Seminar in Statistical Language Modeling Project Report

Rhett D'souza
Northwestern University
RND7074

Amit Adate
Northwestern University
ASA5078

Abstract

We attempted to use simple word embeddings (like GloVe embeddings) of image captions as a condition to a Conditional Generative Adversarial Network (CGAN) to generate images that closely represent the description in the caption. As a baseline/proof of concept, we first, made use of the Fashion MNIST dataset with a simple condition as the output label, rather than word embeddings, to show that a conditional vector generates the desired results, and then attempted to use the caption-image pairs from the COCO dataset to generate images that might resemble the caption. We found that this task is much harder and the high variance and complexity in the captions resulted in noisy images being generated with extremely sparse realistic representations.

1 Introduction and Motivation

Generative Adversarial Networks (GANs) are an extremely recent development, that involves training a deep neural network to map pure noise inputs to a specific ground-truth space. The training scheme of a GAN involves a generator network progressively learning to generate samples that can successively fool a discriminator network trained to classify the fake images from the generator, and ground truth samples from the real-life dataset accordingly. There have been many cases of samples being generated from the generator network that appear to be realistic and plausible, but have no real ground truth.

While the GAN architecture and scheme generated an interesting and amazing set of samples, it offered no way of controlling the nature of the samples we received from the generator network. In response Conditional Generative Adversarial Networks (CGANs) were developed. By concatenating a condition with the input noise, the CGAN is able to learn a defined conditional mapping from the noise space to the generative space. The condition can be any dataset-wide consistent concept, like an output label, a set of determining labels, etc, that can help the generator decide a definitive output to be generated. We have shown the proof of concept of this basic task with Fashion MNIST dataset, where the output label of the Fashion MNIST dataset was concatenated with the input noise to generate realistice images of specific labels, like "T-shirt", "Dress", etc.

The basic structure of a CGAN offers us the capability to provide simple conditional labels to the GAN input, but however, rarely do we ever have the capability to describe an entity with a single word or a set of simple descriptors. Motivated by the practicality of image description and captioning, we attempted to concatenate a set of word embeddings derived from a caption with the input noise and train a generative network to generate samples that closely resemble the description mentioned in the caption. While the leap from basic numeric label encoding conditions to embedding conditioning was large, it appeared to be worth testing.

In this report we describe the methods used for the proof of concept/baseline using the Fashion MNIST, and then we describe the method for the COCO dataset with captions encoded using GloVe embeddings. We then describe and display the results we observed for both cases. Following that, we analyze the results and provide out conclusions.

Note: We have not provided any links, or major references, as requested from the report specifications. We can however provide our sources if requested.

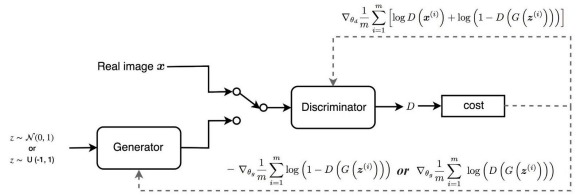

Figure 1: GAN Traininig Mechanism

2 Background of CGANs

Adding into the conductibility aspect of the latent space of a vanilla Generative Adversarial Network (Goodfellow et al., 2014), a variant conditional generative adversarial network (cgan) was introduced. Structurally, a Conditional Generative Adversarial Network (Mirza and Osindero, 2014) contains 2 networks, a generative and a discriminator network. The generative network is responsible for generating samples that fool the discriminator. In turn, the discriminator is designed to tell the fake images from generator apart from the ground truth images from the dataset. The output of the generator network should match the desired dimension of the ground truth (in this case, however, down-sampling and up-sampling is possible to rectify dimension issues) samples that will be fed into the discriminator. The input to the generator is the conditional vector concatenated with the input noise. The input to the discriminator is the output of the generator concatenated with the conditional vector.

The training scheme involves 2 phases. The first phase involves generating a batch of samples from the generator. The samples are fed into the discriminator network and the loss (how well the discriminator did, i.e how well the discriminator did not get fooled) is registered and used to backpropagate through the generator. Following that, a batch generated from the generator is fed along with the ground truth (appropriately labelled) to the discriminator. The loss is registered and used to update the discriminator. This back and forth continues until the loss of the generator gets satisfactorily low.

Note regarding GAN training (Mescheder, 2018), the GAN training mechanism is a min-max training algorithm that is defined over the objective function is demonstrated in Figure 1.

3 Dataset Description

In this section we cover the two datasets on which we conducted experiments, Fashion MNIST dataset and the COCO dataset. One of the key reasons that we chose datasets which are low dimensional is due to the existing time constraints.

3.1 Fashion MNIST

We chose the Fashion MNIST dataset (Xiao et al., 2017) because it is able to capture varied features of conventional clothing in a greyscale format. Hence the feature maps that are generated within the conditional-gan are relatively sparse in organization in comparison to the feature maps generated by a traditional three channel RGB image. For our experiments, the fashion mnist dataset was acquired from kaggle*. The training entails labeled 60000 images each of them being greyscale 28*28 (784) pixels. The training set contains 60000 data points and the test set contains 10000 data points.

3.2 COCO Dataset

The COCO dataset (Lin et al., 2014) is an excellent object detection dataset with 80 classes, 80,000 training images and 40,000 validation images. We subsampled the COCO dataset to 2000 Samples due to its base volume being 18GB, beyond our compute capability. COCO stands for Common Objects in Context. As hinted by the name, images in COCO dataset are taken from everyday scenes thus attaching context to the objects captured in the scenes. There are 91 object categories in COCO, as of the 2017 release of the dataset. The dataset is released in two categories, them being labeled images and segmented images.

4 Model/Implementation

4.1 Conditional GAN for Single Output Label Conditioning

For the Fashion MNIST dataset we used the entire training dataset, with a batch size of 1 in order to train on each image and condition pair. Further, after a lot of tuning on which layers to use we went with fully connected layers to learn all features possible from the input image condition pair. To summarize the generator, it is built with four fully connected layers with monotonically increasing number of linear units per layer. The activation function between the layers is leaky RELU and our model ends with 784 (28*28)

nodes with tanh being the final activation.

The discriminator is built with four linear (fully connected) layers with a sigmoid activation in the end as the discriminator is supposed to perform classification between the input image and condition. The loss function used to train the Gan is Binary Cross Entropy between the target and the output * and the optimizer used is ADAM*. For the fmnist dataset we went upto 20 epochs only but the training took 3 hours as we are using fully connected layers. The number of parameters in the generator and discriminator are 1,489,012 and 1,470,565 respectively. As we chose to use fully connected layers, the training time per epoch is a lot as close to 1.4M weights are updated each backward pass.

4.2 Conditional GAN for Caption Embedding Conditioning

For the COCO dataset we made use of 2000 samples of the captioning task set. Here, we opted to use only 2000 due to resource constraints. Uploading to the Google Drive and importing into Google Colaboratory, followed by training, was resource and time constrained.

The captions were encoded into GloVe word embeddings (100 dimension). These embeddings were concatenated with noise and used as inputs. Ground truth images from the COCO dataset were normalized and resized to 32x32x3 and used for discriminator training and loss calculation for the generator.

The generator for the COCO dataset comprised of 2 dense layers to ingest the noise and word embeddings, followed by 2 convolutional layers. The discriminator comprised of a dense layer followed by a convolutional layer with max-pooling, followed by another convolutional layer with max pooling and then a final dense layer, before a sigmoid activation for the binary cross-entropy calculation at the output. The entire pipeline had tanh activations. The generator had a batch normalization layer after the second dense layer. This second dense layer is important, as its output is reshaped to contain the dimensions of the output image to be generated from the generator. The number of parameters in the generator and discriminator are 9,571,959 and 5,034,881 respectively. We used the standard Stochastic Gradient Descent optimizer with a learning rate of 0.0005 and momentum of 0.9, with a batch size of 128. The training for the COCO dataset ran for a total of 1000 epochs.

5 Evaluation Metrics/Results

In order to measure the performance of our GAN, we need to measure the performance of the generator and the discriminator independently. The performance metric for the discriminator is the loss, how often is the discriminator fooled given that the input to the discriminator is a set of ground truth images along with the generated image. The performance metric of the generator is it's loss, i.e. how often it was unable to fool the discriminator. Ideally, we want the discriminator to be a good classifier (therefore it should converge overtime), but still be fooled by the generator. A comparatively higher discriminator loss is favorable and a comparatively lower generator loss is favorable, indicating that the images generated are closer to ground-truth. As the task is a real-fake classification task, we have computed binary cross entropy loss as our performance measure for both networks for the experiment.

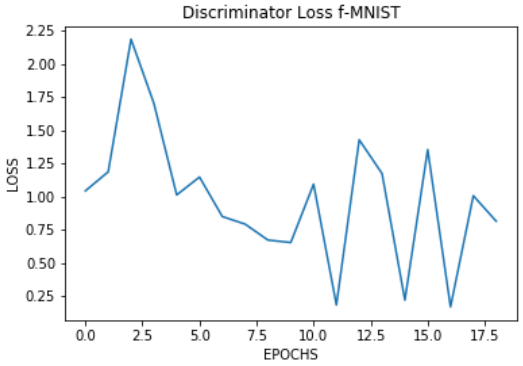

Figure 2: Discriminator Loss f-MNIST

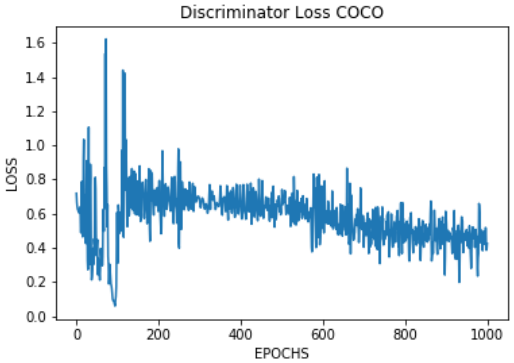

Figure 3: Discriminator Loss COCO

As per the GAN training mechanism, it is a constant adversarial evaluation based optimization ongoing between the generator and the discriminator. For the f-MNIST, we used a fully connected neural network and less number of epochs. We observed that within 20 epochs (3.5 hours of training time) the generator started performing better and the discriminator was struggling to converge. For the COCO dataset however, the discriminator was able to converge must faster, and the generator struggles to reduce it's loss comparatively.

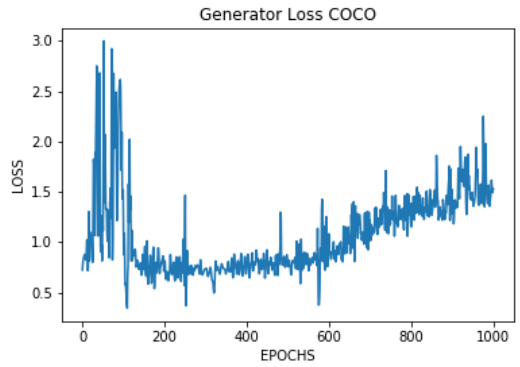

Figure 4: Generator Loss COCO

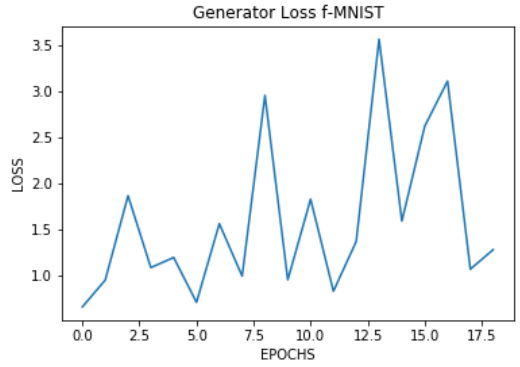

Figure 5: Generator Loss f-MNIST

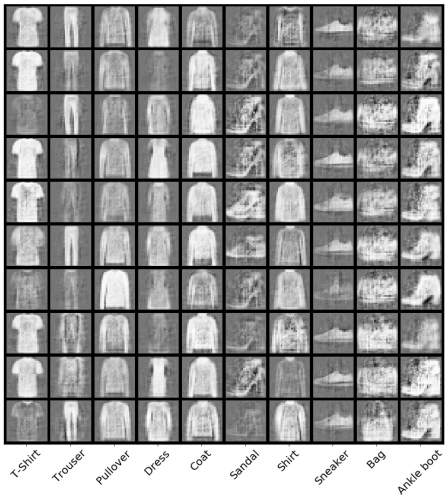

Figure 6: CGAN-f-MNIST results

To note, this is a single text label that was conditioned. As per the images in Fig. 6, they were generated by a simple forward passes through the trained architecture on the corresponding label mentioned one the x axis.

Model	***D-Loss Final***	***G-Loss Final***
c-gan-fmnist	0.8140	1.2817
c-gan-coco	0.4264	1.5314

Table 1: Final Loss Reported

6 Analysis/Interpretation

We found that the images generated from our baseline, the Fashion MNIST dataset well represent the images present in the main ground truth dataset. We also see that this is not the case with the COCO dataset. In both cases the loss of the generator appears to dip aggressively initially and then rise gradually. The loss of the discriminator in both cases appear to fall gradually. We also see that the final losses of both datasets appear to be comparable to one another.

We can hypothesize a number of reasons why the quality of images for the Fashion MNIST dataset are much better than the images generated for the COCO dataset. Straightaway, the complexity of handling a sequence of word embeddings, is much higher than just a conditional label for the respective CGANs. Additionally, the complexity of the captions in the COCO dataset do the task no favors, for example, "A train sits on the track at a

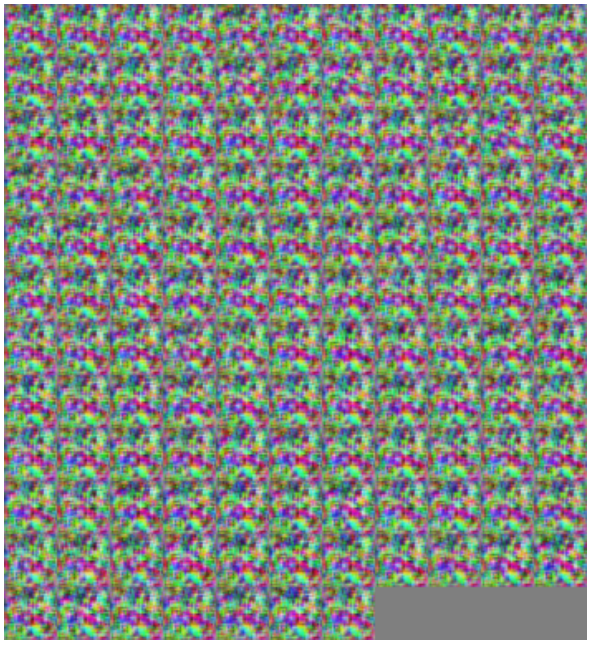

Figure 7: CGAN-COCO Epoch = 0

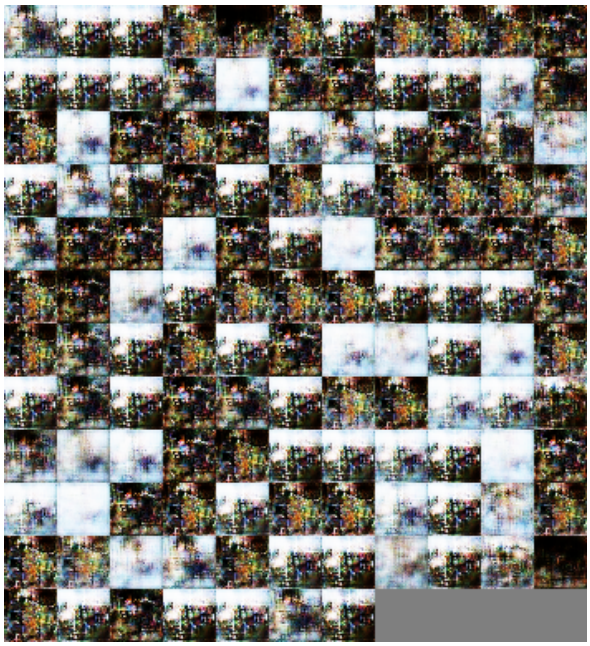

Figure 8: CGAN-COCO Epoch = 982

deserted station overlooked by a tower" is a much more complex sentence than just a label encoding. More sophisticated word embeddings, like Bengio's neural embeddings, embeddings from autoencoders, LSTMs and Self-Attention models (BERT) might have made the task less complex for the CGAN pipeline. Additionally, the complexity of the images are not comparable. The Fashion MNIST images are greyscale images with clear solid backgrounds. The COCO dataset has multiple RGB elements in its background and foreground, as well a number of different foregrounds.

An attempt to justify the difference, based on the training curves, we can see the loss value of the discriminator is slightly lower for the COCO dataset than the loss of the Fashion MNIST dataset towards the end. Similarly, the loss of the generator appears to be slightly higher for the COCO dataset towards the end (we can also see this in Table 1). We can say that the discriminator of the COCO dataset appears to be doing slightly better than the Fashion MNIST dataset, and vice versa for the generators for the 2 datasets. This points to the idea that the Fashion MNIST generator is better at fooling the discriminator than in the case of COCO. Hence we get better quality images for the Fashion MNIST dataset.

Finally, the number of samples we used for the COCO dataset were much smaller than those used for the Fashion MNIST dataset. This means that certain types of captions and images appear very few times and do not offer a substantial set of examples for the CGAN to learn from.

As a conclusion, the CGAN for the COCO dataset was able to learn sparse features from the dataset as can be seen from the difference in the training snapshots from the first epoch with pure noise (Figure 7) and the snapshot with sparse features from epoch number 982 (Figure 8). Therefore, given some improvements and more complex architectures, we are confident the images generated for the COCO dataset will be of higher quality.

References

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie,

Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Lars M. Mescheder. 2018. On the convergence properties of GAN training. *CoRR*, abs/1801.04406.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.