Final Presentation Script

Slide 1:

So pictured here we see what Leonardo just discussed: the architecture for a GAN that produces fake images. SO just to quickly recap, into our Generator we input a noise vector and it out puts fake images. These fake images along with some real images are fed into our discriminator. The discriminator will classify these images as either being real or fake. Based on this, we will repeat the process updating the generator's weights to increase the classification error and update the discriminator's weights to decrease classification error.

So basically, the process I just described can be represented by this equation here. The Discriminator D and the Generator G play this two-player mini-max game with value function V(G,D). I know this is a lot of variables, so lets try to break it down.

Slide 2:

Let's look at what these terms and variables represent. Z is the noise vector that we input into our Generator G. G(Z) then is the output of G given z, that is our fake images. X is a sample taken from our set of real images. Both X and G(Z), our real and fake images get input into our Discriminator D. D then outputs predicted labels, that is the probability that a sample came from the set of real images. So D(x) is the probability that x is a real image. And D(G(Z)) is the probability that G(Z) is a real image, as determined by the discriminator.

Slide 3:

So if we look at this first term, we see that it corresponds to the real images x. Basically for x from our set of real images we take log of D(X). And this second term corresponds to our fake images. Basically, for our noise vector z, from some random distribution, we plug in g(z), the fake image generated by inputting z into G into this term right here.

Slide 4:

So as we said, D of some input represents the probability that an input came from the real data rather than the generator. SO what the discriminator D is trying to do is maximize this value function. SO if you notice, D appears in both these terms. This first term is maximized when D(x) = 1. That is when D identifies a real sample x as being a real sample. This second term is maximized when D(G(Z)) = 0. That is when D identifies a fake sample as a fake sample. Thus the goal is to train D to maximize the probability of assigning the correct label to both training examples and samples from G.

Slide 5:

So what G is trying to do is minimize the value function. We see that G only appears in this second term. So this second term is minimized when D(G(z)) is close to 1. So basically when the discriminator D identifies a fake example G(Z) as being real. So the goal is to train the generator G into fooling the Discriminator, that is generating fake images the discriminator identifies as real. So this is basically what we went over with the analogy of the counterfeiter and the officer, we want to train the Discriminator to correctly identify real and fake samples while simultaneously training the Generator to fool the Discriminator into identifying fake samples as real.

Slide 6:

So on this slide we see the actual algorithm for training a GAN. It consists of two nested loops, the inner loop including the steps for training the discriminator and the outer loop the steps for training the generator. In the inner loop, to train the discriminator, we take m noise samples and m real samples then we update the discriminator by ascending its stochastic gradient according to the gradient of the objective. In the outer loop, to train the generator, we take m noise samples then we update the generator by descending its stochastic gradient according to the gradient of the objective.

Slide 7:

So basically what this algorithm is doing is exactly what we talked about a few slides ago. The discriminator and the generator are playing a mini-max game with value function V(G,D). The goal of the discriminator is to maximize V. Which is what's happening in this inner loop. On every iteration, we're ascending, that is going up the discriminator's stochastic gradient trying to find the values that maximize V.

Slide 8:

The goal of the generator is to minimize V. And that's what's happening in the outer loop. On every iteration, we're descending, that is going down the generators stochastic gradient trying to find the values that minimize V. After a sufficient amount of iterations, that is a sufficient amount of training, the generator will have minimized V, that is the generator will have learned to produce realistic samples that can fool the discriminator.

Slide 9:

So I wanted to touch on a common issue that can arise during the training process for GANs. When using gradient descent to update our model parameters, local minima, points with low loss in a specific region, can cause something caused mode collapse. As an analogy, you can think of an artist that creates a well-received piece of art. If this artwork becomes popular, the artist may become adverse to taking risks and will just start producing similar works. Similarly, in a GAN, the generator might start focusing on producing a limited set of data patterns that deceive the discriminator. As a result of this, the samples the generator produces become repetitive. This is called mode collapse. On the bottom here we see an example of this. In this picture here we see the results of a GAN that was trained to produce samples based on the MNIST dataset, which is a dataset of hand written digits. On the first iteration, before training, it produces random samples that resemble noise. After 100k iterations, we see the GAN is producing more believable samples, but the digit 1 is over-represented. By a million iterations, the GAN is only producing 1's. It collapsed to a single mode. Which is bad because we want the GAN to generate diverse samples. SO there are some ways to mitigate this.

Slide 10:

The first strategy is the use of Wasserstein GANs. Wasserstein GANs or WGANs are an extension of traditional GANs that use the Wasserstein distance as the loss function instead of traditional cross-entropy. By using Wasserstein distance, WGANs provide a more stable and informative training signal, allowing for smoother learning and reduced mode collapse. As a result of this WGANS are effective in handling mode collapse and generating more diverse and realistic samples.

Another Strategy is the use of Unrolled GANs. Unrolled GANs use a generator loss function that takes into account not only the current discriminator's classifications but also the outputs of future discriminator versions. This prevents the generator from over-optimizing for a single discriminator and reduces the likelihood of mode collapse.

Slide 11:

So now we're going to look at the adversarial training process. So, we recall that the generator is trained to maximize the final classification error and the discriminator is trained to minimize the final classification error. For this reason, the classification error is the reference metric for the training of both networks. The ultimate goal of this system is for the generator to learn to produce fake data that the discriminator will be unable to distinguish from real data. So this diagram represents the training during the first iterations, at least an abstraction of it. Real data would be a lot more complex than this, but this diagram will at least give us some intuition on how the training process would unfold. In the forward propagation, noise is fed into the generative network from some distribution. Then the generative network produces fake data from these random inputs. These orange points represent the fake data and the blue points the real data. We note that at this point the generator hasn't been trained, so the fake data doesn't resemble the real data at all yet. Both the real data and the fake data are fed into the discriminator, and the discriminator separates the real data from the fake data. In the back propagation, we update the discriminator's weights to decrease the classification error and also update the generators weights to increase classification error.

Slide 12:

After several iterations our system might look like this. Now after inputting noise into our generator, we get some fake data points with a distribution closer to our real data points. At this point it becomes harder for our discriminator to correctly separate the fake data and the real data. Thus the classification error has increased.

Slide 13:

ON the final iterations, our system will look like this: now when putting noise into our generator, it outputs fake data that closely follows the distribution of the real data. Now when inputting this fake data and the real data into the discriminator, it will be unable to distinguish the two. Ideally, at this point it will predict true or fake with a probability of ½ for any point it receives. In other words, its prediction will be no better than guessing meaning the generator is able to produce fake data that appears real which is our goal. Thus training will be complete. Now I'll pass it over to my group member who will talk about some variations on standard GANs.