

Deep Learning for Computer Vision
Professor Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Generative Adversarial Networks: Part 02

(Refer Slide Time: 00:13)



GANs: Evaluating Optimality

$$\text{Objective: } \min_G \max_D \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]$$

$$\Rightarrow \min_G \max_D \int_x (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))) dx$$

(change of variable, expanding expectation)

$$\Rightarrow \min_G \int_x \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))) dx \quad (\text{taking max inside})$$

$$\text{Let } y = D(x); a = p_{data}; b = p_G$$

$$\text{Then, } f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}; f'(y) = 0 \Rightarrow y = \frac{a}{a+b} \quad (\text{local max})$$

$$\text{Optimal Discriminator: } D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$



Vineeth N B. (IIT-H)

§10.2 GANs

10 / 24

So, to evaluate optimality for training such a network, let us reconsider the objective, min over G, max over D, the expectation for data coming from the training distribution, $\log D(x)$ plus expectation for the generated samples, $\log(1 - D(G(z)))$, where z comes from your Gaussian prior for instance. Now, if we expanded out your expectation, you have min over G, max over D, integral over x, $(p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))) dx$.

We assume that the x in the second term is generated data, which we define as $p_G(x)$. So, this is simply an expansion of the Expectation term in terms of an integral. So, let us now take the max inside the integral. So, the only difference between step 1 and step 2 is that the max in step 1 has gone inside; otherwise, the rest of the terms are exactly the same.

Now, to understand what the max of such a function would be, let us try to write out using some variables. In general, let us say $y = D(x)$, $a = p_{data}$, and $b = p_G$. So, then you can write this entire integrand here as $f(y) = a \log y + b \log(1 - y)$. We ideally need to know when do you attain the maximum over D or maximum over y for such a function.

To find the maximum of a function, you need to take its derivative and set it to 0. So, let us do that \hat{f} , or the derivative of $f(y)$ is $a/y - b/(1 - y)$ that follows from derivatives of $\log y$ and $\log(1 - y)$ and setting this to 0, we get that the maximum is obtained at $y = \frac{a}{a+b}$.

What does that mean when we substitute back the variables? For a given x , the optimal discriminator is $D_G^*(x) = p_{data}(x) / (p_G(x) + p_{data}(x))$. Let us keep this in mind and continue to look at the objective. So, that is the optimal discriminator, which does not end the story because we also have a generator to think about.

(Refer Slide Time: 03:29)



GANs: Evaluating Optimality

$$\begin{aligned} & \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x))) dx \\ & \min_G \int_X \left(p_{data}(x) \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + p_G(x) \left[\log \left(1 - \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right) \right] \right) dx \\ & \min_G \int_X \left(p_{data}(x) \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + p_G(x) \left[\log \left(\frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) \right] \right) dx \\ & \min_G \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) \right] \right) \end{aligned}$$



Vineeth N B (IIT-H)

§10.2 GANs

11 / 24



GANs: Evaluating Optimality

$$\text{Objective: } \min_G \max_D \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]$$

$$\Rightarrow \min_G \max_D \int_x (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))) dx$$

(change of variable, expanding expectation)

$$\Rightarrow \min_G \int_x \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x))) dx \quad (\text{taking max inside})$$

$$\text{Let } y = D(x); a = p_{data}; b = p_G$$

$$\text{Then, } f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}; f'(y) = 0 \Rightarrow y = \frac{a}{a+b} \quad (\text{local max})$$

$$\text{Optimal Discriminator: } D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$



Vineeth N B. (IIT-H)

§10.2 GANs


10 / 24

Let us now look at the generator side of the optimality. So, you have a min of G, of $p_{data}(x) \log D_G^*(x)$, the optimal discriminator, let us assume, the max of D has been evaluated, and we substitute for a max of D inside the integral, plus $p_G(x) (1 - D_G^*(x)) dx$. This can now be written as, we replace D_G^* , the optimal discriminator, as $p_{data}(x) / (p_{data}(x) + p_G(x))$, which we got from the previous slide, which is replacing for that and we also replaced for that in the second term.

Now, from here, we can see that the second term, $1 - p_{data}(x) / (p_{data}(x) + p_G(x))$, can now be rewritten as $p_G(x) / (p_{data}(x) + p_G(x))$, a simple arithmetic operation on top of the earlier expression. Now we are going to get back this expression in terms of expectations.

We are going to bring back expectations from integral. So, the integral over the entire term dx can be now rewritten as, expectation over x coming from p_{data} , $\log [p_{data}(x) / (p_{data}(x) + p_G(x))]$, plus, the expectation of x coming from p_G , which is the generated distribution, $\log [p_G(x) / (p_{data}(x) + p_G(x))]$. What do we do with this?

(Refer Slide Time: 05:13)



GANs: Evaluating Optimality

$$\min_G \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{2 * (p_{data}(x) + p_G(x))} \right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{2 * p_G(x)}{2 * (p_{data}(x) + p_G(x))} \right) \right] \right)$$

$$\min_G \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right) \right] - \log 4 \right)$$

$$\min_G \left(\text{KL} \left(p_{data}(x), \frac{p_{data}(x) + p_G(x)}{2} \right) + \text{KL} \left(p_G(x), \frac{p_{data}(x) + p_G(x)}{2} \right) - \log 4 \right)$$


Kullback-Leibler Divergence

$$\text{KL}(p, q) = \mathbb{E}_{x \sim p} \left[\frac{\log p(x)}{\log q(x)} \right]$$

Jenson-Shannon Divergence

$$\text{JSD}(p_{data}, p_G) = \frac{\text{KL}(p_{data}, \frac{p_{data} + p_G}{2})}{2} + \frac{\text{KL}(p_G, \frac{p_{data} + p_G}{2})}{2}$$

$\min_G (2 * \text{JSD}(p_{data}, p_G) - \log 4) \implies$ expression is minimized when $p_{data} = p_G$
 (Recall $\text{JSD} \geq 0$ by definition)



Vineeth N B (IIT-H)
§10.2 GANs
12 / 24


Let us now multiply and divide both of these terms by 2. Once we do that, we can take the denominator's 2 and add those two terms up and get a minus. Using the first 2, you would get a minus $\log 2$. Using the second 2, you will get a minus $\log 2$. When you put these two together, you will have a minus $\log 4$ term. Now, the first term can be written as a KL-divergence between $p_{data}(x)$ and $(p_{data}(x) + p_G(x))/2$.

Remember, this would be $\log(p/q)$, and you will be left with $p_{data}(x)$ and $(p_{data}(x) + p_G(x))/2$. Similarly, the second term would be the KL divergence between $p_G(x)$ and $(p_{data}(x) + p_G(x))/2$. You still have the minus $\log 4$. Now, let us briefly review some standard notations and definitions. Remember, KL divergence is given by, in this case, to simplify things, the expectation of x belonging to p , instead of $p \log(p/q)$, I can write it as expectation of x belonging to p , $\log p$ by $\log q$.

Jenson-Shannon Divergence is another divergence measure to measure the distance between two probability distributions. Given two distributions p_{data} and p_G , the Jenson-Shannon divergence between those distributions is given by $\text{KL}(p_{data}, (p_{data} + p_G)/2)/2$ plus $\text{KL}(p_G, (p_{data} + p_G)/2)/2$. So, this would be the Jenson-Shannon divergence between these two distributions, which means now we can replace the KL divergences in our case.


So remember, you would have had a by 2 and by 2. So, to ensure that this now becomes min over G, $2 * JSD(p_{data}, p_G) - \log 4$. Remember that Jensen-Shannon divergence, just like KL is also a non-negative quantity by definition. Now, what does this mean? Let us put all things together now.

(Refer Slide Time: 07:53)



Optimality of GANs: Conclusions


- $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$ (Optimal Discriminator for any G)



Vineeth N B (IIT-H) §10.2 GANs 13 / 24

We already saw that the optimal discriminator is given by $p_{data}(x) / (p_{data}(x) + p_G(x))$.

(Refer Slide Time: 08:03)



GANs: Evaluating Optimality

$$\min_G \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{2 * (p_{data}(x) + p_G(x))} \right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{2 * p_G(x)}{2 * (p_{data}(x) + p_G(x))} \right) \right] \right)$$

$$\min_G \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + \mathbb{E}_{x \sim p_G} \left[\log \left(\frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right) \right] - \log 4 \right)$$

$$\min_G \left(\text{KL}(p_{data}(x), \frac{p_{data}(x) + p_G(x)}{2}) + \text{KL}(p_G(x), \frac{p_{data}(x) + p_G(x)}{2}) - \log 4 \right)$$


Kullback-Leibler Divergence

$$\text{KL}(p, q) = \mathbb{E}_{x \sim p} \left[\log \frac{p(x)}{q(x)} \right]$$

Jensen-Shannon Divergence

$$\text{JSD}(p_{data}, p_G) = \frac{\text{KL}(p_{data}, \frac{p_{data} + p_G}{2})}{2} + \frac{\text{KL}(p_G, \frac{p_{data} + p_G}{2})}{2}$$

$\min_G (2 * JSD(p_{data}, p_G) - \log 4) \implies$ expression is minimized when $p_{data} = p_G$
 (Recall $JSD \geq 0$ by definition)



Vineeth N B (IIT-H) §10.2 GANs 12 / 24

This particular term, minimizing over G , $2 * JSD(p_{data}, p_G) - \log 4$, because Jensen-Shannon divergence is a non-negative quantity. This would be minimized when $p_{data} = p_G$ because then this quantity will become 0, and you will be left with minus $\log 4$. So, the generator G is obtained when $p_{data} = p_G$. We knew this intuitively. But we also now see this mathematically.

(Refer Slide Time: 08:44)



Optimality of GANs: Conclusions

- $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$ (Optimal Discriminator for any G)
- $p_{data}(x) = p_G(x)$ (Optimal Generator for any D)
- $D_G^*(x) = \frac{p_G(x)}{p_G(x) + p_G(x)} = \frac{p_{data}(x)}{p_{data}(x) + p_{data}(x)} = \frac{1}{2}$



Vineeth N B. (IIT-H)



§10.2 GANs

13 / 24

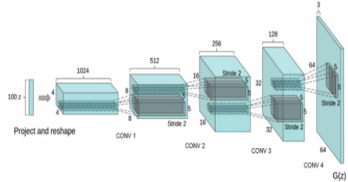
So, let us bring that back. We also know that at optimality for a generator, p_{data} , or the probability distribution of the training data is equal to p_G , which is the distribution of the generator. Now, putting the two together, it states that the optimal discriminator is also $p_G(x) / (p_G(x) + p_G(x))$ because $p_{data} = p_G$ at optimality. And that can also be written as $p_{data}(x) / (p_{data}(x) + p_{data}(x))$, both of which equate to half. At optimality, the discriminator should give you an output half to maintain the balance between generator and discriminator.

We do not want the discriminator always to give one or always give zero. Suppose it outputs half for any sample that is provided to it. We think it has been fooled because it cannot distinguish between a real sample and a fake sample.

(Refer Slide Time: 09:51)





Deep Convolution GAN (DCGAN)³



- Bridge the gap between success of CNNs for supervised learning and unsupervised generative models
- Best practices to enable stable training of GANs with deep architectures
- Smooth high-dimensional Interpolations; Vector arithmetic
- Demonstrate unsupervised feature learning capabilities of GANs and its applications

(Representation Learning)

 ³Radford et al, Unsupervised Representation learning with deep convolutional GANs, NeurIPS 2016
Vineeth N.B. (IIT-H) §10.2 GANs 14 / 24

So, that is about GANs. A follow-up architecture developed in 2016 by Radford et al. was called the Deep Convolution GAN, or the DCGAN. DCGAN was a landmark achievement in improvements over GANs because it came up with a few different ways of improving the generation quality of images using GANs. The main idea of DCGAN was to bridge the gap between the success of CNNs for supervised learning and unsupervised generative models.

It brings best practices of training CNNs to train GANs with deep architectures. They also show that you can play with the latent space representations, the z vectors, you sample from a Gaussian and do what is called Vector Arithmetic, which we will see soon. They also show how using such an architecture for GANs gives you strong feature learning capabilities of the network. Let us see each of these one by one.

(Refer Slide Time: 11:08)



DCGAN: Training Practices for Deeper GANs⁴

- Replaces deterministic spatial pooling functions (e.g maxpooling) with **strided convolutions** → allows to learn spatial downsampling
- **Remove fully connected hidden layers** for deeper architectures
- **Batchnorm in G and D** → prevent generator collapse/enable gradient flow in deeper architectures; not applied at output of G and input of D
- **Non-Linearity**: ReLU for generator, Leaky-ReLU (0.2) for discriminator
- **Output Non-Linearity**: tanh for generator, sigmoid for discriminator



⁴Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Vineeth N.B. (IIT-H)

§10.2 GANs

15 / 24





So, in terms of training practices to train good GANs for generation, DCGAN introduced a few strategies. It replaced deterministic spatial pooling functions such as maxpool with strided convolution, which allowed to learn spatial downsampling. It removed fully connected hidden layers for deeper architectures. So, just convolution layers, nothing more. It introduced batch normalization in both the generator and the discriminator.

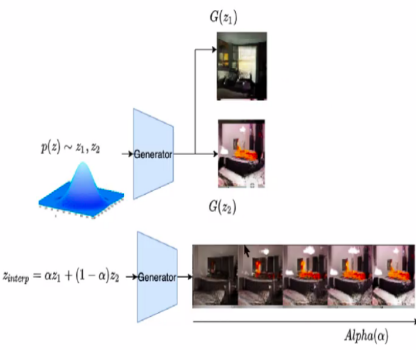
This helps prevent generator collapse or helps gradient flow in deep architectures. However, batch normalization was not applied at the output of G and the input of D. It used a ReLU, non-linearity for the generator and a leaky ReLU non-linearity for the discriminator. And finally, for output for the generator, it used a *tanh* non-linearity that was the final output of the generator. And, as before, a sigmoid activation was used in the output layer of the discriminator. That is to say, whether an input is real or fake.

(Refer Slide Time: 12:29)

Walking the Latent Space



- We can interpolate between two points in latent space and visualize
- Smooth transition in generated image space by changing latent (perceptual) features is a sign of a good model



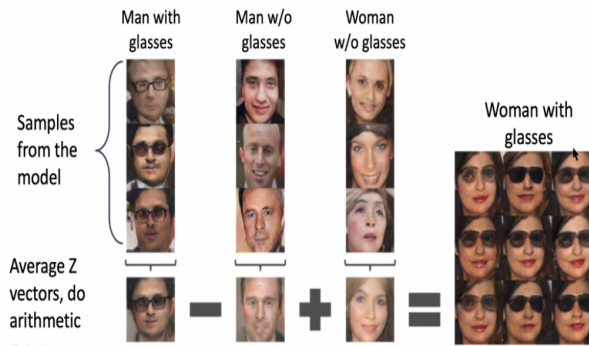
Vineeth N.B. (IIT-H) §10.2 GANs 16 / 24

So, we talked about playing with the latent space to generate different kinds of images. So, this is an interesting experiment. What DCGANs demonstrated is that if you had two latent samples, z_1 and z_2 , that you sampled from the Gaussian. You sent it through the generator. You would get two different outputs because that is what two latent are sampled for. If you now interpolated between those outputs, so if you did $\alpha * z_1 + (1 - \alpha) * z_2$, you end up getting a smooth transition of generated images from image $G(z_1)$ to $G(z_2)$.

(Refer Slide Time: 13:20)



Vector Arithmetic in Latent Space⁵



⁵Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Vineeth N B. (IIT-H)

§10.2 GANs

17 / 24

You could now look at it as doing vector arithmetic in latent space. If you had a set of images of a man with glasses, a set of images of a man without glasses, and a set of images of women without glasses. You can take the latent vectors corresponding to all men without glasses and average it. Similarly, take an average vector for all men without glasses and women without glasses.

Now, if you do arithmetic on those average z vectors or latent vectors, if you say a man with glasses minus the latent vector corresponding to man without glasses, plus the latent z vector corresponding to a woman without glasses, and use the resultant latent vector and pass it through the generator, you end up getting images of a woman with glasses. This is interesting to understand how the latent variable is interpolated, and the generator learns what the equivalent interpolation in the image space should be.

(Refer Slide Time: 14:43)



Pose Transformation⁶

- z_{left} : averaged latent vector of faces looking left
- z_{right} : average latent vector of faces looking right
- $z_{turn} = z_{right} - z_{left}$
- $z_{new} = z + \alpha z_{turn}$
- $G(z_{new}) \Rightarrow$ Transformed image



⁶Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Here is another example of a similar idea for Pose transformation. So, given a set of images looking left, and its corresponding average latent vector to be z_{left} , a set of images looking right, corresponding latent vectors averaged to form z_{right} . If you consider, $z_{turn} = z_{right} - z_{left}$, the difference between the extreme vectors, and z_{new} , a new latent sample.

Remember that z 's are all inputs to the generator. z_{new} is $z + \alpha \cdot z_{turn}$, and now you provide the z_{new} to G , you get transformed images, with various poses between the right and the left pose.

(Refer Slide Time: 15:39)



Feature Learning⁷

Table 1: CIFAR-10 classification results using our pre-trained model. Our DCGAN is not pre-trained on CIFAR-10, but on Imagenet-1k, and the features are used to classify CIFAR-10 images.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% ($\pm 0.7\%$)	4800
3 Layer K-means Learned RF	82.0%	70.7% ($\pm 0.7\%$)	3200
View Invariant K-means	81.9%	72.6% ($\pm 0.7\%$)	6400
Exemplar CNN	84.3%	77.4% ($\pm 0.2\%$)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% ($\pm 0.4\%$)	512

- Train DCGAN on ImageNet-1k
- Use discriminator's convolution features for supervised tasks i.e. classify images from CIFAR-10 dataset
- DCGAN never trained on CIFAR-10 \Rightarrow representation power and domain robustness of learned features
- Competitive results when compared with other feature learning methods



⁷Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

As we mentioned earlier, this work also showed how GANs learn good features that can also be used for classification. This was demonstrated by training the DCGAN on ImageNet-1k and then using the discriminator's convolution features for images from another data set called the CIFAR-10 dataset. So, the GAN is not trained on CIFAR-10; after it is trained on ImageNet, you take CIFAR-10 dataset images. CIFAR-10 is another data set.

You pass those images through the discriminator of the GAN, and you now take its features at a particular layer of the discriminator and use these features with an SVM to classify those CIFAR-10 images into ten different classes. CIFAR-10 stands for a dataset with ten different classes. That is what is done in this particular experiment. You see that the result obtained is fairly competitive to many other contemporary methods at that time in 2016. So, this shows the robustness of the features learned by the discriminator.

(Refer Slide Time: 17:05)



GANs: How to evaluate?

- **Human judgement:** Good generator \implies images with distinctly recognizable objects; Semantically diverse samples
 - **Recognizable objects** \implies classifier predicts class of generated sample correctly with high confidence
 - **Semantic diversity** \implies generated samples must be evenly from all classes in train set
- **Prediction power:** How ImageNet pre-trained Inception Network V3 performs on these generated images
- Still an open research problem...



Vineeth N B. (IIT-H)

§10.2 GANs

20 / 24

Now, for the final discussion of this lecture, is how do you evaluate GANs? Because so far, for supervised learning, we could use accuracy. But in GANs, how do you evaluate these models? One is you could use Human judgment. How would you use Human judgment? You would say a good generator is one, which can generate images with distinctly recognizable objects, and it also generates semantically diverse samples.

How do you measure this? Recognizable objects would mean that an independent classifier would take these generated images and predict the class with high confidence. Semantic diversity would mean that the generator or the GAN generate samples of various classes in the training set, ideally, all classes in the training set. That is one way of evaluating GANs.

Another way of evaluating GANs is by looking at the prediction power. Say you take an ImageNet pre-trained Inception network V3 and see how it performs on the generated images to understand the quality of generated images. If it performs well, perhaps the images are fairly representative of a dataset, such as ImageNet. However, it is also to be kept in mind that the evaluation of generative models is still an open research problem. Some metrics are popularly followed, but there is still scope for improvement of these metrics.

(Refer Slide Time: 18:49)

Inception Score⁸

- Correlates with human judgement
- $p(y|x)$: Inception model/classifier
- $p(y)$: Label/class distribution of generated samples



⁸Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016

Vineeth N B. (IIT-H)

§10.2 GANs



21 / 24

One such metric that is popularly used is the Inception score, which is intended to correlate with human judgment. You consider two quantities, $p(y | x)$, the softmax output over all class labels of an Inception model, given a data point x . You also have $p(y)$, the generated samples' class distribution. What are you looking for here? We ideally want $p(y | x)$ to have a pointed distribution.

So, we would like one class label, the correct class label to be very high in the distribution, and all other class labels to have very low probabilities. That is what we want $p(y | x)$ to be. We ideally want such a distribution to be as far away from a uniform distribution. That tells us that the inception model is highly confident in recognising certain objects generated in a given image. But is this correct? Not necessarily.

We do not want it to be far away from a uniform distribution. We want it to be far away from how class labels are distributed among those generated images. That is the baseline for us because it is possible that the GAN model generated more images of a certain class and lesser images of another class. So, how the marginal distribution of the class labels in your generated images looks is the baseline. And you want the class distribution of a specific data point x to be as far away as from that baseline distribution for your generated images.

(Refer Slide Time: 20:56)




Inception Score⁸

- Correlates with human judgement
- $p(y|x)$: Inception model/classifier
- $p(y)$: Label/class distribution of generated samples
- **Inception Score (IS)**: $= \exp(\mathbb{E}_{x \sim p_g} [D_{KL}(p(y|x) || p(y))])$
 $= \exp(\mathbb{E}_{x \sim p_g, y \sim p(y|x)} [\log(p(y|x)) - \log(p(y))])$
 $= \exp(H(y) - H(y|x))$

$H(y)$: entropy of generated sample class labels (**Semantic diversity = high $H(y)$**)
 $H(y|x)$: entropy of class labels from classifier (**Distinctly Classifiable = low $H(y|x)$**)

- Higher IS \implies better generative model

⁸Radford et al, Unsupervised Representation Learning with Deep Convolutional GANs, NeurIPS 2016
Vineeth N B. (IIT-H) §10.2 GANs 21 / 24



So, the inception score is given by a KL divergence between $p(y | x)$ and $p(y)$. It can be written as $\log(p(y | x)) - \log(p(y))$, which can also be written as the entropy of y , $H(y)$, minus the entropy of y given x , $H(y | x)$. What are these quantities? $H(y)$ is the entropy of generated sample class labels. Remember, semantic diversity would mean that you have high entropy, $H(y)$. So, you are generating samples from all different classes.

High entropy means equally distributed class labels in your generator data distribution. $H(y | x)$ is the entropy of class labels from the classifier for different data points. If it is distinctly classifiable, $H(y | x)$ will be very low, which means one class label will dominate the output of the softmax activation function rather than all labels. What are we looking for? We are looking for a higher Inception score. We would like $p(y | x)$ to be as far away from $p(y)$ as possible.

(Refer Slide Time: 22:24)



Fréchet Inception Distance (FID)⁹

- **Drawback of IS:** Statistics of real-world samples are not used to compare with statistics of synthetic samples
- FID enables us to capture more subtle differences; measures diversity better
- **Intuition:** Distance between real-world data distribution and generating model's data distribution serves as performance measure for generative models; how?
- Embed image x into feature space (activations of Inception-v3 pool3 layer), and compute the Fréchet distance between two multivariate Gaussians:

$$d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + \text{Tr}(C + C_w - 2(C \cdot C_w)^2)$$

where m is mean, C is covariance of generated image features; m_w is mean, C_w is covariance of original image features

- Lower FID \implies better generative model



⁹Heusel et al, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NeurIPS 2017
Vineth N B (IIT-H) §10.2 GANs 22 / 24

While the Inception score is good, one of the limitations of the inception score is it does not consider the real data at all. Its metric is purely based on the generated distribution alone. But we know that the purpose of GAN was to ensure that the generated distribution is close to the real distribution. So, how do we come up with a metric to measure this? This is done by Fréchet Inception distance called FID score, which tries to address this need.

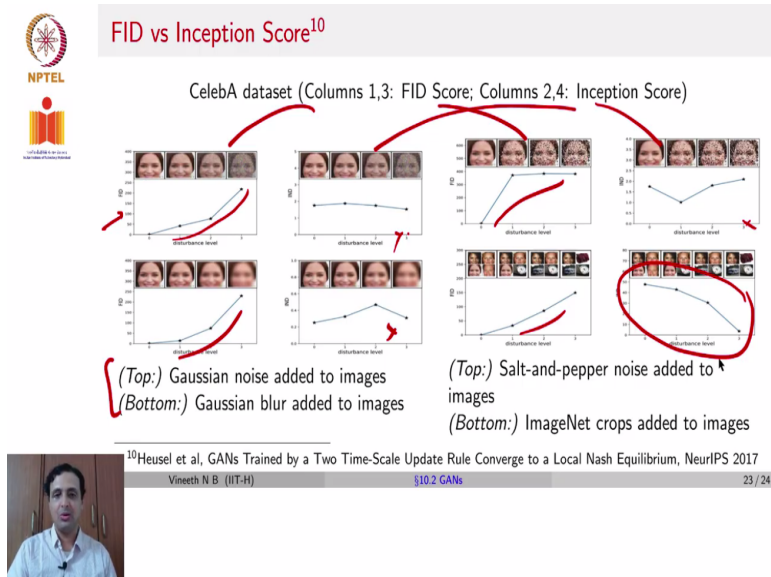
So, we need to find the distance between the real world data distribution and the generating models' data distribution. How do we do this? You take your real images, and you take your generated images and embed them in the feature space of an Inception V3 model. So, whatever images you have, your real training images, and your generated images pass all of them through an Inception network. You would get activations of the pool three layer of Inception v3.

Now, you compute the Fréchet distance between two multivariate Gaussians. Once you get those two sets of features, you compute the Fréchet distance. You are given a gaussian distribution with mean, m , covariance, C , and another distribution with mean m_w , and covariance C_w . The Fréchet distance is defined by the first mean minus the second mean 2 norm whole square plus trace of the first covariance plus the second covariance minus 2 into the 2 covariances squared.

So, that is the definition of the Fréchet distance between two multivariate Gaussians. So, given these two distributions of real and generated samples, you compute their means, compute the

covariances. And then, you can compute the Frechet distance using this formula. So, in this particular case, the lower FID, the better. Lower FID tells you that the generated distribution is close to the real distribution.

(Refer Slide Time: 24:56)



Let us see an example of how these metrics look when you use them. So, these are some results on a data set known as the CelebA, which contains face images of celebrities. In this particular example, the first column and the third column are FID scores, and the second column and the fourth column are Inception scores. So, remember the lower FID, the better, the higher Inception, the better.

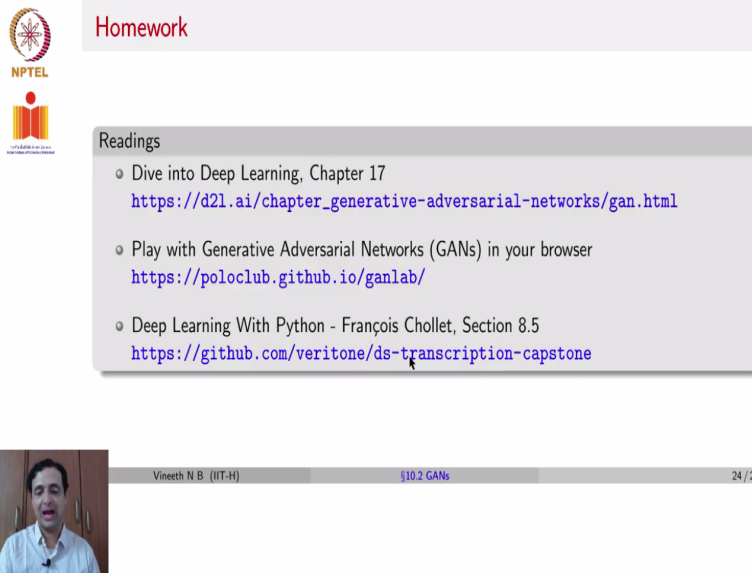
So, you can see here, in this particular case, the top row denotes images with added Gaussian noise. You can see that the FID score keeps increasing, which means it is getting worse and worse as you add more and more noise that is expected because your distribution is changing. Similarly, here too, when Gaussian Blur is added, the FID keeps increasing. Similarly, when salt and pepper noise is added, the FID keeps increasing.

And when ImageNet crops are added to the CelebA dataset, you take some images from ImageNet, crop certain portions and keep adding them to your CelebA dataset. Once again FID score goes up. On the other hand, you see the Inception score, which in this particular case, it is

evident that when the ImageNet crop is added, the inception score goes down over time, which is once again expected.

Remember, the higher Inception score is good. So, as more noise comes in, the inception score drops. However, for Gaussian noise, Gaussian Blur, and salt and pepper noise, you see that the Inception score does not show too much sensitivity, sensitivity to these kinds of noises in the distribution. There is some variation. It is not as stark as for other images.

(Refer Slide Time: 26:58)



The screenshot shows a presentation slide with a white background. On the left side, there is a logo for NPTEL (National Programme on Technology Enhanced Learning) featuring a stylized sun and the text 'NPTEL'. Below the logo is the text 'National Programme on Technology Enhanced Learning'. The main content of the slide is a grey box with a white border. At the top of this box, the word 'Homework' is written in red. Below it, the word 'Readings' is written in black. There are three bullet points, each with a small circle icon. The first bullet point is 'Dive into Deep Learning, Chapter 17' followed by the URL https://d2l.ai/chapter_generative-adversarial-networks/gan.html. The second bullet point is 'Play with Generative Adversarial Networks (GANs) in your browser' followed by the URL <https://poloclub.github.io/ganlab/>. The third bullet point is 'Deep Learning With Python - François Chollet, Section 8.5' followed by the URL <https://github.com/veritone/ds-transcription-capstone>. At the bottom of the slide, there is a small video window showing a man speaking, and a footer with the text 'Vineth N B (IIT-H) §10.2 GANs 24 / 24'.

With that, your readings for this GAN lecture is a very nice dive into Deep learning on GANs provided on this particular link. If you would like to play with GANs in your browser, here is a nice link called <https://poloclub.github.io/ganlab>. And finally, implementation of GAN on this particular link, which you can use to play with the code of GAN again.