Investigating GAN and VAE to Train DCNN

Soundararajan Ezekiel, Larry Pearlstein, Abdullah Ali Alshehri, Adam Lutz, Jackson Zaunegger, and Waleed Farag

***Abstract*—The Convolutional Neural Network (CNN) is a class of deep artificial neural network and has recently gained special attention after demonstrating breakthrough accuracies in various classification tasks. CNNs have shown remarkable performance in machine vision tasks such as image classification, natural language processing and speech recognition. There is evidence that the depth of a CNN plays an important role in performance of CNNs. However, we investigated the feasibility of improving the performance of shallow networks via fusion of the features computed by a homogenous and heterogeneous set of pre-trained networks. We also explored a recently developed framework called the Generative Adversarial Network (GAN), in which we simultaneously train two models, a Generator and a Discriminator. The Generator attempts to produce data that mirrors the probability distribution of the "true" dataset. The Discriminator is trained to distinguish between the true dataset and the counterfeit data produced by the Generator. Our work involves the application of a GAN for generation and fine tuning of synthetic data to be used to train a deep CNN. Specifically, we investigate the use of a synthetic data generator along with a GAN to create an unlimited quantity of labeled training data, without the need for hand-labeling images. We apply this technique to the detection and localization of various vehicles. We attempt to distinguish between military trucks and other types of vehicles. A successful outcome could lead to improvements in addressing security threats rapidly, and cost-effectively. We also investigate an alternative method for generating synthetic data, the Variational Auto-Encoder (VAE). Variational auto-encoders are trained to encode then decode input vectors and can also be useful for generating new training data. VAEs are capable of dimensionality reduction and synthesizing data. Finally, we evaluate our multiplicative fusion method compared to the fusion methods that we investigated previously.**

***Index Terms*—DCNN, GAN, VAE, synthetic data, data augmentation, machine learning.**

I. Introduction

Recent advancements in the capabilities of machine learning have led to its widespread adoption across areas such as predictive analysis, natural language processing, image classification, and object detection [1]. With the development of substantially more powerful graphics processing units, neural networks can be trained faster to solve increasingly complex problems [2]. Despite the increased adoption of machine learning applications, practical applications of algorithms are still rudimentary. Neural networks currently require exceptionally large sets of pre-labeled data, as well as exponentially large amounts of computational power to achieve the performance required of the growing industry [3].

Generally, these networks are limited in their scope to individual tasks, making them less versatile than a human learner. In order to compensate for the scarcity of viable data sets, techniques such as resizing, rotating or flipping images, adjusting the lighting and contrast, the addition of noise, and perspective transformations are employed to supplement existing data sets [4]-[6]. By accounting for a wider variation in the test set, these methods can improve the accuracy and versatility of DCNNs [7]. However, despite improving the classification accuracy in training and testing environments, these methods are limited in their practical applications. Deep convolutional neural networks are discriminative networks, which attempt to match features to labels. When given a data sample, the network categorizes the input based on the features of the data sample.

Another technique used to supplement existing data sets is the generation of synthetic data to train machine learning models [8]. Generative machine learning models, as opposed to discriminative, essentially do the opposite. When given a label, generative algorithms attempt to produce a sample with the features that are associated with that label. By applying these algorithms to a comprehensive simulation of real-world conditions, synthetic data can be systematically generated. This training data is labeled automatically instead of the arduous task of manually labeling individual images [9]-[12]. As with augmentation techniques, synthetic data can perform well while training and testing; however, due to oversights in the creation of the DCNN and real-world outliers, they can encounter problems in practical applications [13]-[15]. One solution to this issue is the use of another type of neural network which uses a generative architecture to train a learning model that can generate new samples for a given label [16], [17]. Two prevalent types of generative networks are Generative Adversarial Networks [18] (GANs) and Variational Autoencoders (VAEs). GANs employ the use of two neural networks, a generator which attempts to create synthetic realistic samples and a discriminator which attempts to differentiate between real and synthetic data. The generator network receives random numbers which are converted into an image. These generated images are then supplied to the discriminator network along with real images from the data set. The discriminator then returns a probability for each image of the likelihood of being real. By training these two networks against each other in tandem, each network helps to improve the other, with the generator being able to create realistic samples based on features gathered from real data, while the discriminator adapts to identifying progressively more realistic samples [19]-[22]. Upon

Manuscript received July 18, 2019; revised October 24, 2019.

S. Ezekiel is with the Computer Science Department, Indiana University of Pennsylvania, Indiana, PA USA (e-mail: sezekiel@iup.edu).

L. Pearlstein is with the Electrical & Computer Engineering Department, The College of New Jersey, Ewing, NJ USA.

A. Alshehri is with the Electrical Engineering Department, King Abdulaziz University, Jeddah, KSA.

doi: 10.18178/ijmlc.2019.9.6.872

completion of the training of the neural networks, the generator network can produce new synthetic samples that can be used to augment a smaller data set. By implementing a GAN, a data set that is too sparse to properly train a DCNN can be augmented with any number of pre-labeled synthetic images to accurately train it. VAEs, another method of generating synthetic images, are comprised of a machine learning model that receives and decompresses encoded data. When training a VAE, the raw signal features of input images are compressed and reduced to a lower dimensional latent space. Once compressed, the latent space is then sampled and decompressed into new samples [23]-[28]. By utilizing these generative architectures, synthetic data can be produced based on real-world data in order to supplement data sets.

This paper is organized as follows: Section II presents an overview of the concepts and technical background for synthetic data, Generative Adversarial Networks, Variational Autoencoders, and the image datasets. Section III provides the empirical framework for our analysis and underlying methodology which includes the modeling and training of both a GAN and VAE to train a DCNN. In Section IV, our results are displayed and discussed, including the accuracy of both methods. Each method is analyzed and compared against the other to determine the optimal technique. Section V focuses on the significance of our techniques for the creation of image sets as well as applications and areas of future research.

II. Technical Background

A. Deep Convolutional Neural Networks

Deep Convolutional Neural Networks (DCNNs) are learning models which are able to learn the aspects of their input in order to classify them into labels [1], [3]. They take advantage of a feed-forward architecture in which several layers are stacked onto each other, with the output of one layer being connected to the input of the next. The convolutional layer consists of filters which convolve over the entire input, computing an activation map which contains the output of each convolution. As the network progresses, the filters that activate when a feature is detected will be learned. Each convolutional layer contains a set of filters, each of which produce their own activation map which are stacked. The pooling layers are responsible for downsampling the size of the representation in order to reduce the number of parameters and prevent overfitting. The pooling layer also is used to map the location of features in an image relative to other features [29]. The pooling layer is typically used after a convolutional layer. The fully connected layer of a network is used for the actual classification in the network, the input of which is all the activation mappings of the previous layer which contain the features of the input. The network is first initialized with all of its weights and filters as random values. The network then takes an image dataset and performs a forward propagation step in which the probabilities for each class are computed. A summation of the error at the output layer over all the classes is then calculated. A backpropagation step is utilized with respect to the determined weights of the network in order to update the filter values and weights using gradient descent. The filter matrix and connection weights are adjusted, with the weights being updated in proportion to their error. This process optimizes the parameters of the network in order to classify images from the training set. In the 2012 ILSVRC, a DCNN known as AlexNet was submitted, which won the competition with an error of only 15.3%, which was over 10.8% lower than the runner up. AlexNet is utilized in this study for classification with the layers shown in Fig. 1a [30]. Another network architecture utilized are VGG16 and VGG19. Both were designed by the Visual Geometry Group and use a deeper layer architecture than AlexNet. The main difference between the two are the number of hidden layers, 16 and 19 respectively, shown in Fig. 1b. By adding more layers, the networks can extract more features from the input dataset at the cost of increased computational intensity and training time. Generally, these networks can provide higher classification accuracy than AlexNet.

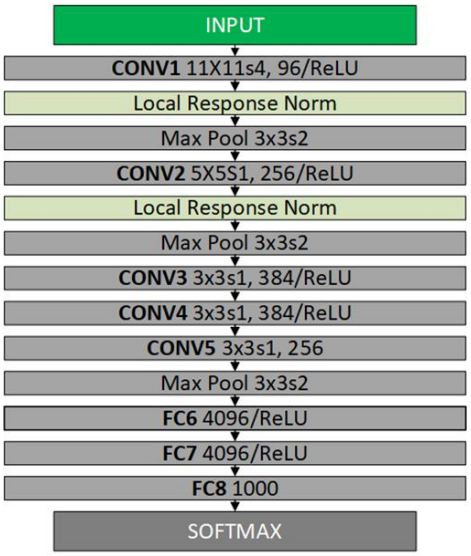

Fig. 1a. Topology of AlexNet.

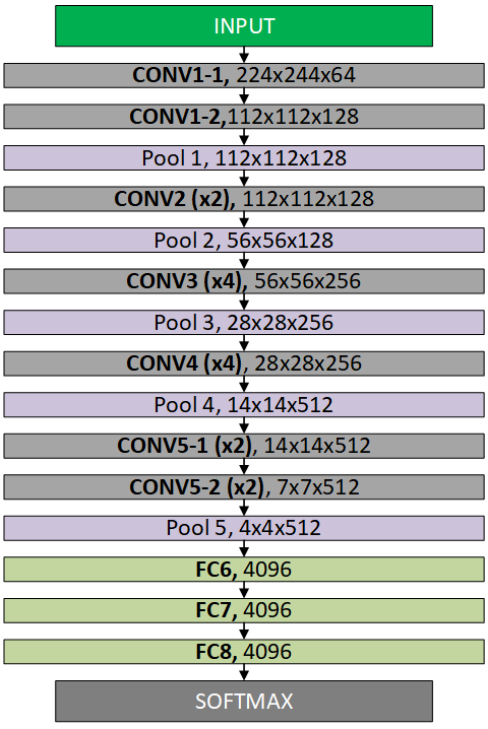

Fig. 1b. Topology of VGG19.

B. Data Augmentation

Algorithm 1

```
procedure AUGMENTDATA(dataset)
    for i in len(imgs) do
        rotatedImg(i) ← rotate(imgs(i))
        colorShiftedImg(i) ← COLORSHIFT(imgs(i)
        colorFilledImg(i) ← COLORFILL(imgs(i)
        blurredImg(i) ← BLUR(imgs(i)
        sharpenedImg(i) ← SHARPEN(imgs(i)
        brightnessAdjImg(i) ← ADJBRIGHTNESS(imgs(i)
        constrastAdjImg(i) ← ADJCONSTRAST(imgs(i)
    augmentedDataset ← rotatedImg + colorShiftedImg + colorFilled +
blurredImg + sharpenedImg + adjBrightImg + adjContrastImg
```

Fig. 2. Pickup truck images that have been sharpened, rotated, recolored, and blurred.

Many datasets do not contain the necessary amount nor variety of images to effectively train a model. One possible solution to this problem is through data augmentation techniques in which transformations are performed on the labeled images, such as rotating, translating, or flipping samples. Images can also be scaled inward or outward. In scaling an image outward, the resulting image will be larger than the original and is subsequently sampled with the size being equal to the original image. The original image can also be subsampled and resized to the original image size, allowing features to be emphasized. Translating the image vertically or horizontally ensures that the neural network is not trained to only look for objects in a single area of the image. Other adjustments such as contrast color shifting, and brightness can also be done, with an overall process shown in Algorithm 1. In doing this, the dataset can be artificially enlarged to better train the model. A well-designed CNN model is translationally invariant to an image so that objects can be recognized from different positions, angles, and lightings as shown in Fig. 2. Augmenting the dataset by performing transformations on the images allows the model to be trained with varying orientations that weren't originally available [31]. Real-world scenarios will often involve images in a dataset being captured under limited conditions that don't properly include objects of interest at all possible perspectives. This is accounted for by training the neural network with synthetically modified data. This modified data helps prevent the neural network from being over-fitted and from learning high frequency features that are not necessarily useful. High frequency features can also be distorted by applying Gaussian noise throughout the image. Because this can also distort low frequency features which might be of use, finding a proper balance of noise to apply to enhance the learning capabilities of the network is necessary. Adding salt and pepper noise is similar to the effect produced by Gaussian noise but to a lesser extent of distortion. Since real world data can exist in a variety of conditions that cannot be properly accounted for through simple transformations, more advanced augmentation techniques can be applied in order to simulate a multitude of conditions.

C. Synthetic Data

One of the most cumbersome aspects of gathering a suitable dataset is the manual capturing and labeling of data. Through the use of synthetic data, this issue can be largely mitigated by creating data that is automatically labeled [11]. This allows models to be trained to a high accuracy without the need to manually capture and label data. Often when using real-world datasets, it's impossible to capture them under ideal circumstances. In the case of outliers, which involve physical sensors being used in uncommon environments or capturing chance events, artificial scenarios can be designed in order to run simulations. Despite the benefits of using synthetic data, it still must be used in conjunction with real-world data for the validation purposes of ensuring the data accurately represents desired real-world scenarios. By supplementing real-world data with synthetic data, a generalized model can be created and calibrated with a smaller set of real-world data

D. Generative Adversarial Networks

Algorithm 2

```
procedure TRAINGAN
    procedure TRAINGENERATOR(augmentedDataset, initParams )
        trainingParams ← initParams
        while (fid >80) do
            procedure TRAININGOPS((augmentedDataset,trainingParams)):
                G_Img ← GENERATEIMG(augmentedDataset, trainingParams)
                loss ← TRAINDISCRIM(augmentedDataset, G_Img)
                G, D ← UPDATENETWORKS(loss)
                fid ← PERFORMMETRICS(G, D)
                if (curIteration mod 1000) = 0 then
                    checkpoint ← SAVECHECKPOINT(G, D)
```

Fig. 3. GAN generated vehicles.

Introduced in 2014 by Goodfellow, *et al*. [18], Generative Adversarial Networks utilize generative models, which, given a label, attempt to create new samples. Normally discriminative network models attempt to map a high dimensional input to a class. Generative models essentially perform the opposite where given a label they attempt to predict features. Whereas discriminative models map the relations between labels and features, generative models are concerned with the probability of a feature when provided a class and how these features are determined [32]. GANs are comprised of two neural networks working in tandem, a

generator and discriminator, summarized in Algorithm 2. The generator, maps a vector of pseudo-random input data to an image, shown in Fig. 3. The discriminator on the other hand is a binomial classifier which attempts to discriminate between real and artificially created images [33], [34].

The images produced by the generator are passed to the discriminator along with real images from the dataset. The discriminator then returns a probability of whether each image is real or fake. In doing this, a feedback loop is created in which the discriminator is use d to improve the capabilities of the generator to produce images closer to the naturally occurring images, and the discriminator is trained to differentiate between increasingly realistic synthetic images.

Algorithm 3

```
procedure TRAINVAE(dataset, hiddenSize)
    [trainData, testData] ← SPLITDATA(dataset, '0.6', 'randomized')
    while (i <maxEpochs) && (gradient >0) && (performance >0) do
        autoenc ← TRAINENCODER(trainData,hiddenSize,maxEpochs)
        encodedImgs ← ENCODER(trainData)
        decodedImgs ← DECODER(encodedImgs)
        performance ← RMSE(decodedImgs, trainData)
    reconstructed ← PREDICT(autoenc,testData)
```

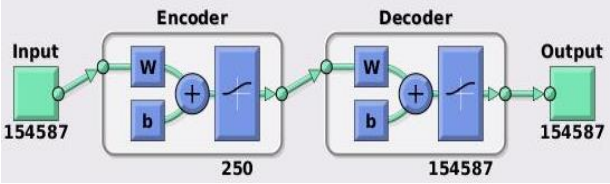

Fig. 4. Architecture of the auto encoder.

Fig. 5. VAE generated vehicle images, sampled from the latent space.

E. Variational Autoencoders

Autoencoders encode input data by reducing the dimensionality of images in order to lessen the effects of any noise present. Autoencoders are implemented using two neural networks, an encoder and a decoder, outlined in Algorithm 3. Autoencoders are specific to the data to which they are trained, as they are trained to attempt to copy input to its output. Due to the compression involved, the output of autoencoders will contain loss and will be degraded compared to its input [35]. The components of an autoencoder are an encoding function, decoding function, and a loss function between the information loss of the compressed representation and the decompressed representation, shown in Fig. 4. Using Stochastic Gradient Descent, the encoding and input into two parameters in a latent space. From the latent normal distribution, similar points that are meant to represent the data are randomly sampled. The decoder then maps the points in latent space to reconstruct the input data, shown from the synthetic data generated in Fig. 5. Two loss functions are used in order to train the parameters of the model. A reconstruction loss which are used to make the decoded sample match the initial input as well as a KL divergence between the prior distribution and the learned latent distribution, which normalizes the data [28].

III. Methodology

The general process followed is outlined in Fig. 6. Vehicles of several categories that would be suitable for training our models were handpicked to be added to our original image dataset. Namely, selecting images that were taken from similar perspectives, in order to assure unique features would be properly learned. After the images were collected, they were organized into their respective categories in order to be used to train a General Adversarial Network and Variational Autoencoder. First, the image set was resized to 227 by 227 in order to maintain consistency in resolutions and the aspect ratio needed to train the model. The resized images are then split based on their labels and used to train the autoencoder, using root mean-squared error as the loss function to validate the reconstructed images with the original set. The images are used as input for the encoder network, in which they are deconstructed into a compressed representation. The encoded representations of the images are then passed through the decoder network, in which the latent space is sampled and reconstructed back to their original sizes. After being fully trained, the latent space of the model can be randomly sampled in order to create an arbitrary number of images which are used to augment the original dataset. The Generative Adversarial Network used in this study is NVIDIA's StyleGAN, which is a TensorFlow based GAN that is able to separate and learn high-level features of images. The images are first resized to a power of two, 256 by 256, the closest resolution to the resolutions used by VGG and AlexNet. Next, data augmentation techniques such as rotation or recolor mapping are performed on each of the images in the dataset in order to increase the size of the training set. The images are then formatted to TFRecords, which serializes them into a binary format. Models are then trained for each type of vehicle, with the first model being trained initially using completely random noise. Subsequent models are trained using snapshots of previous models, which improves the learning rate by utilizing transfer learning. After both the VAE and GAN are trained, they are used to create three datasets in addition to the original, one with GAN generated images, one with images sampled from the VAE, and one with both. Each of the datasets are resized to be used as input for the three DCNNs, AlexNet, VGG16, and VGG19. The image sets are split into 60% training data and 40% test data to be used for validation. Feature vectors of each network are extracted from their respective FC7 layers. The extracted feature vectors are then heterogeneously fused in order to create a fused feature vector. The fused vector then has dimensionality reduction techniques such as PCA and t-SNE to create a representation that retains the most

significant features. The reduced feature vector is then validated against the test set, along with the individual networks' features using a Support Vector Machine. Each SVMs' classification accuracy is then determined for the four datasets.

Algorithm 4

```
procedure SYNTHETIC DATA AUGMENTATION(dataset)
    GANset ← TRAINGAN(originalImages, initParams)
    VAEset ← TRAINVAE(originalImages, hiddenSize, regularzation)
    combinedSet ← GANset + VAEset
    datasets ← [original, VAEset, GANset, combinedSet]
    net1 = AlexNet
    net2 = VGG16
    net3 = VGG19t
    [trainSet, testSet] ← SPLIT(datasets{i}, 0.6)
    for i = 1:len(datasets) do:
        procedure TRAINFEATEXTRACTION(trainSet, net1,2,3)
            trainFC7(A) ← EXTRACTFEATURES(net1, trainSet)
            trainFC7(V16) ← EXTRACTFEATURES(net2, trainSet)
            trainFC7(V19) ← EXTRACTFEATURES(net3, trainSet)
        procedure TESTFEATEXTRACTION(testSet, net1,2,3)
            testFC7(A) ← EXTRACTFEATURES(net1, testSet)
            testFC7(V16) ← EXTRACTFEATURES(net2, testSet)
            testFC7(V19) ← EXTRACTFEATURES(net3, testSet)
        fusedTrainFeatures ← MAX,MIN,SUM,MEAN(trainFC7)
        fusedTestFeatures ← MAX,MIN,SUM,MEAN(testFC7)
        reducedTrainFeatures ← PCA,T-SNE(fusedTrainFeatures)
        reducedTestFeatures ← PCA,T-SNE(fusedTestFeatures)
        procedure CLASSIFYIMAGES(reducedTrainFeatures,reducedTestFeatures)
            classifier ← reducedTrainFeatures, labels
            predY ← PREDICT(classifier, testFC7(A))
            acc = MEAN(predY, testSet.labels)
            predY ← PREDICT(classifier, testFC7(V16))
            acc = MEAN(predY, testSet.labels)
            predY ← PREDICT(classifier, testFC7(V19))
            acc = MEAN(predY, testSet.labels)
            predY ← PREDICT(classifier, reducedTestFeatures)
```

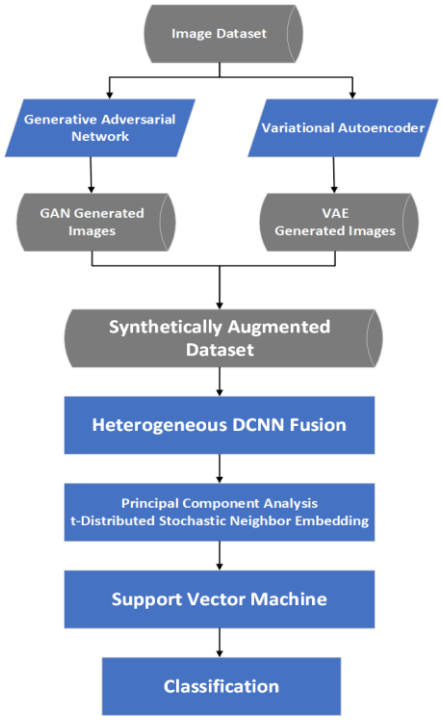

Fig. 6. Methodology.

IV. Results

Trials were performed on two overarching datasets, one containing the three categories for which synthetic images were generated and another containing ten categories, including categories that synthetic data was not generated for, in order to see the effect of the synthetic data on untouched labels. The accuracy of the three individual networks are then compared to each of the four fusion methods, sum, min, max, and average. It should be noted that each of the test sets only have the original images to validate the classification accuracy of only real images and not synthetic. None of the original images in the test set were in the training set, ensuring that none of the networks were trained using images that would be used for validation. The results for each of the trials performed on the smaller dataset are shown in Tables 1-4. Overall, performing heterogeneous DCNN fusion had more consistent high accuracy results for both the augmented and original datasets. The dataset that was augmented with only the VAE images appear to have better performance than the non-augmented dataset using heterogeneous DCNN fusion. From the data, on average the datasets containing GAN augmented images appear to have slightly worse performance overall, with the combined and GAN augmented only datasets having comparable performances, shown in Fig. 7. In regard to the non-augmented dataset, all methods of heterogeneously fusing the feature vectors of the networks outperformed all of the individual networks on their own. The best performing on average for both the individual neural networks as well as fusion methods were direct concatenation and the average of the features. The worst performing fusion method was finding the minimum, however even this method on average had a higher classification accuracy than the best performing individual network.

TABLE I: Original Images, No Synthetic

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.73333	0.91667	0.8416	0.9666	0.9583	0.9083	0.9666
2	0.9	0.91667	0.875	0.9833	0.95	0.925	0.9833
3	0.88333	0.94167	0.9	0.9916	0.975	0.975	0.9916
4	0.81667	0.94167	0.8416	.99	0.9416	0.8666	.99
5	0.76667	0.94167	0.8416	0.9833	0.95	0.9	0.9833
6	0.68333	0.94167	0.8166	0.9916	0.9333	0.8083	0.9916
7	0.75833	0.86667	0.75	0.9666	0.9416	0.95	0.9666

TABLE II: VAE Augmented Dataset

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.78889	0.95278	0.93889	1	0.99722	0.88333	1
2	0.86667	0.93333	0.86944	1	0.99722	0.98056	1
3	0.86111	0.94444	0.89167	1	0.98056	0.94444	1
4	0.87222	0.96111	0.9	1	0.99167	0.93889	1
5	0.875	0.96111	0.89444	1	1	0.96944	1
6	0.88333	0.96111	0.89444	1	0.98889	0.95833	1
7	0.80278	0.94167	0.91944	1	0.99444	0.91389	1

Training the networks with the VAE augmented dataset appears to have produced the best results, with both the individual networks and heterogeneous DCNN fusion methods having higher classification accuracies than the non-augmented dataset. As with the non-augmented dataset, sum and average had the best performance. Due to the computational intensity of both the feature extraction and training of the VAE, the VAE augmented dataset was the smallest of the augmented datasets. Because of the size of the dataset, the classification accuracy is likely abnormally high. By increasing the number of images in both the training and testing sets, a more accurate performance can be measured.

The GAN augmented dataset had lower classification accuracy than the other sets, however it should be noted that

the fused features still had accuracies that were comparable to the individual AlexNet of the un-augmented dataset. The cause of the lower accuracy is likely due to overfitting of the networks during their training, causing it to be trained too closely to the training set, and thus have lower accuracy when presented with previously unseen images.

TABLE III: GAN AUGMENTED DATASET

	Alexnet	VGG 16	VGG 19	Sum	Max	Min	Avg
1	0.7453	0.76137	0.70679	0.81079	0.7832	0.79745	0.81079
2	0.72741	0.74166	0.70255	0.76713	0.80594	0.70497	0.76683
3	0.73469	0.75743	0.68557	0.8117	0.80594	0.6607	0.8117
4	0.73438	0.76774	0.745	0.80837	0.80807	0.78775	0.80837
5	0.76653	0.77047	0.73014	0.80958	0.80534	0.77987	0.80958
6	0.73802	0.73772	0.68921	0.77805	0.80291	0.7268	0.77805
7	0.74833	0.7359	0.65888	0.80443	0.80685	0.76198	0.80412

TABLE IV: COMBINED DATASET

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.7709	0.7576	0.7223	0.8203	0.8171	0.7545	0.8203
2	0.7501	0.768	0.7516	0.8183	0.8183	0.8031	0.8183
3	0.7522	0.7493	0.7254	0.8186	0.8146	0.8082	0.8186
4	0.7182	0.721	0.6912	0.7441	0.816	0.6464	0.7441
5	0.7432	0.7513	0.7602	0.8169	0.8166	0.7726	0.8169
6	0.7815	0.7504	0.7110	0.8189	0.8194	0.7867	0.8189
7	0.7547	0.703	0.7053	0.77	0.7924	0.7455	0.77

The combined dataset also had worse performance than the VAE augmented dataset, again likely due to overfitting occurring during the training step. The combined dataset appears to have slightly higher accuracy than the GAN only augmented dataset, meaning that despite overfitting of the networks, adding the VAE images resulted in slightly higher performance. Averaging each of the trials found that the VAE augmented dataset had the highest performance, followed by the original non-augmented dataset, shown in Fig. 7. The combined and GAN only datasets had comparable results, however the combined dataset appears to have slightly better performance.

TABLE V: ORIGINAL IMAGES, NO SYNTHETIC

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.7106	0.85	0.8455	0.9924	0.9636	0.903	0.9924
2	0.6833	0.85909	0.8424	0.9894	0.9818	0.9121	0.9894
3	0.6333	0.86667	0.8712	0.9894	0.9652	0.9364	0.9894
4	0.6530	0.83182	0.8394	0.9727	0.9439	0.8879	0.9727
5	0.6561	0.8697	0.8561	0.9955	0.9652	0.8849	0.9955
6	0.6364	0.85152	0.8394	0.9924	0.9303	0.8955	0.9924
7	0.6546	0.83485	0.8636	0.9909	0.9697	0.8803	0.9909

In order to see the effect synthetic data would have on the classification accuracy of categories without any synthetic data in them, trials were performed on a dataset containing the three augmented categories as well as seven categories that have not had any data augmentation performed on them. Seven trials were performed on each of the datasets under the same conditions as the smaller set, with all test images removed from the training set in order to ensure none of the images are seen during testing. Overall, as with the smaller set, the VAE augmented dataset performed best on average, followed by the non-augmented set, however the difference is less than that of the smaller dataset, indicating the methods were about the same. The combined and GAN augmented datasets however, performed worse than with the smaller dataset, shown in Table V-Table VIII. Additionally, even with heterogeneous DCNN fusion, the datasets performed on average worse than the individual networks, shown in Fig. 8. A possible explanation for this is due to the model being verfitted, as the datasets with the GAN images contained more images, which would cause the model to learn the details of the training set too closely and as a result fails to accurately classify previously unseen images. As with the smaller dataset, performing any method of heterogeneous DCNN fusion results in a higher accuracy than any of the individual networks alone. Additionally, the classification accuracy of the individual networks was slightly worse with the dataset compared to the individual networks of the smaller dataset.

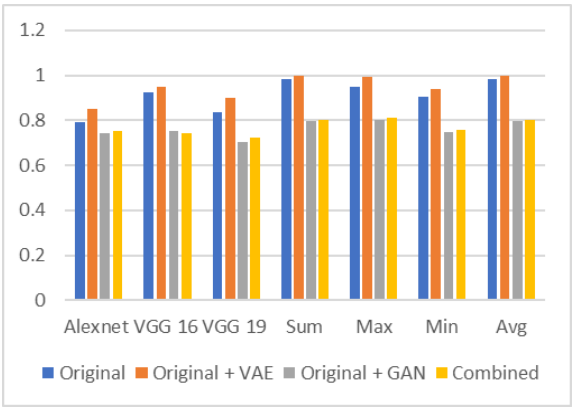

Fig. 7. Average classification accuracy.

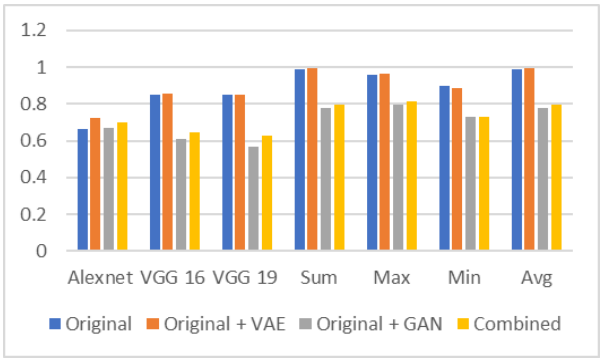

Fig 8. Average classification accuracy.

The VAE augmented dataset was found to have the same if not slightly better performance than the non-augmented dataset. Additionally, although the accuracy of both sets was similar for the fused features, the VAE augmented dataset had higher classification accuracy using the individual networks, especially in the case of AlexNet. Augmenting the dataset with synthetically generated VAE images appears to have a beneficial effect even when classifying images that did not have any synthetic data generated.

As with the smaller dataset, the GAN augmented dataset had worse performance than the VAE dataset, likely due to overfitting of the model. However, the sum, average, and max fusion methods appear to have had better performance than the individual AlexNet when compared to the non-augmented dataset.

The combined dataset was found to have had better performance than the GAN only dataset, indicating that

despite the model being overfit, the inclusion of VAE images had a positive effect regarding classification.

TABLE VI: VAE AUGMENTED DATASET

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.7691	0.812	0.8452	0.9893	0.9631	0.9167	0.9893
2	0.7191	0.8536	0.8571	0.9952	0.9536	0.8452	0.9952
3	0.7321	0.8417	0.8476	0.9952	0.962	0.9357	0.9952
4	0.6905	0.8643	0.8488	0.9905	0.9583	0.8274	0.9905
5	0.737	0.8964	0.8512	0.9941	0.9607	0.8929	0.9941
6	0.6976	0.8655	0.8607	0.9917	0.9714	0.8691	0.9917
7	0.7298	0.8441	0.8452	0.9905	0.9643	0.9107	0.9905

TABLE VII: GAN AUGMENTED DATASET

	AlexNet	VGG 16	VGG 19	Sum	Max	Min	Avg
1	0.7247	0.6531	0.5809	0.8278	0.7901	0.773	0.8278
2	0.6189	0.6074	0.5383	0.7236	0.7545	0.7009	0.7236
3	0.7176	0.6446	0.6017	0.8267	0.8207	0.8095	0.8267
4	0.6903	0.6296	0.5249	0.7447	0.8144	0.6318	0.7447
5	0.7307	0.6115	0.613	0.8261	0.8185	0.81001	0.8261
6	0.5492	0.5599	0.5607	0.7449	0.7717	0.6985	0.7449
7	0.6525	0.5705	0.5394	0.7362	0.7868	0.706	0.7362

TABLE VIII: COMBINED DATASET

	AlexNet	VGG16	VGG19	Sum	Max	Min	Avg
1	0.7311	0.6305	0.6076	0.7843	0.8288	0.6923	0.7843
2	0.6701	0.6691	0.6144	0.7718	0.8278	0.7236	0.7718
3	0.6949	0.6199	0.6123	0.771	0.7911	0.7199	0.771
4	0.6985	0.6079	0.6191	0.809	0.8299	0.7348	0.8088
5	0.7374	0.6488	0.6186	0.8056	0.8158	0.746	0.8056
6	0.6694	0.6509	0.6459	0.83	0.8082	0.758	0.8296
7	0.7038	0.6860	0.6571	0.7978	0.8121	0.7512	0.7978

V. CONCLUSIONS AND FUTURE WORK

Based on the results, supplementing existing datasets with synthetically created images shows promise in increasing the classification accuracy of neural networks and support vector machines. The synthetic images created by the variational autoencoder appear to have had the best performance in the case of the smaller dataset, outperforming the original image set across all classification methods. In the case of the larger dataset, which only contained synthetic images for three categories, the VAE augmented dataset in most cases performed on par or better than the non-augmented set, especially in the case of the individual AlexNet. A possible explanation for the better performance is that the VAE was able to map the latent features of the images and could reconstruct a compressed approximation which was comprised of these features. By fine tuning the parameters of the VAE, it is reasonably possible that the classification accuracy can be further increased. A well-tuned VAE which can better compress the features into a lower dimensional space would create images which are less influenced by background noises or other factors which that would otherwise interfere with the training. Although the GAN and combined datasets usually had worse performance on average than the non-augmented dataset, it doesn't necessarily mean that their use is impractical. A likely cause of the lowered performance is overfitting of the network. Because the GAN generated output which is meant to replicate its input, coupled with the fact that the number of GAN generated images outnumbered the number of VAE images in the dataset, meant that the network was trained too closely to its input and was unable to classify unseen images. Additionally, although the combined set on average had worse performance than the non-augmented set, there were cases in which performing heterogeneous fusion resulted in better performance accuracy of the GAN and combined sets than the individual networks.

Our future work in this area will include further fine tuning both the parameters of the GAN and the VAE in order to create representations of images which contain better approximations of the features of their input, as well as in order to create entirely new images that would not cause the models to become overfit.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

A. Lutz conducted the research and wrote the paper, J. Zaunegger assisted with research, S. Ezekiel and W. Farag analyzed the data, A. Alshehri and L. Pearlstein assisted with the final draft; all authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to extend their gratitude to Indiana University of Pennsylvania's students David Kornish for the integration of the MATLAB code and work on the project.

REFERENCES

[1] A. Torralba, K. Murphy, W. Freeman, and M. Rubin, "Context based vision systems for place and object recognition," in *Proc. ICCV*, 2003.
[2] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv:1605.07678*, 2016.
[3] R. Waseem, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29.
[4] V.-D. Hoang, "Deep CNN and data augmentation for skin lesion classification," *Intelligent Information and Database Systems*.
[5] S. Justin, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 2, issue 3, 2017.
[6] W. Jason, *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*, Stanford University, 2018.
[7] D. Jack, "Fake it 'Till you make it: Synthetic datasets assisting machine learning in data scarce environments," *Intel Artificial Intelligence*, 2018.
[8] L. Pearlstein, M. Kim, and W. Seto, "Convolutional neural network application to plant detection, based on synthetic imagery," in *Proc. 2016 IEEE Applied Imagery Pattern Recognition Workshop*, October 2016, pp. 1-4.
[9] M. Nikolaus, "What makes good synthetic training data for learning disparity and optical flow estimation," *IJCV*, 2018.
[10] B. J. Pablo, "Synthetic data for open and reproducible methodological research in social sciences and official statistics," *AStA Wirtsch Sozialstat Arch.*, 2017.
[11] W. Jason, *The Effectiveness of Data Augmentation in Image Classification Using Deep Learning*, Stanford University, 2018.
[12] T. Jonathan, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. CVPR*, 2018.
[13] M. Nikolaus, "What makes good synthetic training data for learning disparity and optical flow estimation," *IJCV*.
[14] J. Goldberger, H. Greenspan *et al.*, "Synthetic data augmentation using GAN for improved liver lesion classification," in *Proc. 2018 IEEE 15th International Symposium on Biomedical Imaging*, pp. 289-293.
[15] S. Justin, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 2, issue 3, 2017.
[16] G. Rohith, "Generative adversarial networks-explained," *Towards Data Science*, 2018
[17] *Generating Large Images from Latent Vectors*, Otoro.net.2016.
[18] I. J. Goodfellow, "Generative adversarial nets," in *Proc. Neural Information Processing Systems Conference*, 2014.

[19] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *arXiv:1812.04948*, 2018.
[20] S. Reed, Z. Akata, X. Yan, *et al*., "Generative adversarial text to image synthesis," *arXiv:1605.05396*, 2016.
[21] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *Proc. European Conference on Computer Vision*, Springer, Cham, October 2016, pp. 318-335.
[22] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *arXiv:1602.05110*, 2016.
[23] Y. C. Pu, "Variational autoencoder for deep learning of images, labels and captions," in *Proc. Neural Information Processing Systems Conference.*
[24] K. Diederik, *Auto-Encoding Variational Bayes. Computing Research Repository*, 2014.
[25] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.
[26] F. Kevin, *Variational Autoencoders Explained.kvfrans.com.*, 2016
[27] L. Anders, "Autoencoding beyond pixels using a learned similarity metric," *Computing Research Repository*, 2016.
[28] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.
[29] V. Vukotić, C. Raymond, and G. Gravier, "Multimodal and crossmodal representation learning from textual and visual features with bidirectional deep neural networks for video hyperlinking," in *Proc. the 2016 ACM workshop on Vision and Language Integration Meets Multimedia Fusion*, October 2016, pp. 37-44.
[30] K. Alex, S. I. Hinton, and E. Geoffrey, *ImageNet Classification with Deep Convolutional Neural Networks*, 2010.
[31] V.-D. Hoang, "Deep CNN and data augmentation for skin lesion classification," *Intelligent Information and Database Systems*, pp. 573-582, 2018.
[32] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
[33] A. Dosovitskiy, and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," *Advances in Neural Information Processing Systems*, pp. 658-666, 2016.
[34] A. Shrivastava, T. Pfister, O. Tuzel *et al*., "Learning from simulated and unsupervised images through adversarial training," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2107-2116.
[35] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

Soundararajan Ezekiel received his M.A and PhD degrees from the Department of Mathematics, University of Pittsburgh, Pittsburgh, PA USA. He also received his MSc degree in mathematics at Loyola College, post graduate diploma in operations research at Anna University, and his M. Phil at Madras Christian College, India.

He is currently a professor in computer science at Indiana University of Pennsylvania, PA.

Professor Ezekiel is the recipient of three-time SFFP fellow and seven-time VFRP fellow. His research includes image processing, signal processing, wavelet analysis, artificial intelligence, machine vision, deep learning, and cyber security.

Larry Pearlstein received the BSEE from Drexel University in 1982, and the Ph.D. degree in electrical engineering from Princeton University in 1987. He served as the chairperson of the Video Specialists Group within the Advanced Television Systems Committee (ATSC) and led the effort to document the video compression portion of the ATSC Digital Television Standard. While working for ATI, AMD, and Broadcom, he architected video subsystems for consumer electronics chips. He holds 71 US patents in the areas of digital television and consumer electronic. He is currently an associate professor of electrical and computer engineering at The College of New Jersey and conducts research in the areas of deep learning and connected devices.

Abdullah A. Alshehri received his B.S. in 1993 in electrical engineering from University of Detroit, Detroit, MI. USA. He received his M.S. and Ph.D. in electrical engineering from University of Pittsburgh, PA in 1999 and 2004 respectively. From 2005 to 2010 he worked as an assistant professor in the College of Telecom and Electronics CTE and Jeddah College of Technology JCT, Saudi Arabia. In December 2010, he joined the Electrical Engineering Department at King Abdulaziz University-Rabigh KAU, Saudi Arabia and is now an associate professor. His areas of interests are in advanced signal and image processing such as time-frequency, wavelet transform, neural networks, and statistical signal processing. Dr. Alshehri is a member of IEEE since 1992 and a member of the Saudi Engineers Council SEC since 2005.