

## Outline

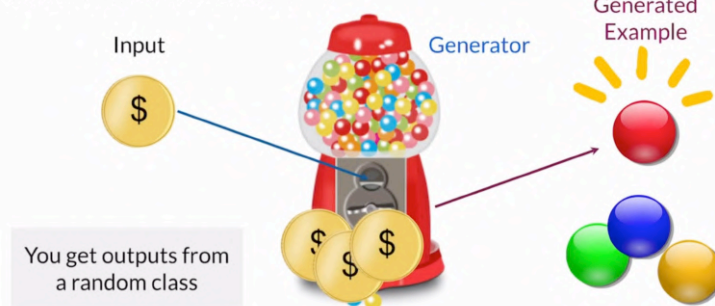
- Unconditional generation
- Conditional vs. unconditional generation

In the last couple of weeks, you've seen how GANs work and how to build them to produce examples that mimic your training dataset. During this week, you will see how to show you how to control the output and get examples from a particular class or for those examples to take on certain features. This is pretty fun.

In this section, you will review what unconditional generation is, and it's actually the one that you've been using this whole time and you'll also be introduced to conditional generation by comparison of them both.



### Unconditional Generation

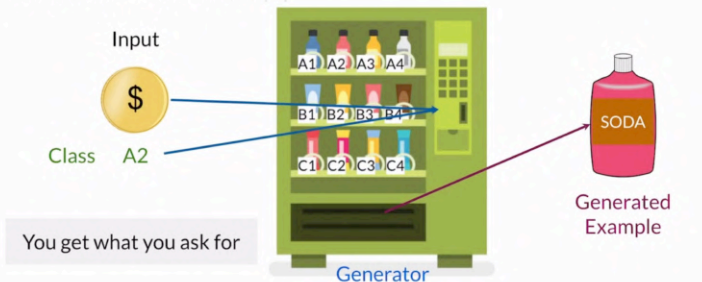


As a quick recap with unconditional generation, you get outputs from random classes. You can think of those as a gumball machine, where you input a coin and you get a random color gumball. If you want to get a gumball of a specific color, say red, you have to keep spinning coins until you get it.

In this example, the coin is like the random noise vector that your GAN uses for generation and the gumball machine is like the generator. Then the gumballs are the random outputs, those images you get.

You can see what color gumballs you might get, just like how you know what your GAN is trained on. You can't control what exact color of output you will get.

### Conditional Generation



On the other hand, conditional generation allows you to ask for an example from a specific class and you get what you ask for. It's like a vending machine.

You input a coin similar to the gumball machine but you input a coin along with a code for an item that you want. For example, A2 for a red soda. But note that you still don't control certain features of the soda bottle. You can't get the one with the latest expiration date or the bottle that's least damaged or the one that's filled up the most, you just get a random red soda. But it is a red soda, not a blue candy bar.

Here the coin and the code are the inputs for the GAN and the vending machine is the generator and the soda is the generated output.

With a conditional GAN, you get a random example from the class you specify. That class is this A2 soda here.

## Conditional vs. Unconditional Generation

Conditional	Unconditional
Examples from <b>the classes you want</b>	Examples from <i>random classes</i>
Training dataset needs to be <b>labeled</b>	Training dataset <i>doesn't need to be labeled</i>

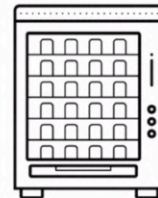
Now you have an idea of how conditional versus unconditional generation are related. Let's compare them a bit. With conditional generation, you can get generated examples from the classes you decide while with unconditional generation, you get examples from a random class.

As a result of that, with conditional generation, you actually have to train your GAN with labeled datasets and those labels are on the different classes you want while unconditional generation doesn't need any labels. You've seen this in previous weeks from the course, that you don't need any labels you just need a pile of real examples. You see how to modify your model for this conditional generation in the following lectures.

## Summary

- Conditional generation requires labeled datasets
- Examples can be generated for the selected class

What you should take away from this section is that conditional generation requires labeled datasets for training in order to learn how to produce examples from desired classes. Coming up, you will see how the labels from your dataset are fed to the generator and discriminator in order to train your GAN and produce examples from the desired class, like selecting the red soda from a vending machine.



deeplearning.ai

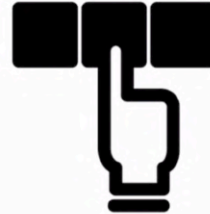
## Conditional Generation: Inputs

Conditional generation lets you produce examples from the classes you want. For doing that, you need to have a label data-set and somehow pass that class information to both the generator and the discriminator during training.

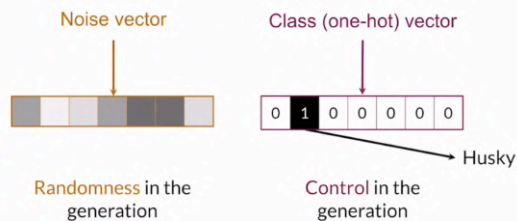
## Outline

- How to tell the generator what type of example to produce
- Input representation for the discriminator

Here you'll learn how to tell the generator which class to produce and how declassifying an example is passing through the discriminator along with the image to be classified.



### Generator Input



You've seen already with unconditional generation, the generator needs a noise vector to produce random examples. For conditional generation, you also need a vector to tell the generator from which class the generated examples should come from. Usually this is a one-hot vector, which means that there are zeros in every position except for one position corresponding to the class you want.

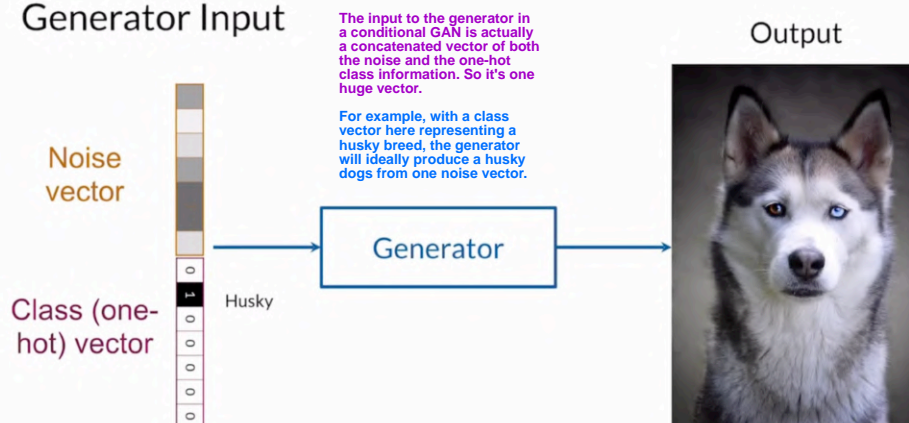
Remember that the noise vector is random values as well, but it doesn't have to be zero or one.

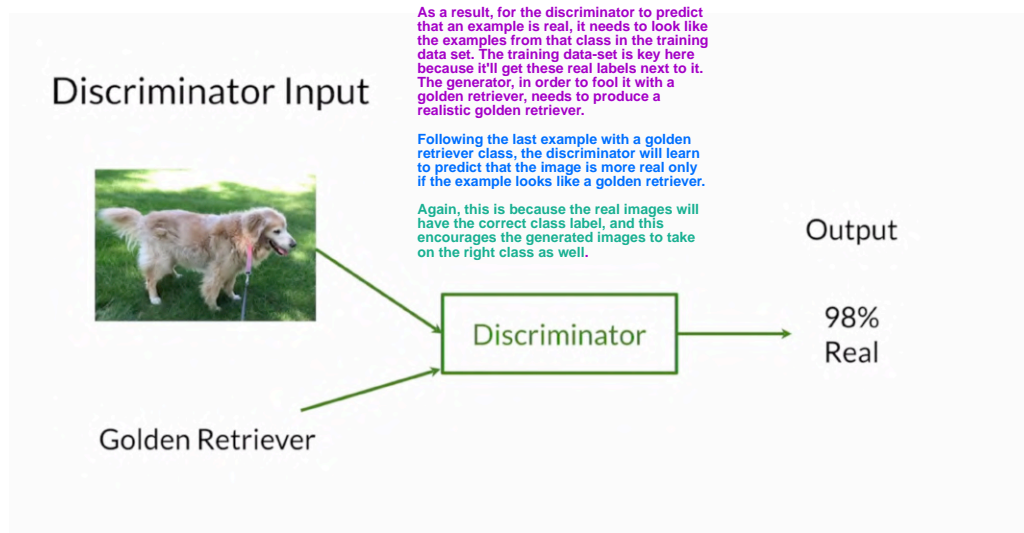
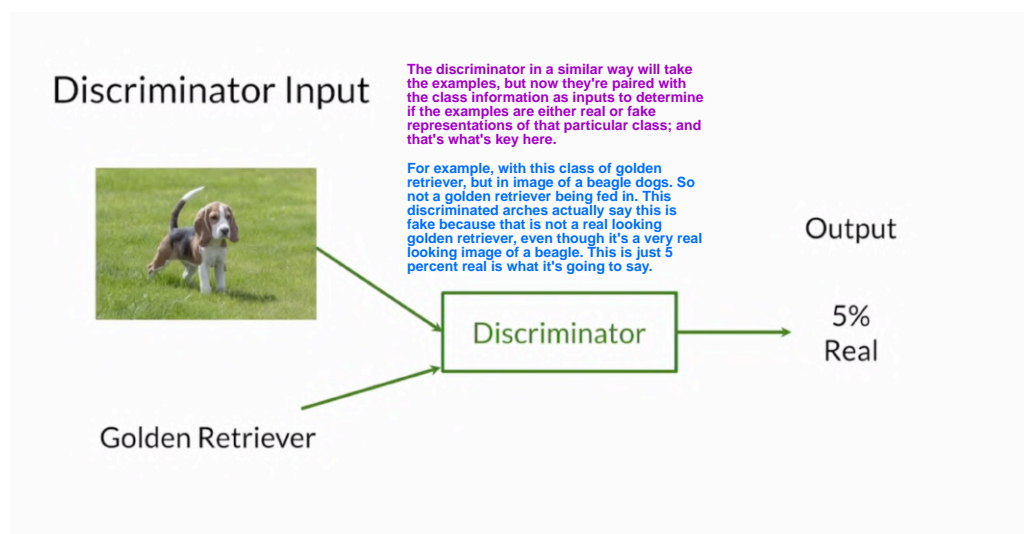
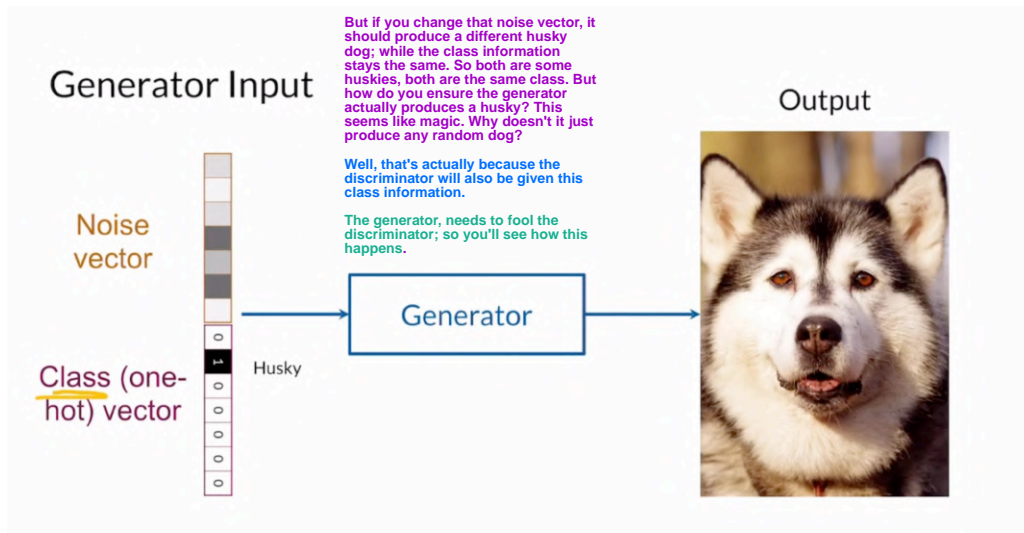
As an example, let's say all of these different values in the one-hot class vector are different dog breeds; and this second cell here corresponds to a husky. Specifying a position of one here means that you want the generator to produce a husky. If you had a one somewhere else and a zero on the husky cell, then you would want it to produce a different class; for example, a golden retriever.

Here the noise vector is the one that adds randomness in the generation, similar to before; to let you produce a diverse set of examples. But now it's a diverse set within the certain class, conditioned on the certain class and restricted by the second class.

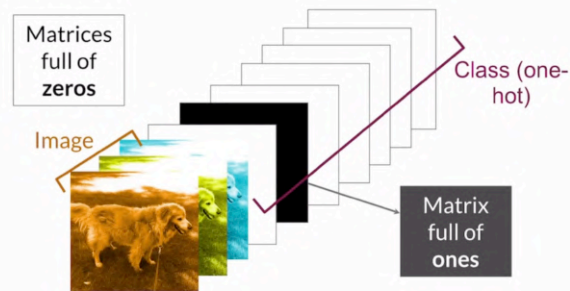
There's one-hot vector is now what lets you control what class to generate.

### Generator Input





## Discriminator Input



Digging a little bit deeper, the input to the discriminator is an image. How do you go about adding that class information? Well, the image is fed in as three different channels, RGB, red, green, blue, or just one channel if it's a gray-scale image. So that's this image component.

Then the one-hot class information could also be fed in as additional channels where all the channels take on values of all zeros; so all these white blocks of the same height and width as the image take on values of all zeros. Whereas this black one here will take on values of ones. In contrast to that one-hot vector, these are typically much larger matrices where each channel is full of zeros at every position where it's not that class.

There are many other less space consuming ways of doing this, like compressing this information in another format. You can even create a separate neural network head to do that for you, which would be prudent if you had many different classes.

But this approach definitely works for the seven or so classes you have here; and the model will learn that information.

## Summary

- The class is passed to the generator as one-hot vectors
- The class is passed to the discriminator as one-hot matrices
- The size of the vector and the number of matrices represent the number of classes

To sum up in conditional generation, you pass the class information to both models.

To the generator it's typically a one-hot vector concatenated with your noise vector to the discriminator when the desired output of the GANs are images, it's one-hot matrices representing the channels.

The size of the class vector and the number of extra channels for the class information is just the same as the number of classes you'll be turning on.



deeplearning.ai

## Controllable Generation

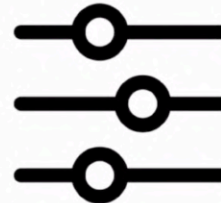
Another way to control the outputs produced by gans, is after it's been trained, or more generally, controllable generation. While conditional generation leverages labels during training, this section will focus on controlling what features you want in the output examples, even after the model has been trained.



## Outline

- What is controllable generation
- How it compares to conditional generation

You'll learn about controlling specific features, and how it compares with conditional generation you learned about in the previous section.



## Controllable Generation



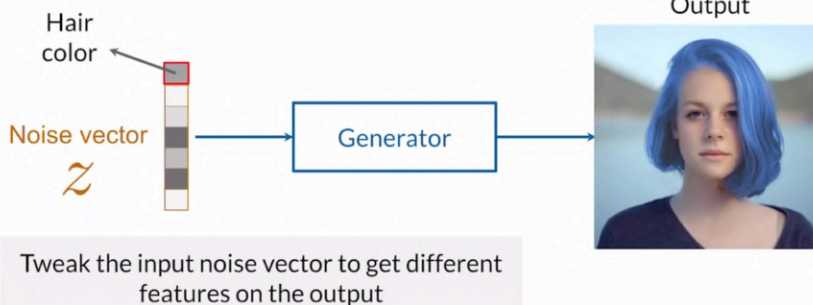
Change specific features of the output

Controllable generation allows you to control some of the features that you want in your output examples.

For instance, with a gan that performs face generation, you could control the age of the person's looks in the image or if they have sunglasses or the direction they're looking at in the picture, or they're perceived gender.

Shen, Y., Gu, J., Tang, X., & Zhou, B. (2020). Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9742-9752)

## Controllable Generation



You can do this by actually tweaking the input noise vector  $Z$ , that is fed to the generator after you train the model. For example, with the input noise vector  $Z$ , maybe you get this picture of a woman with red hair.

Let's say you tweak one of the features from this input noise vector here and Maybe now you get the same woman but with blue hair this time. Maybe its because this first element here, represents changing hair color. That would be super cool. You'll learn exactly how to tweak the  $Z$  in the following lecture.

## Controllable Generation vs. Conditional Generation

Controllable	Conditional
Examples with the <b>features that you want</b>	Examples from <i>the classes you want</i>
Training dataset <b>doesn't need to be labeled</b>	Training dataset <i>needs to be labeled</i>
<b>Manipulate the z-vector input</b>	<i>Append a class vector</i> to the input

But first, to get a better sense of control generation, I'll make a quick comparison with conditional generation. I'll use these terms here because that's typically what researchers mean if you check out the papers, though it's not as clearly delineated. Sometimes controllable degeneration can definitely include conditional generation because you're still controlling the gan in some way.

With controllable generation, you're able to get examples with the features that you want, like faces from people who look older with green hair and glasses.

With conditional generation, you get examples from the class that you want, like a human or bird. Of course, it could also be, I want to person with sunglasses on as well. So far, they're a bit similar.

But controllable generation typically means you want to control how much or how little of a feature you want. They're typically more continuous features like age.

Conditional generation on the other hand, allows you to specify what class you want to a very different type of thing. For this, you need to have a label data set and implemented during training typically. You probably don't want to label every hair length value, so, controllable generation will do that for you and it's more about finding directions of the features you want. That can happen after training.

Of course, controllable generation, you will sometimes also see it happening during training as well. To help nudge the model in a direction where it's easier to control.

Finally, as you just learned, controllable generation works by tweaking that input noise vector  $Z$  that's fed into the generator, while with conditional generation, you have to pass additional information representing the class that you want appended to that noise vector.

## Summary

- Controllable generation lets you control the features of the generated outputs
- It does not need a labeled training dataset
- The input vector is tweaked to get different features on the output

In summary, Controllable generation lets you control the features in the output from your gan. In contrast, with conditional generation, there's no need for a labeled training dataset. To change the output in some way with controllable generation, the input noise vector is tweaked in some other way. In following videos, I'll dig deeper into how exactly that works.



deeplearning.ai

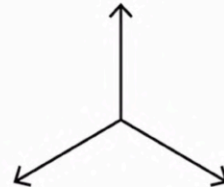
## Vector Algebra in the Z-Space

As you've seen in the previous video, controllable generation is achieved by manipulating the noise vector  $z$  that's fed into the generator. In this section, you will see the intuition behind that process.

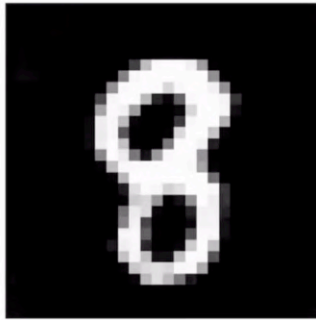
## Outline

- Interpolation in the z-space
- Modifying the noise vector  $z$  to control desired features

Review how to interpolate between two GAN outputs first and you'll learn how to manipulate noise vectors in order to control desired features in your outputs.



## Interpolation Using the Z-Space

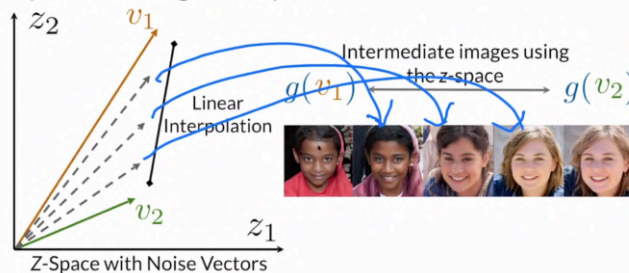


Controllable generation and interpolation are somewhat alike.

With interpolation, you get these intermediate examples between two generated observations. In practice, with interpolation, you can see how an image morphs into another, like in this GIF, where each digit from zero to nine morphs into the following one.

How an image morphs into another

## Interpolation Using the Z-space



What happens is that you get intermediate examples between the targets by manipulating the inputs from Z-space, which is just the name for the vector space of the noise vectors. You'll see later that this is the same idea behind control bot generation.

Just to be clear here,  $z_1$  and  $z_2$  are the two dimensions in this Z-space that you're looking at right now. As an example, there's a noise vector  $V_1$  and a noise vector  $V_2$ , where  $V_1$  could have a  $z_1$  value of, let's say five and a  $z_2$  value of let's say 10.

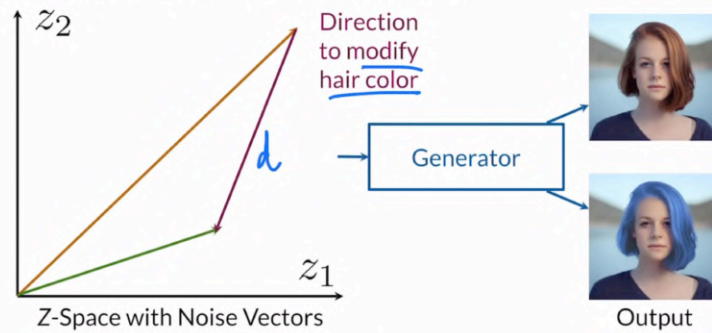
Then this is the vector 5,10 and then  $V_2$  has a smaller value, so four and two. So it's the vector 4, 2. That's what  $z_1$  and  $z_2$  are, just dimensions on the Z-space and the actual vectors  $V_1$  and  $V_2$  are going to represent concrete vector values in this Z-space.  $V_1$ , when you feed it into the generator, will produce this image here, and  $V_2$  when you feed it into the generator, will produce this image there.

If you want to get intermediate values between these two images, you can make an interpolation between their two input vectors,  $V_1$  and  $V_2$  in the Z-space, actually. This interpolation is often a linear interpolation. Of course, there are other ways to interpolate between these two vectors.

Then you can take all these intermediate vectors and see what they produce from the generator. The generator takes this vector and produces that image, this vector, that image, and this vector, that image to get this gradient between these two images.



## Z-Space and Controllable Generation



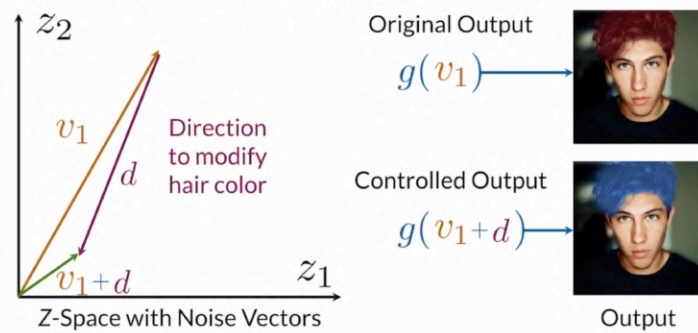
Controllable generation also uses changes in the Z-space and takes advantage of how modifications to the noise vectors are reflected on the output from the generator.

For example, with the noise vector, you could get a picture of a woman with red hair and then with another noise vector, you could get a picture of the same woman but with blue hair.

The difference between these two noise vectors is just this direction in which you have to move in Z-space to modify the hair color of your generated images. In controllable generation, your goal is to find these directions for different features you care about. For example, modifying hair color. But don't worry about finding that exact direction yet, I'm going to show you in the following lectures.

With this known direction  $d$ , let's call this direction  $d$ , in your Z-space, you can now control the features on the output of your GAN, which is really exciting.

## Z-Space and Controllable Generation



This means that if you then generate an image of a man with red hair produced by the same Generator  $g$ , with this input noise vector here,  $v_1$ , you can modify the hair color of this man in the image by adding that direction vector  $d$  you found earlier to the noise vector, creating this new noise vector here,  $v_1 + d$ , passing that into your generator and getting a resulting image where his hair is now blue.

## Summary

- To control output features, you need to find directions in the z-space
- To modify your output, you move around in the z-space

To sum up, in controllable generation, you need to find the directions in the Z-space related to changes of the desired features on the output of your GAN.

With known directions, controllable generation works by moving the noise vector in different directions in that Z-space. Up next, you'll learn some challenges related to controllable generation and how to find directions on the Z-space with known effects on the generated outputs.





deeplearning.ai

# Challenges with Controllable Generation

Controllable generation makes it so that you can decide the features and the output of a GAN, like hair color or hair length in the GAN that produces pictures of people.

## Outline

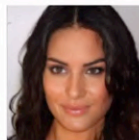
- Output feature correlation
- Z-space entanglement

However, controllable generation has a couple of challenges that you'll see in this section. Specifically, you'll learn about feature correlation and the alpha space in Z space entanglement.



## Feature Correlation

Uncorrelated Features



Add beard



Correlated Features



Add beard  
Make more masculine

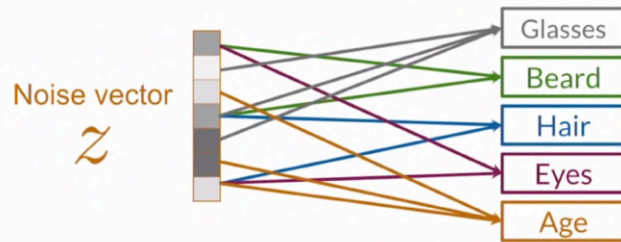


For example, ideally you want to be able to control single features like the amount of beard on a person's face produced by your GAN and if features in the data set don't happen to have a high correlation, you'd be able to take this picture of a woman and add a beard to her by moving in some direction in Z space.

However, it's very likely that in the data set you use for training features like the presence of a beard and how masculine the face looks will be strongly correlated. If you want to add a beard to the picture of a woman, you'd end up modifying many more features on the output.

But that perhaps is not desirable because you want to be able to find these directions where you can just change one feature about someone. That way you can reliably edit images.

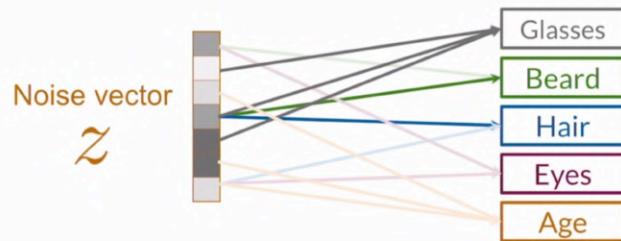
## Z-Space Entanglement



Another challenge faced by controllable generation is known as entanglement in the Z space.

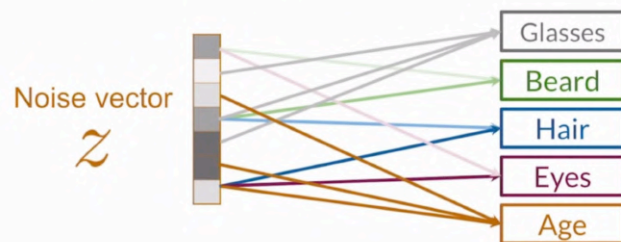
When the Z space is entangled, movement in the different directions has an effect on multiple features simultaneously in the output. Even if those features aren't necessarily correlated in your training data set. This is just how the noise space was learned; to be very entangled.

## Z-Space Entanglement



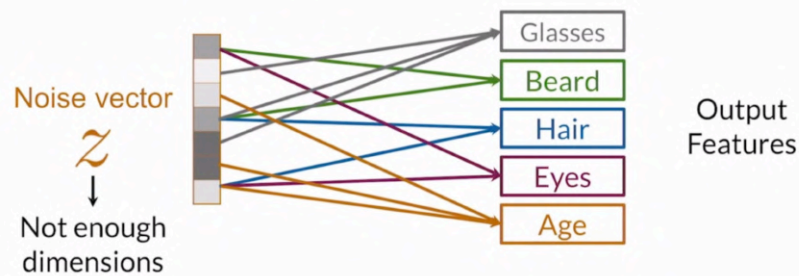
In this entangled Z space, when you control if a person in the output has glasses, for example, you also end up modifying whether she has a beard in her hair

## Z-Space Entanglement



Or when you try to modify her apparent age, you'll end up also changing her apparent eyes and hair color too. That's not quite desirable. The same happens with other uncorrelated features.

## Z-Space Entanglement



It is not possible to control single output features

This means that changes in some of the components of the noise vectors change multiple features in the output at the same time and this makes it difficult, if not impossible, to control the output.

This is a very common problem when the Z space doesn't have enough dimensions relative to the number of features you want to control in the output because then it actually can't map things one-to-one.

This is also just generally an issue when training generative models.

## Summary

- When trying to control one feature, others that are correlated change
- Z-space entanglement makes controllability difficult, if not impossible
- Entanglement happens when z does not have enough dimensions

To recap, controllable generation faces several challenges. If features in your data set has a high correlation with each other and you don't account for this in some way, then when you try to control the output of your GAN, you end up changing multiple features at a time.

Even if the features you want to control don't have a high correlation with each other in the training data set, control of a generation is also difficult if your Z space is entangled. Which will happen commonly if the number of dimensions in your Z space is not large enough but there are a host of other reasons of why that happens as well.



deeplearning.ai

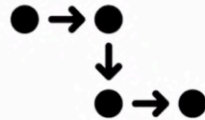
## Classifier Gradients

Trouble generation works by moving into the z-space, according to directions that correspond to desired features, such as lengthening hair or shortening hair.

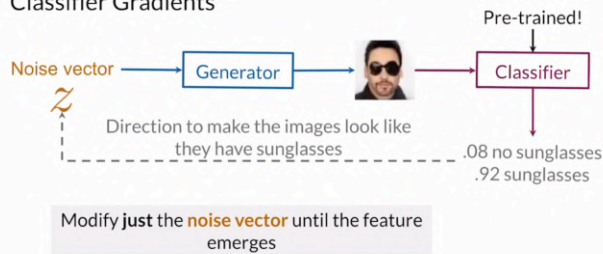
## Outline

- How to use classifiers to find directions in the z-space
- Requirements to use this method

In this section, you'll learn a very simple method used to find that direction using the gradient of trained classifiers, you'll see what the requirements are for this method.



### Classifier Gradients



So to find the direction in the z-space that modifies certain features on the output, say the presence of sunglasses. You could use a train classifier that identifies if a person in a picture has that said feature.

So to do so, you could take a batch of noise vector  $Z$ , that goes through the generator to get some images.

You then pass these images through a sunglasses classifier, which will tell you at the outputs correspond to people with or without sunglasses.

Then you use them information to modify your  $Z$  vectors, and this is without modifying the weights of the generator at all. So the generator weights are frozen. You're done training, and you modify your  $Z$  vectors by moving in the direction of the gradient with the costs. That penalizes the model for every image classified as not having sunglasses.

So then you repeat this process until the images are classified as people with sunglasses.

So this method is very simply inefficient, and some might argue, even lazy, because you're using this classifier that's already there for you. But I think it's great, that you're taking advantage of a pre-training classifier.

However, of course there's also the downside of that. You need a pre-trained classifier that accurately detects the feature that you want to control with four hand. You can also train your own of course, so if you wanted to classifier that detected beards, for example, you might have to train that on your own if that isn't available off the shelf. Be you should always check if something is available out there, because this could be a really simple and cool way to allow you to start controlling your.

## Summary

- Classifiers can be used to find directions in the z-space
- To find directions, the updates are done just to the noise vector

So you should take away from this video that pre-trainer classifiers can be used to find directions in the z-space associated with features in the output of. And to find those directions using the gradients of the classifiers, you need to modify the noise vectors without changing the generator. So remember all of this happens after training.







deeplearning.ai

# Disentanglement

Previously, you saw what entanglement and the z-space means and why that's problematic in controllable generation. In this section, you'll review disentanglement and some ways to encourage your model to achieve it.

## Outline

- What a disentangled z-space means
- Ways to encourage disentangled z-spaces

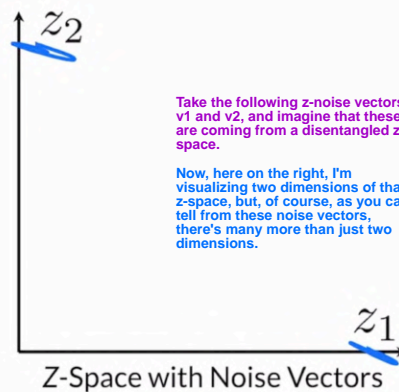
First, you'll review what entangled versus disentangled z-space means, then I'll mention some of the most popular ways to encourage your model to have a disentangled z-space.



## Disentangled Z-Space

$$\underline{v_1 = [1, 2, 3, \dots]}$$

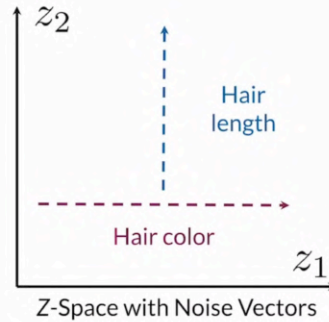
$$\underline{v_2 = [5, 6, 7, \dots]}$$



## Disentangled Z-Space

$$v_1 = [ \overset{z_1}{1}, \overset{z_2}{2}, 3, \dots ]$$

$$v_2 = [ \underset{\text{Hair color}}{5}, \underset{\text{Hair length}}{6}, 7, \dots ]$$



Now, if this is a disentangled z-space, then each of these positions would correspond to a single feature in the output.

For example, this first element, this first dimension of the noise vector would correspond to hair color, and the second dimension would correspond to hair length, and so on.

If you wanted to change the hair color in a picture, you would literally just change the first element on its noise vector, or move in the  $z_1$  direction on that z-space; and that's all you have to do, change the hair color. Then for hair length, you just move in the  $z_2$  dimension. That's just changing these values here in the second position.

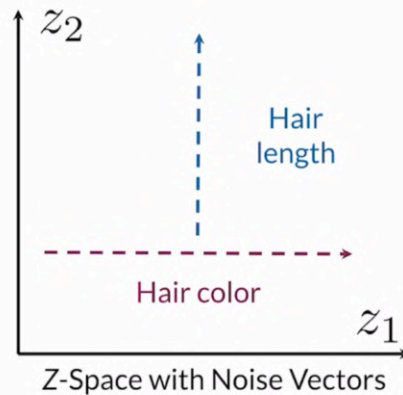
Also, it's quite possible that these noise vectors are disentangled with respect to just these two different features and the rest of the values don't mean anything. This is because you typically want the size of your noise vector to be larger than the number of features you want to control. This makes it easier for your model to learn because it allows these values to take on different things over time during training.

For example, if you only wanted to control these two features like hair color and hair length, then it's probably prudent to make your noise vector larger than just two-dimensions, but here you see three or more. These other values won't control a specific feature, they just help the model adapt to in training.

## Disentangled Z-Space

$$v_1 = [ \overset{z_1}{1}, \overset{z_2}{2}, 3, \dots ]$$

$$v_2 = [ \underset{\text{Hair color}}{5}, \underset{\text{Hair length}}{6}, 7, \dots ]$$



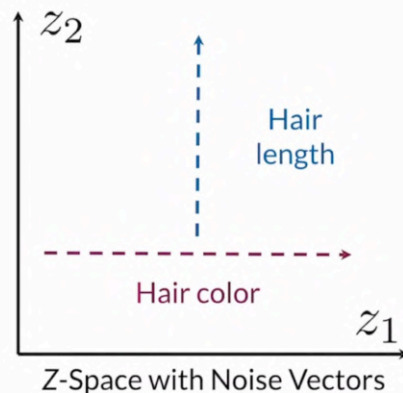
Because the components of the noise vectors in the disentangled z-space allow you to change those features that you desire in the output, they're often called latent factors of variation, where the word latent comes from the fact that the information from the noise vectors is not seen directly on the output, but they do determine how that output looks. Sometimes you might hear noise vectors being referred more generally as latents.

Latent factors of variation

## Disentangled Z-Space

$$v_1 = [ \overset{z_1}{1}, \overset{z_2}{2}, 3, \dots ]$$

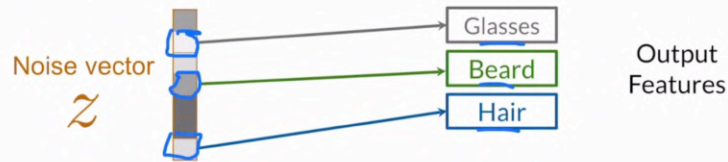
$$v_2 = [ \underset{\text{Hair color}}{5}, \underset{\text{Hair length}}{6}, 7, \dots ]$$



Then factors of variation means that these are just different factors like hair color and hair length that you want to vary, and only that one factor, that one feature that you're varying when you're varying it, not anything else.

Latent factors of variation

## Disentangled Z-Space

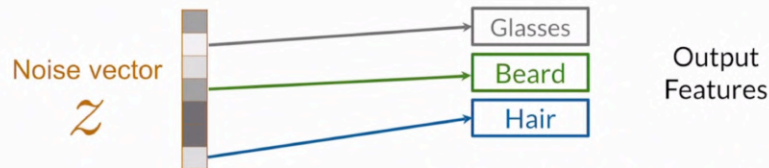


Essentially, a disentangled z-space means that there are specific indices on the noise vectors, these specific dimensions that change particular features on the output of your GAN. For instance, if a person on the image generated by a GAN has glasses or not, or whether she has a beard or some features of her hair.

Each of these cells will correspond to something that you desire to change, and you can just change the values of that dimension in order to adapt glasses, beard, or hair.

A crucial different between a disentangled z-space and an entangled z-space is that, here, with a disentangled z-space, when you control one of the features on the output, for example, glasses, the other features remain the same. The beard and hair will remain the same. Or if I change whether someone has a beard, the glasses and hair will stay the same.

## Disentangled Z-Space

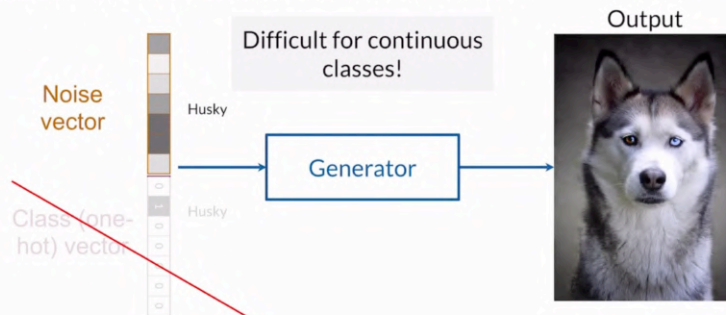


Changes to one feature don't affect the others



What this means is that with a disentangled z-space, you're much more likely to be able to add a beard to a person that looks very feminine without changing her hair or her facial features.

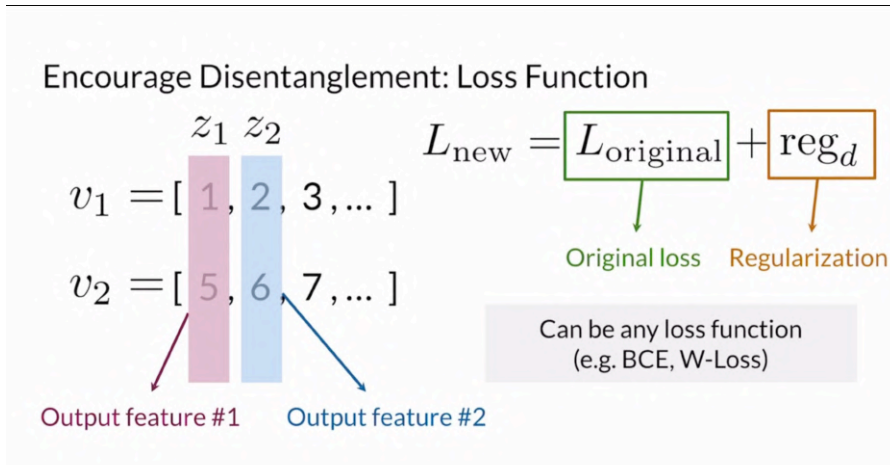
## Encourage Disentanglement: Supervision



One way to encourage your model to use disentangled z-spaces is to label your data and follow a similar process, the one used for conditional generation.

But in this case, the information from the class is embedded in the noise vector; so you don't need this extra one-hot class information or class vector.

However, using this method could be problematic for continuous classes. Imagine having to label thousands of human faces with the length of their hair. Of course, if you do this, even having a few different classes, a few different buckets, might actually nudge your generator in the right direction.



Another way to encourage your model is to use a disentangled z-space without labeling any of your examples.


Instead, you add a regularization term to the loss function of your choice like BCE or W-loss to encourage your model to associate each index from the noise vectors, two different features on the output.

This regularization could come from classifier gradients and they're also much more advanced techniques to do this in an unsupervised way, meaning without any labels.

### Summary

- Disentangled z-spaces let you control individual features by corresponding z-values directly to them
- There are supervised and unsupervised methods to achieve disentanglement

In summary, disentangled z-spaces let you control individual output features by corresponding certain z-values directly to desired features you want to control. To encourage your model to use disentangled noise vectors, you can use both supervised and unsupervised learning methods.



## (Optional) An Example of a Controllable GAN

Want to learn more about controlling GAN generations using latent space? Check out this paper!

Interpreting the Latent Space of GANs for Semantic Face Editing (Shen, Gu, Tang, and Zhou, 2020):  
<https://arxiv.org/abs/1907.10786>

## Works Cited

All of the resources cited in Course 1 Week 4, in one place. You are encouraged to explore these papers/sites if they interest you—the first paper has already been included as an optional reading! They are listed in the order they appear in the lessons.

From the videos:

- Interpreting the Latent Space of GANs for Semantic Face Editing (Shen, Gu, Tang, and Zhou, 2020):  
<https://arxiv.org/abs/1907.10786>

From the notebooks:

- MNIST Database: <http://yann.lecun.com/exdb/mnist/>
- CelebFaces Attributes Dataset (CelebA): <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

