Machine Learning Engineer Nanodegree

Capstone Project

Chris Aasted
February 23, 2017

I. Definition

Project Overview

Collecting sufficiently large datasets of labeled data can be one of the most expensive, or entirely prohibitive, aspects of training machine learning algorithms that generalize well. Fitting a model using a dataset that isn't sufficiently representative of the potential combinations of inputs can lead to over-fitting, poor performance on novel data (including the reserved test set), or even cause difficulty minimizing validation error. Collecting vast quantities of data has become increasingly popular as the cost to do so has dropped over the last couple decades, but the expense to label this data and make it useful for many algorithms can still be very high. Generative adversarial networks (GANs)[1] are one area of unsupervised learning that can make use of these large datasets and potentially lower the amount of labeled data required to train and test new models by learning concepts within the data in an unsupervised fashion[2].

One of the popular[3] datasets for evaluating the performance of algorithms for image recognition, as well as for training GANs to generate life-like images, is the CIFAR-10 collection of images[4]. This dataset contain 50,000 images for training and 10,000 images for testing, with 10 classes of images. The performance of various algorithms on this set of images is readily available, so I am proposing to use it in examining how GANs can be utilized to enhance image classification using combinations of supervised and unsupervised learning.

Problem Statement

Can generative adversarial networks be used to make an image classifier more robust? Using the CIFAR-10 dataset I will evaluate whether a set of generative adversarial networks can be utilized in such a way as to achieve higher accuracy than an equivalent classification algorithm using traditional supervised learning.

Evaluation Metrics

The performance metric for this proposal will be categorical accuracy on a set of test data. The accuracy of the GAN-based architecture will be compared to the accuracy of a non-GAN trained set of discriminators, a standard multi-class CNN architecture, as well as the current state-of-the-art methods for these datasets[3].

II. Analysis

Data Exploration

The CIFAR-10 dataset that will be used in this project is available directly through a Keras import statement, which downloads the images available at https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz. CIFAR-10 contains 10 classes, with 5000 samples per class. Each sample is an image with 32 pixels, by 32 pixels, by 3 color channels, resulting in a 32x32x3 array, which will need to be converted from unit8 to floating point values in the range [0, 1]. The GAN's generator will accept 100 random floating-point values [0, 1] as input and produces a 32x32x3 tensor with values in the range [0, 1]. The GAN's discriminator accepts either the output of the generator or images from the CIFAR-10 dataset and classifies them as being either "real" or "generated," where "real" images came from the CIFAR-10 dataset of a particular image class and "generated" images came from the generator. The GAN training process produces an adversarial relationship between these networks and ideally settles at the Nash equilibrium[1].

Alex Krizhevsky's webpage for the CIFAR-10 and CIFAR-100 datasets provides a random sampling of images such as those seen in **Figure 1**, as well as the additional clarification that "The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. 'Automobile' includes sedans, SUVs, things of that sort. 'Truck' includes only big trucks. Neither includes pickup trucks."

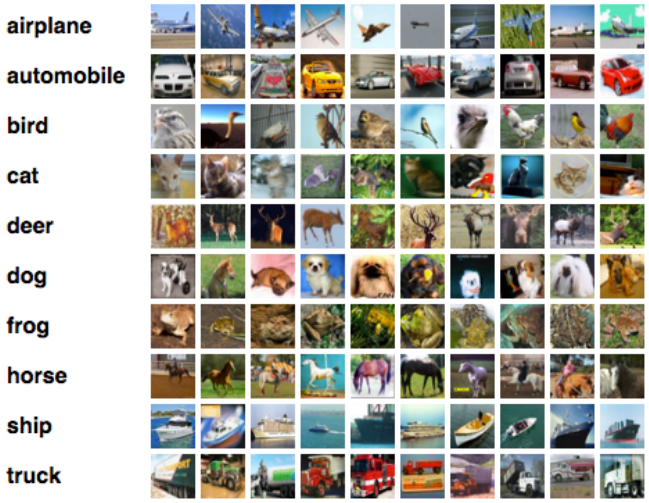


Figure 1: A random sampling of images from each of the CIFAR-10 image classes.

Statistical analysis of the data shows that the training images have a mean of 120.7 and standard deviation of 64.1, while the test images have a mean of 121.5 and standard deviation of 64.1. These values will need to be scaled down to the range [0, 1] in order to be paired with the sigmoidal outputs of a generative network. There is also some variation between classes, with the class means being represented by the following array: [142.366, 116.698, 119.406, 116.232, 111.775, 117.394, 106.562, 118.89, 133.476, 124.277]. However, this can be seen as a distinguishing feature between images of each of these types (such as color) and will not be modified. No other data processing is required and no data will be excluded as outliers.

Exploratory Visualization

To verify that the image types do not contain a defining characteristic that trivializes the classification task, I have rendered the average pixel values for all of the training images of each type in **Figure 2**. It is visually apparent that no obvious trends exist within any of the image labels.

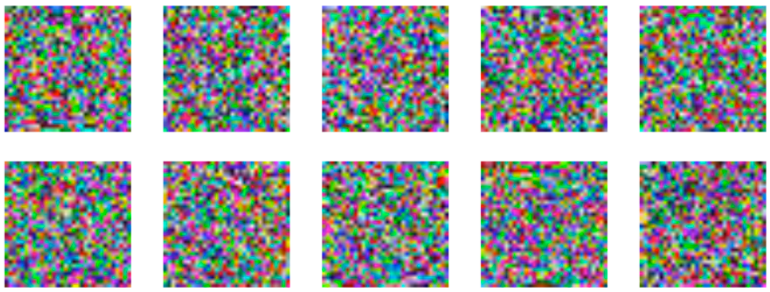


Figure 2: The mean pixel values for each class of the CIFAR-10 dataset.

Algorithms and Techniques

Based on the premise that GANs produce a generator and discriminator pair that are capable of creating and differentiating the underlying concepts of a target class, I believe it is possible to train a set of discriminators that each evaluate the likelihood that new images belong to their respective classes, and as a collection evaluate the most likely class. Since the GAN training process distinguishes a real instance of its class from an imitation, this process won't simply pick out distinguishing features **between classes** but rather **of each class**. The success of this approach will be determined by whether the algorithm is able to achieve higher accuracy on test data than a comparable network trained using a traditional method.

I will train up to ten GANs to generate and distinguish classes of images from a publically available dataset. Using the resulting discriminators, I will build a parallel architecture that evaluates new images using all of these networks and selects the class of the image to be the output with the highest confidence in it matching a real image of that particular class. The performance of this architecture will be evaluated based on its accuracy at determining class labels against a test dataset.

Examples of existing applications of GANs and code, are available from multiple sources [5, 6, 7, and 8].

Benchmark

To evaluate the utility of the GAN-based classifier, I will also train a deep convolutional neural network (CNN) using the same parallel networks architecture as the GAN, but with an output node for non-class images instead of generated images. I will also evaluate the performance of a multi-class training approach. The performance of these benchmark models will be evaluated on the same test data.

III. Methodology

Data Preprocessing

For use in this application, the image data is first converted to floating point values (from uint8). Next the values are scaled to the range [0, 1] by dividing by 255. For the single-class discriminators, the images are then split into two groups, representing the current class being trained and a second set for all others. In the single-class discriminator case, the image labels are only used for this separation step, and then new labels are produced to represent the current "real" image class, the "generated" image class, and the "other" images class. For use with multi-class discriminators, the image labels are converted from integer values to categorical values, meaning each label is converted to an array of binary values with length 10, where a "1" represents the class label.

Implementation

To successfully complete this project, I will separate it into four major parts. The first is to design a GAN capable of learning to generate images. The second is to assemble up to ten GAN-trained discriminators into a larger multi-class labeling system. The third is to train discriminators for each image class without including the generative component and to build a traditional CNN and train it to perform classification on the same data. The fourth is to compare the performance of these three approaches to each other and against the current state-of-the-art solutions.

Designing a GAN to learn to generate and discriminate images will be the most difficult and time consuming of these tasks. I will begin by creating a GAN capable of successfully generating realistic images for a single class in the CIFAR-10 dataset. Once a successful design has been found, it will be individually applied to the other nine classes of the dataset. Provided that each generator is now able to create convincing images (human evaluated), the discriminators will then be deemed suitable for use in the classification task.

Assembling up to ten discriminators to jointly evaluate the class of an image is a simple process, but is not as computationally efficient as running each network in serial, storing their output as an array, and then evaluating their outputs as a group after each network has processed all of the test images. I will use this simulated parallel network approach for evaluating the performance of the composite networks.

Refinement

After experimenting with a number of generator and discriminator architectures, I was able to successfully implement a GAN training process that was capable of creating realistic looking images for the "car" image type. During the process of repeating the training for each of the other image classes, it occurred to me that the "real" vs. "generated" output labels would produce a scenario where the discriminator for any given class was very likely to indicate that novel images of its image type should be classified as being a generated image of their class, since they are similar to, but not an exact match for the original training data. This results in a scenario where when the discriminators have been trained by producing very good generative models, the accuracy of the composite classifier is actually zero percent. Since I used a softmax activation function as the output

for each of the discriminators, combining the probability of the real and generated output nodes wasn't feasible.

As a result, I reformulated the discriminator architecture in the GAN training process to use output labels for "real," "generated," and "other" images. This allows the GAN process to backpropagate "real" labels for the generator to learn from, while also allowing for the use of sum of the "real" and "generated" probability to assess membership of an image type during classification using the discriminator. This also introduced the use of an effectively unlabeled set of data for all "other" images.

Another major refinement that had to be made was the number of epochs the GAN was allowed to train for. For as long as the GAN process remains stable, the quality of images that the generator produces continues to increase. However, when my early models had been trained for 1,000 epochs, the discriminators often over-fit to the original images to the point that they assigned probability values of 0 or 1 for many of the test cases. Since 10 discriminators are being using in parallel, these extreme values prohibit determining which class has the highest probability since multiple classes return the same value, resulting in nominal classification accuracy. As a result, the networks needed to be re-trained using fewer epochs, at the expense of generated image quality.

IV. Results

Model Evaluation and Validation

The models for the GAN and regular classifier networks in this project were implemented in Python using the Keras frontend for TensorFlow. All of the code for the models and training processes discussed below are available at https://github.com/caasted/electric-sheep.

The generator model architecture (illustrated in **Figure 3**) is heavily based on the model used by Salimans et al.[9] In summary, it utilizes one fully connected layer and three convolutional layers, with 2-dimentional up-sampling before each convolution, which reshapes a 100x1 noise vector into a 32x32x3 tensor with values ranging from zero to one, just like the CIFAR-10 images after pre-processing. The generator uses batch normalization to enhance its ability to learn quickly.

The discriminator model architecture (illustrated in **Figure 4**) was developed after testing a variety of deep convolutional models based on common architectures, but most closely resembles the Keras example deep CNN architecture for classifying images from the CIFAR-10 dataset, available at https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py. However, my architecture uses eight convolutional layers, two large dense layers, and Leaky ReLU activation for most layers. This architecture was chosen because it produced convincing image generation results very quickly.

Building the GAN model is actually just a matter of stacking the generator and discriminator networks into a single model (**Figure 5**), disabling training of the discriminator layers during GAN training, and then backpropagating error from the "real" image output node to cause the generator to progressively produce outputs that more closely resemble images from the training data. The most complicated aspect of this process is finding a combination of model architectures, learning

rates, batch sizes, and training data presentation, such that neither the generator nor discriminator begins to overpower the other. In my experience if the generator is too aggressive it will only learn a single image, and if the discriminator overpowers the generator it can prevent the generator from learning effectively. My solution to this problem was to use the Adadelta[10] optimizer, which negates the need for adjusting learning rates, and instead I created a process to allow the GAN training cycle the option to repeat training on batches when the loss is too high. By ramping up the number of allowed repeat batches as training progresses, I have had success at keeping the generative loss in a region where learning still occurs at a competitive rate, but not so much so that the process becomes unstable. As a result, the hyper-parameters I used to balance the training process became the size of the batches used to train the generator, the base number of repeat batches, and the maximum number of repeat batches allowed. I found that small batches, as low as 10 samples, with a high number of allowed repeats, as high as 20, resulted in the most stable training, but had the worst image quality. Increasing the batch size to 100, increased the quality of the images, but often increased the number of repeat batches that actually occurred and occasionally caused the network to become unstable.

Throughout the development of the GAN, the entire set of "automobile" training images was used as the "real" image set, with no data allocated for validation. Instead the goodness of the model was evaluated by whether or not the generator became capable of producing images that resembled cars. Once the network reached sufficient performance on this very subjective metric, the training process was repeated for the other nine classes of images and their joint accuracy was assessed on the test images set. For clarification, the joint output classification of ten discriminators was considered to be the network that had the highest combined probability that a test sample belonged to either the "real" or "generated" class of that network.

To assess the performance of the discriminative networks produced by the GAN process, two benchmark models were developed as well. The first was a set of ten classifiers with the same architecture as the GAN discriminator, but with two output nodes instead of three. The networks were each trained by performing the same training set separation used for training the GANs, so the output classes became "real" images and "other" images. The joint accuracy of these ten networks was then assessed using the test data, where the class of a test image was considered to be the network with the highest probability that the sample belonged to the "real" classification from the network.

The second benchmark model used the same architecture as the first, but with ten output nodes instead of two, resulting in a single node for each class of image. As a result, the training data was not separated into multiple groups and only one network was trained. When evaluated on the test data set, the class of a sample is simply the node with the highest softmax output (probability). The inclusion of this model is intended to demonstrate how a more conventional multi-class solution performs on this classification task.

Since I am not seeking to design a classifier for a specific task, but rather assess the utility of a GAN training process at enhancing the robustness of a discriminator, the performance of each of these models was assessed after every ten epochs of training. This provides visualization for the training speed of each method as well as the relative performance of each model after sufficiently converging on the training set. This also resolves the inequality of only withholding a validation set for the benchmark models, but not the GANs, and the conclusions that I reached when using a validation set, which indicated that the benchmark models did not require as many epochs of training to reach sufficiently low training loss. The performance of the GAN composite classifier and the benchmark models are illustrated in **Figure 6**.

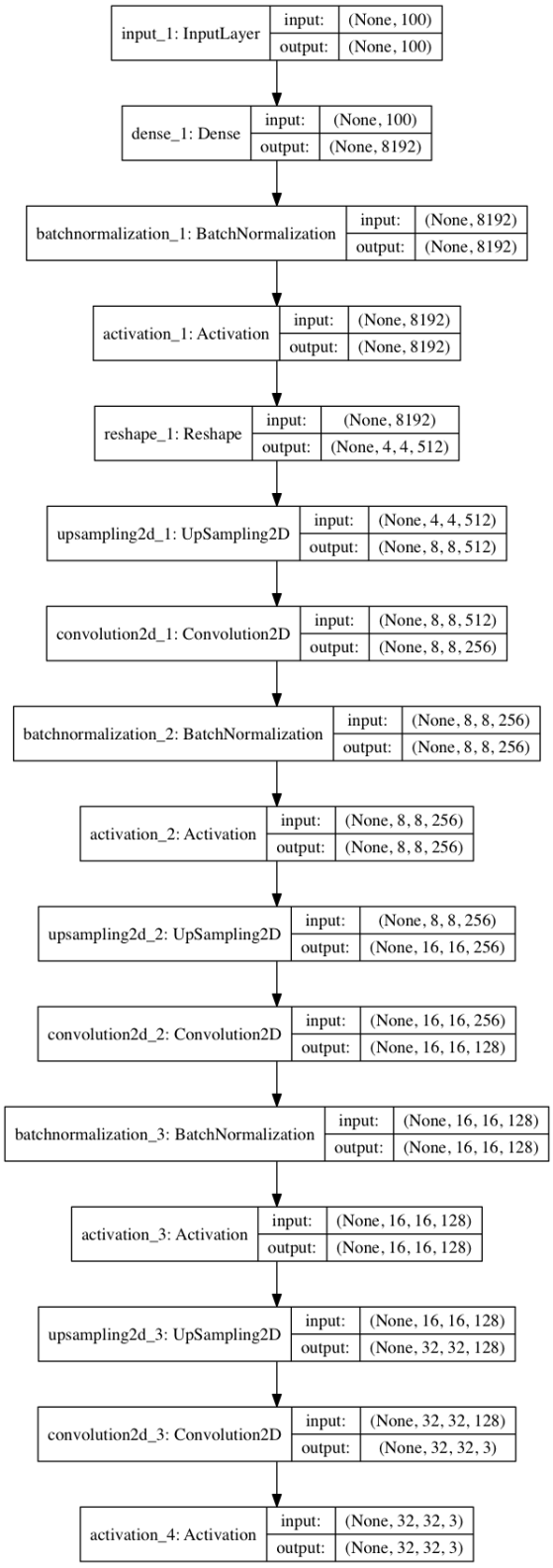


Figure 3: The model architecture for the generative network used in this project.

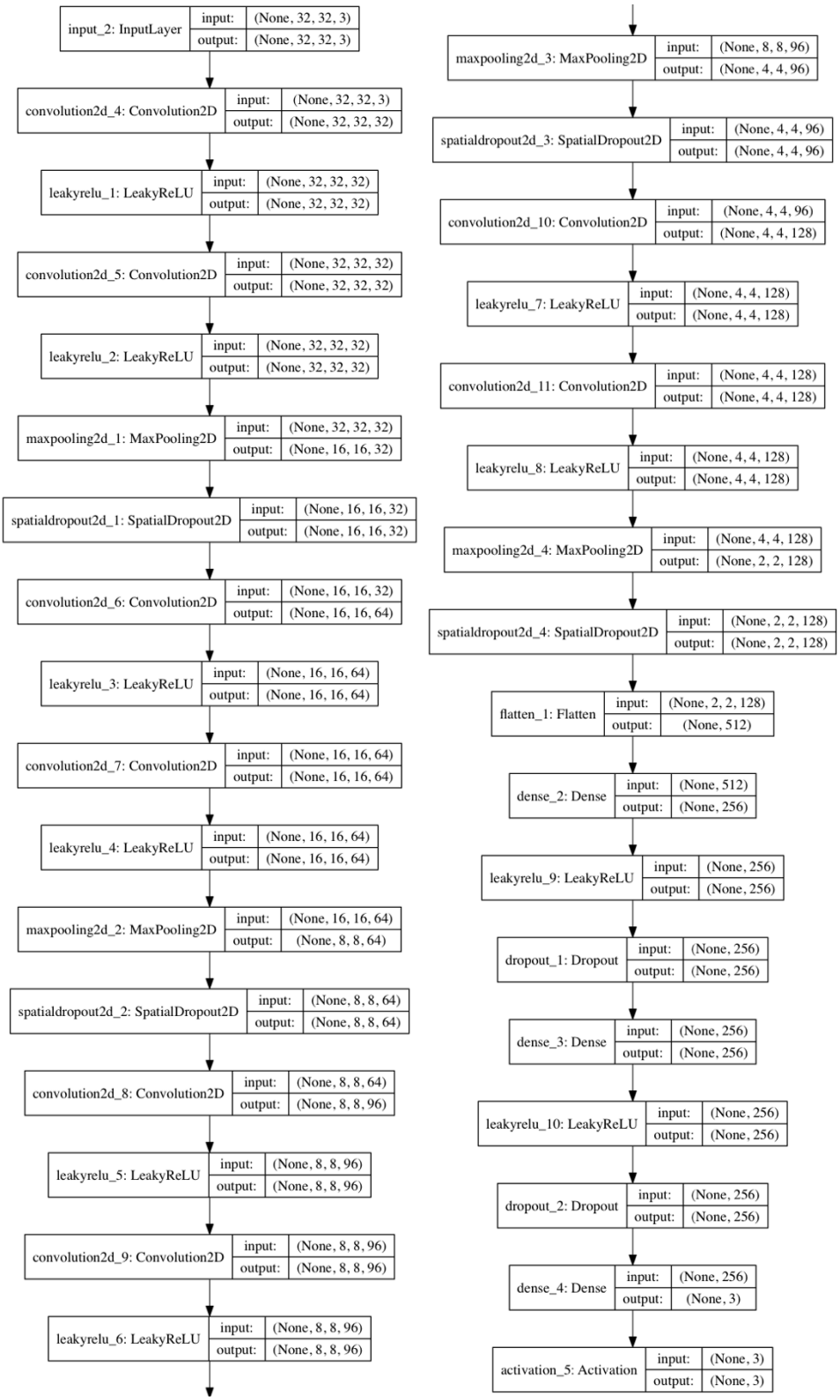


Figure 4: The model architecture for one of the discriminative networks used in this project.

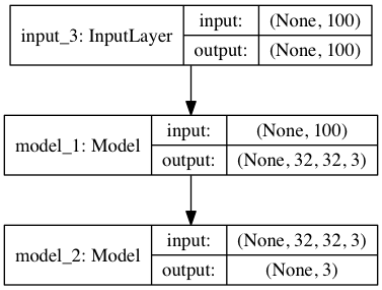


Figure 5: GAN model architecture.

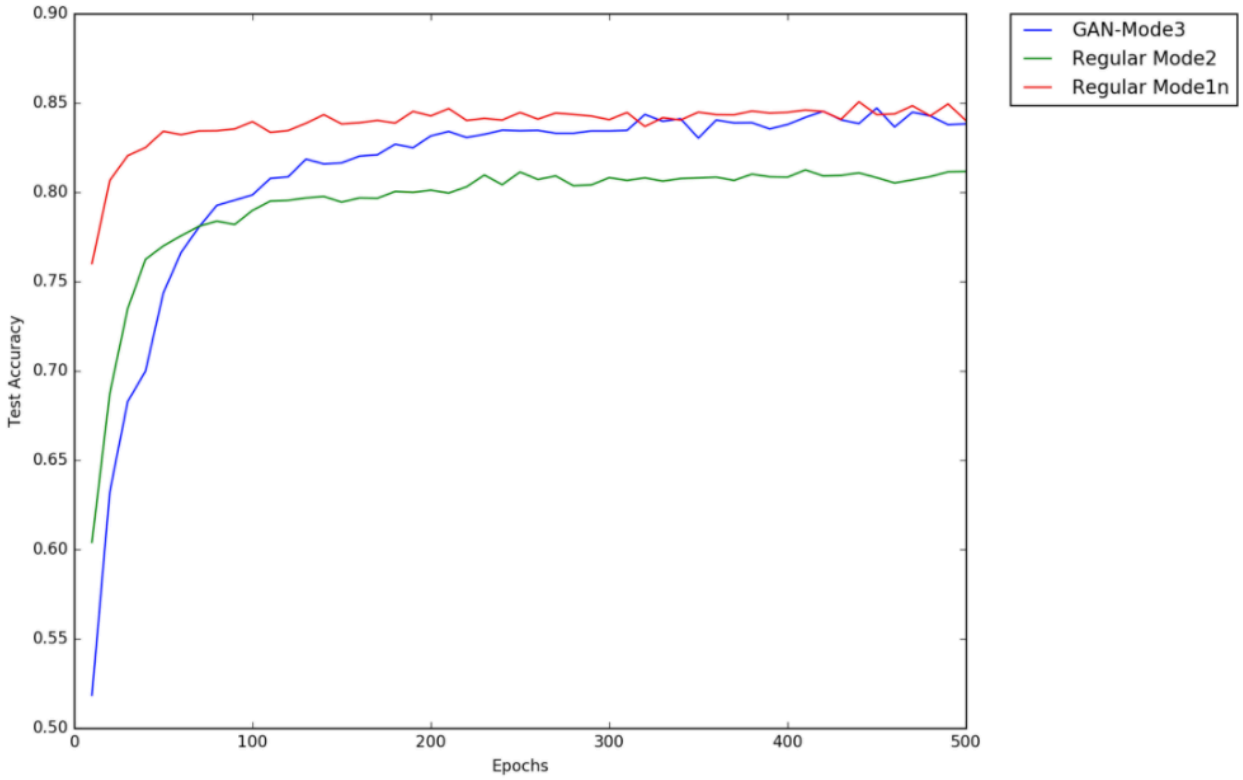


Figure 6: Test accuracy trend over time for the GAN and two benchmark models. GAN-Mode3 represents the 3-output, GAN-based, composite classifier (blue). Regular Mode2 represents the 2-output, composite benchmark classifier (green). Regular Mode1n represents the 10-output, multi-class benchmark classifier (red).

Justification

As illustrated in **Figure 6**, while the benchmark composite classifier barely reaches 81% accuracy on the test set, the GAN-based composite classifier approaches the nearly 85% accuracy that can be achieved through training a multi-class version of the same network. These results can be interpreted to indicate that the GAN training process significantly increased the performance of the composite classifier, but did not exceed a more traditional multi-class approach. This isn't

surprising considering patching together several independently trained classifiers isn't a common approach because of this decrease in performance. However, this suggests that developing a GAN training system that is able to label images as being "real" or "generated" for each class of a multi-class discriminator could yield even better results. It also seems likely that increasing the length of time the GAN spends generating new images that are similar to real images, the more robust the discriminator may become, provided that the multi-class GAN prevents the softmax outputs from saturating on test images, as I experienced with the single-type GAN training process.

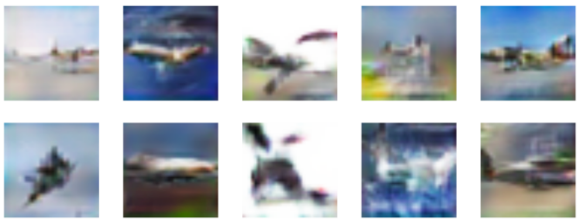
None of these models approach the level of accuracy that has been reported through methods such as data augmentation (over 90%), but this could represent an alternative, or supplemental, method for performing data augmentation through continuously generating new images and perhaps taking advantage of additional unlabeled data for improving the generative model.

V. Conclusion

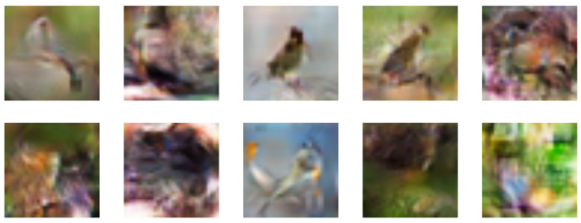
Free-Form Visualization

Some of the more interesting outcomes of this project are the generated images that the discriminators utilized to train their representation of class-like images. Below are samples for each of the image classes taken at the 500th epoch. While the generated images at the 500th epoch are not very photo-realistic, they represent a set of images that machine-learning algorithms may mistake for true images.

Class 0 – Airplane

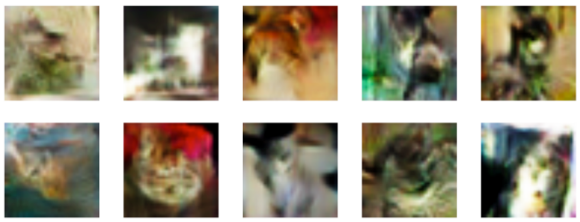


Class 2 – Bird

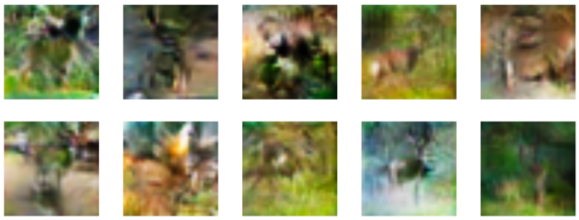


Class 1 – Automobile

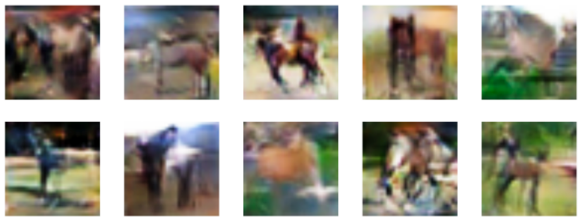
Class 3 – Cat



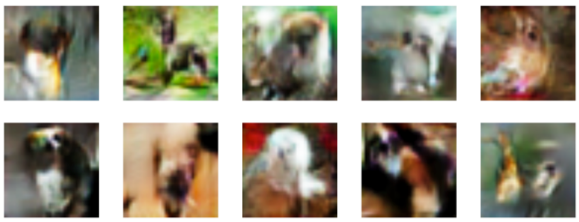
Class 4 – Deer



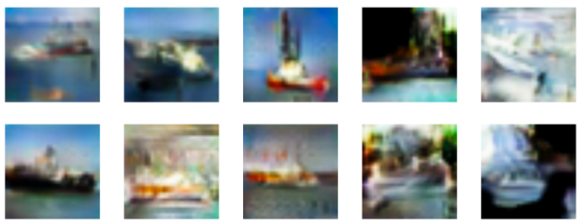
Class 7 – Horse



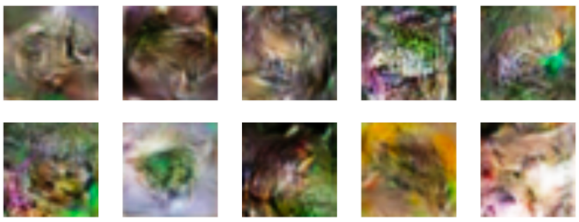
Class 5 – Dog



Class 8 – Ship



Class 6 – Frog



Class 9 – Truck

Reflection

In this project I used deep learning, convolutional neural networks, and supervised learning to assess the utility of generative adversarial networks for improving performance on an image classification task. I balanced generative and discriminatory networks to deliberately reach a point where they weren't at a Nash equilibrium, but tended towards allowing the discriminator to fully distinguish generated images as a class of their own, but as a moving target that increasing resembled the true image data. Using a set of ten discriminators trained on each of the image classes in the CIFAR-10 dataset, I created a composite classifier that used the joint output of the ten discriminators to infer the class of new test images. To evaluate the utility of this method, I also trained traditional image classifiers using the same model architecture and compared their efficacy to the GAN-based classifier's performance.

The most interesting aspect of this project was probably the process of finding a generator-discriminator pair that could remain stable during training. In the end, the repeat-batch training process that I used to maintain balance for longer periods of time, ultimately may have simplified the model matching process considerably.

The most difficult aspect of the project was probably the development cycle time. Many of the models that I tested took hours of training, despite the dedicated use of an NVIDIA GeForce 980ti graphics card, just to determine that they weren't going to produce adequate results. The final GAN implementation takes about 18 hours to train on the ten classes of images and because of the repeat batch loss control procedure, the time to train for higher numbers of epochs can increase non-linearly.

While the models used in this project are useful at illustrating a potential new application of GANs, the performance they achieve is not remarkable in itself. Applying a GAN-enhancement process to a state of the art image classifier could produce much more impressive results, but will likely require significant work to accomplish as the adversarial training process is inherently unstable and may be easier to tune when using simple networks.

Improvement

There is evidence to believe that developing a multi-class conditional GAN[11] could provide a significant improvement over the results presented here. This would produce a discriminator capable of labeling real and generated images for all ten classes in a single network. A multi-class discriminator would alleviate the issues from over-fitting experienced by the individual class discriminators that were trained to 1000 epochs. Continued adversarial training would still most likely result in over-fitting to the training data, but the generative class labels would very likely continue to improve, as the generated images become more realistic looking. Additionally, a single classifier network has the advantage of being much more efficient than a composite of ten networks. However, creating this network is beyond the scope of the proposed project and may take considerably longer to develop.

---

[1] Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil;
[2] Chen, Xi; Duan, Yan; Houthooft, Rein; Schulman, John; Sutskever, Ilya; Abbeel, Pieter (2016). "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". **arXiv:1606.03657v1**
[3] Benenson, Rodrigo. "Classification datasets results." github.io. February 22, 2016. Web. January 23, 2017.
[4] Krizhevsky, Alex. "Learning Multiple Layers of Features from Tiny Images" (2009). Available at https://www.cs.toronto.edu/~kriz/cifar.html.
[5] https://github.com/osh/KerasGAN
[6] https://openai.com/blog/generative-models/
[7] http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/
[8] https://github.com/openai/improved-gan
[9] Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi (2016). "Improved Techniques for Training GANs." **arXiv:1606.03498v1**

[10] Zeiler, Matthew D. (2012). "ADADELTA: An Adaptive Learning Rate Method." **arXiv:1212.5701v1**
[11] Mirza, Mehdi and Osindero, Simon (2014). "Conditional Generative Adversarial Nets". **arXiv:1411.1784v1**