2020

Troika Generative Adversarial Network (T-GAN): A Synthetic Image Generator That Improves Neural Network Training for Handwriting Classification

Joe Anthony M. Milan

Proceso L. Fernandez Jr

Philippine Journal of Science
149 (3-a): 841-856, October 2020
ISSN 0031 - 7683
Date Received: 10 Mar 2020

Troika Generative Adversarial Network (T-GAN): A Synthetic Image Generator That Improves Neural Network Training for Handwriting Classification

Joe Anthony M. Milan[1]* and **Proceso L. Fernandez**[2]

[1]College of Computer Science, Don Mariano Marcos Memorial State University – South La Union Campus (DMMMSU-SLUC), Agoo, La Union, Region I 2504 Philippines
[2]Department of Information Systems and Computer Science
Ateneo De Manila University (ADMU), Quezon City, National Capital Region 1108 Philippines

Training an artificial neural network for handwriting classification requires a sufficiently sized annotated dataset in order to avoid overfitting. In the absence of sufficient instances, data augmentation techniques are normally considered. In this paper, we propose the troika generative adversarial network (T-GAN) for data augmentation to address the scarcity of publicly labeled handwriting datasets. T-GAN has three generator subnetworks architectured to have some weight-sharing in order to learn the joint distribution from three specific domains. We used T-GAN to augment the data from a subset of the IAM Handwriting Database. We then compared this with other data augmentation techniques by measuring the improvements brought by each technique to the handwriting classification accuracies in three types of artificial neural networks (ANNs): deep ANN, convolutional neural network (CNN), and deep CNN. The data augmentation technique involving the T-GAN yielded the highest accuracy improvements in each of the three ANN classifier types – outperforming the standard techniques of image rotation, affine transformation, and combination of these two – as well as the technique that uses another GAN-based model, the coupled GAN (CoGAN). Furthermore, a paired t-test between the 10-fold cross-validation results of the T-GAN and CoGAN, the second-best augmentation technique in this study, on a deep CNN-made classifier confirmed the superiority of the data augmentation technique that uses the T-GAN. Finally, when the generated synthetic data instances from the T-GAN were further enhanced using the pepper noise removal and median filter, the classification accuracy of the trained CNN and deep CNN classifiers were further improved to 93.54% and 95.45%, respectively. Each of these is a big improvement from the original accuracies of 67.43% and 68.32%, respectively of the 2 classifiers trained on the original unaugmented dataset. Thus, data augmentation using T-GAN – coupled with the mentioned two image noise removal techniques – can be a preferred pre-training technique for augmenting handwriting datasets with insufficient data samples.

Keywords: artificial neural networks (ANN), data augmentation, generative adversarial network (GAN), handwriting classification, synthetic data, troika GAN

*Corresponding Author: joeanthony.milan@dmmmsu-sluc.com

INTRODUCTION

Handwriting classification involves processing images of a handwritten text to accurately produce the matching digital form of the text. It has been a topic of much interest for many years, and the current state-of-the-art techniques involve deep learning neural network models (Abiodun *et al.* 2018). However, training such models, because of the sizable number of parameters, normally requires a large amount of annotated data samples in order to avoid overfitting.

With the scarcity of publicly labeled handwriting datasets in the global market, the expensive collection of correctly labeled images, and the lack of access to substantial amounts of data, it is a challenge to train a deep neural network that performs robustly. Thus, it is sensible to explore ways to increase the instances of a limited dataset through the generation of synthetic data to enable achieving good accuracy at handwriting classification. This process of generating additional training data is referred to as data augmentation.

For most image-processing applications, data augmentation is normally done through the utilization of annotation-preserving transformations on the input data from the existing dataset (Goodfellow *et al.* 2016). These transformations include randomly rotating, shearing, translating, cropping, flipping, or deforming the image. Through the random nature of data augmentation, an endless (in theory) supply of training data can be generated. Data augmentation, however, is not universally applicable to all problem domains (Neff *et al.* 2018). For example, in handwriting classification, the horizontal flips (mirroring) and 180-degree rotations cannot be applied because they would produce invalid samples.

In 2014, Goodfellow *et al.* introduced the generative adversarial network (GAN). It offers a promising method of automatically learning a generative model by just training standard deep neural networks. A GAN consists of two adversarial subnetworks – a generator and a discriminator. The generator synthesizes data from an input noise vector, aiming to fool the discriminator into misclassifying the synthetic data as a real instance. The discriminator is a standard classification network that receives both real and generator-synthesized instances and aims to perfectly classify each input image as either real or synthetic.

Since the emergence of GANs with impressive outcomes in various domains, numerous variants of GAN have been proposed. These include the deep convolutional GAN (Radford *et al.* 2015), CycleGAN (Chu *et al.* 2017), Wasserstein GAN (Arjovsky *et al.* 2017), DeLiGAN (Gurumurthy *et al.* 2017), DeblurGAN (Kupyn *et al.* 2017), SimGAN (Dilipkumar 2017), DualGAN (Yi *et al.* 2017), CoGAN (Liu and Tuzel 2016), InfoGAN (Chen *et al.* 2016), conditional GAN (Mirza and Osindero 2014), and more.

These GANs have been used purposively in many data generation tasks such as image generation (Marchesi 2017; Wang and Jiang 2016), domain-transfer (Bousmalis 2017), auto-painter (Liu *et al.* 2017b), synthetic data generation (Zhu *et al.* 2018; Nazki *et al.* 2018; Frid-adar *et al.* 2018), and text to photo-realistic image synthesis (Zhang *et al.* 2017). Though GANs show impressive results when trained on large datasets, how GANs perform when trained on a small amount of data is still a topic of active research.

The proposal of CoGAN opens the first attempt to tackle the generation of data from two domains (Bang and Shim 2018). The incorporation of autoencoder, encoder, decoder, classifiers, and loss function modifications are some of the enhancements or variations performed to the CoGAN architecture. Liu *et al.* (2017a) extended CoGAN by integrating variational autoencoders for GAN to achieve image translation by mapping the data between two domains into a shared latent space through the shared weighted encoder. To attain the unpaired and unsupervised image-to-image translations, CycleGAN (Zhu *et al.* 2017) and DiscoGAN (Kim *et al.* 2017) use a cyclic consistent loss term in addition to the adversarial loss, while DualGAN (Yi *et al.* 2017) takes advantage of dual learning by expanding the basic GANs into two CoGANs, two generators, and two discriminators. On the other hand, triangle GAN (Gan *et al.* 2017) consists of four neural networks, two generators, and two discriminators, whereas triple GAN (Li *et al.* 2017) comprises two conditional GANs with a generator, a discriminator, and a classifier to perform semi-supervised learning. Bang and Shim (2018) introduced the resembled GAN, which employs a feature statistic matching algorithm and implicitly induces two generators to match feature covariance from both domains leading to share semantic attributes. The multitask CoGAN (Lin *et al.* 2018) consists of CoGAN networks for scene Chinese character style transfer and classifier networks trained by the style-transferred data generated by the CoGAN. More recent studies that extend or tweak the CoGAN architecture include the Conditional CoGAN (Wang and Gupta 2019), which implements CoGAN as a conditioning model to capture the joint distribution of dual-domain samples in two different tasks. These include the spatio-temporally CoGAN (Qi *et al.* 2020) for predictive scene parsing, which employs both CNNs and convolutional long short-term memory in the encoder-decoder architecture; and TriGAN (Roy *et al.* 2020) for data-generation from multiple source domains, which is composed of a generator network comprising of an encoder and decoder, and a discriminator network that is based on the projection discriminator.

Most of the image-to-image translation or domain adaptation works that cited the CoGAN architecture focus on datasets of characters or digital images, biomedical images, artworks, paintings, sketches, animal images, human images, human poses, road scenes, vehicle images, cityscapes, object images, 3D and 2D images, music, and videos. Although existing works have achieved promising results, in most of them, there was an ample amount of instances to perform multiple domains or one-to-one domain adaptations. Our extensive literature review revealed no previous work that focused on a limited size of handwritten text datasets and added a third branch to CoGAN for data augmentation.

In this paper, we extend the CoGAN architecture and propose a new GAN variant – the T-GAN. The T-GAN contains a group of three GANs working together to learn good models for generating synthetic data from three specified domains (original handwriting image data, randomly rotated data, and affine transformed data). The generated image data are then used to increase the number of available instances to improve the training of various ANN classifiers, as demonstrated by significant improvements in the accuracies of these classifiers. The rest of the paper details the proposed T-GAN further and also describes the preparations, experiments conducted, the results produced, and the insights from the analyses of these results.

METHODOLOGY

Initial Dataset for Handwriting Classification

The IAM Handwriting Database (Marti and Bunke 2002) is a large publicly available database of handwritten English text written by 657 writers. This database is commonly used to train and test handwritten text recognizers. It has a handwriting sample of 115,320 images of isolated and labeled words scanned at a resolution of 300 dpi and saved as PNG images with 256 gray levels.

In this paper, a small subset of the IAM Handwriting Database was used in order to simulate the usual scenario where only a small-sized dataset is available. This enables us to investigate the effectiveness of using T-GAN as a data augmentation technique for situations involving very limited real data instances. Previous studies (Dilipkumar 2017, Wigington 2017) on handwriting classification also used a subset of the IAM Handwriting Database, but where some words actually contain a good enough number of samples. To add challenge in our study, we used only 439 images corresponding to the 20 least frequently occurring words from this set. A list of these words with their corresponding frequencies is presented in Table 1.

Table 1. Frequency of handwritten words used in the study.

Vocabulary words	Frequency
alone	22
answer	22
became	22
believe	22
charming	22
cure	22
enjoy	22
event	22
except	22
extraordinary	22
fire	22
foreign	22
heavy	22
hoped	22
master	22
met	22
Mr	22
rest	22
shops	22
stop	22
Grand total	**439**

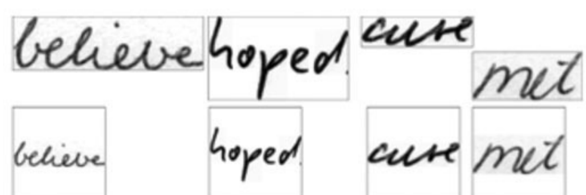

Figure 1. Original image samples and their corresponding resized versions

Segmented instances of handwritten words were already provided in the database, so no segmentation technique was needed in this work. However, the dataset contained images of different sizes, whereas the neural network – because of its fixed-sized input layer – required all input images to have the same dimensions. Thus, we resized each original image to 100 x 100 pixels while preserving the aspect ratio of the image of the word. Figure 1 displays the sample size of the original image (first row) with the corresponding resized versions (second row).

Handwriting Classification Using ANN Models

Three architecturally different ANNs were considered as classifiers to establish the baseline in the study. This is to

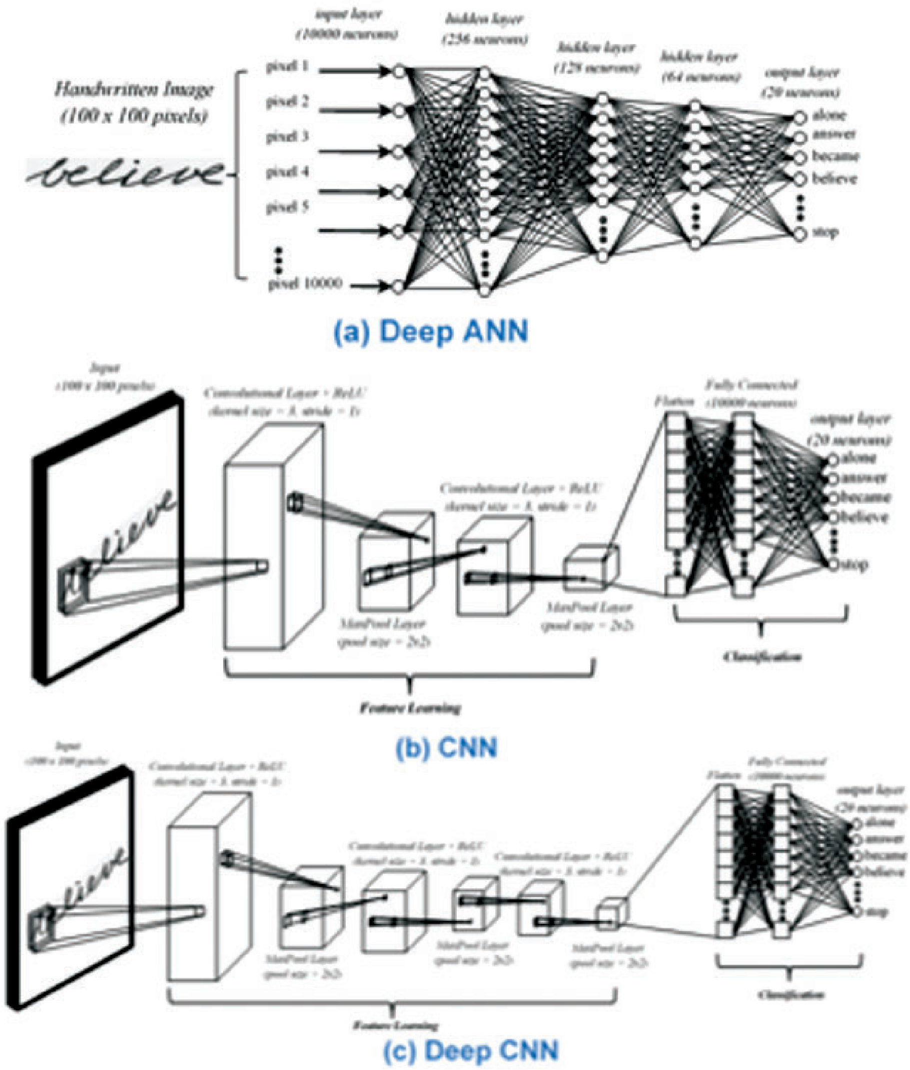

Figure 2. Architectures of the handwriting classification models.

Table 2. Tables of the deep ANN handwriting classification model.

Layer (type)	Output shape	Param. #
dense_1 (dense)	(None; 256)	2560256
dense_2 (dense)	(None; 128)	32896
dense_3 (dense)	(None; 64)	8256
dense_4 (dense)	(None; 20)	1300

Total params.: 2,602,708
Trainable params.: 2,602,708
Non-trainable params.: 0

Table 3. Details of the CNN handwriting classification model.

Layer (type)	Output shape	Param. #
conv2d_1 (Conv2D)	(None; 98, 98, 16)	160
max_pooling2d_1 (MaxPooling2D)	(None; 49, 49, 16)	0
conv2d_2 (Conv2D)	(None; 47, 47, 8)	1160
max_pooling2d_2 (MaxPooling2D)	(None; 23, 23, 8)	0
flatten_1 (flatten)	(None; 4232)	0
dense_1 (dense)	(None; 10000)	42330000

Total params.: 42,531,340
Trainable params.: 42,531,340
Non-trainable params.: 0

Table 4. Details of the deep CNN handwriting classification model.

Layer (type)	Output shape	Param. #
conv2d_1 (Conv2D)	(None; 98, 98, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None; 49, 49, 32)	0
conv2d_2 (Conv2D)	(None; 47, 47, 16)	4624
max_pooling2d_2 (MaxPooling2D)	(None; 23, 23, 16)	0
conv2d_3 (Conv2D)	(None; 21, 21, 8)	1160
max_pooling2d_3 (MaxPooling2D)	(None; 10, 10, 8)	0
flatten_1 (flatten)	(None; 800)	0
dense_1 (dense)	(None; 10000)	8010000
dense_2 (dense)	(None; 20)	200020

Total params.: 8,216,124
Trainable params.: 8,216,124
Non-trainable params.: 0

demonstrate that the data augmentation technique is able to bring forth improvements in accuracy regardless of the type of neural network classifier. These three ANNs are the deep ANN, the CNN, and the deep CNN. The architectures of the three different ANN models implemented in the study are presented in Figure 2 and detailed in Tables 2, 3, and 4, respectively.

All the three ANN types were trained using the ADAM optimizer and the categorical cross-entropy was used as the loss function. Stratified 10-fold cross-validation was done, with the average accuracy in the 10 folds used as the baseline accuracy for each of the three ANN classifiers.

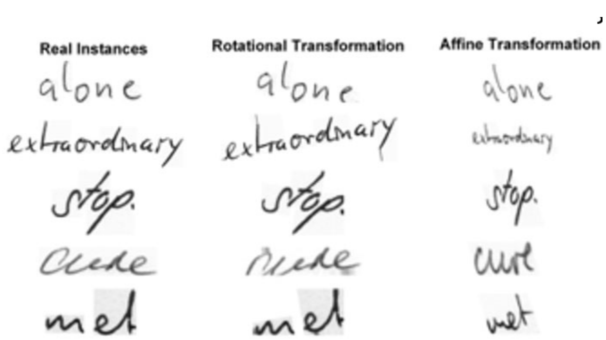

Figure 3. Generated synthetic handwriting data using augmentation techniques.

Standard Image Data Augmentation Techniques

To investigate the improvements in the classification accuracy induced by the application of data augmentation, we first explored the following standard techniques: image rotation, affine transformation, and the combination of these two. The rotation was implemented using an ImageDataGenerator class, a Keras library class that generates batches of tensor image data with real-time data augmentation. The angles of rotation were all random and between –10 and 10 degrees. The affine transformation was also implemented by applying shearing and rotation attributes using the AffineTransform() method in the scikit-image library for the Python programming language. Figure 3 shows some examples of synthetic data produced using standard data augmentation techniques.

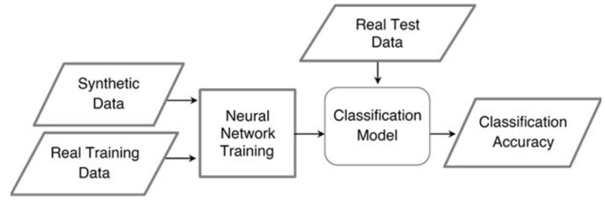

Figure 4. Training procedures of the handwriting classification models

Measuring the Classification Accuracy Improvements from Data Augmentation

The training procedure for each of the three handwriting classification models is illustrated in Figure 4. While the baseline models (deep ANN, CNN, and deep CNN) used only real images for training during the first phase of the experiments, these models were trained using augmented dataset (real training data + synthetic data) in the second phase. This allows us to measure how much improvement, if any, is induced by a specific data augmentation technique.

Similar to what was done in the first phase, stratified 10-fold cross-validation was implemented in the second phase of the experiment. We made sure that the images used for generating synthetic data came only from the nine training folds and all the generated images were used only for training the ANN classifier. The remaining fold, having no synthetic test data, served as the test set to evaluate the model. Iteratively, each of the models was gauged 10 times to calculate the accuracy rating for each fold. The average classification accuracy from the 10 folds was used as an estimate of the model's performance.

Using only the original dataset for training, the deep ANN yielded poor results while the CNN and deep CNN handwriting classifiers were able to produce relatively good results, as shown in Table 5. As shown also in the same table, the baseline accuracies from these three ANNs have been improved with the application of a data augmentation technique compared against the baseline results.

Table 5. Accuracy results of the handwriting classification models.

Training data	Dataset size	Deep ANN classification accuracy	CNN classification accuracy	Deep CNN classification accuracy
Without data augmentation	439	4.78%	67.43%	68.32%
With rotational transformation	1317	5.47%	69.02%	73.13%
With affine transformation	1317	5.24%	67.64%	68.56%
With rotational + affine transformations	1317	5.01%	69.93%	70.86%
With $CoGAN_1$	1317	5.24%	77.89%	81.11%
With $CoGAN_2$	1317	5.01%	67.73%	80.87%
With $CoGAN_1 + CoGAN_2$	1317	5.24%	79.05%	83.62%
With T-GAN	1317	5.92%	83.38%	87.27%

Improving the Augmentation Technique Using GAN-based Models

We explored different architectures of GAN models to determine which can produce good quality synthetic images that can be used in training more accurate neural network classifiers. One such GAN is the CoGAN proposed by Liu and Tuzel in 2016. CoGAN is designed to learn a joint distribution of images in two different domains. It consists of a pair of GANs (GAN_1 and GAN_2), each of which is responsible for synthesizing images in one domain. These two share the weights in the first few layers (responsible for decoding high-level semantics) of the generative models. They also share the weights in the last few layers of the discriminative models (see Figure 5a). This weight-sharing constraint allows CoGAN to learn a joint distribution of images. Both the weight-sharing constraint and adversarial training are essential in learning this distribution even without correspondence supervision. Figure 5 illustrates the architectures of the two networks.

In the original paper, the MNIST (handwritten digit) dataset was used to train CoGAN for two specific tasks. An experiment for learning a joint distribution was performed in which the first domain consisted of the original handwritten digit images, and the second domain consisted of the digits 90-degree in-plane rotation.

To our knowledge, data augmentation using CoGAN has not yet been explored for handwriting classification. In our experiments, the CoGAN architecture (Liu and Tuzel 2016) was modified to receive and produce images with size 100 x 100 pixels. We constructed CoGAN using a deep ANN unlike in the original paper where CoGAN was structured using deep CNN. Training a GAN made with deep CNN takes a long time and so, we modified the construction of CoGAN using a deep ANN to reduce the time consumed for training. The 90-degree in-plane rotation used in a previous paper (Liu and Tuzel 2016) was also not adopted because doing so will produce generally unrealistic handwriting images. We, therefore, paired the original dataset (first domain) with images randomly rotated from –10 to 10 degrees angle, which served as the second domain. The parameters of the generators and discriminators were also adjusted to generate promising synthetic handwriting images. After the training, we kept the generator models for each class to generate synthetic data to be added to the real training data of the handwriting classification networks.

There were actually three different CoGAN-based data augmentation techniques experimented on in this study. $CoGAN_1$ was trained using the original dataset and the randomly rotated instances of the handwriting samples. Another model of CoGAN was trained on original dataset and affine transformed data, and we refer to this as $CoGAN_2$. The last model was the combination of $CoGAN_1$ and $CoGAN_2$, which we represented as $CoGAN_1 + CoGAN_2$ in the paper. These variations of CoGAN were made to have a reasonable comparison of the improvements induced in the classification accuracy when compared to our proposed synthetic image generator.

Since the results of using CoGAN looked promising, we explored extending the architecture by including an additional subnetwork, thus having a group of three GANs (GAN_1, GAN_2, and GAN_3) working together to synthesize images from three different domains. We refer to this new GAN variant as the Troika GAN (T-GAN).

The T-GAN was trained to add the third domain obtaining an instance of the original data, which are generated using affine transformation to the first and second domains used to train CoGAN. Afterward, the generator models from this data augmentation technique were kept for each class and utilized in generating synthetic data for handwriting classification.

The generative models (generators) were structured with five layers. Batch normalization processing with a momentum of 0.8 and LeakyRelu having an alpha of 0.2 were employed in all layers of the models, whereas the output layer used the tanh activation function. As shown

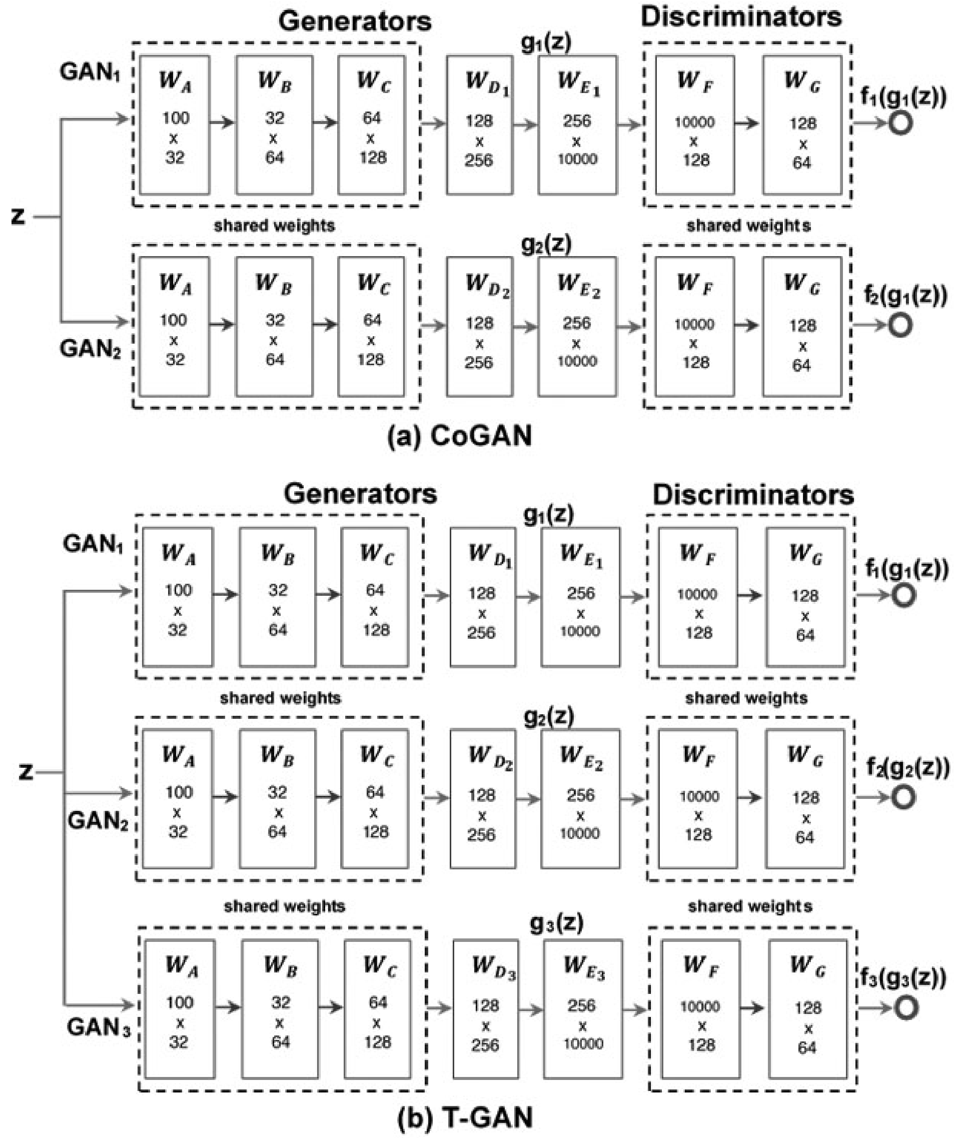

Figure 5. (a) CoGAN and (b) T-GAN Architectures: GAN1 is used to generate synthetic data from the original dataset,GAN2 is responsible of synthesizing rotaed images, and GAN3 learns to synthesize synthetic data featuring affine transformations.

in Figure 5b, the training parameters were shared in all layers (W_A, W_B, and W_C) except for the last hidden layer (WD_1, WD_2, and WD_3) and the output layer (WE_1, WE_2, and WE_3). Output images were generated having a random noise vector of size 100 as an input (z).

For the discriminative models (discriminators), two layers were shared – namely, the input layer (W_F) and a hidden layer (W_G) – except for the output layer (f_1,f_2, and f_3). The models implemented LeakyRelu having a 0.2 alpha and dropout of 0.4, while the output layer employed sigmoid as the activation function. Batches containing output images from the generative models and images from the training subsets (each pixel value is normalized to be in the range [–1, 1]) were the inputs to the discriminator models.

For training the CoGAN and T-GAN, we used the ADAM optimizer setting the learning rate to 0.0002 and momentum parameter to 0.5. Since we have a limited dataset, batch size was set to 20. The number of iterations to train the two GANs was extended arbitrarily until the model produces synthetic handwriting images that have merits to the handwriting classification networks. We visually checked the produced synthetic images to determine if it is a realistically looking handwriting sample. Once it did so at 30000 epochs, we then stopped the training and employed the generated models for each class to determine the impact of using CoGAN and T-GAN as data augmentation approaches.

Generator Networks
Input = 100 random noise

Input Layer:
32 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)
1st Hidden Layer:
64 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)
2nd Hidden Layer:
128 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)

(1st Domain):
3rd Hidden Layer:
256 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)
$g_1(z)$:
10000 neurons, activation = tanh

(2nd Domain):
3rd Hidden Layer:
256 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)
$g_2(z)$:
10000 neurons, activation = tanh

(3rd Domain):
3rd Hidden Layer:
256 neurons, LeakyRelu(alpha=0.2)
BatchNormalization(momentum=0.8)
$g_3(z)$:
10000 neurons. activation = tanh

Discriminator Networks
Flatten Input: 100 x 100 x 1

Input Layer:
128 neurons, LeakyRelu(alpha=0.2)
DropOut(0.4)
1st Hidden Layer:
64 neurons, LeakyRelu(alpha=0.2)
DropOut(0.4)

(1st Domain):
$f_1(g_1(z))$:
1 neuron, activation=sigmoid

(2nd Domain):
$f_2(g_2(z))$:
1 neuron, activation=sigmoid

(3rd Domain):
$f_3(g_3(z))$:
1 neuron, activation=sigmoid

Figure 6. The discriminator and generator networks used for CoGAN and T-GAN.

```
Initialize the shared network parameters connection weights:
g1(z), g2(z) = same values
f1(g1(z)), f2(g2(z)) = same values
for i=0 to 30000 do
    z = random noise vector
    img1 = randomly selected samples from the first domain
    img2 = randomly selected samples from the second domain
    f1(g1(z)) = gradients of parameters
    f2(g2(z)) = gradients of parameters
    average_gradientsDisc = 0.5 * [f1(g1(z)) + f2(g2(z))]
    Calculate the f1^{i+1} and f2^{i+1} according to the gradients
    g1(z) = gradients of parameters
    g2(z) = gradients of parameters
    average_gradientsGen = 0.5 * [g1(z) + g2(z)]
    Calculate the g1^{i+1} and g2^{i+1} according to the gradients
end for
```

```
Initialize the shared network parameters connection weights:
g1(z), g2(z), g3(z) = same values
f1(g1(z)), f2(g2(z)), f3(g3(z)) = same values
for i=0 to 30000 do
    z = random noise vector
    img1 = randomly selected samples from the first domain
    img2 = randomly selected samples from the second domain
    img3 = randomly selected samples from the third domain
    f1(g1(z)) = gradients of parameters
    f2(g2(z)) = gradients of parameters
    f3(g3(z)) = gradients of parameters
    average_gradientsDisc = [f1(g1(z)) + f2(g2(z)) + f3(g3(z)] / 3
    Calculate the f1^{i+1}, f2^{i+1} and f3^{i+1} according to the gradients
    g1(z) = gradients of parameters
    g2(z) = gradients of parameters
    g3(z)) = gradients of parameters
    average_gradientsGen = [g1(z) + g2(z) + g3(z))] / 3
    Calculate the g1^{i+1}, g2^{i+1} and g3^{i+1} according to the gradients
end for
```

Figure 7. Pseudocode implemented for the CoGAN (left) and T-GAN.

The training period took approximately 24.80 min and 35.54 min, respectively, for CoGAN and T-GAN. Although the two GAN architectures took a long time for training, the generation of synthetic data is much faster.

A summary of the network architectures of CoGAN and T-GAN is presented in Figure 6. This details the parameters used in the experiments in which the generator and the discriminator contain networks shared by each domain. Note that the highlighted sections using broken lines indicate the shared layers of the networks, while the shaded parts of the figure emphasize the added GAN network for T-GAN. Furthermore, Figure 7 shows the pseudocode implemented for the CoGAN and T-GAN, and the highlighted statements denote the added processes employed to devise T-GAN.

Implementation Details

The different augmentation techniques and the ANN classification models were implemented using open source frameworks such as a) Python 3.6.4, the programming language; b) OpenCV 3.3.1, the image processing library; c) scikit-image 0.13.1, a Python package dedicated for image processing; d) Keras 2.1.6, an open-source neural network library that runs using Tensorflow as the backend; and e) Jupyter Notebook 5.4.0, an open-source web application program interface. On the other hand, experiments were performed on an ASUS laptop with Intel Core i7-8750H CPU and NVIDIA GeForce GTX 1050 Ti GPU running at 2.20GHz~2.21GHz using 8.00 GB RAM and Windows 10 Home Single Language 64-bit Operating System.

RESULTS AND DISCUSSION

Sample Images of Augmented Data

Figure 8 shows some sample images of generated synthetic data using the limited real dataset applying the rotation, affine transformations, $CoGAN_1$ (trained using the original dataset and rotational transformation), $CoGAN_2$ (trained using the original dataset and affine transformation), and T-GAN as data augmentation techniques. The resulting synthetic instances of the handwriting dataset offer diversity and are expected to contribute to better classification accuracy through the improvement of the training of the different ANN classifiers. Table 5 shows the comparison of classification accuracy results after 10 epochs of training the three models (deep ANN, CNN, and deep CNN) with the augmented dataset as opposed to the non-augmented dataset. The table also displays the size of datasets that served as training data for each model in which training-set size was increased using rotational transformation, affine transformations, CoGAN, and T-GAN augmentation techniques.

Figure 8. Sample images of the generated handwriting data using image rotation, affine transformations, CoGAN1, CoGAN2, and T-GAN.

Classification Accuracies

The first row of Table 5 shows the baseline accuracies of 4.78% (deep ANN), 67.43% (CNN), and 68.32% (deep CNN) for the three ANN classifiers using the original dataset only. The much higher accuracies of CNN-based models are expected since CNN models have generally been shown in the literature to work very well with image datasets.

The next three rows show the results of the three ANN classifier types trained on datasets that are augmented with the specified traditional techniques. Observe that the size of the dataset is three times that of the original dataset. For all but one case, the baseline accuracy for each ANN classifier type was improved with the application of data augmentation. The improvements, though, are generally small.

The last four rows of Table 5 show the results of using GAN-based data augmentation. Note that for all three ANN classifier types, the highest improvements from the baseline accuracies were obtained when the T-GAN data augmentation was performed. Table 6 further details these improvements, showing significant increases (*i.e.* more than 23% *relative* to the baseline) across all 3 ANN classifier types.

We next investigated if the T-GAN is statistically superior to the second-best data augmentation technique in our experiments. For this, we performed a paired t-test of the accuracies in the 10-fold cross-validations of the T-GAN and the $CoGAN_1 + CoGAN_2$. As shown in Table 7, the difference is significant ($\alpha = 0.05$), with a p-value of 0.0233. Thus, there is strong evidence of the superiority of the T-GAN as a synthetic image generator.

Table 6. Accuracy Improvements in using T-GAN for data augmentation.

ANN classifier type	Baseline accuracy	T-GAN induced accuracy	Absolute increase in accuracy	Relative increase in accuracy
Deep ANN	4.78%	5.92%	1.14%	23.85%
CNN	67.43%	83.38%	15.95%	23.65%
Deep CNN	68.32%	87.27%	19.84%	27.74%

Table 7. Paired t-test results comparing the two best-performing data augmentation techniques for the Deep CNN handwriting classification model.

Fold number	$CoGAN_1 + CoGAN_2$	T-GAN
1	88.64%	86.36%
2	79.55%	81.82%
3	81.82%	90.91%
4	86.36%	93.18%
5	81.82%	86.36%
6	93.18%	90.91%
7	70.55%	84.09%
8	86.05%	88.64%
9	88.64%	88.64%
10	79.55%	81.82%
Mean	**83.62%**	**87.27%**
***p*-value**	**0.0233 <= α, therefore, significant**	

Legend: α = 0.05

Caveats: Data Augmentation Time and Model Training Time

While the previous results show that the use of T-GAN induced the biggest accuracy improvements across the different ANN classifier types, it did come at the cost of higher data augmentation time. There is a possibility that the seemingly better augmentation technique of T-GAN can be overcome by the other techniques by simply spending the "extra" time available to generate more synthetic instances. We show that this is unlikely to be the case by performing another set of experiments. Specifically, for each data augmentation technique, the dataset size was expanded further until the accuracies in all three ANN classifier types showed a decreasing trend or increased just marginally. Table 8 summarizes the results related to this. Observe that for all three ANN types, the best accuracies (shown in the bold and italicized font) were still obtained with the use of the T-GAN data augmentation. Equally important, a trend analysis of the effect of increasing the size of the dataset showed that the accuracy in each of the other data augmentation techniques tend to plateau at lower levels (refer to Figure 9). From this, we infer that allotting even much more time to other augmentation techniques to produce more synthetic data will not produce results that are better than those garnered from the T-GAN augmentation.

We also performed a trend analysis on the effects of the training time. Figure 10 shows that the deep CNN classifier has relatively similar training times after 10 epochs on the different data augmentation techniques; however, the T-GAN data augmentation technique obtained the best accuracies on the different dataset sizes.

Finally, we compare the performance of deep CNN classifier without any augmentation against the performance of deep CNN with T-GAN data augmentation. This is to determine whether or not the time spent for training the T-GAN for data augmentation could have been used to simply provide more training time for the deep CNN classifier (without any augmentation). The results (see Figure 11) shows a clear advantage for the use of T-GAN. Allotting more training time (up to 12000 epochs) for the deep CNN classifier (without any augmentation) does not produce better results than those acquired by the classifier trained (10 epochs) with T-GAN augmentation. Though the time spent for training the deep CNN classifier (without any augmentation) is still shorter than the time spent using T-GAN, its accuracy starts to plateau at a lower level. Setting too much training time for the deep CNN with the limited dataset can expose it to high risk of overfitting (Ying 2019; Srivastava *et al.* 2014). The data augmentation using T-GAN to expand the training set can alleviate the overfitting encountered by the deep CNN handwriting classifier, thus improving the performance of the classifier.

Post Processing: Noise Reduction

Since the sample images generated by both the CoGAN and T-GAN appeared a bit noisy, we performed some post-processing techniques to improve the generated images further. Specifically, we applied pepper noise removal to reduce the extraneous dots, and then afterward applied the median filter to intensify the black and white pixels. Samples of post-processed synthetic handwriting images are shown in Figure 12.

Using the CNN and deep CNN based handwriting classifiers, the implementation of the post-processing technique yielded improvements in performance, as shown in Table 9. The T-GAN achieved the highest accuracy of 93.54% with a 3.78% increase from the previous accuracy result (89.76%) for the CNN-built handwriting classifier, while a 1.14% increase was carried for the deep CNN (from 94.31% to 95.45%). The post-processing technique also enhanced the performance of the handwriting classification models when applied to the generated synthetic images using the three variants of CoGAN ($CoGAN_1$, $CoGAN_2$, and combination of both).

A paired t-test was again conducted (refer to Tables 10 and 11), and this revealed that the data augmentation using T-GAN, together with the post-processing noise reduction technique, yields (statistically) significantly better accuracies. The resulting *p*-values were 0.00676 for CNN and 0.0261 for the deep CNN, both of which are significant (α = 0.05).

Table 8. Accuracy results of the handwriting classification models with more images.

Training data	**Dataset size**	**Augmentation time (s)**	**Deep ANN classification accuracy**	**CNN classification accuracy**	**Deep CNN classification accuracy**
Without data augmentation	439	–	4.78%	67.43%	68.32%
With rotational transformation	1317	9.30	*5.47%*	69.02%	73.13%
	2195	18.60	5.01%	69.23%	72.89%
	3073	28.10	4.78%	69.01%	73.41%
	3951	37.23	5.01%	70.60%	72.88%
	4829	51.25	5.01%	70.61%	74.03%
	5707	55.96	5.24%	*71.75%*	*74.93%*
	6585	65.35	4.56%	69.25%	73.81%
With affine transformation	1317	17.80	5.24%	63.56%	68.56%
	2195	35.63	*5.47%*	63.55%	68.11%
	3073	53.45	4.78%	64.24%	69.92%
	3951	71.24	4.78%	*65.61%*	*70.17%*
	4829	89.42	5.01%	65.61%	67.44%
With rotational + affine transformations	1317	15.55	5.01%	69.93%	70.86%
	2195	27.10	*5.47%*	*72.91%*	*73.58%*
	3951	62.28	5.01%	66.05%	72.19%
	5707	93.36	4.56%	68.78%	71.53%
With $COGAN_1$	1317	107.40	5.24%	77.89%	81.11%
	2195	222.12	*5.47%*	82.01%	84.06%
	3073	325.92	5.47%	*84.28%*	86.57%
	3951	432.72	5.24%	83.59%	87.47%
	4829	534.12	5.01%	81.55%	87.02%
With $COGAN_2$	1317	126.00	5.01%	67.73%	80.87%
	2195	248.40	*5.24%*	78.37%	82.79%
	3073	372.60	5.01%	*80.64%*	83.83%
	3951	504.00	5.24%	80.41%	84.74%
	4829	631.20	4.79%	79.96%	*85.66%*
With $COGAN_1$ + $COGAN_2$	1317	116.70	*5.24%*	79.05%	83.62%
	2195	235.26	5.01%	80.41%	85.21%
	3073	349.26	5.24%	81.32%	*86.12%*
	3951	468.36	5.24%	*82.46%*	85.44%
	4829	584.46	5.01%	82.01%	85.88%
With T-GAN	1317	167.40	***5.92%***	83.38%	87.27%
	2195	355.80	5.47%	88.15%	90.89%
	3073	523.20	5.01%	88.16%	92.72%
	3951	691.80	5.70%	***89.76%***	93.64%
	4829	859.20	5.24%	89.53%	93.86%
	5707	1032.10	5.24%	88.61%	***94.31%***

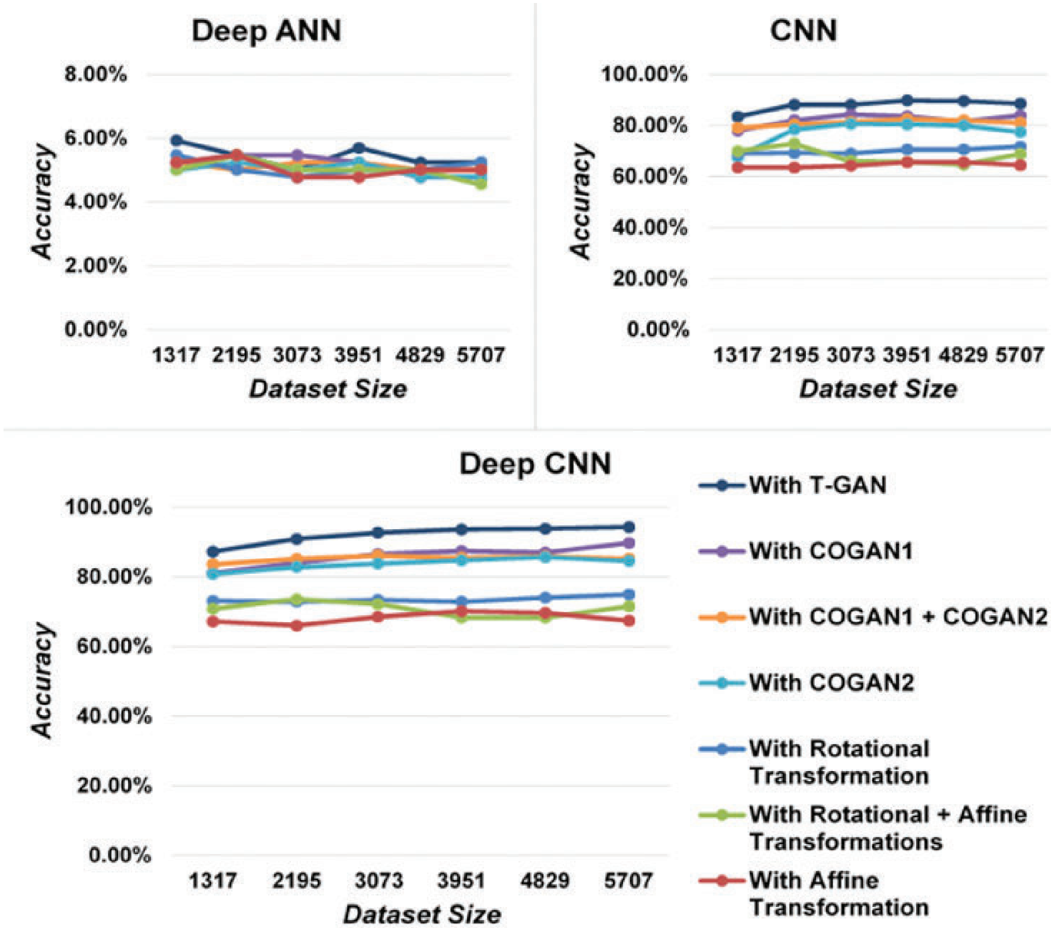

Figure 9.. Trend analysis of the effect of increasing the size of the dataset.

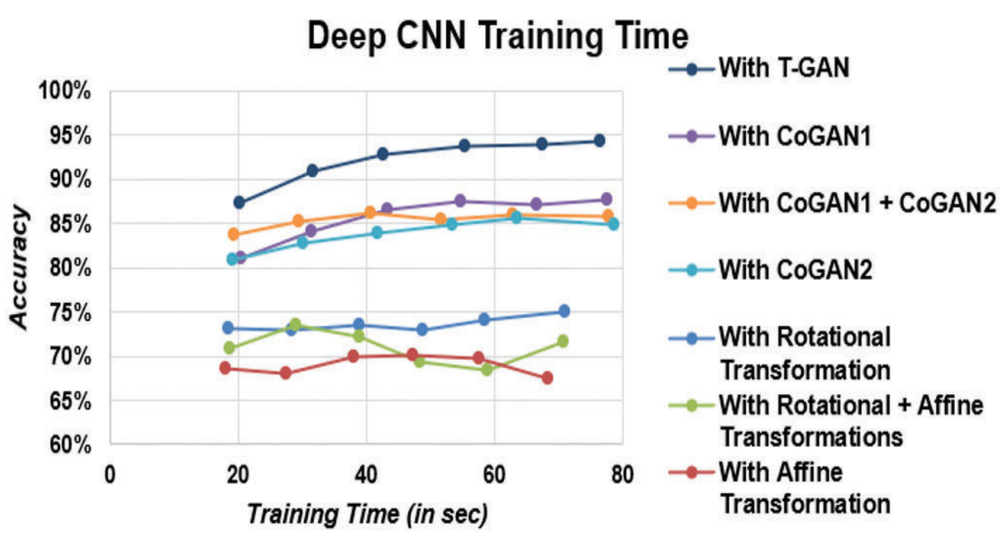

Figure 10.. Trend analysis of the effect of training time on Deep CNN classifer.

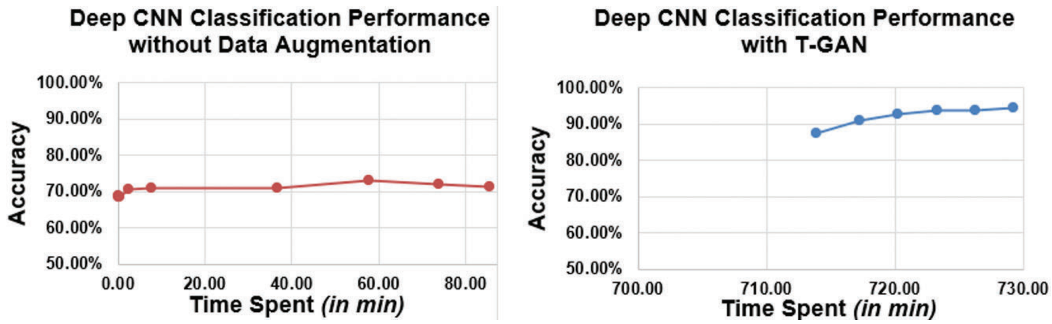

Figure 11.. Comparison of Deep CNN without data augmentation and with TGAN augmentation.

Figure 12.. Sample images after the implementation of the post processing technique. Post Processing: Noise Reduction

Table 9. Accuracy results of the CNN handwriting classification models after post processing the generated synthetic data.

Data augmentation technique	Dataset size	CNN classification accuracy	Deep CNN classification accuracy
$CoGAN_1$	1317	82.91%	85.42%
	2195	85.42%	87.47%
	3073	86.36%	89.06%
	3951	88.64%	89.29%
	4829	86.59%	89.06%
$CoGAN_2$	1317	80.64%	83.89%
	2195	81.55%	85.66%
	3073	81.78%	87.93%
	3951	82.24%	86.10%
	4829	81.55%	86.57%
$CoGAN_1 + CoGAN_2$	1317	80.87%	84.75%
	2195	81.33%	86.56%
	3073	82.92%	87.02%
	3951	80.41%	87.49%
	4829	82.46%	86.80%
T-GAN	1317	83.84%	87.92%
	2195	91.12%	92.94%
	3073	92.03%	94.76%
	3951	93.17%	94.98%
	4829	93.18%	94.99%
	5707	**93.54%**	**95.45%**

Table 10. Significant difference between the accuracy results of implementing post processing techniques and without post processing using T-GAN data augmentation approach for a deep CNN handwriting classification model.

Fold number	T-GAN (without post processing)	T-GAN (with post processing)
1	95.45%	95.45%
2	90.91%	93.18%
3	93.18%	95.45%
4	90.91%	93.18%
5	97.73%	97.73%
6	95.45%	95.45%
7	93.18%	93.18%
8	97.67%	97.67%
9	97.73%	97.73%
10	90.91%	95.45%
Mean	**94.31%**	**95.45%**
***p*-value**	**0.0261 <= α, therefore, significant**	

Legend: α = 0.05

Table 11. Significant difference between the accuracy results of implementing post processing techniques and without post processing using T-GAN data augmentation approach for a CNN handwriting classification model.

Fold number	T-GAN (without post processing)	T-GAN (with post processing)
1	95.45%	97.73%
2	90.91%	93.18%
3	84.09%	90.91%
4	88.64%	93.18%
5	90.91%	99.24%
6	90.91%	90.91%
7	86.36%	95.45%
8	93.02%	93.02%
9	86.36%	93.18%
10	90.91%	88.64%
Mean	**89.76%**	**93.54%**
***p*-value**	**0.00676 <= α, therefore, significant**	

Legend: α = 0.05

CONCLUSION

In this paper, we have presented T-GAN, a new GAN variant inspired from CoGAN, to generate synthetic data from a limited handwriting image dataset. This data augmentation technique involves a group of three GANs working together, by sharing learned weights, to generate synthetic images (that appear quite realistic) from three different domains. In this study, the three domains selected were the original handwritten word images, random rotations, and random affine transformations.

Using T-GAN as a synthetic image generator has brought forth the biggest improvements in the training of the different ANN classifier models, as manifested in the resulting superior classification accuracies. The superiority was statistically established by conducting a paired t-test. It was also demonstrated that the superiority of T-GAN was not due to the longer computational time required to learn the generative model and produce synthetic images, but is rather intrinsic to the nature of this data augmentation technique.

Finally, it was shown that the T-GAN, together with the (image) post-processing techniques of pepper noise reduction and median filter, is a viable data augmentation combination. When used to generate synthetic images for the training of a deep CNN model, the resulting classifier had the best accuracy score of 95.45%, which is a big improvement from the baseline (unaugmented training set) accuracy of 68.32%.

For future studies, it would be interesting to find out how the proposed T-GAN data augmentation technique can be helpful in developing a system with capabilities to decipher other insufficiently labeled handwriting datasets, such as those of prescriptions of doctors.

ACKNOWLEDGMENTS

We extend our heartfelt gratitude to: (a) the Commission on Higher Education under the K-12 Transition Program Management Unit for a graduate scholarship grant to JA Milan, (b) the ADMU for the never-ending support and guidance, and (c) the DMMMSU-SLUC (the scholar's sending higher education institution).

REFERENCES

ABIODUN OI, JANTAN A, OMOLARA AE, DADA KV, MOHAMED NA, ARSHAD H. 2018. State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11): e00938.

ARJOVSKY M, CHINTALA S, BOTTOU L. 2017. Wasserstein GAN. arXiv preprint arXiv:1701.07875.

BANG D, SHIM H. 2018. Resembled generative adversarial networks: two domains with similar attributes. arXiv preprint arXiv:1807.00947.

BOUSMALIS K, SILBERMAN N, DOHAN D, ERHAN D, KRISHNAN D. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1(2): 7.

CHEN X, DUAN Y, HOUTHOOFT R, SCHULMAN J, SUTSKEVER I, ABBEEL P. 2016. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. p. 2172–2180.

CHU C, ZHMOGINOV A, SANDLER M. 2017. CycleGAN: a master of steganography. arXiv preprint arXiv:1712.02950.

DILIPKUMAR D. 2017. Generative adversarial image refinement for handwriting recognition. Carnegie Mellon University, Pittsburgh, PA.

FRID-ADAR M, KLANG E, AMITAI M, GOLDBERGER J, GREENSPAN H. 2018. Synthetic data augmentation using GAN for improved liver lesion classification. In: IEEE 15th International Symposium. p. 289–293.

GAN Z, CHEN L, WANG W, PU Y, ZHANG Y, LIU H..., CARIN L. 2017. Triangle generative adversarial networks. In: Advances in neural information processing systems. p. 5247–5256.

GOODFELLOW I, BENGIO Y, COURVILLE A. 2016. Deep learning. Retrieved from http://www.deeplearningbook.org

GOODFELLOW I, POUGET-ABADIE J, MIRZA M, XU B, WARDE-FARLEY D, OZAIR S, COURVILLE A, BENGIO Y. 2014. Generative adversarial nets. In: Advances in neural information processing systems. p. 2672–2680.

GURUMURTHY S, SARVADEVABHATLA RK, BABU RV. 2017. DeLiGAN: generative adversarial networks for diverse and limited data. In: CVPR. p. 4941–4949.

KIM T, CHA M, KIM H, LEE JK, KIM J. 2017. Learning to discover cross-domain relations with generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning 70: 1857–1865.

KUPYN O, BUDZAN V, MYKHAILYCH M, MISHKIN D, MATAS J. 2017. DeblurGAN: blind motion deblurring using conditional adversarial networks. arXiv preprint arXiv:1711.07064.

LI CX, XU T, ZHU J, ZHANG B. 2017. Triple generative adversarial nets. In: Advances in neural information processing systems. p. 4088–4098.

LIN Q, LIANG L, HUANG Y, JIN L. 2018. Learning to generate realistic scene Chinese character images by multitask coupled GAN. In: Chinese Conference on Pattern Recognition and Computer Vision (PRCV). Cham: Springer. p. 41–51.

LIU MY, BREUEL T, KAUTZ J. 2017a. Unsupervised image-to-image translation networks. In: Advances in neural information processing systems. p. 700–708.

LIU Y, QIN Z, LUO Z, WANG H. 2017b. Auto-painter: cartoon image generation from sketch by using conditional generative adversarial networks. arXiv preprint arXiv:1705.01908.

LIU MY, TUZEL O. 2016. Coupled generative adversarial networks. In: Advances in neural information processing systems. p. 469–477.

MARCHESI M. 2017. Megapixel size image creation using generative adversarial networks. arXiv preprint arXiv:1706.00082.

MARTI UV, BUNKE H. 2002. The IAM-database: an English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition 5(1): 39–46.

MIRZA M, OSINDERO S. 2014. Conditional generative adversarial networks. Manuscript: https://arxiv. org/abs/1709.02023, 9, 24.

NAZKI H, LEE J, YOON S, PARK DS. 2018. Synthetic data augmentation for plant disease image generation using GAN. Future Convergence Contents Realization the 4th Industrial Revolution. p. 459–460.

NEFF T, PAYER C, ŠTERN D, URSCHLER M. 2018. Generative adversarial networks to synthetically augment data for deep learning based image segmentation. In: Proceedings of the OAGM Workshop 2018: Medical Image Analysis. Verlag der Technischen Universität Graz.

QI M, WANG Y, LI A, LUO J. 2020. STC-GAN: spatio-temporally Coupled Generative Adversarial Networks for Predictive Scene Parsing. IEEE Transactions on Image Processing 29: 5420–5430.

RADFORD A, METZ L, CHINTALA S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

ROY S, SIAROHIN A, SANGINETO E, SEBE N, RICCI, E. 2020. TriGAN: Image-to-Image Translation for Multi-Source Domain Adaptation. arXiv preprint arXiv:2004.08769.

SRIVASTAVA N, HINTON G, KRIZHEVSKY A, SUTSKEVER I, SALAKHUTDINOV R. 2014. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1): 1929–1958.

WANG J, JIANG J. 2019. Conditional Coupled Generative Adversarial Networks for Zero-shot Domain Adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. p. 3375–3384.

WANG X, GUPTA A. 2016. Generative image modeling using style and structure adversarial networks. In: European Conference on Computer Vision. Cham: Springer. p. 318–335.

WIGINGTON C, STEWART S, DAVIS B, BARRETT B, PRICE B, COHEN S. 2017. Data augmentation for recognition of handwritten words and lines using a cnn-lstm network. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) 1: 639–645.

YI Z, ZHANG H, TAN P, GONG M. 2017. Dualgan: unsupervised dual learning for image-to-image translation. In: Proceedings of the IEEE international conference on computer vision. p. 2849–2857.

YING X. 2019. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series 1168(2): 022022.

ZHANG H, XU T, LI H, ZHANG S, HUANG X, WANG X, METAXAS D. 2017. Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint.

ZHU JY, PARK T, ISOLA P, EFROS AA. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. p. 2223–2232.

ZHU X, LIU Y, LI J, WAN T, QIN Z. 2018. Emotion classification with data augmentation using generative adversarial networks. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Cham: Springer. p. 349–360.