Dissertation Notes:

There is a very long tail of creative applications that are difficult to anticipate but useful once they have been discovered. All of these applications (insert some applications) of generative models provide compelling reasons to invest time and resources into improving generative models.

Need one network for generating images and another for replacing a product in a frame of a video(an image). DCGAN for image generation? SRGAN for inserting into image?

SRGAN for deepfaking the product into the frame? SRGANs understand that there are multiple correct outputs (possible frames) rather than averaging over many answers to impose the single best output. (like a standard GAN)

There are many possible outputs for a frame. The model should choose an image that is a sample from the probability distribution rather than an image that is an average of all the possible images as it would yield a poor result.

Object transfiguration  
InstaGAN? (instance aware GAN) – object swapping  
CycleGAN?  
Recycle-GAN?

Generating synthetic images with DCGAN. Using the Keras Sequential API with Tensorflow 2 as the backend.

DCGAN:

Generator network – Generating fake images to be tested in the discriminator network  
Discriminator network – Takes a set or real images and fake from the generator network. Testing if the image generated by the generator network is real or fake

Minimax game theoretical formulation

This is a cycle. They train each other to get better results

Architecture guidelines for stable Deep Convolutional GANs:

* Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
* Use batchnorm in both the generator and the discriminator.
* Remove fully connected hidden layers for deeper architectures.
* Use ReLU activation in generator for all layers expect for the output, which uses Tanh.
* Use LeakyReLU activation in the discriminator for all layers.
* ^ other than output where a sigmoid is used for binary classification

CycleGAN:

Input dataset is made up of unpaired data, so to enforce that the network learns the correct mapping, an acceptable loss function must be used. Possible cycle consistency loss.

A two GAN system to deal with mode collapse. (When the network keeps outputting the same thing because the discriminator keeps classifying it as real)

For a cycleGAN to work the output needs to be like the input rather than creating an entirely new image.

Assume I am trying to convert an image containing a Pepsi can into an image containing a coke can

General Structure of the CycleGAN:

* Generator network to generate image of coke can
* Discriminator network to tell if the generated image is real or fake
* Second generator network to turn it back into a Pepsi can
* Second discriminator to tell if the Pepsi can is real or fake

The second generator is used to make sure the model is outputting the same image as the original rather than creating a separate image.

Image input  
 Generator G (create coke can)  
 Discriminator (test if coke can is real) (loss function)  
 Generator F (turning coke can back into Pepsi)  
 Discriminator (test if Pepsi can is real) (loss function)

The final loss function is to get the difference between the original and the image from Generator F

Essentially, one does the process of translating and one undoes it.

Creating the dataset:

Need approximately 1000 images per training set. (1k coke cans, 1k pepsi cans) 100-200 in the test sets. Using google images to acquire as many as possible although has proven a challenge to gather enough. Planning on taking pictures under different backgrounds and conditions and applying random image quality post processing in tensorflow with tf.image.random\_jpeg\_quality. I can also process the images in other ways to get a larger training and test set. Examples include transforms, noise, random cropping.