Generative adversarial networks for increasing the veracity of big data

Conference Paper · December 2017

DOI: 10.1109/BigData.2017.8258219

2 authors, including:

Conrad Tucker
Pennsylvania State University
96 PUBLICATIONS **493** CITATIONS

SEE PROFILE

Generative Adversarial Networks for Increasing the Veracity of Big Data

Matthew L. Dering
Computer Science and Engineering
Penn State University
University Park, Pennsylvania 16802
Email: mld284@cse.psu.edu

Conrad S. Tucker
Engineering Design and Industrial Engineering
Penn State University
University Park, Pennsylvania 16802
Email: ctucker4@psu.edu

***Abstract*—This work describes how automated data generation integrates in a big data pipeline. A lack of veracity in big data can cause models that are inaccurate, or biased by trends in the training data. This can lead to issues as a pipeline matures that are difficult to overcome. This work describes the use of a Generative Adversarial Network to generate sketch data, such as those that might be used in a human verification task. These generated sketches are verified as recognizable using a crowdsourcing methodology, and finds that the generated sketches were correctly recognized 43.8% of the time, in contrast to human drawn sketches which were 87.7% accurate. This method is scalable and can be used to generate realistic data in many domains and bootstrap a dataset used for training a model prior to deployment.**

***Keywords*-Generative Models, Big Data, Deep Learning, GANs, Sketches**

I. Introduction

The utility of automatic, realistic, synthetic data generation in big data problems has been demonstrated in both Velocity [1] and Variety [2]. Generative models are also frequently employed in big data settings for capturing trends within data [3], [4]. A generative model is one which models data as a distribution, or combination of distributions, which can then be sampled. In some cases, this ability to be sampled is a byproduct of the technique used [5]. In other cases, sampling this distribution is the goal [6], as obtaining new or unique data is critical to many applications.

Typically big data problems apply a filter or transform to reduce the size of their data, and gain insights, or other benefits. While finding the signal in the noise has many benefits, this work considers the benefits of generating realistic data with known characteristics prior to implementing a big data system. This method will have benefits in bootstrapping as well. For instance, a company that wishes to put in place a robust "CAPTCHA" style human verification system might use such a model to generate a large number of images. These generated images could be used to either fool non-human actors, or could conceivably be produced by a human actor and thus used for training. This generated data would be based on a much smaller amount of data, and thus could be more secure from the beginning, and less prone to bias in the training set.

For this reason, the authors of this work describe how generating large amounts of human quality data could be used for training a big data platform before deployment. These ideas have been incorporated in the past for building more diverse models [7], but it is becoming increasingly important as model sizes grow. Adversarial training of models has shown to be helpful [8], [9], [10] in providing a model that is more robust to error. Big data settings have a vested interest in handling edge cases in predictable ways. By incorporating generated data, these edge cases can be predicted ahead of time. This approach is valuable in assuring the *veracity* of data, which has long proved to be a substantial challenge when drawing inferences from high dimensional data such as text [11] or images [12].

The main contributions of this work are as follows:

- A Generative Adversarial Network (GAN) is proposed which can quickly generate human-readable visual images objects.
- These images have use in both training of a big data model, pre-deployment, and in big data decision making, post-deployment.

II. Related Work

A. Generated Big Data

Generated data has reached the field of big data in unusual ways. Some work has examined how to integrate generating realistic data into big data pipelines, so that the pipelines can be tested for load, accuracy, or other metrics without using random data [1]. Recent work has explored generated data as an attack vector [13]. Like other attack vectors, an understanding of the problem can lead to increased security [14], [15]. Even in big data situations where security is not critical, these insecure outcomes are due to systems which have not been properly trained.

It is important to contrast this work with other synthetic data techniques as well. Randomly generated data may be appropriate in some settings, but this data can quickly become unfeasible or unreasonable once the dimensionality of the data grows (for instance, text or image data). Synthetic data generation techniques focus on solving class imbalance problems, with approaches such as SMOTE [16]. More recent techniques are driven by sampling and synthesizing from the training data as well [17], [18]. These ideas are helpful when

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

dealing with lower dimensional data or a class imbalance problem, but suffer from the same problems as random data when considering high dimensional data.

This work proposes to use generated data in a big data pipeline during training, so that the models are resistant to inaccuracies due to new data. This work proposes that by generating new novel data like that which the model may be exposed to in the real world, while still understanding what training outcomes are desired for this generated data, the resulting model is more robust to varying levels of data veracity.

B. Generative Models

There are many types of generative models. As stated above, generative models are simply models which can be sampled and used to generate new data. For instance, Latent Dirichlet Allocation [19], a tool popular for topic modeling in text which is commonly used in social networks [20], [21]. Sampling from these can lead to a representation of topic words within a text corpus. Gaussian Mixture Models are a type of model which assumes that the data is a linear combination of several Gaussian distributions in space. Sampling from these distributions is often used as a method of clustering, where the center of the distribution is treated as the centroid of the cluster. These methods are very common for modeling and understanding numerical data [22].

The advent of deep learning, a type of machine learning that involves stacked non-linear combinations of neurons, has brought about new types of generative models. While not strictly generative, these types of model often contain millions of parameters, and convincing results have been obtained by simply optimizing the input to generate a desired output [23]. More explicitly, a type of network called a Recurrent Neural Network can be used to generate text [24], caption images [25], generate images [26], and even generate images from captions [27], [28]. However, because of their high quality outputs, the most popular generative models are Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). Both model types work by discovering a lower dimensional representation of a high dimensional data type (for instance, images). A VAE works by encoding an input into a low dimensional vector of zero mean and unit variance. This vector is then decoded back into the original image, and by generating random vectors of zero mean and unit variance, new data can be produced. On the other hand, a GAN challenges a network (the discriminator) to determine if a given input is generated by a neural network (the generator), or drawn from the dataset. As with the VAE, the input to this generator is also a random vector with zero mean and unit variance. The work presented here employs a GAN which is augmented by a class prior as well as a random prior.

As shown above, there are many types of generative models, suited to many purposes. This work considers how automated image generation would provide value in training big data pipelines, as well as bootstrapping small datasets.

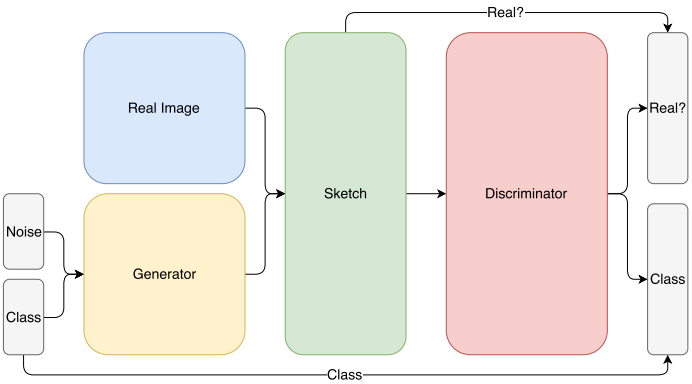

Fig. 1: An outline of the GAN in this work

III. Method

This section is organized as follows: Section III-A outlines the data used by this work. Section III-B describes the class of network used. Section III-C outlines the training process and Section III-D puts forth how the results are evaluated.

A. Data description

The data considered in this work is a set of labeled sketches of various objects. Image data consists of a dense matrix of pixel values, and serves as partial input data for the Discriminator network. For example, an image of $h \times w$ pixels and n color channels is represented as a $h \times w \times n$ matrix. Since this method concerns black and white sketches, in this case $n = 1$, though this method could generalize to RGB data, in which case $n = 3$ (one channel each for Red Green and Blue). A sample sketch from the dataset can be seen in Fig. 2a, and how a small portion of a sketch can be viewed as pixels can be seen in Fig. 2b. Since sketches are a simplified image modality, the values can be represented by a binary value so that the value at a given coordinate is 0 if that pixel is colored in, and 1 otherwise. This methodology assumes sketches are of a fixed size.

Hand drawn sketch data differs from color image modality in several ways. First, they are sparse in content. In contrast to a photograph, where even the content other than the subject (i.e. the background, or context), is meaningful and important. Sketches are changes (that is, marks) made by a designer given a blank space. Second, there is an expectation of continuity of these marks. Generally, sketches are made using long continuous strokes, rather than short small ones, and very rarely contain detailing such as dots. Finally, the choice of adding detail to sketches is not made lightly by the artist, and generally is only made when semantically meaningful. For instance, consider the object *tennis racket*. Without the laces running across the frame, it would be difficult to determine the difference between a racket and a spoon.

Each sketch also has a class label (such as *cup*, or *airplane*), represented by a binary vector c in length, where c is the number of classes in the dataset. This vector is all zeroes except in the position corresponding the given class.

(a) A sketch of an axe from the dataset.

1	1	1	1	1	1
1	1	0	0	0	1
1	1	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1

(b) A $6 \times 6 \times 1$ subset of the right top edge of the ax blade.

Fig. 2: Image modalities considered in this method.

B. Generative Adversarial Network

A GAN has two main components: The generator G and the discriminator D. G is a function which maps a class C, and a random latent variable Z to a sketch S ($G : \mathbb{R}^C \times \mathbb{R}^Z \rightarrow \mathbb{R}^S$). Z is intended as a random prior, sampled uniformly from a unit norm space across z dimensions. While other distributions may be used, the purpose of this is to learn to map a uniform low dimensional vector to a range of images, so using non-uniform distributions would result in lower diversity of output. Similarly, D is a function which maps a sketch S to a class C and a binary variable, which is 1 if the sketch and photos match, and 0 otherwise ($D : \mathbb{R}^S \rightarrow \{0, 1\} \times \mathbb{R}^C$).

Neural networks are layers of simple mathematical units called *neurons*. A neuron has three main components: weights, biases, and activation function. Typically weights and biases are learned, while the activation function is fixed. The operation performed by these neurons can be seen in Equation 1:

$$y = f(w \times x + b) \tag{1}$$

where y is the output of a neuron, f is the activation function, w are the weights, x is the input, and b is the bias. Each network layer is distinguished by type as well, which primarily changes what is considered input to the neurons in a given layer. For instance, for a fully connected layer, the input x will be every neuron of the previous layer. As such, these layers have high memory demands, since each of these inputs requires a learned weight. In the instance of a convolution layer, the input is a small locally adjacent region of the previous layer close to the location of the neuron in the new layer.

The primary activation function used in this network is the leaky Rectified Linear Unit (LeakyReLU), which is given by equation 2:

$$f(x) = \begin{cases} x & x > 0 \\ \alpha \times x & x \leq 0 \end{cases} \tag{2}$$

where x is the output of the neuron prior to activation, and α is a set parameter, generally less than 1 (many works suggest values of 0.1 or 0.2). This operation adds non-linearity to the network, by allowing positive gradients to pass through to the following layer, while squashing negative gradients without discarding them completely (as in the case of ReLU, where $\alpha = 0$).

1) Generative Network: Generative networks have a diverse set of designs, but briefly each will learn to generate output based on some prior input. Some instances use random noise as their only prior, others may use metadata such as object class or textual description. The network proposed in this work uses as a prior a random vector which represents the features of a sketch of the object, and a class vector.

It is important to note the exact nature of the task performed by this network. This network must learn to draw convincing sketches by searching the input space, given only random noise and a class vector, using only a binary variable and a class prediction as feedback. By convincing, this means it must generate output which is confused by a trained network with samples drawn from the training set. The structure of the network, similar to the reference network proposed by Odena et al˙ [29] can be seen in Fig. 3.

Because of the nature of sketches discussed in Section III-A, it is important that this generator network be able to accommodate these concerns. Possible issues include overfitting due to an overly large network (especially problematic in the case of the discriminator network). Since these networks are very large, and the target modality is simple in nature (the dataset in the case study reports a median number of strokes of 14), the risk of overfitting and failing to generate realistic output is significant.

Another one of the key aspects of this work is the use of *embeddings* for class identification. Embeddings are maps of data in high dimensional space, often used for text information. Using these embeddings, classes can be represented not as one-hot vectors (i.e. this is a cup), but by some learned vector in space. For example, in a two dimensional space, a hammer may be located at $(-1, 1)$, while a cup may be at $(1, -1)$. Since these locations are learned by the network, these locations have meaning. For example, an axe is similar to a hammer (e.g. both have handles and heads), so it may be closer to the hammer than the cup in space. This has a few advantages. For one, this allows these vector locations to be changed to explore the design space *in between* classes (i.e. 90% cup, 10% hammer, 80% cup, 20% hammer and so on). Next, these embeddings can be fuzzed, or manipulated slightly, as a form of data augmentation, such as in [28].

2) Discriminator Network: The other half of a GAN is the discriminator. In the game-like arrangement that constitutes a GAN, the discriminator's role is to learn the difference between generated input and real input. This network will also learn to classify input into a class label (e.g. cup, car).

This network consists of three parts. The first part is a feature extractor of the image, which decomposes the image into a vector of fixed length based on its contents. Using this vector as input, the second part makes a decision about if the generated input is generated or a member of the dataset. The final portion uses this same vector to make a classification of the input into an object type. More detail can be seen in Fig. 4.

The goal of the discriminator is not to perfectly learn the

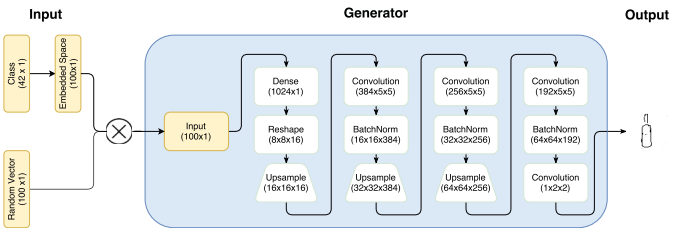

Fig. 3: Generator Network architecture. After each batch Normalization there is a ReLU. The sizes shown below each unit represent output size, except for the convolutions which represent kernel size and channel count.

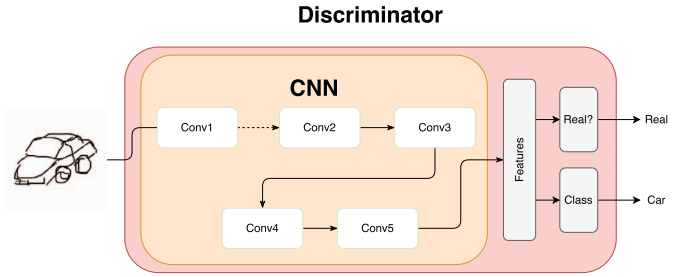

Fig. 4: Discriminator architecture. Orange denotes the CNN. White boxes in the CNN denote Convolutional Layers. Solid lines denote LeakyReLU with Batch Normalization. Dotted lines denote LeakyReLU, no Batch Normalization. Grey boxes denote vectors.

difference between data from the dataset and generated data. Instead, its primary function is to provide a differentiable loss which can be used to tune the weights of the generator. While having a highly accurate or fully featured discriminator may prove beneficial, the current state of the art emphasizes that they avoid overfitting instead. As discussed in Section III-B1, the sketch data modality presents unique challenges. The output of the discriminator has no practical benefit to this method, apart from defining a loss space which the generator explores via the generated input. The primary concern is finding a balance in network size between large networks and small networks. If the network is too large, it may overfit and learn the entirety of the dataset. This prevents the generator from ever "winning", which will slow or even prevent the generator from training. Conversely, if the network is too small, the generator may be able to fool the discriminator too easily, and hinder its ability to create convincing output.

C. Training And Evaluation

The training process for an Auxiliary Classifier GAN (AC-GAN) involves training the networks both separately and together. An overview of this can be seen in Algorithm 1. For each batch of input, the generator will generate a random sketch of a random class. This, along with input drawn from the dataset, is used to train the discriminator, which calculates a probability of being drawn from the dataset, as well as a class prediction.

Algorithm 1: Training Process

Input: Sketches S and associated classes C
for *batch* $\mathsf{b} \in \mathsf{S}$ **do**
  Train Generator;
  Generate fake data with generator;
  Train discriminator on real and generated data;
end

One active area of research for these types of networks is addressing their lack of convergence. Intuitively, as the discriminator becomes less likely to be fooled, the generator must become more adept at fooling it. From this arrangement, it is unclear at what point a model can be considered converged and training should cease. In practice, these models oscillate, with the generator occasionally outperforming the discriminator, and the discriminator occasionally outperforming the generator. This is a drawback of a GAN based approach instead of a VAE or FVBN. In this work, the network is trained until the discriminator is able to accurately recognize object classes, as this is a metric which both the generator and discriminator wish to minimize. In determining the convergence point, the classification loss is also considered, as this is a goal of both networks and is not adversarial in nature.

Neural networks are prone to learning very high activations and probabilities, especially in adversarial arrangements [30]. Hence, this network is trained using *One Sided Smoothing*, where the target prediction for real examples is a value less than 1, rather than 1. In this case, the value 0.9 was chosen, as suggested in [30].

D. Evaluation

This network is evaluated by human survey participants who are asked to categorize random objects shown to them drawn by the generator network. Additionally, the ability of this generator to generate new concepts will be proven by generating imputed classes.

The primary evaluation of the generator's ability to generate convincing sketches is made using a survey. This survey was given to participants on Amazon Mechanical Turk, who were compensated monetarily. They are asked to pick from 5 random classes (one of which is correct) as well as a "Do not know/none of the above" option. Importantly, the images selected for this survey must satisfy two conditions: they must be correctly classified by the discriminator, and the discriminator must incorrectly believe that they are genuine images. Since the network uses two priors, a class and random noise, the random noise space is imputed while class is held constant. This is repeated for each class, with the same 20 randomly sampled noise priors for each of the classes.

IV. Case Study

A. Data

A case study is presented in order to evaluate the effectiveness of this method. Sketches are drawn from the

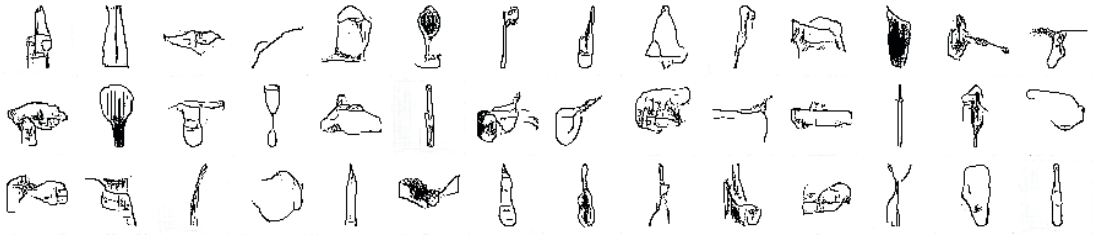

Fig. 5: Random generated samples across all 42 classes.

chair, airplane, alarm clock, axe, bell, bench, bicycle, blimp, cannon, car (sedan), motorcycle, piano, pickup truck,pistol, racket, rocket, sailboat, rifle, scissors, couch, cup, door, eyeglasses, fan, guitar, hammer, harp, helicopter, hot-air balloon, hourglass, knife, shoe, spoon, sword, table, tank, teapot, trumpet, violin, wheelchair, windmill, wine bottle

TABLE I: Classes generated in this case study.

Airplane, Alarm Clock, Bell, Bench, Bicycle, Blimp, Cannon, Car, Chair, Couch, Cup, Door, Eyeglasses, Fan, Guitar, Hammer, Harp, Helicopter, Hot-Air Balloon, Hourglass, Piano, Pickup Truck, Pistol, Racket, Spoon, Table, Tank, Teapot, Trumpet, Violin, Windmill, Wine Bottle

TABLE II: Classes which could fool the discriminator.

Sketchy Database [31]. This dataset contains 75,471 sketches across 125 categories based on 12,500 photos. 10922 of these sketches were removed for reasons cited by the dataset including ambiguity, erroneously including context such as background, incorrect object pose, or artist error.

Additionally, naturally occurring objects were removed from this dataset, leaving 42 classes, which can be seen in Table I. In total, 19,633 sketches were used to train this GAN.

B. Network Description And Training

The ACGAN used in this method is a straightforward network similar to Odena et al. [29], with a few differences. The major difference is to accommodate the smaller generated sketch size. The sketches are originally 256×256 and are scaled down to 64×64 to speed up training.

The model was trained for 100 epochs, with a batch size of 64, using an Adam optimizer with the hyperparameters suggested in the reference work [29].

C. Evaluation

This network is evaluated by a survey. The survey was conducted on Amazon Mechanical Turk, using an open call and offering a monetary reward. For each network, up to 20 images for each class are generated randomly, which are correctly classified but are misclassified by the discriminator as being real (with a probability of > 0.7).

Next, 500 random sets of 5 images are drawn from this sample. For each image, 3 incorrect classes, the correct class, and an option for "Do Not Know/None of the above", are available, and a user must answer. Respondents who "clicked through" and took less than 5 seconds per image are not considered, as the quality of their response cannot be ensured.

V. Results

A. Qualitative Evaluation

Of the 42 classes trained, results were collected for 32 classes. This is primarily due to the difficulty of generating

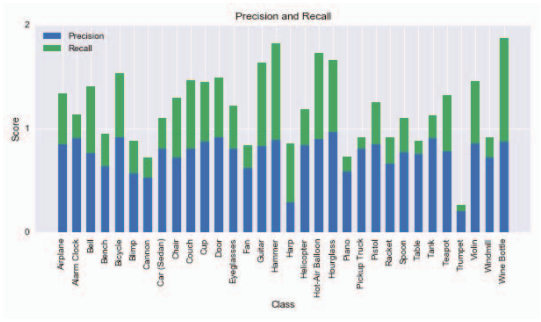

Fig. 6: Precision and Recall of the qualitative assessment

examples which could fool the discriminator. The discriminator outputs a probability of being sampled from the dataset (between 0 and 1), and only samples which had a probability of > 0.7 were considered. Those classes are listed in Table II For the large network, 281 responses were left after filtering out responses which were too fast, and therefore unreliable.

The precision and recall for the qualitative measurement can be seen in Fig. 6. Overall, the predictions were 43.8% accurate. Note that several classes were highly accurate, including Wine Bottle (100%), Hammer (92.8%), Hot Air Balloon (83.6%), Guitar (80.7%), and Hourglass (69.6%). One of the poorest performers, Harp, only had 7 cases included in the evaluation phase (the second least common class was Couch), but is included here for reference. Overall Pickup Truck and Trumpet had the lowest F Score.

Of interest in this study is the ability of these networks to generate recognizable objects of both high and low detail. Among the highest detail objects in the dataset is racket. As discussed, rackets are distinguished by laces in a frame, and rackets without laces are removed from the dataset as ambiguous. The precision of the racket class was among the best.

To better contextualize these results, another survey was conducted. Drawing random images from the training set, users on Mechanical Turk were asked about five random images drawn from the dataset, and given the opportunity

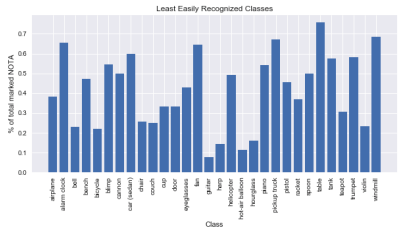

Fig. 7: Classes marked none of the above or do not know

to reply none of the above. From the 500 respondents, 252 responses met the time threshold. Overall, the users were 87.7% accurate at categorizing these sketches. The primary source of these inaccuracies came from 3 classes: Rifle, Sailboat and Rocket. Each of these classes registered a none of the above response over 80% of the time.

B. NOTA Analysis

The users in this survey were given the option of selecting none of the above (NOTA) or do not know as one of the choices. In addition to noting which classes were identified correctly, it is also worth examining which classes were generated that users could not select, even given that the correct choice was presented to them. In other words, these classes were generated poorly enough that the user was unable to recognize the sketch, even when primed with the correct choice.

The results of this can be seen in Fig. 7. The classes which was most easily recognized is guitar, followed by hot air balloon and hourglass. Violin, bicycle, and bell also rarely generated enough confusion for the user to pick NOTA. Two classes were never categorized as NOTA, hammer and wine bottle. Hammer is a distinctive shape, with a clear handle and head, so it is encouraging that the GAN was able to generate clear examples of these. On the other hand, wine bottle is a very simple shape, with most examples having minimal detailing. However, these classes are also not very diverse, both in shape and viewpoint.

In contrast, over 70% of tables were unable to be recognized. This may be due to having a high diversity of viewpoint, making it difficult for the network to learn a consistent model. Tables are also relatively low in detail, and some sketches include extraneous detail such as coloring in the surface. A similar issue seems to have yielded poor performance from the pickup truck class, and car class, as there is very little consistency of viewpoint across these classes.

Another possible source of error is sketch complexity, as measured by mean number of strokes in the training set. Comparing the f-score of the generated sketches with the average number of strokes in the training set, a general downward trend is demonstrated. This can be seen in Fig. 8.

C. Design Exploration

This section demonstrates the proposed network's effectiveness in exploring designs across classes. This is accomplished

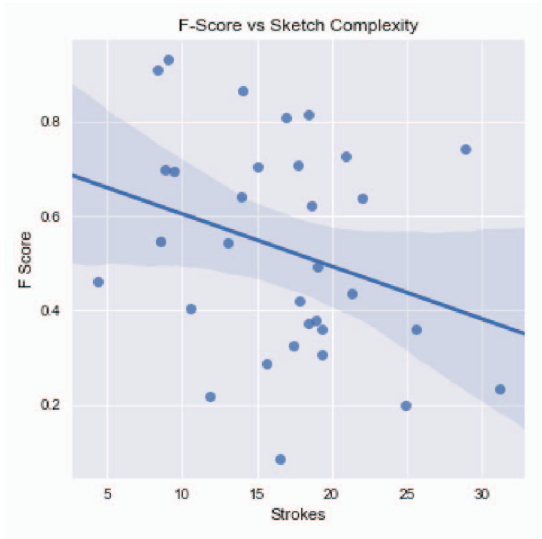

Fig. 8: Class recognition performance vs average sketch complexity

by generating 20 numbers, spaced evenly through the space, and using those as input across each class. The diversity of these designs can be seen in Figure 9. Note that the center row, for which the random number is a 100 dimensional vector of 0s, the output is all the same. This reflects the bias of the network. Since the latent space is 100 dimensions, sampling just 20 points represents a small fraction of the space. A more in depth study of this space may provide a much deeper insight.

A similar method can be used to demonstrate the ability of the objects to be morphed between each other. Figure 10 shows the output of the generator given a constant noise vector, but imputing the space between two classes. Since this network uses embedding, each class is represented by a vector, enabling arithmetic such as addition, subtraction, and averaging. For clarity, 6 pairs of classes are considered, whose class vector values are derived by taking a weighted average of two classes, though this can easily be expanded to three or more classes.

D. Data Quality

The use of GANs converges with data quality issues in several ways. First, it provides a helpful macroscopic method of detecting outliers. Since the discriminator is created and trained to detect if the input is derived from the dataset or is artificial, any erroneous classification of real data as artificial could be considered an outlier. In some cases, this may even guard against possible intrusion or injection of false records by an adversarial actor. In the motivating example of a CAPTCHA style image verification system, these outliers could give historical notifications to previous intrusions by automated actors. That is to say, any misclassification in the dataset due to outliers could potentially trigger an integrity check, and could indicate that the existing human verification system has been compromised.

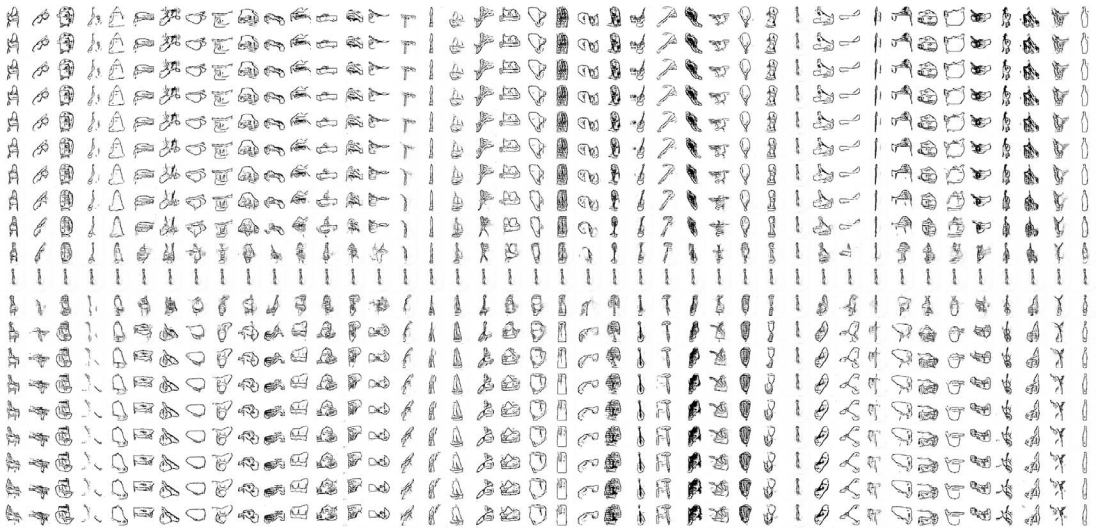

Fig. 9: 20 generated samples for each of the 42 classes

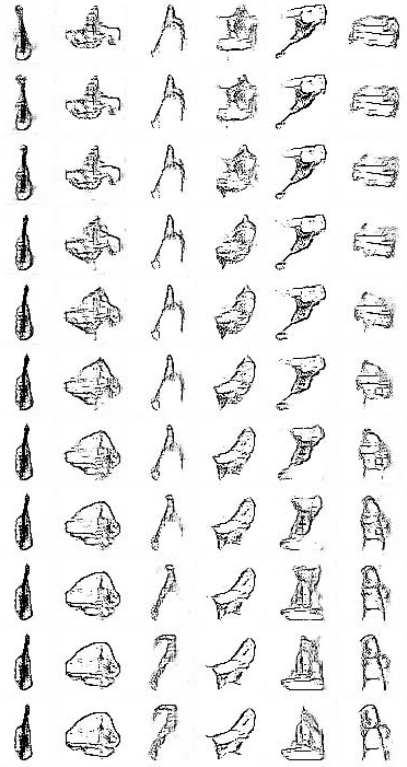

Fig. 10: Exploring the space between classes. Class Pairs (in columns) from left to right, top to bottom: Violin – Guitar; Car – Truck; Hammer - Ax; Airplane - Helicopter; Rifle – Pistol; Chair – Bench. The ratios change by 10% for each row (i.e. 100/0, 90/10 . . . 10/90, 0/100).

VI. Conclusions

This work has outlined how a GAN would integrate in a big data pipeline, and improve the veracity of data. This improvement takes place both prior to deployment during training and after deployment as a filter. This would also allow the adverse impact of low veracity data to be mitigated, since the model would be robust to many more types of data. The utility of this GAN was demonstrated by training it with sketched images of 42 classes of objects. The images generated by the GAN were shown to human observers who were 43.8% accurate in classifying the images, while 87.7% accurate in classifying human generated images. These generated images could be used to augment an existing dataset by providing more input for training, or to guard a system against high quality computer generated images. By ensuring that low veracity data can be handled gracefully and integrated or protected against, a GAN such as this one can be a valuable part of many big data systems.

Acknowledgments

The authors would like to acknowledge the NSF DUE/IUSE #1449650: Investigating the Impact of Co-Learning Systems in Providing Customized, Real-time Student Feedback and DARPA FUN DESIGN #HR00111820008.

References

[1] R. J. Nowling and J. Vyas, "A domain-driven, generative data model for big pet store," in *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*. IEEE, 2014, pp. 49–55.

[2] C. Beecks, M. S. Uysal, and T. Seidl, "Gradient-based signatures for big multimedia data," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2834–2835.

[3] X. Li, M. Cheung, and J. She, "Connection discovery using shared images by gaussian relational topic model," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 931–936.

[4] X. Jia, A. Khandelwal, J. Gerber, K. Carlson, P. West, and V. Kumar, "Learning large-scale plantation mapping from imperfect annotators," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1192–1201.

[5] T. Mukherjee, B. Parajuli, P. Kumar, and E. Pasiliao, "Truthcore: Non-parametric estimation of truth from a collection of authoritative sources," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 976–983.

[6] X. Jia, A. Wang, X. Li, G. Xun, W. Xu, and A. Zhang, "Multi-modal learning for video recommendation based on mobile application usage," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 837–842.

[7] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Information Fusion*, vol. 6, no. 1, pp. 99–111, 2005.

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[10] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[11] T. Bodnar, C. Tucker, K. Hopkinson, and S. G. Bilén, "Increasing the veracity of event detection on social media networks through user trust modeling," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 636–643.

[12] M. A. Schuh and R. A. Angryk, "Massive labeled solar image data benchmarks for automated feature recognition," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 53–60.

[13] M. Wojnowicz, B. Cruz, X. Zhao, B. Wallace, M. Wolff, J. Luan, and C. Crable, "influence sketching: Finding influential samples in large-scale regressions," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3601–3612.

[14] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint arXiv:1602.02697*, 2016.

[15] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[17] A. Pourhabib, B. K. Mallick, and Y. Ding, "Absent data generating classifier for imbalanced class sizes," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 2695–2724, 2015.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 1322–1328.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[20] D. Fried, M. Surdeanu, S. Kobourov, M. Hingle, and D. Bell, "Analyzing the language of food on social media," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 778–783.

[21] T. Bodnar, M. L. Dering, C. Tucker, and K. M. Hopkinson, "Using large-scale social media networks as a scalable sensing system for modeling real-time energy utilization patterns," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[22] H.-Â. Cao, T. K. Wijaya, K. Aberer, and N. Nunes, "Estimating human interactions with electrical appliances for activity-based energy savings recommendations," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1301–1308.

[23] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[24] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[25] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[26] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[27] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating images from captions with attention," *arXiv preprint arXiv:1511.02793*, 2015.

[28] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[29] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585*, 2016.

[30] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[31] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The sketchy database: Learning to retrieve badly drawn bunnies," *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.