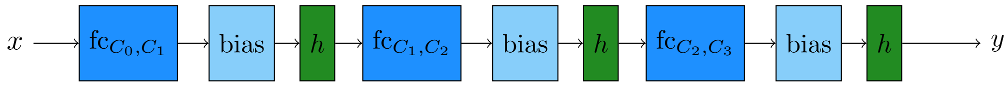

Figure 1: Illustration of a multi-layer perceptron with $L = 3$ fully-connected layers followed by bias layers and non-linearities. The sizes C_1 and C_2 are hyper-parameters while C_0 and C_3 are determined by the problem at hand. Overall, the multi-layer perceptron represents a function $y(x; w)$ parameterized by the weights w in the fully-connected and bias layers.

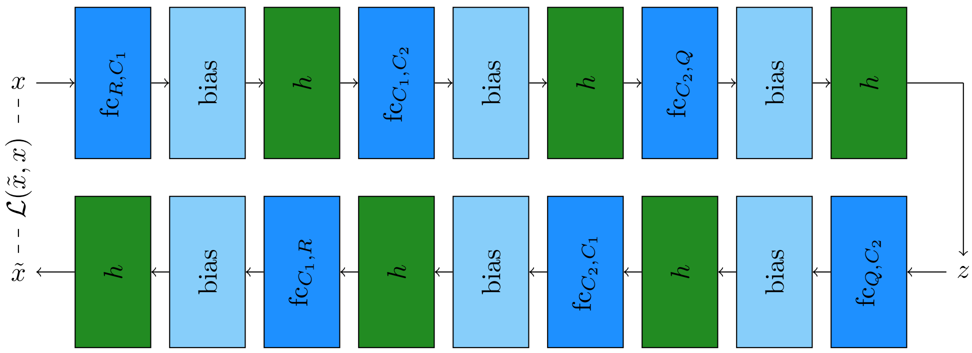

Figure 2: A simple variant of a multi-layer perceptron based auto-encoder. Both encoder (top) and decoder (bottom) consist of 3-layer perceptrons taking an R-dimensional input x. The parameters C_1, C_2, and Q can be chosen; Q also determines the size of the latent code z and is usually chosen significantly lower than R such that the auto-encoder learns a dimensionality reduction. The non-linearity h is also not fixed and might be determined experimentally. The reconstruction loss $\mathcal{L}(\tilde{x}, x)$ quantifies the quality of the reconstruction $\tilde{x}$ and is minimized during training.

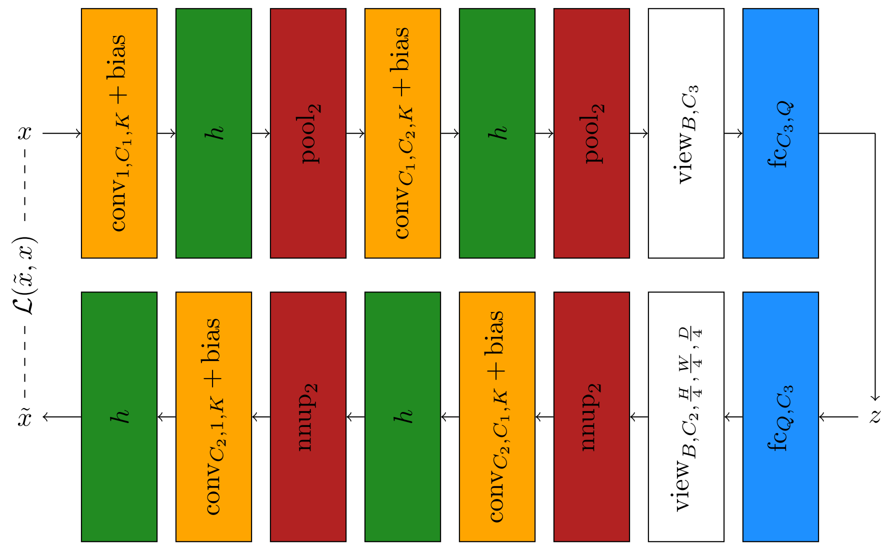

Figure 3: Illustration of a convolutional auto-encoder consisting of encoder (top) and decoder (bottom). Both are modeled using two stages of convolutional layers each followed by a bias layer and a non-linearity layer. The encoder uses max pooling to decrease the spatial size of the input; the decoder uses upsampling to increase it again. The number of channels C_1, C_2 and C_3 as well as the size Q are hyper parameters. We assume the input to comprise one channel. Again, the reconstruction loss $\mathcal{L}(\tilde{x}, x)$ quantifies the quality of the reconstruction and is minimized during training.