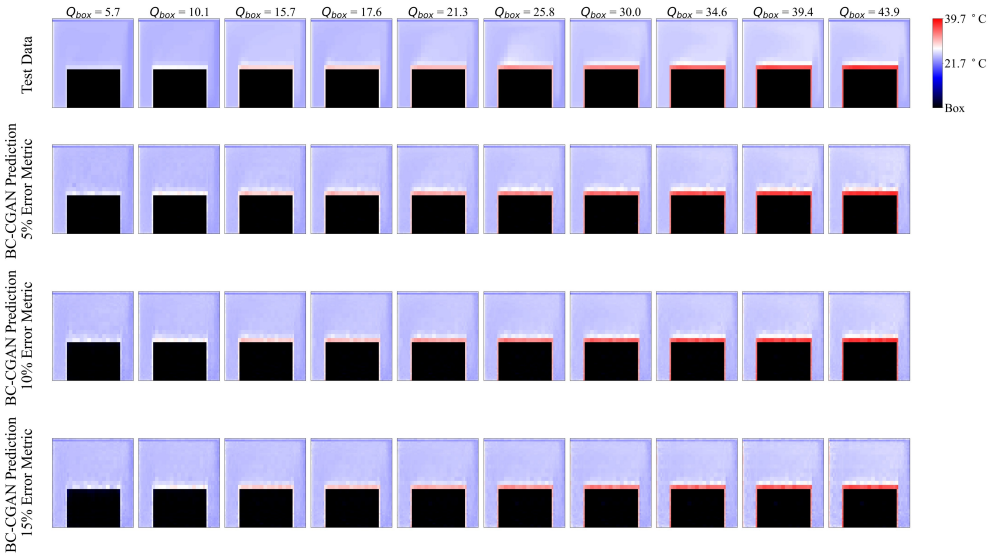
Figure 14: Box plots of the number of epochs required to train the BC-CGAN model for different training datasets and error metrics.

6.5. Evaluation Results

Example evaluation results of the BC-CGAN model for the randomly selected test data points are shown in Figure 15. The BC-CGAN predictions capture the main trends and features of the velocity and temperature distributions at different Q_{box} values. However, the qualitative differences between the BC-CGAN predictions and reference test data are more apparent for this case compared to the previous. This is because the error metric is calculated entirely based on RMSE for this case rather than a combination of RMSE and max error, so locations with higher error may persist in these predictions.



(a) Velocity prediction.

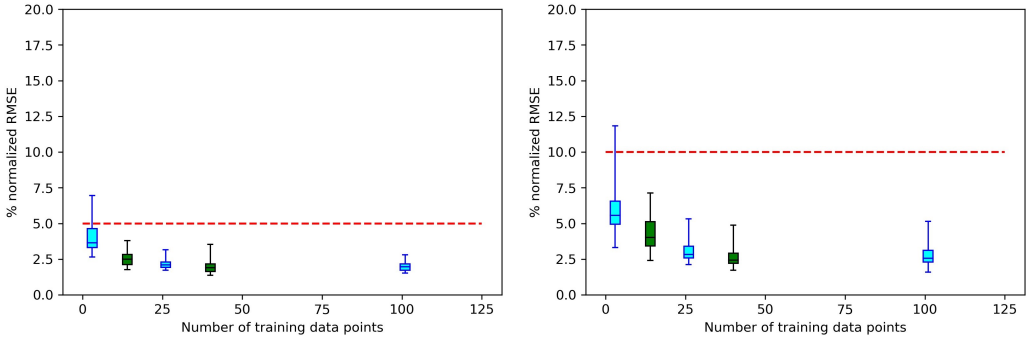


(b) Temperature prediction.

Figure 15: Evaluation results for velocity and temperature prediction for the trained BC-CGAN model with different error metric thresholds.

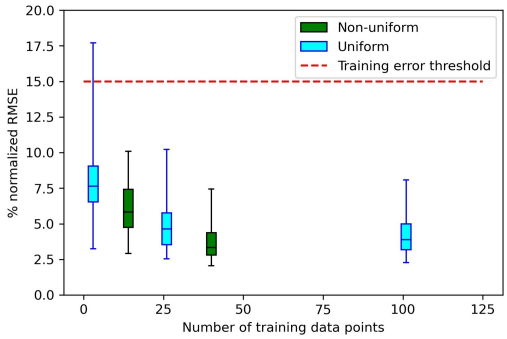
718 The comprehensive quantitative evaluation results for the BC-CGAN

models trained with different datasets is shown in Figure 16. The results show the error threshold is satisfied by the predictions from all the datasets except the smallest training dataset with only three points. This shows the consequences of drastically reducing the amount of training data, since it is not able to capture the trends across the range of Q_{box} as well. Additionally, the non-uniform training dataset with more training data points often performs the best in terms of its predictions. This was also the training dataset that typically required the most epochs to train the models. The results from this case as well as the previous case appear to show a tradeoff between training epochs and prediction performance. While some training datasets may take more time to satisfy the validation criteria, they can perform better on a wider range of test data, perhaps because of the additional training. The difference in error is more apparent for the looser error metric thresholds compared to the 5% error threshold. The change in error among the models trained by the different datasets is almost negligible for this strict error threshold, except for the dataset with only three data points. Tuning either the uniform step of the input parameter or ε for the training datasets may help balance training time and evaluation performance.



(a) 5% error metric.

(b) 10% error metric.



(c) 15% error metric.

Figure 16: Box plots of the % normalized RMSE of the test predictions for the BC-CGAN models trained by different datasets and error metrics.

For this case, the BC-CGAN models can generate predictions with an average speed of 11 ms per prediction, while a CFD simulation took an average of 13.7 minutes. Both the BC-CGAN prediction and CFD simulation times are higher for this case because of the additional complexity compared to the lid-driven cavity flow, with the increase in CFD time being more

significant. The trained BC-CGAN prediction was on average about 75,680 times faster than a CFD simulation for this case. It took an average of 18 minutes to train the models to satisfy the 5% error metric, when considering all the training datasets used in this case except for the one with only three data points, which was found to have poor evaluation performance. When excluding that dataset, it took a minimum of 3.2 hr to generate training data. After accounting for the time to generate data and train the BC-CGAN models, it becomes beneficial to use the BC-CGAN approach over CFD simulations when more than 15 predictions are required for this case. The results for this case show the significant potential of this model for accelerating flow prediction with more complex cases.

7. Conclusion

In this paper, we proposed a new BC-CGAN model for fast prediction of indoor airflow distribution. We extended the original CGAN model to make predictions based on a continuous input parameter, such as a boundary condition, rather than a discrete parameter, like a specific design. We also designed a novel feature-driven algorithm for generating training data for AI models. The algorithm includes training data points based on significant

changes between the flow outputs and inputs, with the goal of minimizing the amount of generated training data while ensuring training quality. The new BC-CGAN model and feature-driven algorithm are evaluated for two benchmark flow cases: an isothermal lid-driven cavity flow and non-isothermal mixed convection flow with a heated box.

The results show the trained model can predict velocity and temperature distribution with less than 5% normalized RMSE and up to 75,000 times faster than reference CFD simulations. For the lid-driven cavity case, the trained models were able to make predictions for the test data with much less than 5% normalized RMSE, even for the higher error metric threshold cases. This is because we could use a stricter error metric that combined RMSE and max error during the training process, which allowed for the predictions to produce images without points of very high error. For the more complex mixed convection flow with heated box case, this type of training error metric could not be used, since it was dificult for the models to converge during training with this method. Despite this, the trained BC-CGAN models for this case make predictions below their error threshold for the test data, except for the models trained by the dataset with only three data points. While reducing the amount of training data often reduces the training time

779 in this paper, drastically reducing the amount of training data caused the
780 BC-CGAN models to perform poorly against the test data.

781 Use of the feature-driven algorithm often reduces the epochs required to
782 train the BC-CGAN models for the lid-driven cavity flow case, since it was
783 able to capture the non-linear trend between the change in flow outputs and
784 inputs. However, the feature-driven algorithm did not always produce this
785 same effect for the mixed convection flow with heated box case, perhaps
786 because there was not a clear non-linear trend between the flow outputs
787 and inputs. For both cases, there is an apparent tradeoff between training
788 time and test performance. The BC-CGAN models that took longer to train
789 often performed better on the test data compared to the BC-CGAN models
790 that were trained quicker. For the lid-driven cavity case, the increase in test
791 prediction error was very small for the models that were trained quicker,
792 particularly since all the models were very accurate because of the use of
793 max error in the training process. The change in error for the predictions on
794 the test data in the mixed convection flow with heated box case was more
795 significant when the error metric threshold was higher. A strict error metric
796 in this case resulted in small changes in test error among the models trained
797 by the different datasets, except for the dataset with only three data points,

798 which performed poorly for this case.

799 Future studies can be conducted based on the work in this paper. First,
800 more practical applications can be studied, for example data center airflow
801 scenarios. One input parameter was used for each of the studies in this paper,
802 but the BC-CGAN models and feature-driven algorithm can be evaluated for
803 applications with multiple input parameters. This is important for expanding
804 the BC-CGAN models to more applications, for example optimizing both
805 supply airflow rate and temperature considering the indoor environment.
806 Additionally, the models in this paper were trained to predict 2D airflow
807 distributions, but a 3D prediction may be necessary for certain applications.
808 The impacts of the additional complexity when considering multiple inputs
809 and 3D outputs on the model training and evaluation needs to be studied in
810 future research. Incremental training, by either expanding the training range
811 to new data or using "online" training when deploying the models [46, 47]
812 can be performed to improve the models over time with new data. Finally,
813 the trained BC-CGAN models can be used for a long-term evaluation or
814 optimization study that requires many realizations to show the computational
815 benefits of using this model over other numerical methods in these scenarios.
816 They can also provide real-time or faster predictions of airflow distribution,

which can be useful for emergency management scenarios.

Acknowledgements

This research was supported in part by the U.S. Defense Threat Reduction Agency and performed under U.S. Department of Energy Contract No. DE-AC02-05CH11231. This research was also partially supported by the National Science Foundation under Awards No. IIS-1802017, CBET-2217410, CNS-2025377, and CNS-2241361.

References

[1] Y. Li, P. V. Nielsen, CFD and ventilation research, Indoor Air 21 (6) (2011) 442–453.

[2] X. Han, W. Tian, J. VanGilder, W. Zuo, C. Faulkner, An open source fast fluid dynamics model for data center thermal management, Energy and Buildings 230 (2021) 110599.

[3] X. Zhu, T. Shi, X. Jin, Z. Du, Multi-sensor information fusion based control for VAV systems using thermal comfort constraints, Building Simulation 14 (4) (2021) 1047–1062.

[4] X. Kong, Y. Chang, N. Li, H. Li, W. Li, Comparison study of thermal comfort and energy saving under eight different ventilation modes for space heating, Building Simulation 15 (7) (2022) 1323–1337.

[5] K. Gangisetti, D. E. Claridge, J. Srebric, M. T. Paulus, Influence of reduced VAV flow settings on indoor thermal comfort in an office space, Building Simulation 9 (1) (2016) 101–111.

[6] J. E. Castellini Jr, C. A. Faulkner, W. Zuo, D. M. Lorenzetti, M. D. Sohn, Assessing the use of portable air cleaners for reducing exposure to airborne diseases in a conference room with thermal stratification, Building and Environment 207 (2022) 108441.

[7] F. Mohamadi, A. Fazeli, A review on applications of CFD modeling in COVID-19 pandemic, Archives of Computational Methods in Engineering (2022) 1–20.

[8] B. Jayaraman, E. U. Finlayson, M. D. Sohn, T. L. Thatcher, P. N. Price, E. E. Wood, R. G. Sextro, A. J. Gadgil, Tracer gas transport under mixed convection conditions in an experimental atrium: Comparison between experiments and CFD predictions, Atmospheric Environment 40 (27) (2006) 5236–5250.

851 [9] S.-J. Cao, Challenges of using CFD simulation for the design and online
852 control of ventilation systems, Indoor and Built Environment 28 (1)
853 (2019) 3–6.

854 [10] L. Wang, Q. Chen, Applications of a coupled multizone-CFD model to
855 calculate airflow and contaminant dispersion in built environments for
856 emergency management, HVAC&R Research 14 (6) (2008) 925–939.

857 [11] W. Zuo, Q. Chen, Real-time or faster-than-real-time simulation of
858 airflow in buildings, Indoor Air 19 (1) (2009) 33.

859 [12] W. Zuo, Q. Chen, Fast and informative flow simulations in a building by
860 using fast fluid dynamics model on graphics processing unit, Building
861 and Environment 45 (3) (2010) 747–757.

862 [13] S. B. Pope, Computationally eficient implementation of combustion
863 chemistry using in situ adaptive tabulation, Combustion Theory and
864 Modelling 1 (1) (1997) 41–63.

865 [14] W. Tian, T. A. Sevilla, D. Li, W. Zuo, M. Wetter, Fast and self-learning
866 indoor airflow simulation based on in situ adaptive tabulation, Journal
867 of Building Performance Simulation 11 (1) (2018) 99–112.

[15] Q. Zhou, R. Ooka, Comparison of different deep neural network architectures for isothermal indoor airflow prediction, Building Simulation 13 (6) (2020) 1409–1423.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems 27 (2014).

[17] R. Barth, J. Hemming, E. J. Van Henten, Optimising realism of synthetic images using cycle generative adversarial networks for improved part segmentation, Computers and Electronics in Agriculture 173 (2020) 105378.

[18] T. Iqbal, H. Ali, Generative adversarial network for medical images (MI-GAN), Journal of Medical Systems 42 (11) (2018) 1–11.

[19] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784 (2014).

[20] G. Antipov, M. Baccouche, J.-L. Dugelay, Face aging with conditional generative adversarial networks, in: 2017 IEEE international conference on image processing (ICIP), IEEE, 2017, pp. 2089–2093.

[21] Y. Ye, M. Strong, Y. Lou, C. A. Faulkner, W. Zuo, S. Upadhyaya, Evaluating performance of different generative adversarial networks for large-scale building power demand prediction, Energy and Buildings 269 (2022) 112247.

[22] Q. Zhang, A. Ferdowsi, W. Saad, M. Bennis, Distributed conditional generative adversarial networks (GANs) for data-driven millimeter wave communications in UAV networks, IEEE Transactions on Wireless Communications 21 (3) (2021) 1438–1452.

[23] D. Chen, X. Gao, C. Xu, S. Chen, J. Fang, Z. Wang, Z. Wang, FlowGAN: a conditional generative adversarial network for flow prediction in various conditions, in: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2020, pp. 315–322.

[24] Y. Wang, W. Wang, G. Tao, H. Li, Y. Zheng, J. Cui, Optimization of the semi-sphere vortex generator for film cooling using generative adversarial network, International Journal of Heat and Mass Transfer 183 (2022) 122026.

[25] L. Gonog, Y. Zhou, A review: generative adversarial networks, in:

2019 14th IEEE conference on industrial electronics and applications (ICIEA), IEEE, 2019, pp. 505–510.

[26] L. Sun, J. Chen, Y. Xu, M. Gong, K. Yu, K. Batmanghelich, Hierarchical amortized gan for 3D high resolution medical image synthesis, IEEE journal of biomedical and health informatics 26 (8) (2022) 3966–3975.

[27] M. D. Cirillo, D. Abramian, A. Eklund, Vox2Vox: 3D-GAN for brain tumour segmentation, in: International MICCAI Brainlesion Workshop, Springer, 2021, pp. 274–284.

[28] X. Zhao, F. Ma, D. Güera, Z. Ren, A. G. Schwing, A. Colburn, Generative multiplane images: Making a 2d gan 3d-aware, in: European Conference on Computer Vision, Springer, 2022, pp. 18–35.

[29] M. El-Kaddoury, A. Mahmoudi, M. M. Himmi, Deep generative models for image generation: A practical comparison between variational autoencoders and generative adversarial networks, in: International Conference on Mobile, Secure, and Programmable Networking, Springer, 2019, pp. 1–8.

[30] S. Mokhtar, A. Sojka, C. C. Davila, Conditional generative adversarial networks for pedestrian wind flow approximation, in: Proceedings of

the 11th Annual Symposium on Simulation for Architecture and Urban Design, 2020, pp. 1–8.

[31] Y. Zheng, Y. Zhang, Z. Zheng, Continuous Conditional Generative Adversarial Networks (cGAN) with Generator Regularization, arXiv preprint arXiv:2103.14884 (2021).

[32] C. Blohm, H. C. Kuhlmann, The two-sided lid-driven cavity: experiments on stationary and time-dependent flows, Journal of Fluid Mechanics 450 (2002) 67–95.

[33] H. Kuhlmann, M. Wanschura, H. Rath, Flow in two-sided lid-driven cavities: non-uniqueness, instabilities, and cellular structures, Journal of Fluid Mechanics 336 (1997) 267–299.

[34] S. Albensoeder, H. C. Kuhlmann, Accurate three-dimensional lid-driven cavity flow, Journal of Computational Physics 206 (2) (2005) 536–558.

[35] O. R. Burggraf, Analytical and numerical studies of the structure of steady separated flows, Journal of Fluid Mechanics 24 (1) (1966) 113–151.

[36] U. Ghia, K. N. Ghia, C. Shin, High-Re solutions for incompressible flow

using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics 48 (3) (1982) 387–411.

[37] M. Khan, N. Delbosc, C. J. Noakes, J. Summers, Real-time flow simulation of indoor environments using lattice Boltzmann method, Building Simulation 8 (4) (2015) 405–414.

[38] T. Chiang, W. Sheu, R. R. Hwang, Effect of Reynolds number on the eddy structure in a lid-driven cavity, International Journal for Numerical Methods in Fluids 26 (5) (1998) 557–579.

[39] N. Ramanan, G. M. Homsy, Linear stability of lid-driven cavity flow, Physics of Fluids 6 (8) (1994) 2690–2701.

[40] M. D. Zeiler, D. Krishnan, G. W. Taylor, R. Fergus, Deconvolutional networks, in: 2010 IEEE Computer Society Conference on computer vision and pattern recognition, IEEE, 2010, pp. 2528–2535.

[41] K. O'Shea, R. Nash, An introduction to convolutional neural networks, arXiv preprint arXiv:1511.08458 (2015).

[42] L. N. Smith, A disciplined approach to neural network

hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay, arXiv preprint arXiv:1803.09820 (2018).

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[44] M. Wang, Q. Chen, Assessment of various turbulence models for transitional flows in an enclosed environment (RP-1271), HVAC&R Research 15 (6) (2009) 1099–1119.

[45] M. Wang, Q. Chen, On a hybrid rans/les approach for indoor airflow modeling (rp-1271), HVAC&R Research 16 (6) (2010) 731–747.

[46] L. C. Jain, M. Seera, C. P. Lim, P. Balasubramaniam, A review of online learning in supervised neural networks, Neural Computing and Applications 25 (3) (2014) 491–509.

[47] B. Pérez-Sánchez, O. Fontenla-Romero, B. Guijarro-Berdiñas, A review of adaptive online learning for artificial neural networks, Artificial Intelligence Review 49 (2) (2018) 281–299.