So far, generative models only generate data that is similar to the training set. Generally, though, we're not interested in generating data from a particular distribution, but rather data with *specific attributes*. Conditional generation does exactly that.

Conditional Generation

General issue: A neural network compute a single function. Can we compute *a family of functions* instead? (a function parametric w.r.t. given attributes). For instance, in generative model, we would like to *parametrize generation* according to *specific attributes* - generate a given digit - generate the face of an old man wearing glasses - generate a red, sports car

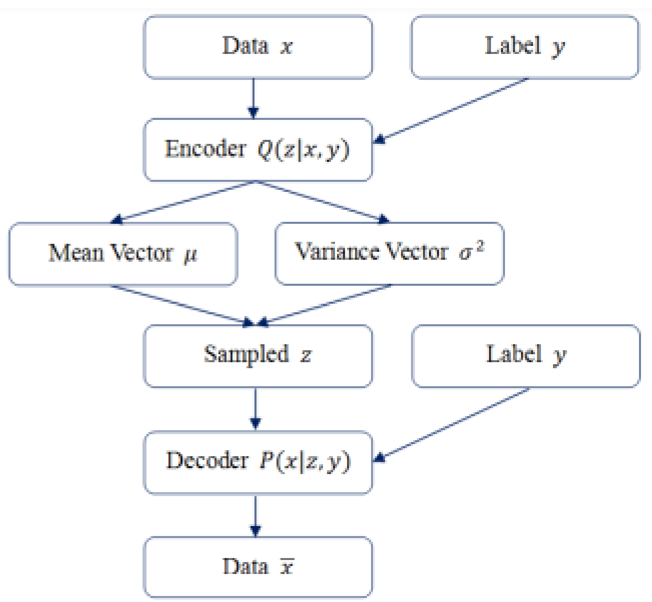
Issues

- Integrate the condition inside the generative model
- Concrete handling of the condition (mixing input and condition)

Conditional VAE (CVAE)

Both the *encoder* $Q(z|X)$ and the *decoder* $P(X|x)$ are now parametrized w.r.t. a given *condition* c: $Q(z|X, c)$ and $P(X|z, c)$. What about the prior? - We can still work with a single, *condition **independent** prior* (e.g. a normal gaussian) - simpler, a little more burden on the decoder side - We are basically assuming that the *prior distribution*, in any case, *does not depend on c*. - We can also *use a different (possibly learned) prior* (e.g. a different Gaussian) *for each condition* - slightly more complex; not clearly beneficial

The architecture of CVAE is this:



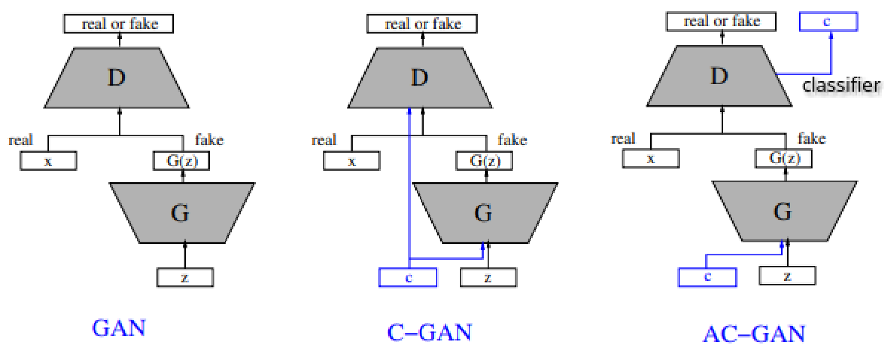
Additional info on CVAE

By giving the label info to bo the encoder and the decoder, they can essentially exploit that information in some ways. In general, for example, ==they can use that info for encoding the information (instead of encoding the data into the latent space)==. In general, the clusters of the latent space become much more defined, since we do not need anymore to distinguish this information in a way.
- To be more precise, if we saw the latent space, they would not be any more clustering, but rather the data of the same class would overlap.

Also, in general, VAE have a more regular latent space wrt to general autoenconder, since we are using in fact using a kind of regularization method, which in fact is the KL distance.

Conditional GANs

The generator takes *in input the condition*, in addition to the noise.



What about the discriminator? - use the condition to discriminate fakes for real of the given class (**Conditional GAN**) - It gives the same condition given to the generator as an additional input to discriminator. - try to classify w.r.t different conditions in addition to true/fake discrimination (**Auxiliary Classifier GAN**) - *couples* the discriminator *with a classifier*, so in addition it also has to guess the label of the image.

Loss function for AC-GANs Notation: - $p^*(x, c)$ is ***true*** *image-condition joint distribution* - $p_\theta(x, c)$ is the *joint distibution of* ***generated*** *data* - $q_\theta(c|x)$ is the **classifier**

In addition to the usual [[2023-04-19 - Generative Models 2#GAN's loss function|GAN objective), we also try to minimize the following quantities:

$$\underbrace{-\mathop{\mathbb{E}}_{p^*(x,c)} ln(q_\theta(c|x))}_{term\ 1}\underbrace{-\mathop{\mathbb{E}}_{p_\theta(x,c)} ln(q_\theta(c|x))}_{term\ 2}$$

- term 1: the *classifier should be consistent with the real distribution* - So, it's the dedicated to the classifier. - term 2: the generator must create images easy to classify by the discriminator.

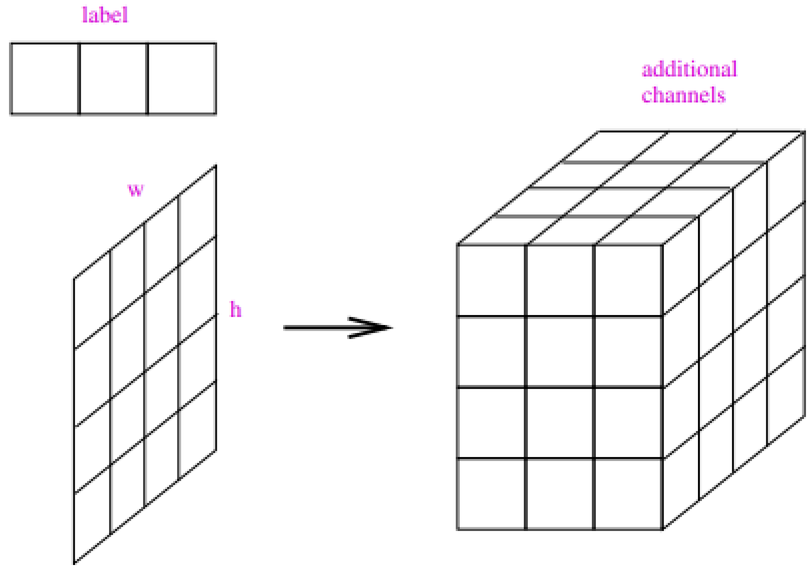
The second term has always been criticized, so in InfoGAN, for example, we only have the *first term*. - The second term helps to generate images far from boundaries between classes, hence, likely more sharp. But what if *real images are close to boundaries*? - This is a problem of almost every GAN: some images are very easy to generate, while others provide a very bad result. - It has also been criticized because the classifier can suffer from the [[2023-04-19 - Generative Models 2#GANs problems|Mode Collapse), too.

Concrete handling of the condition

In conditional networks, we pass the label/condition as an additional input. How is this input going to be processed? If we need to add it to a dense layer, we just concatenate the label to the input. If we need to add it to a convolutional layer, we have two basic ways: - Vectorization - Feature-wise Linear Modulation (FILM)

Vectorization

We essentially *repeat the label* (typically in categorical form) for *every input neu-*

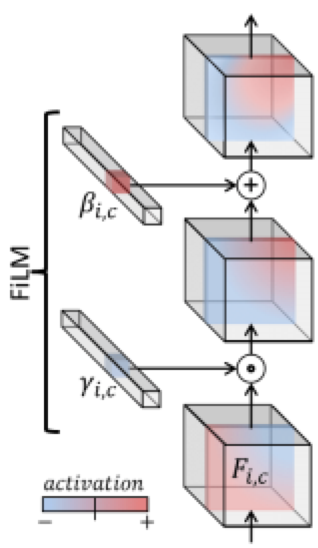


ron, and *stack them as new channels.*

FILM

Idea: use the condition to *give a different weight to each feature* (each channel).

We use the condition to generate two vectors γ and β with size equal to the channels of the layer. Them we rescale layers by γ and add β.



It's less invasive than parametrizing the weights. Nevertheless, Vectorization remains the most typical and the most easy to use though.