CAP6610 Project Report 5

Justin Ho
justinho@ufl.edu

April 17, 2023

1 Introduction

As a reminder from the last report, I was able to successfully craft an experiment design that could be conducted to evaluate the GAN and VAE models that I selected. Recently, I have been training and tweaking the eight models that I will be examining to try to ensure a fair comparison will be had amongst all the techniques which has been a bit tricky given the dataset. Outside of that, with the trained models, I have been evaluating each model using the evaluation metrics discussed previously.

2 Problems with the Cifar-10 dataset & Fine-tuning

Fine-tuning was personally a tricky process for me. Finding the best hyper-parameters and architectures for the GANs in particular proved to be challenging as the CIFAR-10 dataset seems to be an innately difficult dataset for GANs to learn. I suspect this is innate due to the structure within the dataset itself. As a brief reminder the CIFAR-10 dataset consists of 60,000 images of 10 classes. If you break that down, the GAN only really has 6,000 samples to learn from which I suspect is not enough for the GAN architectures to approximate. It is also worth noting that the images within each class of the CIFAR-10 dataset are very different and distinct, from planes, to animals, to other various objects. I suspect due to this difference and the already small amount of samples for each class, the model is suffering from the curse of dimensionality and a significant amount of additional samples is needed for the GANs to perform better. Regardless of this limitation, I did not want to change my dataset as I suspect the problems I listed here offer some interesting insight into the types of datasets that GANs may be weaker to. I also wanted to further explore various tricks and optimizations that can be used to increase the performance of GANs while maintaining roughly the same architecture as VAEs to ensure a fair comparison.

Due to the overall age of GANs, a vast amount of literature and research is available about various optimizations that one can take to improve the performance of GANs. A lot of these optimizations are noted as "best practices" that machine learning practitioners found incidentally. The ones I chose to implement was the use of batch normalization

in both the generator and the discriminator [3], utilizing LeakyReLU to counter act sparsity [3], and adding noise to the real or fake labels [1]. I found that just these simple tricks made the images produced by my GANs far more discernible.

3 Initial Results

Unconditioned Models		
Model Architecture	FID (Lower is better)	IS (Higher is better)
Transposed Convolution GAN	184.16078	4.224521
Upscale Convolution GAN	186.089	5.3463454
Transposed Convolution VAE	324.03116	1.9790983
Upscale Convolution VAE	351.15283	1.4193777

Conditioned Models		
Model Architecture	FID (Lower is better)	IS (Higher is better)
Transposed Convolution GAN	200.14644	3.9170582
Upscale Convolution GAN	201.97002	3.741853
Transposed Convolution VAE	262.48502	2.0973198
Upscale Convolution VAE	268.64572	2.0683987

4 Analysis

The results I got from running my model through the Inception Score and Frechet Inception Distance metrics were interesting. The VAEs increased in performance dramatically for both FID and IS when it was conditioned with the class label. On the other hand, the opposite occurred with GANs, with the GANs overall getting worse performance for both FID and IS when they were conditioned. This was surprising to me as my underlying intuition was that the addition of a class label during the training process should greatly improve the generative abilities of both models not just the VAEs. I suspect that there may be something with my implementation of the CGAN but if I am unable to find anything wrong, then this result is quite interesting. In terms of Upscale Convolutions, the generative models using upscale convolutions performed worse across the board outside of the unconditioned GANs in which it improved the IS. More analysis will have to be done to determine the cause of this but this is surpsing as it goes against previous literature [2].

5 Conclusion and Next Steps

We are approaching the end of this project and in terms of next steps, I will be rereviewing the results and seeing if there are any bugs that could have cause the strange results with conditioned GANs. I will also be wrapping up my paper.

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[2] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[3] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 11 2016.

```
!pip install tensorflow-gan
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Collecting tensorflow-gan
  Downloading tensorflow_gan-2.1.0-py2.py3-none-any.whl (367 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 367.1/367.1 kB 7.4 MB/s eta 0:00:00
Requirement already satisfied: tensorflow-probability>=0.7 in /usr/local/lib/pytho
Requirement already satisfied: tensorflow-hub>=0.2 in /usr/local/lib/python3.9/dis
Requirement already satisfied: protobuf>=3.19.6 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: absl-py in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: gast>=0.3.2 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: dm-tree in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: cloudpickle>=1.3 in /usr/local/lib/python3.9/dist-p
Requirement already satisfied: decorator in /usr/local/lib/python3.9/dist-packages
Installing collected packages: tensorflow-gan
Successfully installed tensorflow-gan-2.1.0
```

```
import tensorflow as tf
import tensorflow_gan as tfgan
from tensorflow import keras
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
batch_size = 120
latent_dim = 110
```

```
### https://blog.tensorflow.org/2022/01/summer-of-code.html

@tf.function
def get_inception_score(images, num_inception_images = 8):
 size = tfgan.eval.INCEPTION_DEFAULT_IMAGE_SIZE
 resized_images = tf.image.resize(images, [size, size], method=tf.image.ResizeMethod.BI

 num_batches = batch_size // num_inception_images
 inc_score = tfgan.eval.inception_score(resized_images, num_batches=num_batches)

 return inc_score


@tf.function
def get_fid_score(real_image, gen_image):
 size = tfgan.eval.INCEPTION_DEFAULT_IMAGE_SIZE

 resized_real_images = tf.image.resize(real_image, [size, size], method=tf.image.Resize
 resized_generated_images = tf.image.resize(gen_image, [size, size], method=tf.image.Re
 num_inception_images = 1
 num_batches = batch_size // num_inception_images
 fid = tfgan.eval.frechet_inception_distance(resized_real_images, resized_generated_ima
 return fid
```

UNCONDITIONAL MODELS

```
# We'll use all the available examples from both the training and test
# sets.
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

test_dataset = tf.data.Dataset.from_tensor_slices((x_test, y_test))
test_dataset = test_dataset.shuffle(buffer_size=1024).batch(batch_size)

all_images = np.concatenate([x_train, x_test])
all_labels = np.concatenate([y_train, y_test])

# Scale the pixel values to [0, 1] range, add a channel dimension to
# the images, and one-hot encode the labels.
all_images = all_images.astype("float32") / 255.0
```

```
all_labels = keras.utils.to_categorical(all_labels, 10)

dataset = tf.data.Dataset.from_tensor_slices((all_images))
dataset = dataset.shuffle(buffer_size=1024).batch(batch_size)

print(f"Shape of training images: {all_images.shape}")
print(f"Shape of training labels: {all_labels.shape}")
```

```
Shape of training images: (60000, 32, 32, 3)
Shape of training labels: (60000, 10)
```

```
def getRealImages():
  for i in dataset:
    return i


realImages = getRealImages()


del x_train
del y_train
del x_test
del y_test

del all_images
del all_labels
del dataset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-9172944f13fc> in <cell line: 1>()
----> 1 del x_train
      2 del y_train
      3 del x_test
      4 del y_test
      5

NameError: name 'x_train' is not defined
```

SEARCH STACK OVERFLOW

```
un_trans_conv_gan = tf.keras.saving.load_model(
    "/content/drive/MyDrive/un_trans_conv_gan"
)

un_trans_conv_vae = tf.keras.saving.load_model(
    "/content/drive/MyDrive/un_trans_conv_vae"
)

un_upsc_conv_gan = tf.keras.saving.load_model(
    "/content/drive/MyDrive/un_upsc_conv_gan"
)

un_upsc_conv_vae = tf.keras.saving.load_model(
    "/content/drive/MyDrive/un_upsc_conv_vae"
)
```

```
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
```

```
z = tf.random.normal(shape=(batch_size, 110))

un_trans_conv_gan_generated_image = un_trans_conv_gan(z)
un_upsc_conv_gan_generated_image = un_upsc_conv_gan(z)
un_trans_conv_vae_generated_image = un_trans_conv_vae(z)
un_upsc_conv_vae_generated_image = un_upsc_conv_vae(z)


print("---------- INCEPTION SCORE ------------")
print("Uncondition Transposed Convolution Gan")
print(get_inception_score(un_trans_conv_gan_generated_image))
print()

print("Uncondition Upscale Convolution Gan")
print(get_inception_score(un_upsc_conv_gan_generated_image))
```

```
print(get_inception_score(un_upsc_conv_gan_generated_image))
print()

print("Uncondition Transposed Convolution VAE")
print(get_inception_score(un_trans_conv_vae_generated_image))
print()

print("Uncondition Upscale Convolution VAE")
print(get_inception_score(un_upsc_conv_vae_generated_image))
print()


print("---------- FRECHET INCEPTION DISTANCE SCORE ------------")
print("Uncondition Transposed Convolution Gan")
print(get_fid_score(realImages, un_trans_conv_gan_generated_image))
print()

print("Uncondition Upscale Convolution Gan")
print(get_fid_score(realImages, un_upsc_conv_gan_generated_image))
print()

print("Uncondition Transposed Convolution VAE")
print(get_fid_score(realImages, un_trans_conv_vae_generated_image))
print()

print("Uncondition Upscale Convolution VAE")
print(get_fid_score(realImages, un_upsc_conv_vae_generated_image))
print()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.9/dist-packages/tensorflow/python/a
Instructions for updating:
back_prop=False is deprecated. Consider using tf.stop_gradient instead.
Instead of:
results = tf.map_fn(fn, elems, back_prop=False)
Use:
results = tf.nest.map_structure(tf.stop_gradient, tf.map_fn(fn, elems))
---------- INCEPTION SCORE ------------
Uncondition Transposed Convolution Gan
/usr/local/lib/python3.9/dist-packages/tensorflow/python/util/nest.py:917: UserWar
  structure[0], [func(*x) for x in entries],
/usr/local/lib/python3.9/dist-packages/keras/legacy_tf_layers/base.py:627: UserWar
  self.updates, tf.compat.v1.GraphKeys.UPDATE_OPS
tf.Tensor(4.224521, shape=(), dtype=float32)

Uncondition Upscale Convolution Gan
tf.Tensor(5.3463454, shape=(), dtype=float32)

Uncondition Transposed Convolution VAE
tf.Tensor(1.9790983, shape=(), dtype=float32)

Uncondition Upscale Convolution VAE
tf.Tensor(1.4193777, shape=(), dtype=float32)

---------- FRECHET INCEPTION DISTANCE SCORE ------------
Uncondition Transposed Convolution Gan
tf.Tensor(184.16078, shape=(), dtype=float32)

Uncondition Upscale Convolution Gan
tf.Tensor(186.089, shape=(), dtype=float32)

Uncondition Transposed Convolution VAE
tf.Tensor(324.03116, shape=(), dtype=float32)

Uncondition Upscale Convolution VAE
tf.Tensor(351.15283, shape=(), dtype=float32)
```

```
del realImages
del un_trans_conv_gan
del un_trans_conv_vae
del un_upsc_conv_gan
del un_upsc_conv_vae

del un_trans_conv_gan_generated_image
del un_upsc_conv_gan_generated_image
del un_trans_conv_vae_generated_image
del un_upsc_conv_vae_generated_image
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-933da2ca6296> in <cell line: 1>()
----> 1 del realImages
      2 del un_trans_conv_gan
      3 del un_trans_conv_vae
      4 del un_upsc_conv_gan
      5 del un_upsc_conv_vae

NameError: name 'realImages' is not defined
```

CONDITIONAL MODELS

```
# We'll use all the available examples from both the training and test
# sets.
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

test_dataset = tf.data.Dataset.from_tensor_slices((x_test, y_test))
test_dataset = test_dataset.shuffle(buffer_size=1024).batch(batch_size)

all_images = np.concatenate([x_train, x_test])
all_labels = np.concatenate([y_train, y_test])

# Scale the pixel values to [0, 1] range, add a channel dimension to
# the images, and one-hot encode the labels.
all_images = all_images.astype("float32") / 255.0
all_labels = keras.utils.to_categorical(all_labels, 10)

dataset = tf.data.Dataset.from_tensor_slices((all_images, all_labels))
dataset = dataset.shuffle(buffer_size=1024).batch(batch_size)

print(f"Shape of training images: {all_images.shape}")
print(f"Shape of training labels: {all_labels.shape}")
```

```
Shape of training images: (60000, 32, 32, 3)
Shape of training labels: (60000, 10)
```

```
def getRealImages():
  for i in dataset:
    return i


realImages = getRealImages()[0]


del x_train
del y_train
del x_test
del y_test

del all_images
del all_labels
del dataset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-46-8d590e495e63> in <cell line: 1>()
----> 1 del x_train
      2 del y_train
      3 del x_test
      4 del y_test
      5

NameError: name 'x_train' is not defined
```

SEARCH STACK OVERFLOW

```
trad_cond_vae = tf.keras.saving.load_model(
    "/content/drive/MyDrive/trad_cond_vae"
)

trans_cond_gan = tf.keras.saving.load_model(
    "/content/drive/MyDrive/trans-cond-gan"
)
```

```
upsc_cond_gan = tf.keras.saving.load_model(
    "/content/drive/MyDrive/upsc-cond-gan"
)

upsc_cond_vae = tf.keras.saving.load_model(
    "/content/drive/MyDrive/upsc-cond-vae"
)
```

```
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
WARNING:tensorflow:No training configuration found in save file, so the model was
```

```
import random

def generateWithCGan(model, z):
  label = keras.utils.to_categorical([random.randint(0, 9) for i in range(batch_size)],
  noise_and_labels = tf.concat([z, label], 1)
  generated_image = model.predict(noise_and_labels)
  return generated_image


def decode(model, z, apply_sigmoid=False):
    logits = model(z)
    if apply_sigmoid:
        probs = tf.sigmoid(logits)
        return probs
    return logits

def sample(model, eps=None):
    label = keras.utils.to_categorical([random.randint(0, 9) for i in range(batch_size)
    if eps is None:
        eps = tf.random.normal(shape=(batch_size, latent_dim))
    return decode(model, tf.concat([eps, label], 1), apply_sigmoid=True)

def generateWithCVae(model, eps):
  return sample(model, eps)


z = tf.random.normal(shape=(batch_size, latent_dim))


trad_cond_vae_generated_images = generateWithCVae(trad_cond_vae, z)
upsc_cond_vae_generated_images = generateWithCVae(upsc_cond_vae, z)

trans_cond_gan_generated_images = generateWithCGan(trans_cond_gan, z)
upsc_cond_gan_generated_images = generateWithCGan(upsc_cond_gan, z)
```

```
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 9ms/step
```

```
print("---------- INCEPTION SCORE ------------")
print("Conditional Transposed Convolution Gan")
print(get_inception_score(trans_cond_gan_generated_images))
print()

print("Conditional Upscale Convolution Gan")
print(get_inception_score(upsc_cond_gan_generated_images))
print()

print("Conditional Transposed Convolution VAE")
print(get_inception_score(trad_cond_vae_generated_images))
print()

print("Conditional Upscale Convolution VAE")
print(get_inception_score(upsc_cond_vae_generated_images))
print()


print("---------- FRECHET INCEPTION DISTANCE SCORE ------------")
print("Conditional Transposed Convolution Gan")
print(get_fid_score(realImages, trans_cond_gan_generated_images))
print()
```

```
print("Conditional Upscale Convolution Gan")
print(get_fid_score(realImages, upsc_cond_gan_generated_images))
print()

print("Conditional Transposed Convolution VAE")
print(get_fid_score(realImages, trad_cond_vae_generated_images))
print()

print("Conditional Upscale Convolution VAE")
print(get_fid_score(realImages, upsc_cond_vae_generated_images))
print()
```

```
---------- INCEPTION SCORE ------------
Conditional Transposed Convolution Gan
tf.Tensor(3.9170582, shape=(), dtype=float32)

Conditional Upscale Convolution Gan
tf.Tensor(3.741853, shape=(), dtype=float32)

Conditional Transposed Convolution VAE
tf.Tensor(2.0973198, shape=(), dtype=float32)

Conditional Upscale Convolution VAE
tf.Tensor(2.0683987, shape=(), dtype=float32)

---------- FRECHET INCEPTION DISTANCE SCORE ------------
Conditional Transposed Convolution Gan
/usr/local/lib/python3.9/dist-packages/tensorflow/python/util/nest.py:917: UserWar
  structure[0], [func(*x) for x in entries],
/usr/local/lib/python3.9/dist-packages/keras/legacy_tf_layers/base.py:627: UserWar
  self.updates, tf.compat.v1.GraphKeys.UPDATE_OPS
tf.Tensor(200.14644, shape=(), dtype=float32)

Conditional Upscale Convolution Gan
tf.Tensor(201.97002, shape=(), dtype=float32)

Conditional Transposed Convolution VAE
tf.Tensor(262.48502, shape=(), dtype=float32)

Conditional Upscale Convolution VAE
tf.Tensor(268.64572, shape=(), dtype=float32)
```

✓ 56s completed at 3:03 PM