COMPSCI 689
Lecture 20: Autoencoders

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

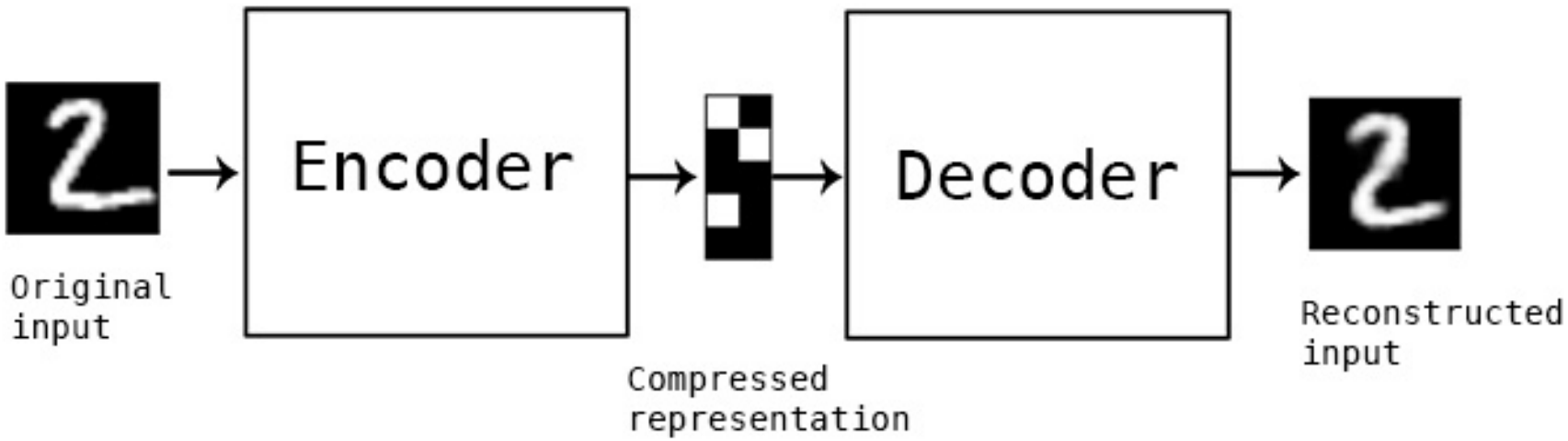
Latent Linear Models

- Latent linear models like factor analysis provide probability densities for linear manifolds in high dimensional spaces.
- Their main limitation is that they can only model linear manifolds.
- To build models for non-linear manifolds, we can apply basis expansions or look at kernelized latent linear models.
- However, these approaches require us to know the right expansion/kernel to apply.

Autoencoders

- An autoencoder is a deterministic model that consists of two components: an encoder function and a decoder function.
- The encoder function $f(\mathbf{x})$ maps a D-dimensional input vector $\mathbf{x} \in \mathcal{X}$ into a K-dimensional code vector $\mathbf{h} \in \mathcal{H}$.
- The decoder function maps the K-dimensional code vector $\mathbf{h} \in \mathcal{H}$ back to a D-dimensional reconstruction of the input vector $\mathbf{r} \in \mathcal{X}$.
- A basic goal is to learn to reconstruct $\mathbf{x}$ as the output of the network while constraining $\mathbf{h}$ in a way that forces it to encode salient features of the input while ignoring noise.

Example: Autoencoder for Digits



Linear Autoencoders

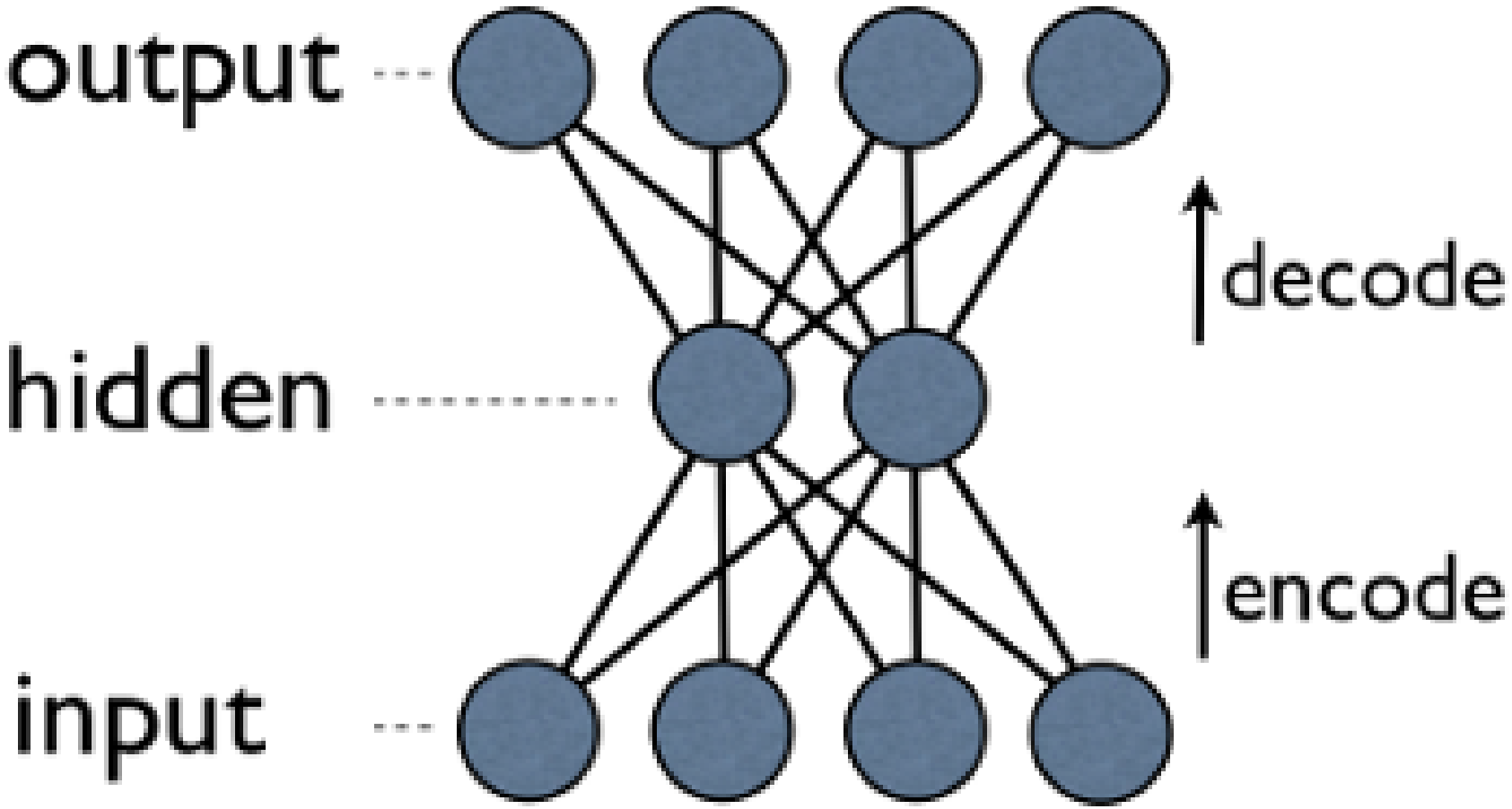
- A linear autoencoder is an autoencoder where the encoder and decoder are linear functions.
- The encoding and decoding functions are:

$$f(\mathbf{x}) = \mathbf{V}\mathbf{x}$$
$$g(\mathbf{h}) = \mathbf{W}\mathbf{h}$$

- Such a linear encoder-decoder model can be learned by minimizing the MSE between the inputs and the reconstructions, often called the *reconstruction error*.

$$\mathbf{V}^*, \mathbf{W}^* = \arg\min_{\mathbf{V},\mathbf{W}} \sum_{n=1}^{N} \|\mathbf{x}_n - g(f(\mathbf{x}_n))\|_2^2$$

Example: Basic Linear Autoencoder Model



Factor Analysis as an Autoencoder

- The factor analysis model can be used as a particularly complex parameterization of a linear autoencoder trained using maximum likelihood estimation (the equations below assume $\mu = 0$)
- The encoder function is simply the mean of $P(\mathbf{z}|\mathbf{x})$:

$$f(\mathbf{x}) = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}[\mathbf{z}] = \mathbf{V}\mathbf{x}$$
$$\mathbf{V} = (I + \mathbf{W}^T \Psi \mathbf{W})^{-1} \mathbf{W}^T \Psi^{-1}$$

- The decoder function is the mean of $P(\mathbf{x}|\mathbf{z})$:

$$g(\mathbf{z}) = \mathbb{E}_{P(\mathbf{x}|\mathbf{z})}[\mathbf{x}] = \mathbf{W}\mathbf{z}$$

Constraints on Linear Autoencoders

- In order for an MSE-minimizing linear autoencoder to learn something useful about the structure of the input data, it is necessary to constrain the code vector in some way.
- If $K = D$ (complete representation) or $K > D$ (overcomplete representation), then we can achieve zero reconstruction error by setting the first D rows of V to the identity matrix and the first K columns of W to the identity matrix (all other entries are zero).
- One way to constrain a linear autoencoder is to require $K < D$ (an undercomplete representation).
- This dimensionality reduction forces the model to extract useful information from the inputs and to discard noise in order to minimize the reconstruction error.

Sparse Overcomplete Autoencoders

- A different way to constrain a linear autoencoder is to allow $K > D$ while constraining the number of non-zero elements of the code vector $\mathbf{h}$.
- *Sparse coding* is a dimensionality reduction model that has a linear decoder and an optimization-based encoder:

$$f(\mathbf{x}) = \arg\min_{\mathbf{h}} \quad \|\mathbf{x} - \mathbf{W}\mathbf{h}\|_2^2 + \lambda\|\mathbf{h}\|_1$$
$$g(\mathbf{h}) = \mathbf{W}\mathbf{h}$$

- The parameters $\mathbf{W}$ are learned on a data set using:

$$\arg\min_{W,\mathbf{h}_1,\ldots,\mathbf{h}_N} \sum_{n=1}^{N} \left(\|\mathbf{x}_n - \mathbf{W}\mathbf{h}_n\|_2^2 + \lambda\|\mathbf{h}_n\|_1\right)$$

Linear Denoising Autoencoders

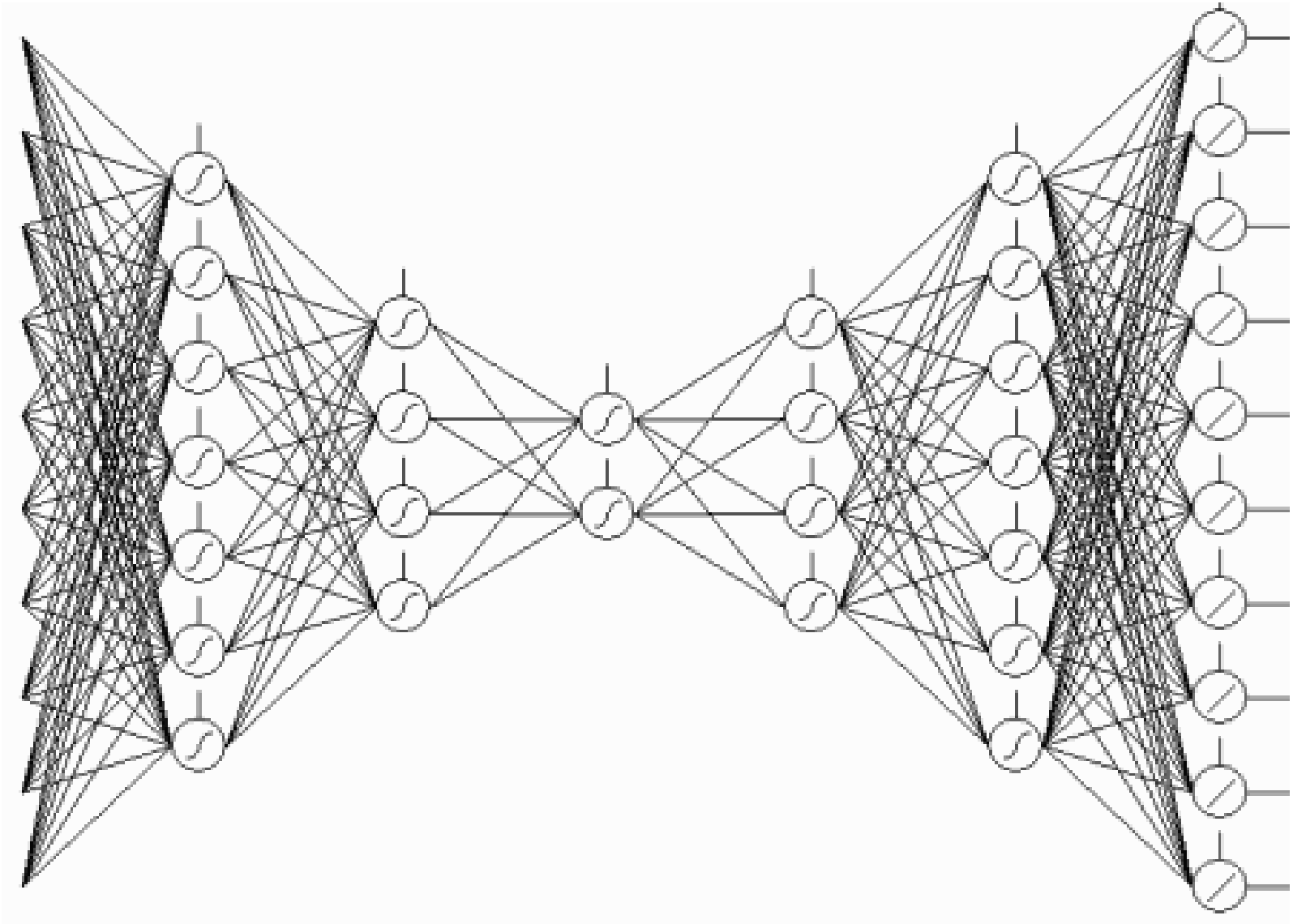
- Yet another way to stop an overcomplete autoencoder from simply learning the identity function is to make the problem that the autoencoder has to solve more difficult.
- A *denoising autoencoder* purposely corrupts the input $\mathbf{x}$ using a sample drawn from a stochastic noise process $q(\mathbf{x}'|\mathbf{x})$.
- It then provides $\mathbf{x}'$ as the input to the autoender while requiring the reconstruction that the model produced match the original $\mathbf{x}$.
- The model can be trained as shown below. The characteristics of the noise process are hyperparameters of model.

$$\mathbf{V}^*, \mathbf{W}^* = \arg\min_{\mathbf{V},\mathbf{W}} \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{x}'|\mathbf{x}_n)} \left[\|\mathbf{x}_n - g(f(\mathbf{x}'))\|_2^2 \right]$$

Non-Linear Autoencoders

- All of the above models and training criteria can only accurately represent data defined on linear manifolds.
- To make an autoencoder non-linear, it suffices to make the encoder and decoder networks non-linear.
- However, the capacity of non-linear autoencoders is no longer limited by the length of the code vector.

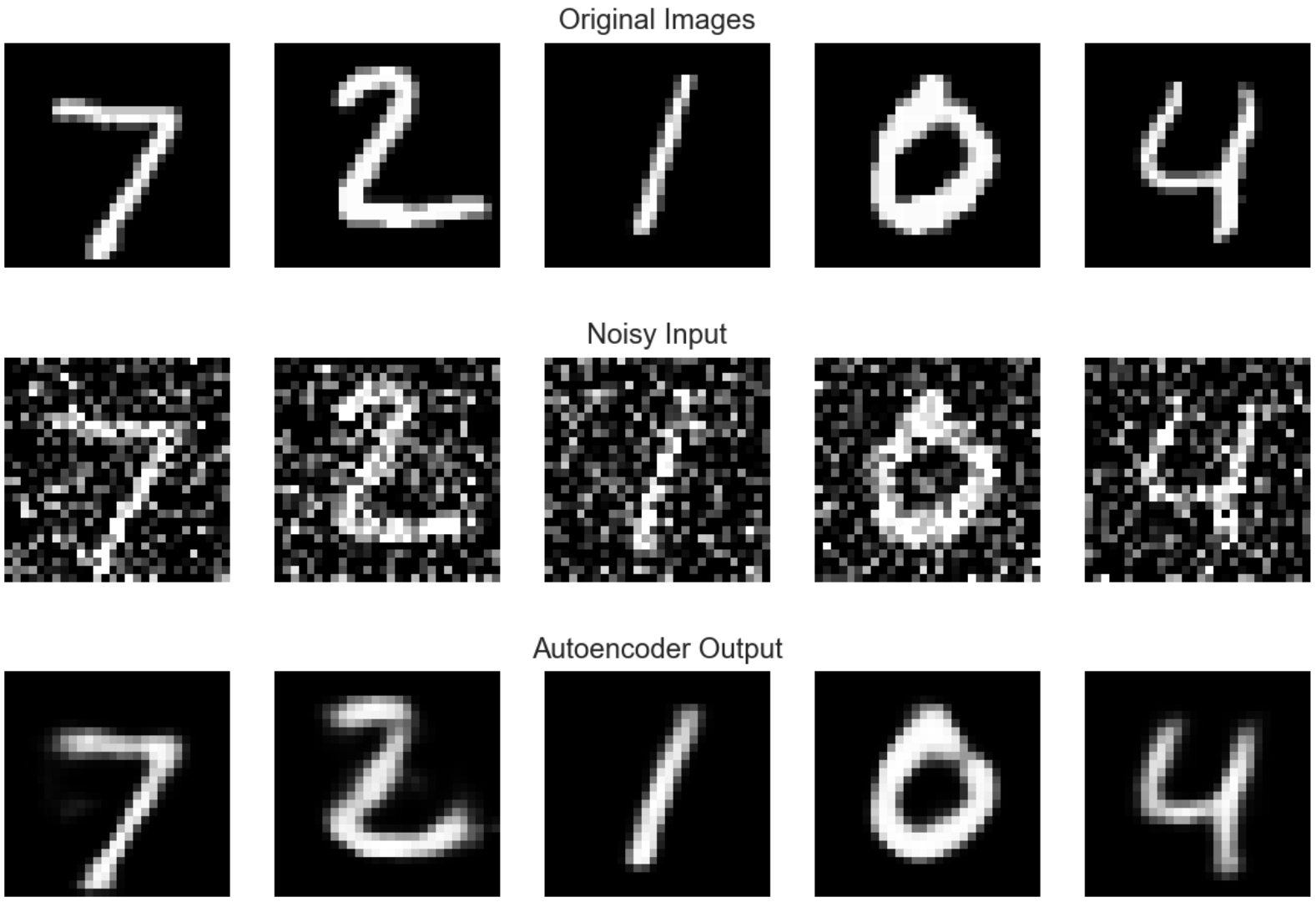
Example: Deep Autoencoders



Constraining Non-Linear Autoencoders

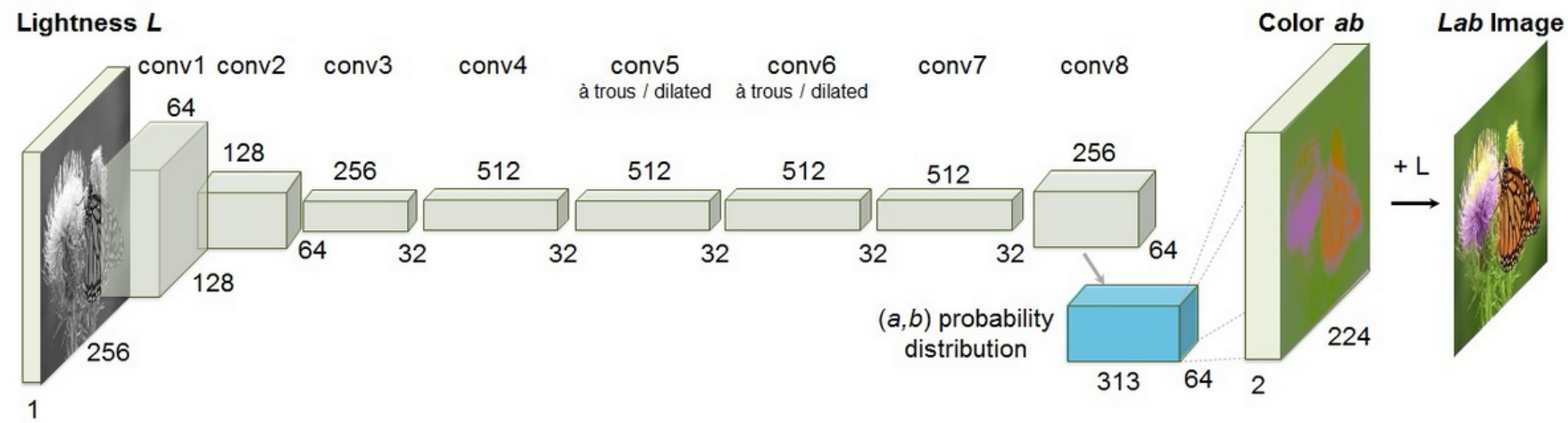
- Like with overcomplete linear autoencoders, nonlinear autoencoders need to have constraints to force them to extract useful information from the data.
- Possible constraints include constraining the width and depth of the encoder and decoder networks and using various forms of the denoising principle.
- As with supervised learning, it is known that some data distributions can be much more efficiently represented with deep, non-linear autoencoders instead of shallow non-linear or linear autoencoders.
- The key is determining the network architecture that gives the best performance for a given down-stream task.

Example: Deep Denoising Autoencoder for Images



Example: Deep Autoencoder for Image Colorization

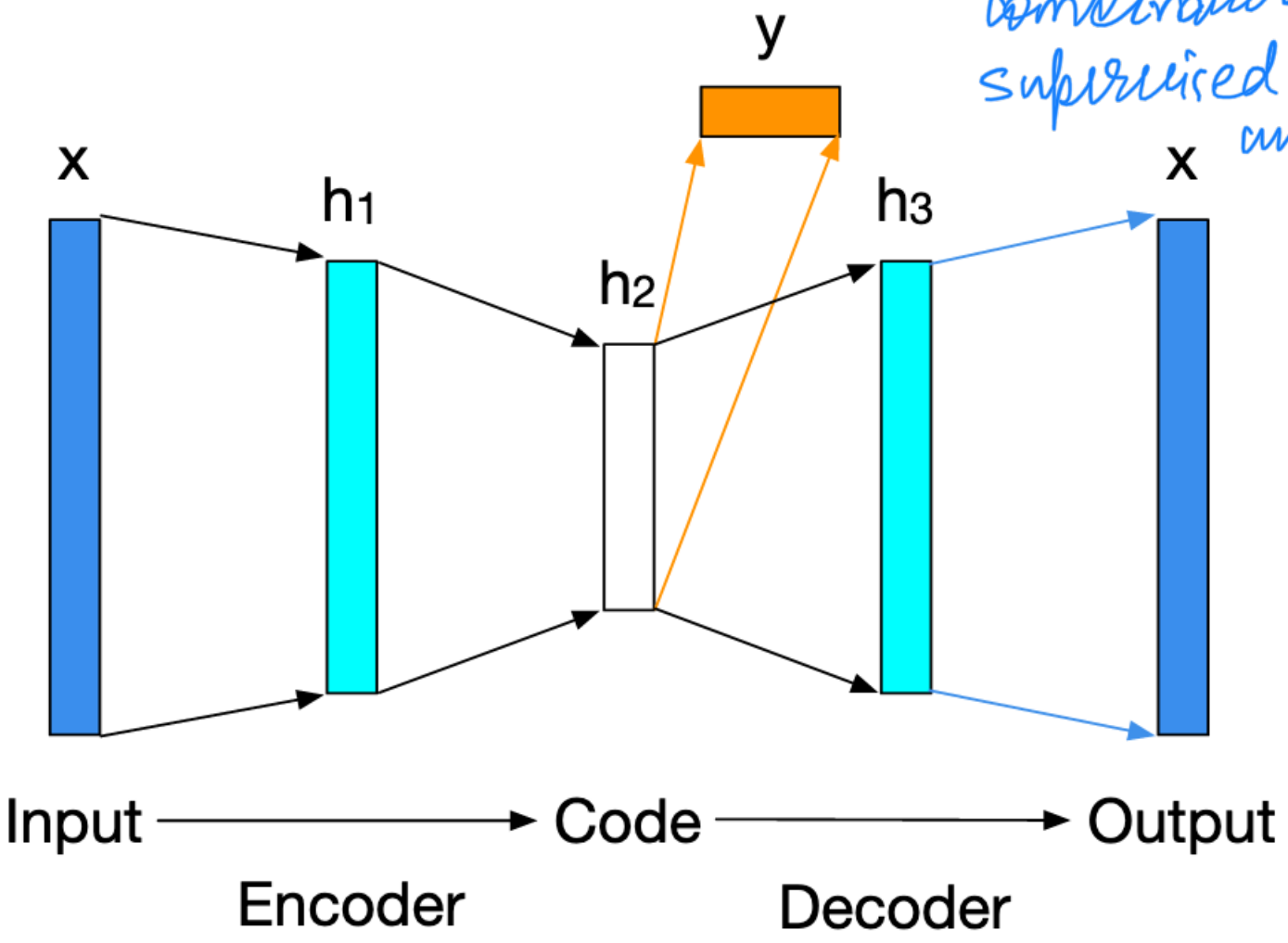
Figure 13: The CAE is trained to colorize the image



Autoencoder Applications

- Autoencoders have many applications including feature extraction, image and signal denoising, and semantic hashing for information retrieval.
- Unsupervised feature extraction (e.g., representation learning) is one of their most common applications and involves learning an encoder-decoder pair on a large unlabeled data set.
- The basic autoencoder architecture can also be modified for semi-supervised learning by including an encoder, a decoder, and a classifier or regression network that uses the code vector as its input.

Example: Semi-Supervised Autoencoder/Classifier



Autoencoder Applications

- The decoder half of the autoencoder can also be used as a generator.
- The main question is how to control the latent codes to produce a desired output.
- Current ML image generators solve this task by mapping natural language sentences (prompts) in to the space of latent codes.