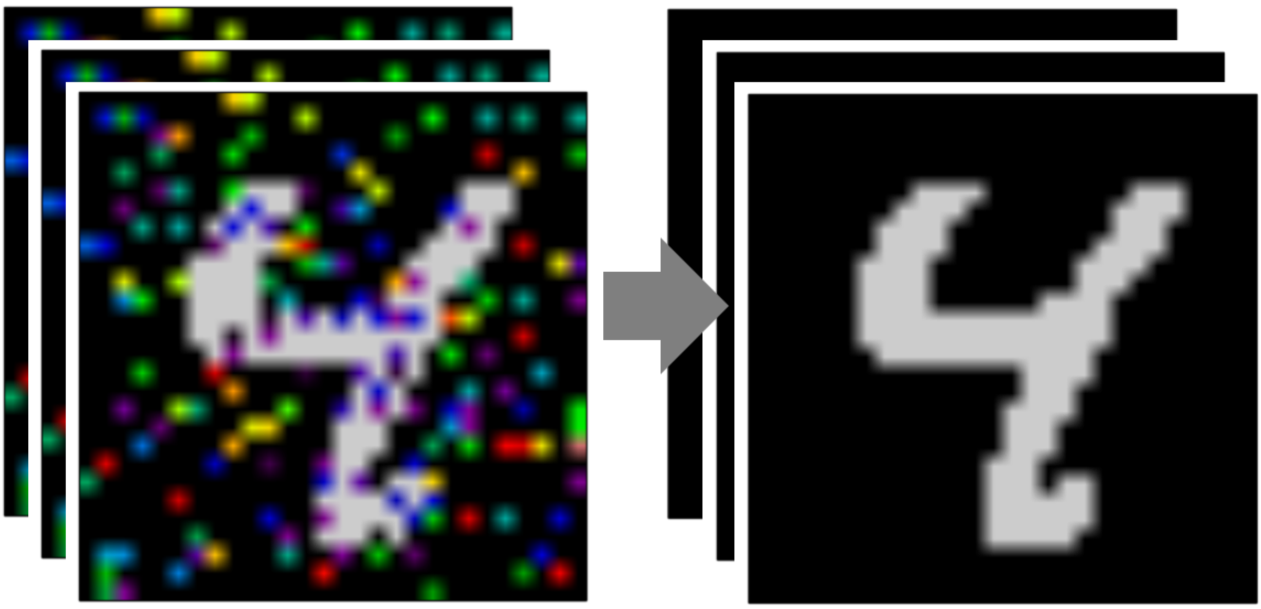
# Autoencoder Zoo – Image correction with TensorFlow

Grant Holtes Dec 17, 2017 · 3 min read



Noise removal using a convolutional autoencoder

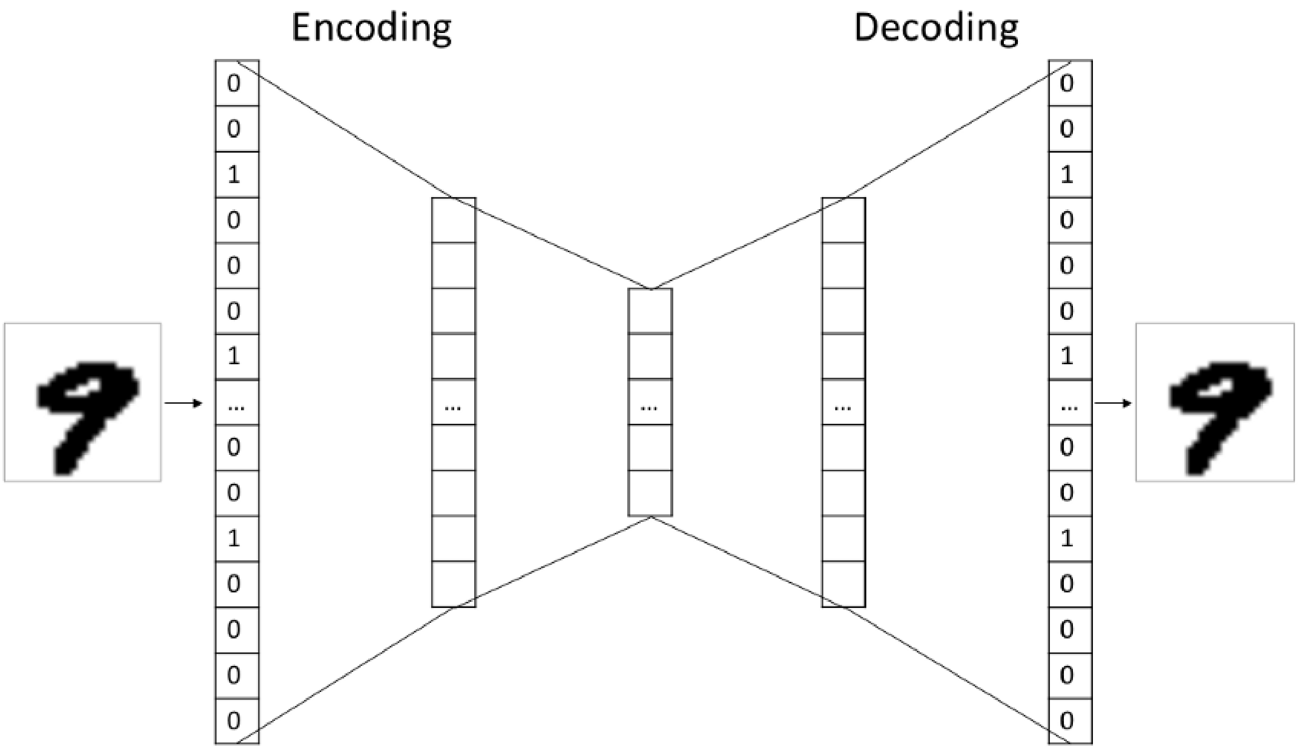
In its vanilla state, an Autoencoder is a function where $f(x) = x$. While this seems superfluous, it has its uses. The interesting bit is that the information in $x$ is compressed, then $x$ is is reconstructed from this compressed state.

The aim of the auto encoder is to learn the most efficient compression. The compressed state typically learns the features of $x$, so doesn't explicitly store $x$. Due to this the

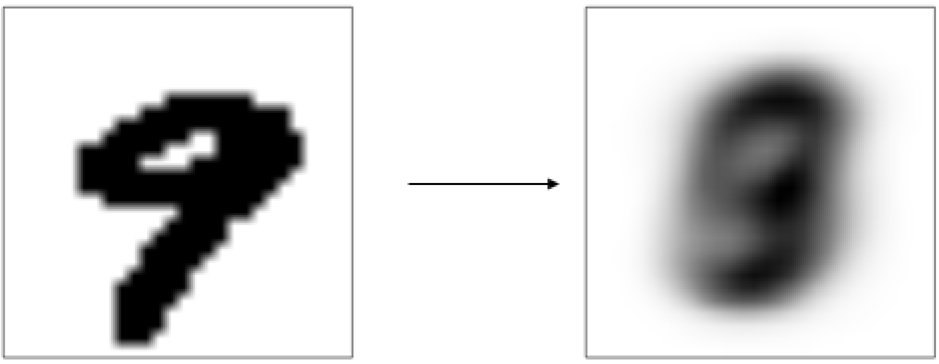
I will be using the MNIST data set partially due to its ubiquity in machine learning, and partially because I have a copy preprocessed.

## Exhibit 1 — Deep Neural Network

The network has 3 deep or hidden layers, which are mirrored in their dimensions. The compressed or encoded state is the middle layer.




This architecture was overly simplistic for the task at hand and failed to capture enough features to be effective, as shown by the output below.

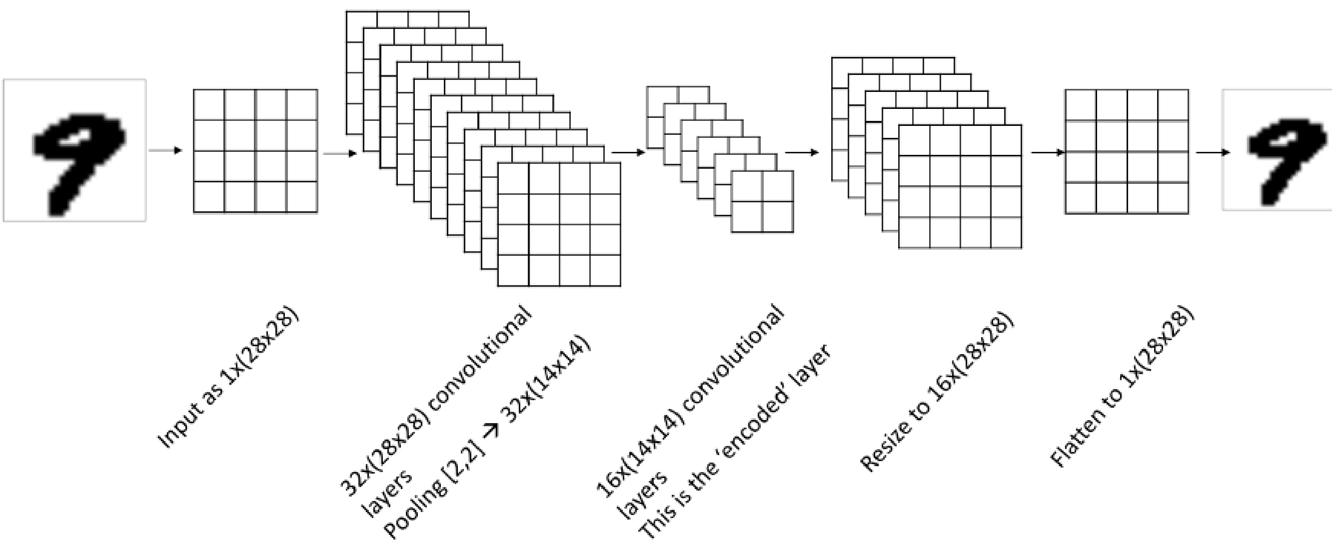


The workings of these networks have been explained in much details in other posts, but I will attempt to make an brief explanation here.

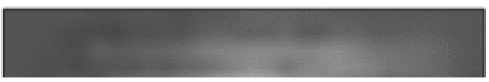
**Convolutional layer** — This layer takes in subsections of the overall image (In this case a 5x5 pixel area), iterating through each possible subsection of the image, using the same weights and biases for each subsection. The number of convolutional filters in a layer are the number of weight and bias 'sets' used. So while the input may be a 1x28x28 image, the output will be (# of layers)x28x28.

**Pooling layer** — The 'max pooling' method aims to reduce the dimensions of the input. For example I used a 2x2 pool size, which means that maximum value in a 2x2 subsection is output, reducing the dimension of that subsection to 1x1. A stride of 2 is used, meaning that each subsection is 2 pixels from the previous one so that none of the subsections overlap. This pooling configuration halves the size / dimensionality of the input.

The following architecture is used:



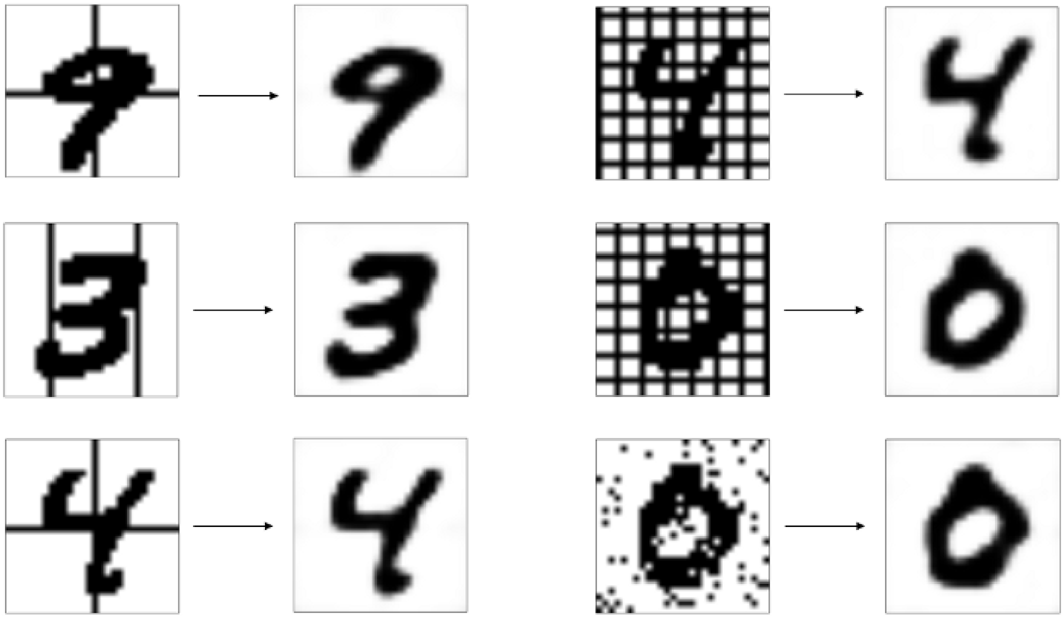

The convolutional architecture is a lot more effective as an Autoencoder, as shown by the accurate reconstruction of the input images.



Get started Open in app

Training process shown with a verification image

As promised, this system holds more uses than simply recreating an input. It can be used to reconstruct noisy or obscured images, using a partial image to reconstruct otherwise missing information. The images below are test images, so have not been used to train the network.



Verification images after 20,000 training iterations

The left hand side represents a static object in the image that the network aims to remove, so the 'error' that needs to be removed is in the same location in each image. The right hand side are random objects — the grid isn't fixed in its location and the noise is randomly generated. The autoencoder is able to learn how to remove undesirable elements and reconstruct the desired information.

## Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter

Image Processing TensorFlow Autoencoder