

Deep Learning
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 57
Variational Autoencoder

Hello, welcome to the NPTEL online certification course on Deep Learning. You remember that whatever we have discussed so far that belongs to a class of the networks, where the purpose was to discriminate among different categories. Or in other words whatever we have learnt or the network has learnt based on that what the network does is given an input image or given an input object, the network tries to classify that input image or the input object to one of the known classes or one of the known categories.

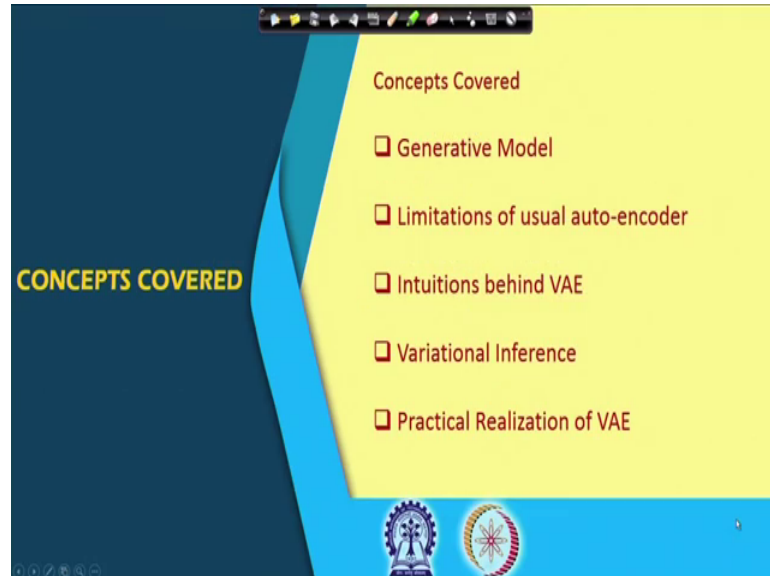
So, the neural network of the deep neural network has learned those different categories during the training process with the by means of a learning algorithm which was back propagation algorithm, wherein the gradient descent operation was followed where the gradient descent operation tried to minimize the loss function, where the loss function indicates that what is the accuracy of the decision that is taken by the neural network.

So, during training we give a pair, you give a training sample or in the training data which is labeled data. So, at the output you feed the data at the input and you find out what the neural network is giving at the output or what is the decision that the neural network is taking. And because we knew that what is the level of that particular data or what is the actual class belongingness of that particular data, so we can find out that whether that class belongingness and whatever the neural network was telling us was giving the decision whether these two decisions are matching or not.

So, if these two decisions match that is the actual class belongingness and the decision given by the neural network, if they match in that case we do not need to train the network anymore because the network is giving the correct decision. But in case the decision taken by the network does not match the actual class belongingness of the input data, then we have to train the neural network, so that the error that you get at the output that is minimized or the error is reduced to 0. And for that the approach that was taken was gradient descent approach and the algorithm was back propagation learning

algorithm. And the networks that we have discussed so far they are discriminatory in nature.

(Refer Slide Time: 03:05)



So, what we are going to discuss today is what is known as generative model. So, unlike the discriminative model, in case of generative model we give some description of the object or some description of the image and we expect that the network will be able to generate or will be able to create the object or will be able to create the image based on the description which has been given.

So, as we expect the network to generate the image, so the model is a generative model and while doing so, we will try to see that a closely related network that we have already discussed before which is an auto-encoder. So, we will try to see what are the limitations of the auto-encoder and the particular generative model that we will discuss today is what is known as variational auto-encoder.

So, we will try to find out or we will try to discuss what is the difference between auto-encoder and of variational auto-encoder, and then we will discuss about an algorithm which is known as variational inference which is used to train or which is the theory behind training of a variational auto-encoder. And then, we will also try to see that how the variational auto-encoder can actually be realized in practice using the neural networks.

(Refer Slide Time: 04:37)

The slide is titled "Generative Model" and features a list of attributes on the left side, each preceded by a checkbox:

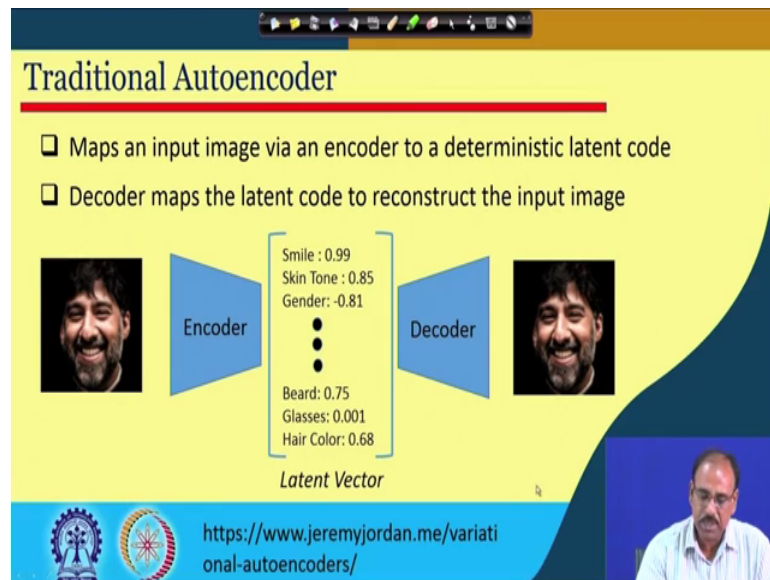
- Big Animal.
- Has four legs.
- Big ears.
- Long trunk.
- A pair of tusks
-

To the right of the list is a photograph of an elephant in a savanna setting. Below the list, the text "Latent Variables" is written in a cursive font, with a blue arrow pointing from it to the "....." checkbox. The slide also includes a navigation bar at the top, a logo at the bottom left, and a small video inset of a man at the bottom right.

So, let us first see that what is this generative model. Suppose, we wanted a network to generate an image of an animal where we give some description of the animal. So, we say that it is a big animal, so the animal has four legs, it has big ears, it has a long trunk, it has a pair of tusks and things like that. So, you can try to describe the animal with the help of these different attributes or these different descriptions.

So, naturally, given this description that that is a big animal, it has four legs, it has big ears, it has long trunk, it has a pair of tusks the kind of animal that will come to once mind immediately that it must be an elephant. So, you try to draw the picture of an elephant, right. So, that is what is the generative model. You are giving some descriptions and based on the descriptions the model tries to generate an image of the animal or the model tries to generate the object. And the descriptions that we are giving these descriptions are what is known as latent variables. So, all these latent variables putting together that becomes a latent vector or a latent descriptor of the object or the image.

(Refer Slide Time: 06:10)



So, once we have this as we have already shown before that a closely related network that we have already discussed is what is the autoencoder or will say call it a traditional autoencoder. So, what this autoencoder does? Given an input image or input data the autoencoder maps this input data to a latent code via an encoder network. So, this encoder network is what we have shown over here. So, here what is shown is the encoder network.

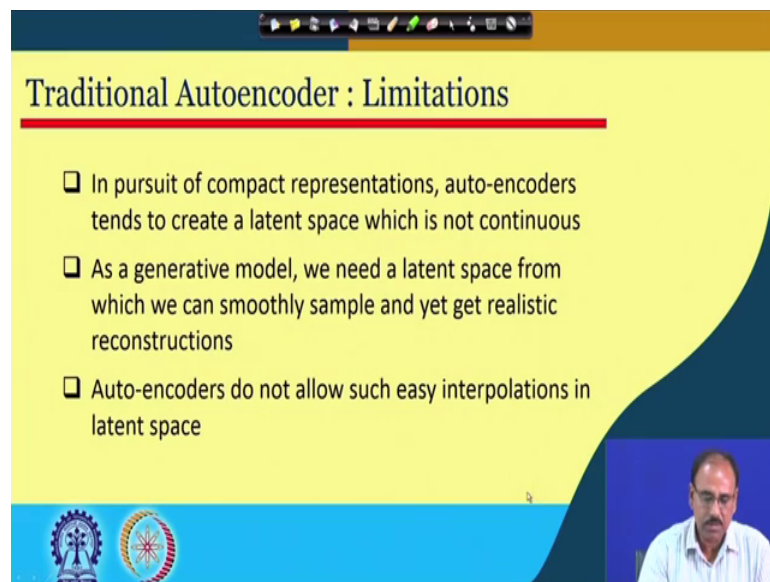
So, the autoencoder is generating a latent code say vector z through this encoder network, and then we have another network which is the decoder network that is on this part. The decoder network regenerates or creates the input image of the input object from this latent or code that has been generated by the encoder. And you remember that when we talked about the autoencoder we said that our purpose is that once you get a latent code from an encoder network, you simply forget the decoder part. So, this latent code is actually used for recognition or for classification purpose.

The decoder network is used for training time only, when you train the autoencoder, so that your latent code becomes a representative of the input data in order to ensure that you use the decoder network. But once your autoencoder is trained you simply forget about the decoder network. So, you simply use the input autoencoder, the autoencoder output which is the latent code and this latent code is later on used for classification purpose and we have said before that we can have different types of classifiers which

takes this latent input, this latent vector as an input and gives its decision as to which of the categories your input data belongs.

So, intuitively we can say that when this encoder is creating this latent vectors, it creates or gets the create the description of different attributes, say that the attributes may be smile, it may be skin tone, it may be gender, it may be beard, whether the person is wearing glasses, glasses, whether the hair color is black and so on. So, all these different descriptors or different attributes for each of these different attributes the autoencoder tries to get a numerical value and this is generated during the training process. So, given such autoencoder let us see that what is missing in this autoencoders.

(Refer Slide Time: 09:11)



The slide is titled "Traditional Autoencoder : Limitations" and features three bullet points. At the bottom left, there are two circular logos, and at the bottom right, there is a small video inset showing a man speaking.

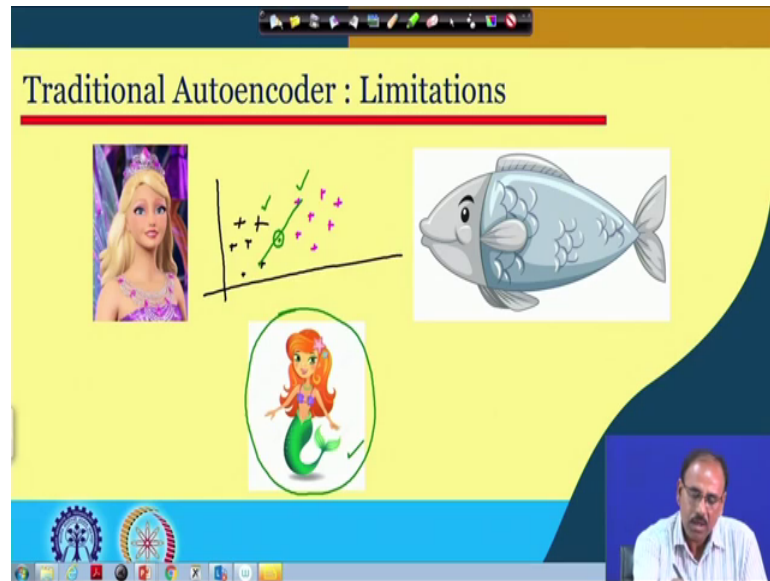
- ❑ In pursuit of compact representations, auto-encoders tends to create a latent space which is not continuous
- ❑ As a generative model, we need a latent space from which we can smoothly sample and yet get realistic reconstructions
- ❑ Auto-encoders do not allow such easy interpolations in latent space

So, basically what the autoencoder is giving? Autoencoder is giving us a code for an input data where this code is a compact representation of the input data, so that using this compact representation we can classify the input data. Whereas, what is required in case of generative model is our generative model is not for classification purpose.

So, in case of generative model we need a latent space, so that we can sample a vector as we can sample a code from that latent space and we can get a realistic reconstruction corresponding to that latent code which is given by the generator or the decoder. Usually or in general the auto encoders do not allow such is interpolation in latent space because the latent space which is generated by the traditional auto encoders are various parts, are not, they are not compact or they are not continuous.

So, this auto encoders are not really meant for generation or for creation of the images or creation of the data, they are actually meant for the classification purpose. Whereas, in case of a generative model I want a decoder, so that this decoder will get an input data or code a latent vector and from that vector it can generate an output which will be our realistic object or a realistic image.

(Refer Slide Time: 10:52)



So, what I mean is let us take this particular example. Suppose, I have an autoencoder model and autoencoder as we have seen that the encoder part of the autoencoder gives you a latent vector or a code which is a compact representation of the input vector. So, suppose we train this autoencoder using a large number of say Barbie dolls and using a large number of say fish, ok.

So, when you go to latent space suppose in the latent space all the Barbie dolls they are coded in a space there is something like this. Whereas, when you are training this autoencoder using fish maybe you are creating a set of vectors somewhere over here, so you find that in both these cases, it is quite sparse all the codes which are learnt by the autoencoder the codes are quite sparse.

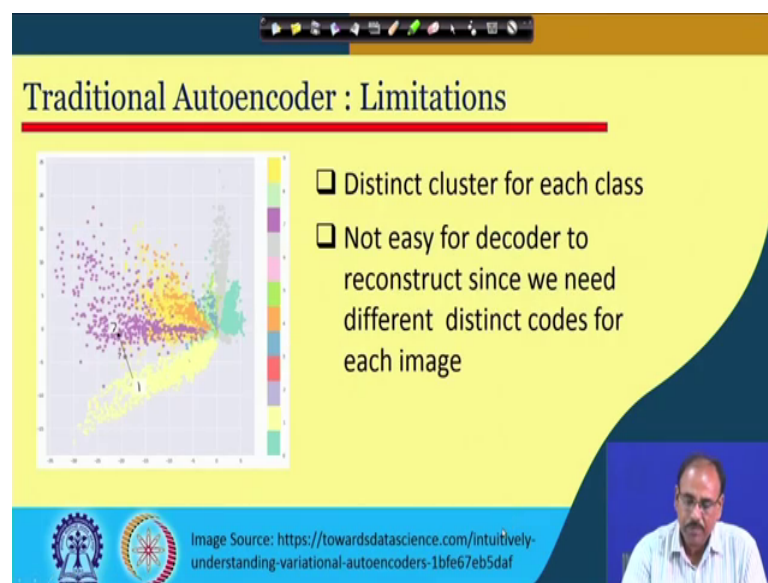
Now, what happens? If I take a point somewhere over here, so this gives me a latent code and I feed this latent code to the decoder to reconstructions. Now, you find that in case of autoencoder the autoencoder has never seen such a data. So, the autoencoder will be at loss to what is to create. So, that is why this big question marks. On the other hand, if I

go for variational autoencoder in that case what is expected of variational auto encoder is something like this, because in case of variational autoencoder it learns the distribution or the probability distribution of the vector z , so I can have a smooth interpolation between any set of vectors, ok.

So, this is just a hypothetical example that maybe if I use a variational autoencoder, then the variational autoencoder will try to interpolate between the Barbie dolls and the set of fishes, ok. So, somewhere over here it is in between fish and Barbie doll and it may create something like a Marbet something like this.

So, this is just an hypothetical example. And what we are trying to, what I am trying to say is that what is expected out of a generative model which in this case what we are discussing is a variational autoencoder.

(Refer Slide Time: 13:50)



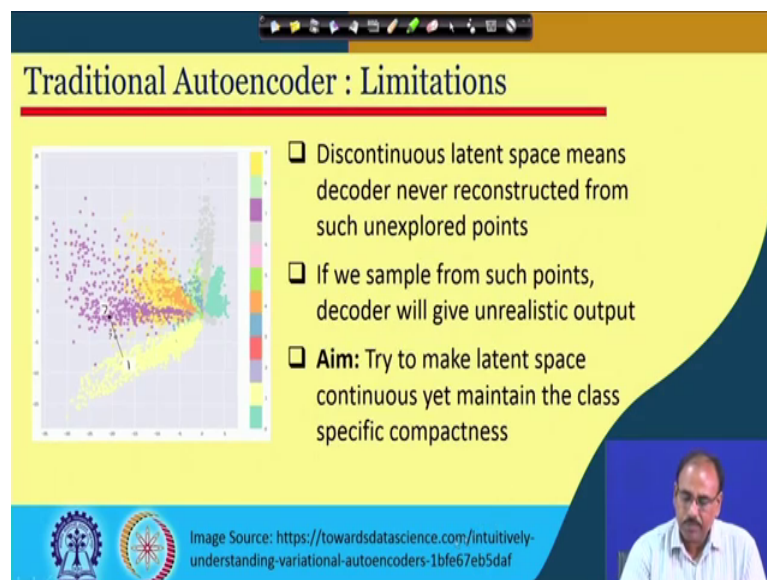
So, given this example now let us see that why autoencoder cannot give us such a kind of reconstruction. So, autoencoders, what they are giving is they are making us or they are giving us distinct cluster for each object class or each image class. So, as shown over here. So, you find that this is an output from the m nested database, where the feature space or the latent vector space is a two-dimensional vector space.

So, here you find that different numerals, m nested database if you remember that those are database of different handwritten numerals. So, you find that when you train an

autoencoder, the autoencoder gives you distinct clusters of the latent vectors z for different classes. So, you find that for one there is one cluster, for two there is another cluster and so on.

And these clusters of these vectors are quite sparse. So, as these are sparse it is not at all easy for a decoder to reconstruct because if I take a point somewhere over here, as shown over here, if I take a point over here which the encoder has never seen because the encoder has never seen this point of this vector it is not possible for that decoder to reconstruct what should be the corresponding object or what should the corresponding image. So, the decoder in case of autoencoder what it needs is it needs a distinct code for each image that is to be generated. So, that is the limitation of from the autoencoder.

(Refer Slide Time: 15:41)



The slide is titled "Traditional Autoencoder : Limitations" and features a scatter plot on the left showing distinct clusters of latent vectors. To the right of the plot is a list of three bullet points:

- ❑ Discontinuous latent space means decoder never reconstructed from such unexplored points
- ❑ If we sample from such points, decoder will give unrealistic output
- ❑ **Aim:** Try to make latent space continuous yet maintain the class specific compactness

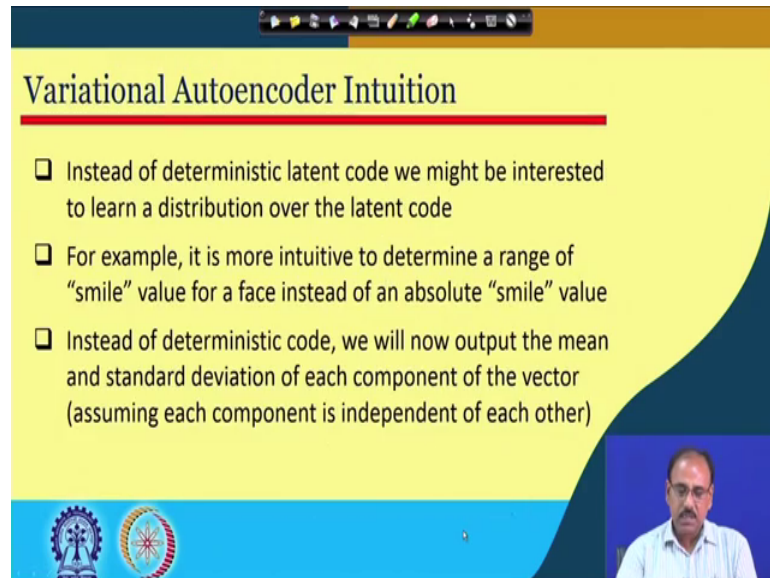
At the bottom of the slide, there is a small video inset of a man speaking, and a footer with logos and the text: "Image Source: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>"

So, as the autoencoder gives you discontinuous latent space so; that means, the decoder cannot reconstruct from such a space from a point which is actually unexplored. And if we sample from such points that decoder may give us some output it will give you some output, but the output may not be realistic at all. So, what we try to do is our aim is that we try to make this latent space continuous unlike in case of autoencoder, where the latent space is not continuous it is false.

So, in case of variational autoencoder, what we try to do is we try to make the latent space continuous and though it is continuous, but still in this continuous latent space for

different distributions of the latent vectors for different classes they maintain their specific class identity or the class compactness. So, let us see what does it mean.

(Refer Slide Time: 16:44)



The slide is titled "Variational Autoencoder Intuition" and features a yellow background with a dark blue curved shape on the right side. At the top, there is a navigation bar with various icons. Below the title, there are three bullet points:

- ❑ Instead of deterministic latent code we might be interested to learn a distribution over the latent code
- ❑ For example, it is more intuitive to determine a range of "smile" value for a face instead of an absolute "smile" value
- ❑ Instead of deterministic code, we will now output the mean and standard deviation of each component of the vector (assuming each component is independent of each other)

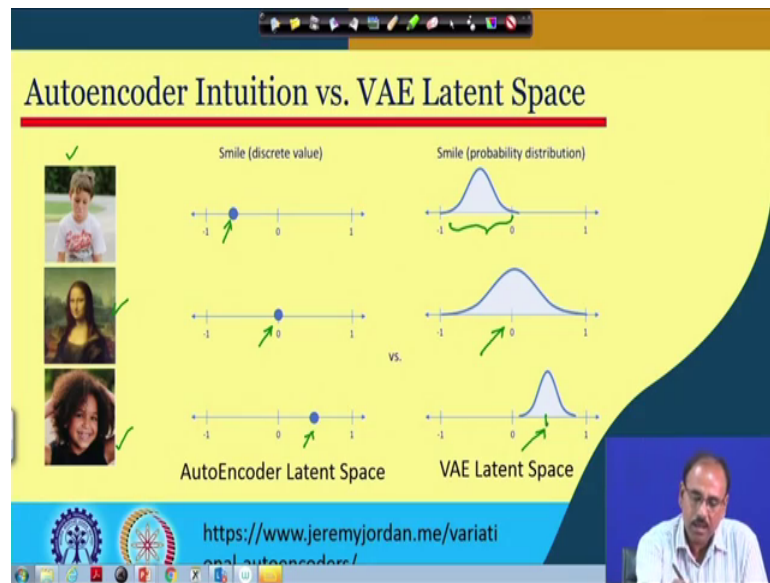
In the bottom right corner, there is a small video inset showing a man with glasses and a light blue shirt speaking. At the bottom left of the slide, there are two circular logos: one with a gear and a person, and another with a star and a gear.

So, what we want is that unlike in case of autoencoder, where autoencoder gives you a deterministic latent code for every input data in case of variational autoencoder we are not interested in the deterministic code, but rather what we are interested in is to learn the distribution over the latent code. So, here instead of giving a unique code for every input data, we want to have the distribution of the latent codes for the class of input data that we have.

For example, it might be more intuitive to determine a range of smile rather than given giving a specific absolute value to thus attribute smile, right. So, instead of deterministic code if this distribution is we assume it to be a Gaussian distribution, then for every class of input data or for every input data instead of getting a deterministic code, we would like to have the mean and standard deviation for each component of the vector the latent vector that you created.

And we can assume that every component is independent, so for a multi-dimensional case what I get is covariance matrix, the covariance matrix will be a diagonal matrix. So, this is what is expected out of our variational autoencoder.

(Refer Slide Time: 18:10)



So, more specifically taking this particular example which gives you the difference between an autoencoder and a variational autoencoder. So, here you find that for the first example case, here it is quite apparent that this kid is not smiling, right. So, what the autoencoder, the traditional autoencoder will give do is it will assign a value which is negative indicating that it is not a smiling face. What the variational autoencoder will do is instead of giving a specific value to this particular attribute smile, it will give you a range of values or it will give you a distribution of the smile variable.

Similarly, over here in this second example you find that the face is neither happy nor very happy. So, you can give autoencoder gives a distinct value which is maybe which may be around 0 to this particular attribute smile whereas, the variational autoencoder gives a range of values with a mean 0 and the distribution of a standard and the variance equal to 1.

Over here, this kid is smiling a lot, so autoencoder gives a positive value, as a positive value to this attribute smile whereas, variational autoencoder will give you a distribution with mean value which is positive somewhere over here and it has a certain distribution. So, this is what your variational autoencoder gives, generates as a latent variable it gives you a distribution. And once you have a distribution, now I can sample a vector from that distribution and that vector is fed to the decoder for reconstruction of the image.

(Refer Slide Time: 20:12)

Variational Autoencoder Intuition

- ❑ With this setup we can represent each latent factor as a probability distribution
- ❑ We can sample from such distribution
- ❑ Then the sampled vector can be passed through Decoder (Generator) to generate an image

So, now it is obvious that every input data is now represented as a for every input data the autoencoder; the variational autoencoder gives a probability distribution. And we can simply sample latent vector from that probability distribution and pass this latent vector to the decoder for reconstruction of the image or the reconstruction of the object. So, this is what is expected out of autoencoder, variational autoencoder.

(Refer Slide Time: 20:47)

Variational Autoencoder Intuition

Smile: [distribution]

Skin tone: [distribution]

Gender: [distribution]

Beard: [distribution]

Glasses: [distribution]

Hair color: [distribution]

Latent attributes

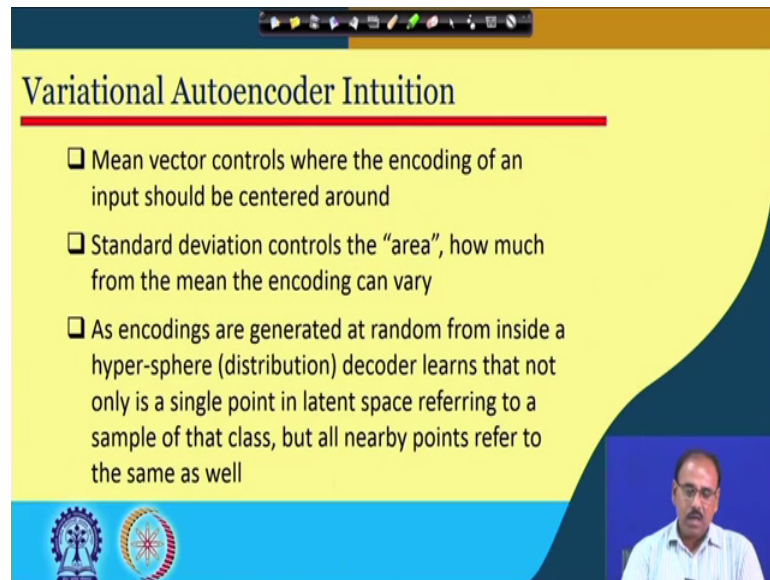
encoder → decoder

<https://www.jeremyjordan.me/variational-autoencoders/>

And this particular diagram simply tells you that the encoder part of the variational autoencoder gives you the distributions of different attributes. From this distribution you

can sample a vector; the vector is fed to the decoder and the decoder generates or the decoder reconstructs your image, and you find that your input image and the output means they are likely to be similar because they have to belong to the same class.

(Refer Slide Time: 21:22)



Variational Autoencoder Intuition

- ❑ Mean vector controls where the encoding of an input should be centered around
- ❑ Standard deviation controls the “area”, how much from the mean the encoding can vary
- ❑ As encodings are generated at random from inside a hyper-sphere (distribution) decoder learns that not only is a single point in latent space referring to a sample of that class, but all nearby points refer to the same as well

The slide features a yellow background with a blue and orange gradient on the right side. At the bottom left, there are two circular logos. At the bottom right, there is a small video inset showing a man with glasses and a white shirt speaking.

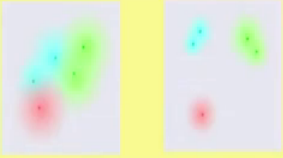
So, when you have this distribution of the latent variables, if I assume that it forms it gives you a Gaussian distribution then I will have a mean and a standard deviation. So, the mean of the vector, it mean vector that controls where the encoding of the input should be centered around, so it simply gives to the position of that particular distribution. And the standard deviation of your seed controls the spread of that particular attribute or it tells you the area or how much from the mean the encoding of that attribute should vary.

And as the encodings are generated at random from inside a hyper sphere as we are assuming that it is a Gaussian distribution, what the decoder learns is not only that a single point within that space for which and images to be generated. But the decoder learns that around that point or within a neighborhood of that point there is a sample belonging to that class and the decoder has to generate that sample. So, it is not from a single point within the within that hyper spherical space, but maybe within a neighborhood of any given point it gets latent vector and generates and output data corresponding to those pair of vectors.

(Refer Slide Time: 23:02)

Variational Autoencoder Intuition

- ❑ For smooth interpolations, ideally, we want overlap between samples that are not very similar too, in order to interpolate between classes.
- ❑ However μ and σ can take any value and learn to cluster the mean vectors of different classes far apart (and minimize σ) to reduce uncertainty for the Decoder



Our goal

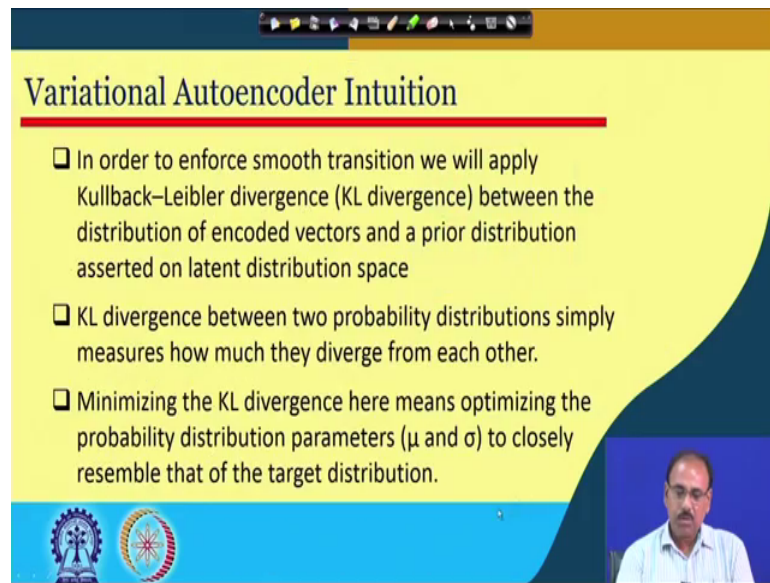
Network might converge to

Image Source: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>

So, now two things can happen. So, you find that we have two objectives; one is for every given data input data we want to generate a distribution the distribution will have a mean and a variance, and we want that the distributions should be very compact. So, that we can have smooth interpolation between samples and the decoder can generate the corresponding output.

So, for this possibly to make this smooth interpolation possible that distribution should be quite compact. However, the mean and the standard deviation that the variational autoencoder learns that may simply put that distributions of different classes far apart or the mean vectors for different classes may be far apart and if it so happens then the interpolation might be difficult, right.

(Refer Slide Time: 24:14)



Variational Autoencoder Intuition

- ❑ In order to enforce smooth transition we will apply Kullback–Leibler divergence (KL divergence) between the distribution of encoded vectors and a prior distribution asserted on latent distribution space
- ❑ KL divergence between two probability distributions simply measures how much they diverge from each other.
- ❑ Minimizing the KL divergence here means optimizing the probability distribution parameters (μ and σ) to closely resemble that of the target distribution.

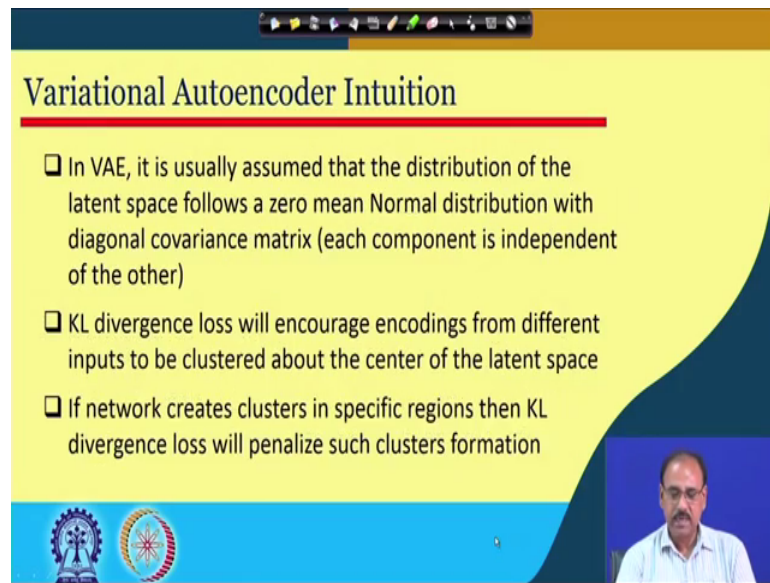
The slide features a yellow background with a red horizontal line under the title. At the bottom left, there are two circular logos. At the bottom right, there is a small video inset showing a man with glasses speaking.

So, what we want is in order to make sure that the distributions become compact, they are not far apart in your feature space in the latent space you can impose a prior distribution. So, by asserting this prior distribution we want that for every category of the objects for which we want the mean and standard deviation, so this distribution should be very close to a prior distribution or in other words we can measure what is the pullback library divergence or KL divergence between these two distributions and that KL divergence we will try to minimize.

What is the KL divergence? The KL divergence between two probability distribution, it simply measures how much is the divergence between those two distributions or we can simply put that it measures that distance between two distributions. We will see a bit later that that if what is the definition of the scale divergence and how we can compute the Leibler divergences.

So, if we minimize the scale divergence, and in most of the cases what we do is this prior distribution that we impose this is usually zero mean unit variance or unit standard deviation. So, by minimizing the KL divergence what we try to ensure is that the distributions of the feature vectors the distributions of the latent vectors for the data belonging to different classes they closely resemble that of the target distribution for the target distribution is a Gaussian distribution or normal distribution with zero mean and unit variance, and as they, as a result they become very compact.

(Refer Slide Time: 26:07)



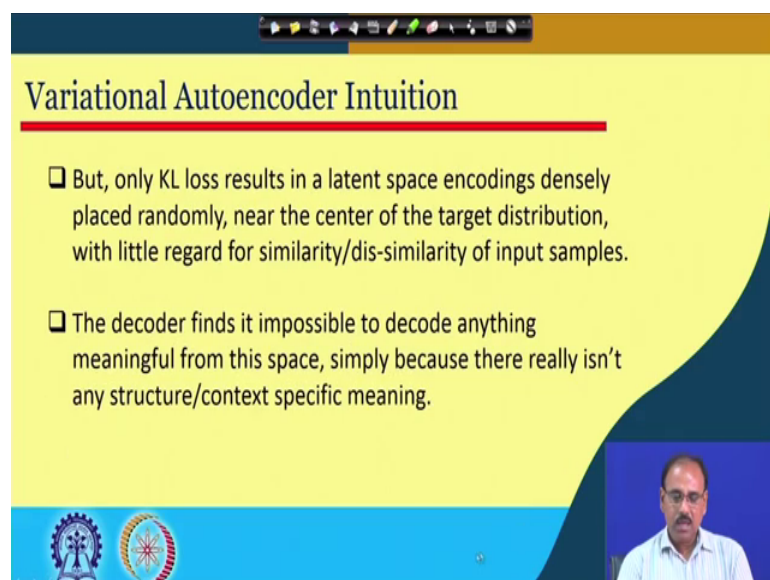
Variational Autoencoder Intuition

- ❑ In VAE, it is usually assumed that the distribution of the latent space follows a zero mean Normal distribution with diagonal covariance matrix (each component is independent of the other)
- ❑ KL divergence loss will encourage encodings from different inputs to be clustered about the center of the latent space
- ❑ If network creates clusters in specific regions then KL divergence loss will penalize such clusters formation

The slide features a yellow background with a blue and orange gradient on the right side. At the bottom left, there are two circular logos. A small video inset in the bottom right corner shows a man with glasses and a mustache, wearing a light blue shirt, speaking.

So, the KL divergence loss by minimizing that it encourages encoding from different inputs to be clustered about the center of the latent space because we have specified the target distribution to be zero mean and unit variance. Whereas, the other loss which is the data loss that will try to maintain the clusters or the compactness of the cluster of the vectors belonging to the same class of data.

(Refer Slide Time: 26:49)



Variational Autoencoder Intuition

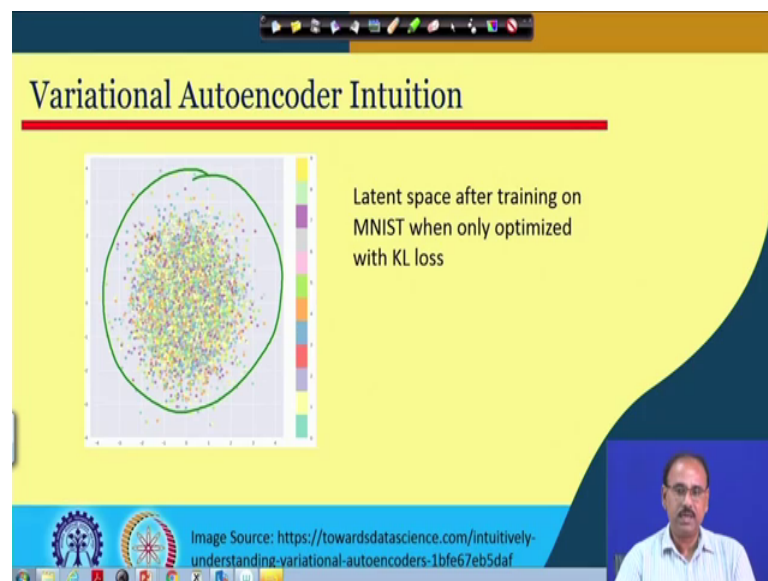
- ❑ But, only KL loss results in a latent space encodings densely placed randomly, near the center of the target distribution, with little regard for similarity/dis-similarity of input samples.
- ❑ The decoder finds it impossible to decode anything meaningful from this space, simply because there really isn't any structure/context specific meaning.

The slide features a yellow background with a blue and orange gradient on the right side. At the bottom left, there are two circular logos. A small video inset in the bottom right corner shows the same man from the previous slide, speaking.

So, we have to use both the KL divergence as well as the data loss together. If we use only one of them, if we use only KL divergence then basically what we are trying to do is

for the data or the encoding for different classes we want all those encoded vectors to have zero mean and unit variance. So, as a result what will happen is the structure that data will lose the structural property of the different classes, and if it loses the structural property it becomes difficult for the decoder to reconstruct (Refer Time: 27:31). Whereas, if we use only the data loss then the compactness of the distributions that we want in case of variational autoencoder or in case of generative model that may not be maintained.

(Refer Slide Time: 27:46)



So, if you use all of them together, so this example simply says this diagram simply shows that if I only use the KL loss or KL divergence loss, and I try to minimize that then you find that the data belonging to all different classes, they are clustered together, all of them are combined together. So, as a result you are losing the class identity or the structural difference between different data sets. So, this is what is not desirable.

(Refer Slide Time: 28:16)

Variational Autoencoder Intuition

- ❑ Optimizing reconstruction loss + KL divergence loss results in the generation of a latent space which maintains the similarity of nearby encodings on the local scale via clustering
- ❑ Yet globally, is very densely packed near the latent space origin

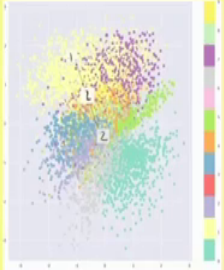
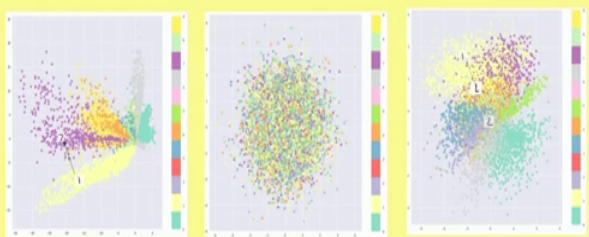


Image Source: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>

This is the one, where if I put both the KL divergence as well as the reconstruction loss and the data loss together, the KL divergence will try to make the distributions compact and the reconstruction loss or the data loss will try to maintain the cross belongingness or try to maintain the separability of the data belonging to different process. And that is what has been shown over here, you find that this set of data belongs to one class, this set of data belongs to one class, this set of data belongs to one class whereas, globally all these distributions form a very compact set of clusters and that is what is required in case of variational autoencoder.

(Refer Slide Time: 29:10)

Variational Autoencoder Intuition



Reconstruction Loss KL Divergence Loss KL Divergence + Reconstruction Loss

Image Source: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>

This diagram shows puts all this different conditions side by side. So, on the leftmost case what we have is when what is given in autoencoder where you use only the reconstruction loss. So, by minimizing the reconstruction loss, the traditional autoencoder gives you a set of latent vectors where the vectors belonging to different category form different clusters, but the clusters are not compact in the latent space.

The middle one shows if you use only the KL divergence loss in which case the data belonging to different classes they are all mixed together. Your cluster becomes very compact, but at the same time you lose the structural information; that means, the different data sets, they do not have, they cannot maintain their own identity. So, as a result this also becomes useless because given a sample from such a kind of distribution to the decoder, the decoder does not have any structural information because all the structural information is lost.

So, the data becomes useless to the decoder. So, what is required in our case is what is given on the right most one that you try to minimize the KL divergence and the reconstruction loss together. So, as a result you get a compact distribution whereas, in every distribution the identity of the data belonging to different classes they are also maintained.

So, we will stop here today. Today, what we have seen is that what we can get from a traditional autoencoder, what are the limitations of the traditional after encoder, why they cannot be used as a generative model, because traditional autoencoder gives you a latent vectors in a latent space or the latent space is very diverse, it is not compact and it is not continuous. So, as it is not continuous, if I sample a vector from that such a latent space and give it to the decoder the decoder does not know what is what to reconstruct construct or what to create.

So, in case of variational autoencoder, instead of generating a code for an input data the variational autoencoder actually gives you a distribution of the codes or distribution of the encoding. So, for reconstruction purpose what you do is your sample a latent vector from that distribution and pass that latent vector to the decoder and using this latent vector the decoder can reconstruct a meaningful data or a meaningful image. So, we will continue more with this variational autoencoder or generative model in our next lectures.

Thank you.