Generative modeling

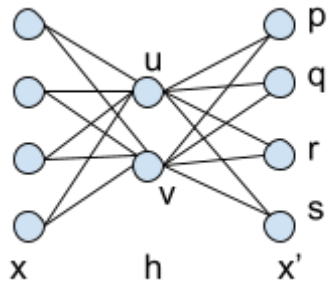
Recall the supervised learning problem: we have training data (x,y) ($x \in R^m, y \in \{-1, +1\}$) generated by P(x,y) where P(x,y) is unknown. We are given test data x' and we want to predict the label y' of x'. We have two fundamental approaches to solving this problem:

Generative approach: here we try to estimate P(x,y) and then we can classify x' by comparing P(x',+1) and P(x',-1).

Discriminative approach: here we don't bother with trying to estimate P(x,y) but instead we focus on P(y|x). We apply Bayes rule to get P(y|x) = P(x|y)P(y)/P(x) where P(x|y) is the likelihood.

Let's get to specifics. How can we generate data with a neural network? For example how can we generate artificial samples for the ionosphere dataset with a neural network?

Consider the network below which is called an autoencoder.



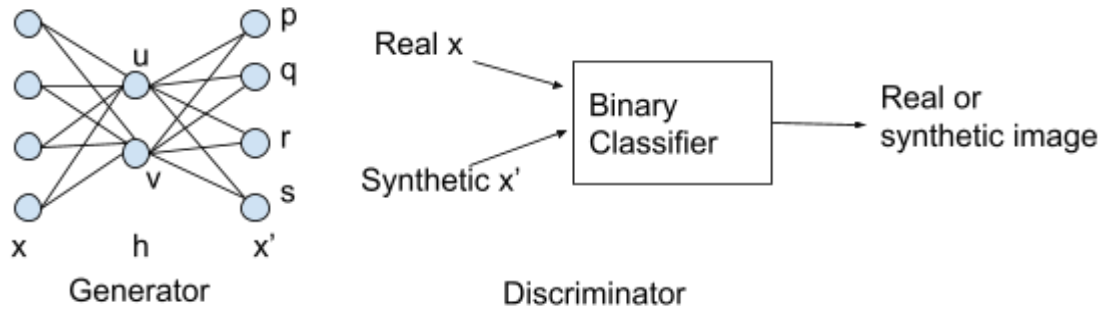
In the above model we reduce our input to a lower dimensional space and then increase it back to the original space. This is called an encoder-decoder model. We can write the loss for the above model as $L = ||x - x'||^2$. Now that we have the loss we can calculate $dL/dp,\ dL/dq,\ dL/dr,\ dL/ds,\ dL/du,\ and\ dL/dv$ and train the network with gradient descent. Note that labels are not required, this is unsupervised learning.

After training the above model we can generate data (at least in principle) by giving it random vectors in the hidden space h. Does this approach actually work in practice?

This approach, while it may seem totally plausible, works poorly due to overfitting. To make it work we regularize the loss as
$L = ||x - x'||^2 + ||p||^2 + ||q||^2 + ||r||^2 + ||s||^2 + ||u||^2 + ||v||^2$. This now then becomes similar to a basic variational autoencoder (VAE). We can add more terms to relieve overfitting and the model ends up somewhat hard to train.

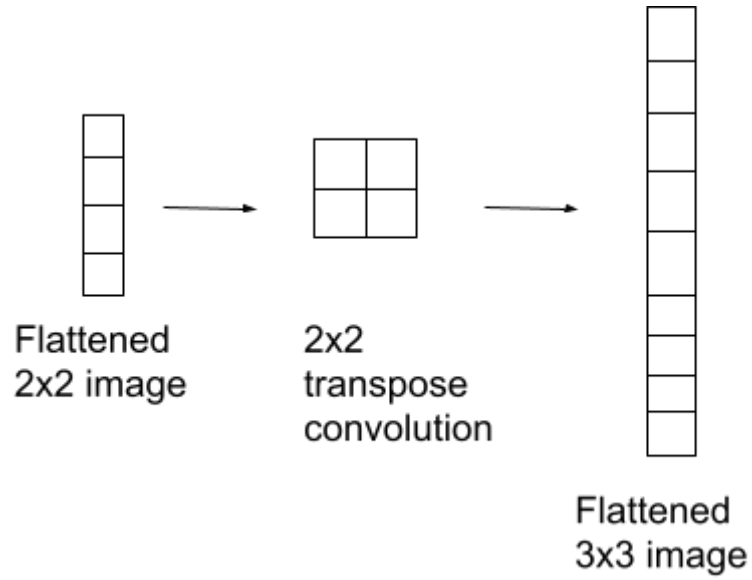
Is there another approach to train the autoencoder to generate better samples? We can use the generative adversarial approach to produce better synthetic samples. Instead of using a least squares loss we evaluate the synthetic sample by giving it a binary classifier that tries

to distinguish between real and synthetic samples. The goal of the generator is to fool the discriminator into thinking the synthetic image is real. This approach is called a generative adversarial network (GAN).

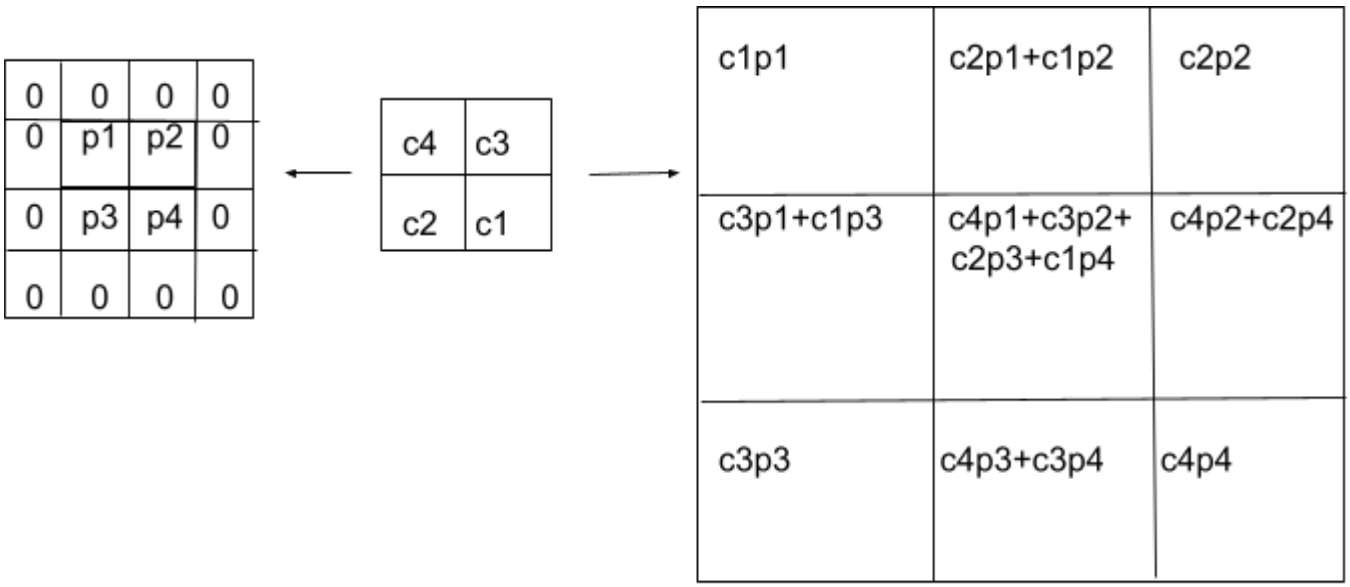


Generative modeling for convolutions

Consider the network below. We apply a 2x2 convolution to a 2x2 image to get a 3x3 image. This is the same as taking the transpose of a 2x2 convolutional matrix applied to a 3x3 image.



Algorithmically we apply the transpose convolution as shown below. We pad the 2x2 image so that the output dimensions are 3x3. The kernel in the middle is applied to the input image padded with zeros.



How can we train the above network? Let us use the least squares loss function (also known as the mean squared error MSE). Suppose we are given a set of images xi for i=0 to n-1. We can write the loss as

$$\sum_{i=0}^{n-1} \left|\left|x_i - x'_i\right|\right|^2$$

Where x'i is the generated image. Suppose the true 3x3 image xi is given in the flattened form as xi = [q1, q2, q3, q4, q5, q6, q7, q8, q9]. Then we can write the loss explicitly as

$$L = \sum_{i=0}^{n-1} (q_1 - c_1p_1)^2 + (q_2 - (c_2p_1 + c_1p_2))^2 + (q_3 - c_2p_2)^2 + (q_4 - (c_3p_1 + c_1p_3))^2 + (q_5 - (c_4p_1 + c_3p_2 +$$
$$+ (q_6 - (c_4p_2 + c_2p_4))^2 + (q_7 - c_3p_3)^2 + (q_8 - (c_4p_3 + c_3p_4))^2 + (q_9 - c_4p_4)^2$$

To train the above loss with gradient descent we need dL/dc1, dL/dc2, dL/dc3, and dL/dc4 which are straightforward to calculate from the above equation. This means we can now train a network to generate images even if the input images [p1,p2,p3,p4] are just random pixels. For example if the target images xi are of cats I can train a generative network to produce images of cats starting from random vectors.

In practice the MSE loss tends to minimize error on the average across many samples. As a result generated images may be blurred. Below we have two examples taken from the NIPS 2016 GANs tutorial.

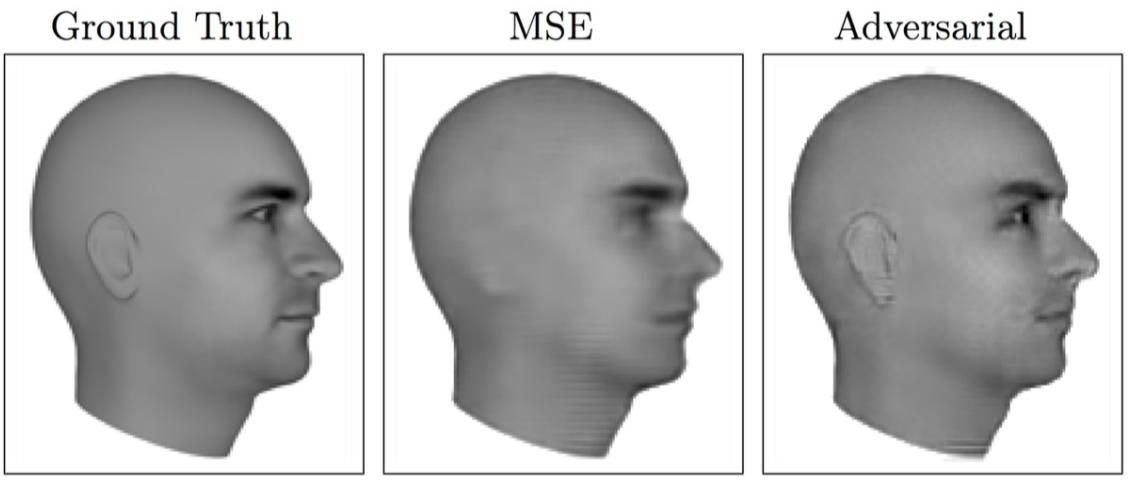


Figure 3: Lotter *et al.* (2015) provide an excellent illustration of the importance of being able to model multi-modal data. In this example, a model is trained to predict the next frame in a video sequence. The video depicts a computer rendering of a moving 3D model of a person's head. The image on the left shows an example of an actual frame of video, which the model would ideally predict. The image in the center shows what happens when the model is trained using mean squared error between the actual next frame and the model's predicted next frame. The model is forced to choose a single answer for what the next frame will look like. Because there are many possible futures, corresponding to slightly different positions of the head, the single answer that the model chooses corresponds to an average over many slightly different images. This causes the ears to practically vanish and the eyes to become blurry. Using an additional GAN loss, the image on the right is able to understand that there are many possible outputs, each of which is sharp and recognizable as a realistic, detailed image.

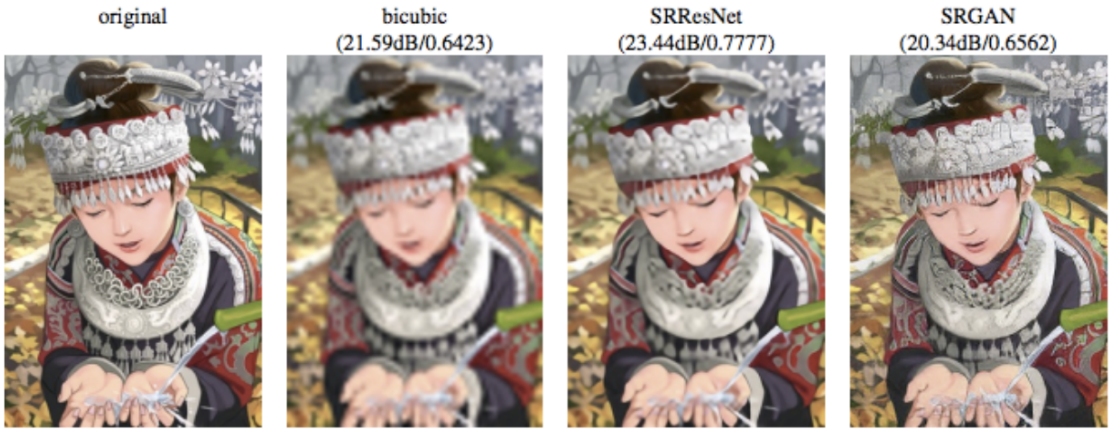


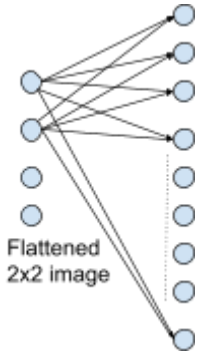
Figure 4: Ledig *et al.* (2016) demonstrate excellent single-image superresolution results that show the benefit of using a generative model trained to generate realistic samples from a multimodal distribution. The leftmost image is an original high-resolution image. It is then downsampled to make a low-resolution image, and different methods are used to attempt to recover the high-resolution image. The bicubic method is simply an interpolation method that does not use the statistics of the training set at all. SRResNet is a neural network trained with mean squared error. SRGAN is a GAN-based neural network that improves over SRGAN because it is able to understand that there are multiple correct answers, rather than averaging over many answers to impose a single best output.

An alternative to the MSE loss is to use an adversarial network loss. In other words instead of evaluating the quality of the output x'i with MSE we compare it to the true image xi using a classifier (also known as discriminator).

$$Loss = \sum_{i=0}^{n-1} classification\ accuracy\ between\ x'_i\ and\ x_i$$

This leads us into generative adversarial networks.

We can generate data without convolutions as well. In the example below we remove the convolutional kernel and go straight from a 2x2 image to a 3x3 image.



We have the option of adding a hidden layer in between as shown below.