Deep Learning for Computer Vision

Lecture 13: Autoencoders, Sparsity (Again!), Contractive Auto Encoders, Embeddings, PCA

Peter Belhumeur

Computer Science
Columbia University

Everything we have built so far has been built using supervised learning.

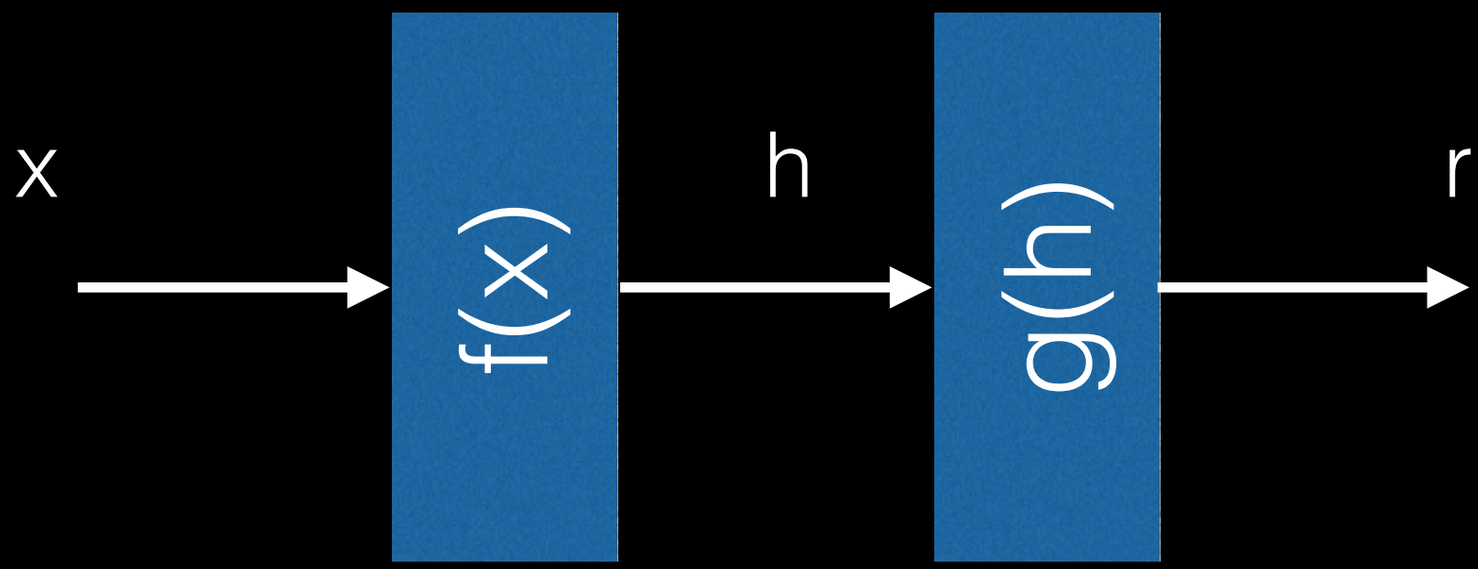
Supervised Learning

- Given inputs X and labels y, train a neural network so the y' = f(X).
- The goal is to find f(X), by minimizing the loss L(y, y'=f(X)).
- And for all of our tasks we were given, or provided, the labels y.
- This is a supervised process.

Let’s look at a type of learning that is NOT supervised.

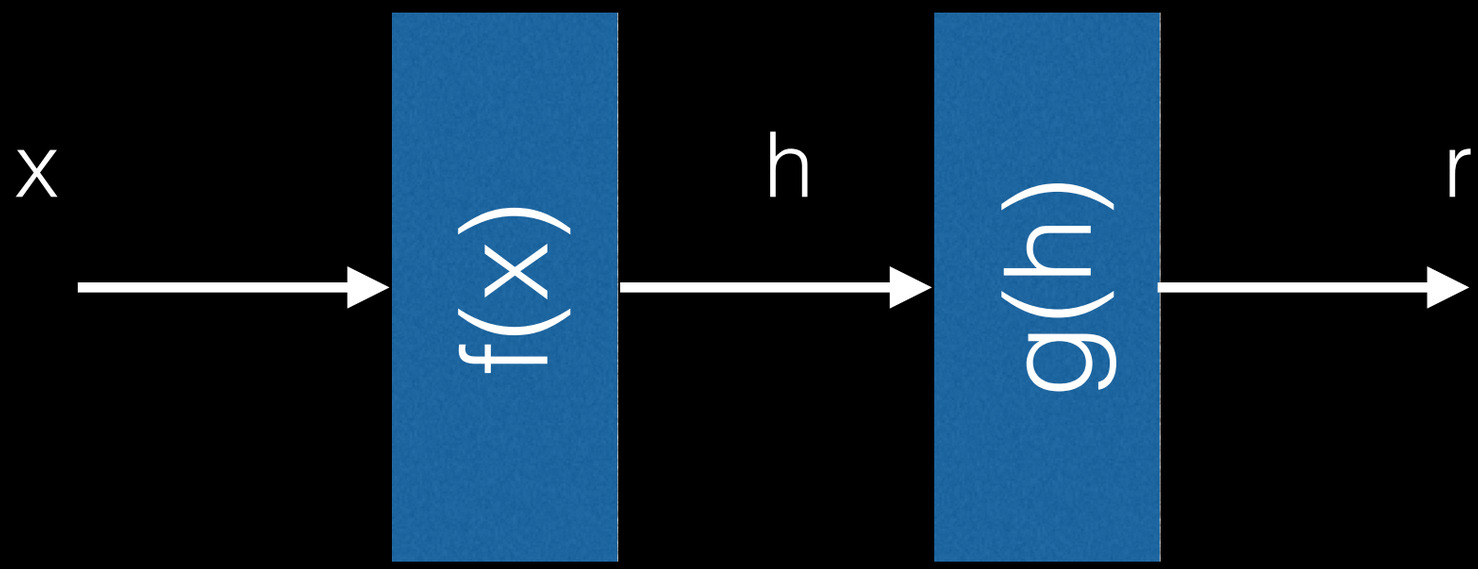
What if we made a neural network whose desired output was its input?

Autoencoder



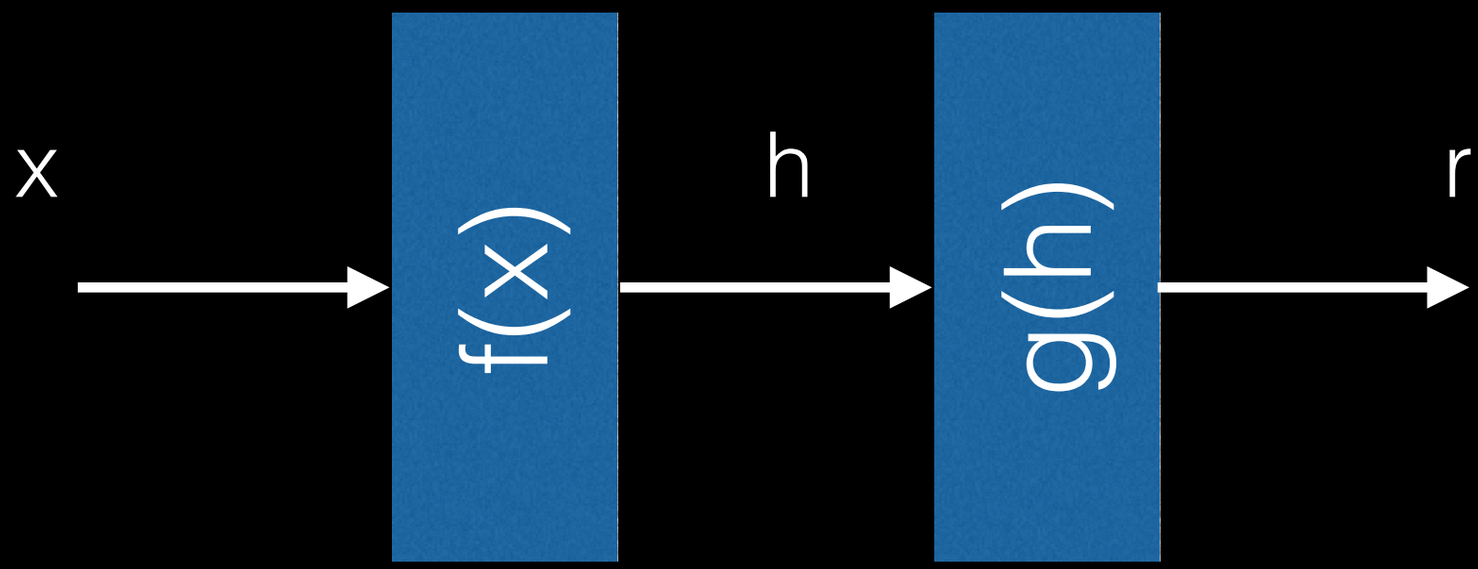
x is the input and r is a **reconstruction** of x.

Autoencoder



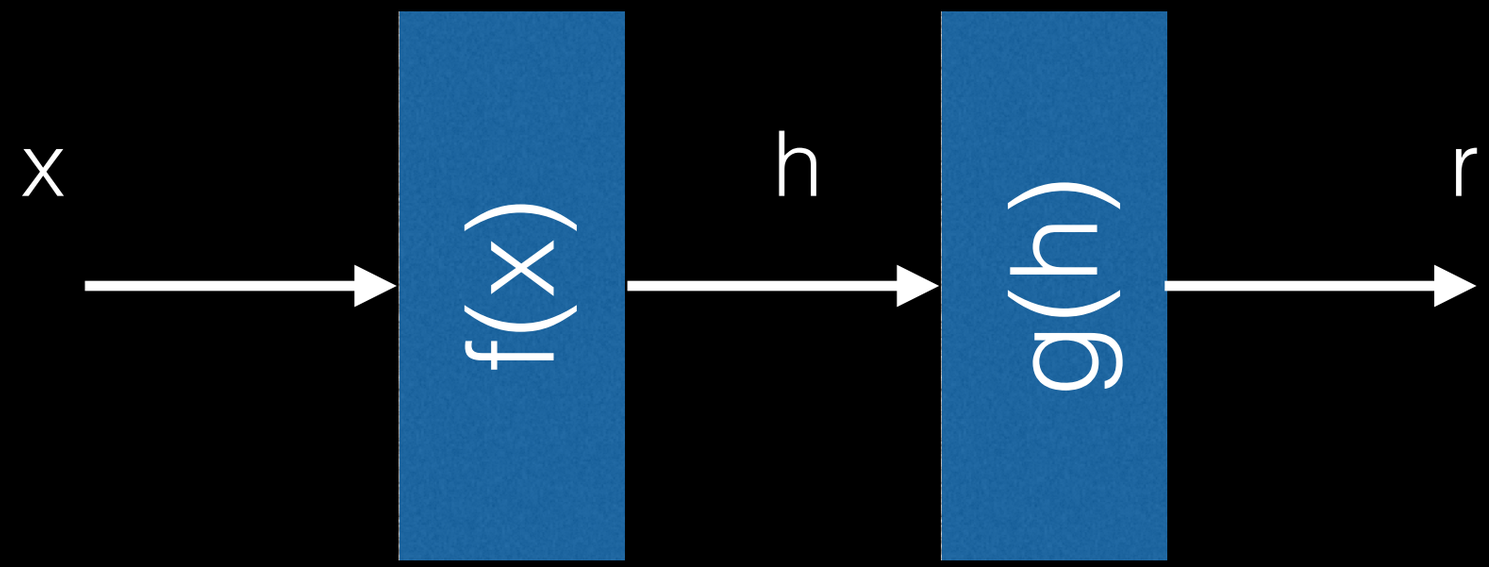
f is the **encoder** and g is the **decoder.**

Autoencoder



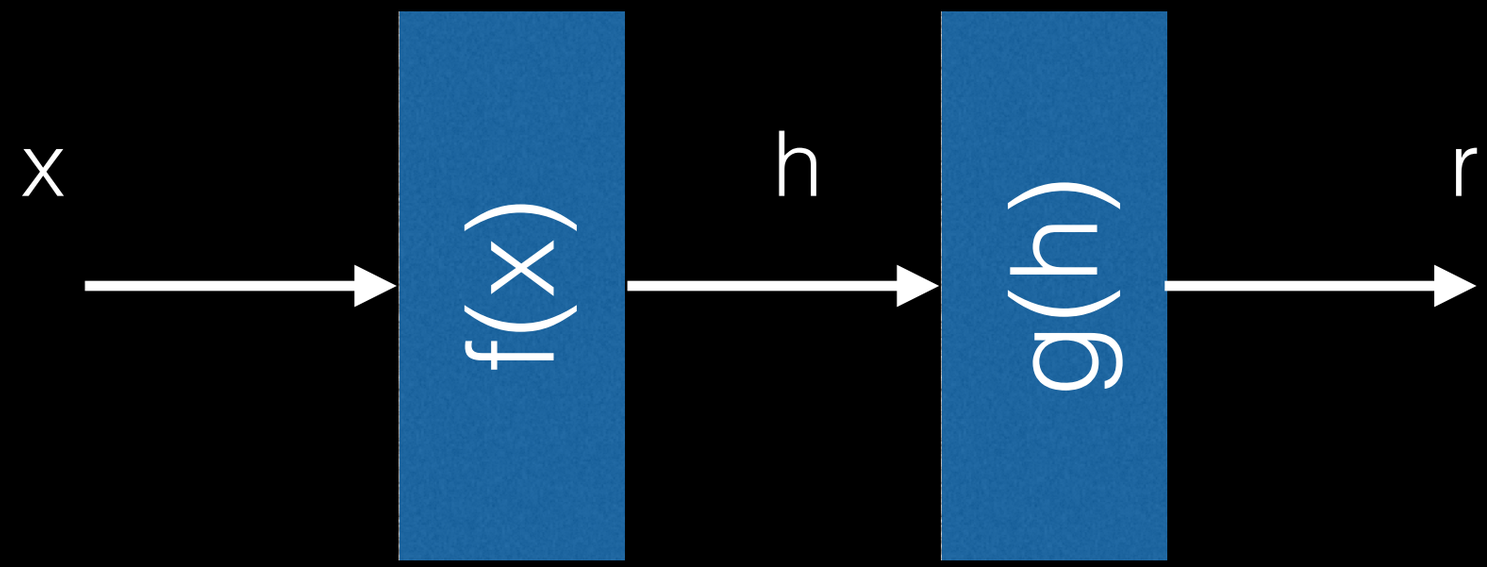
We want r to look like x and enforce this by a loss L(x, g(f(x))).

Autoencoder



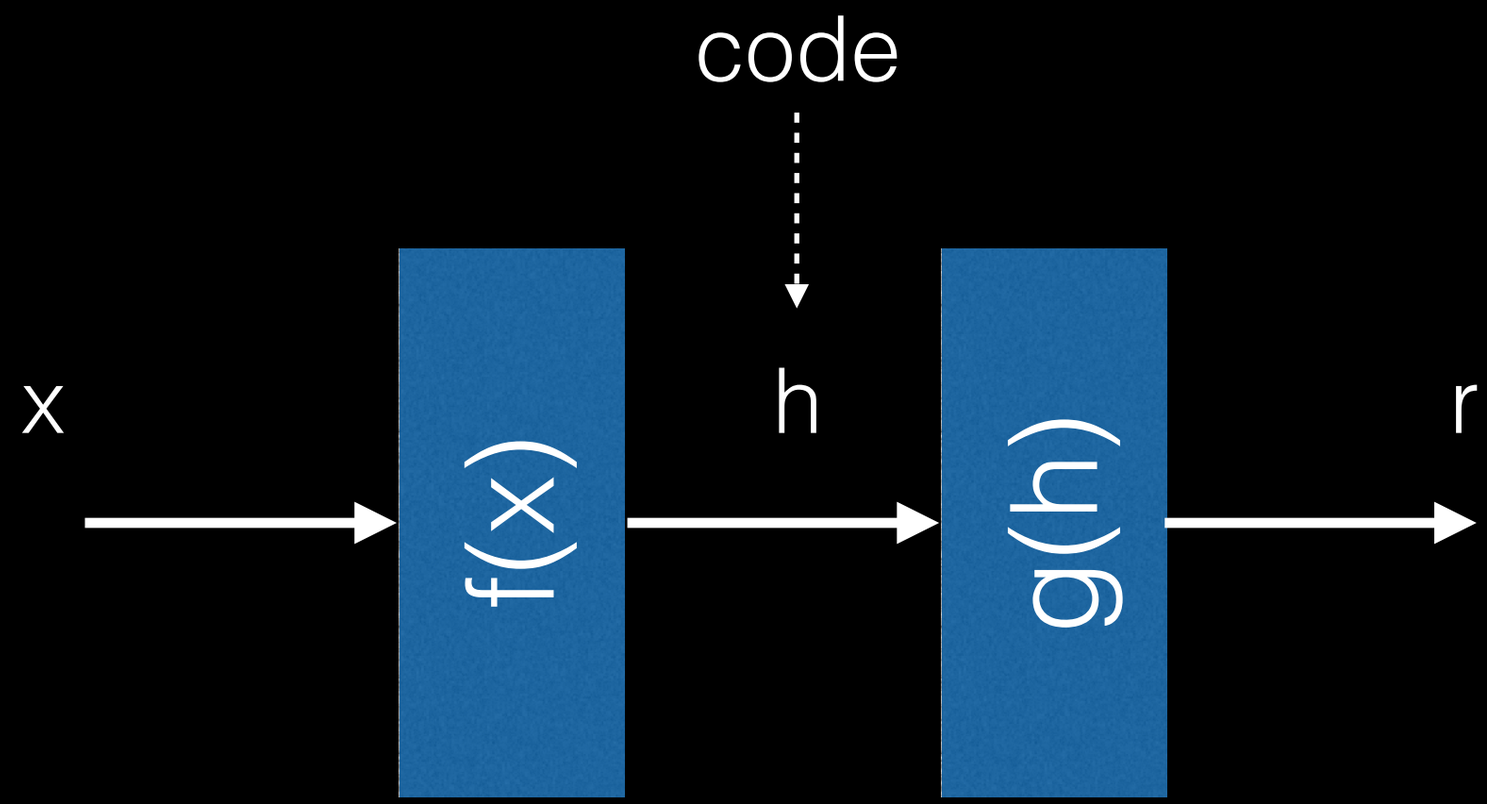
This penalizes g(f(x)) from being dissimilar to x, where we can measure dissimilarity by mean squared error…or some other metric.

Autoencoder



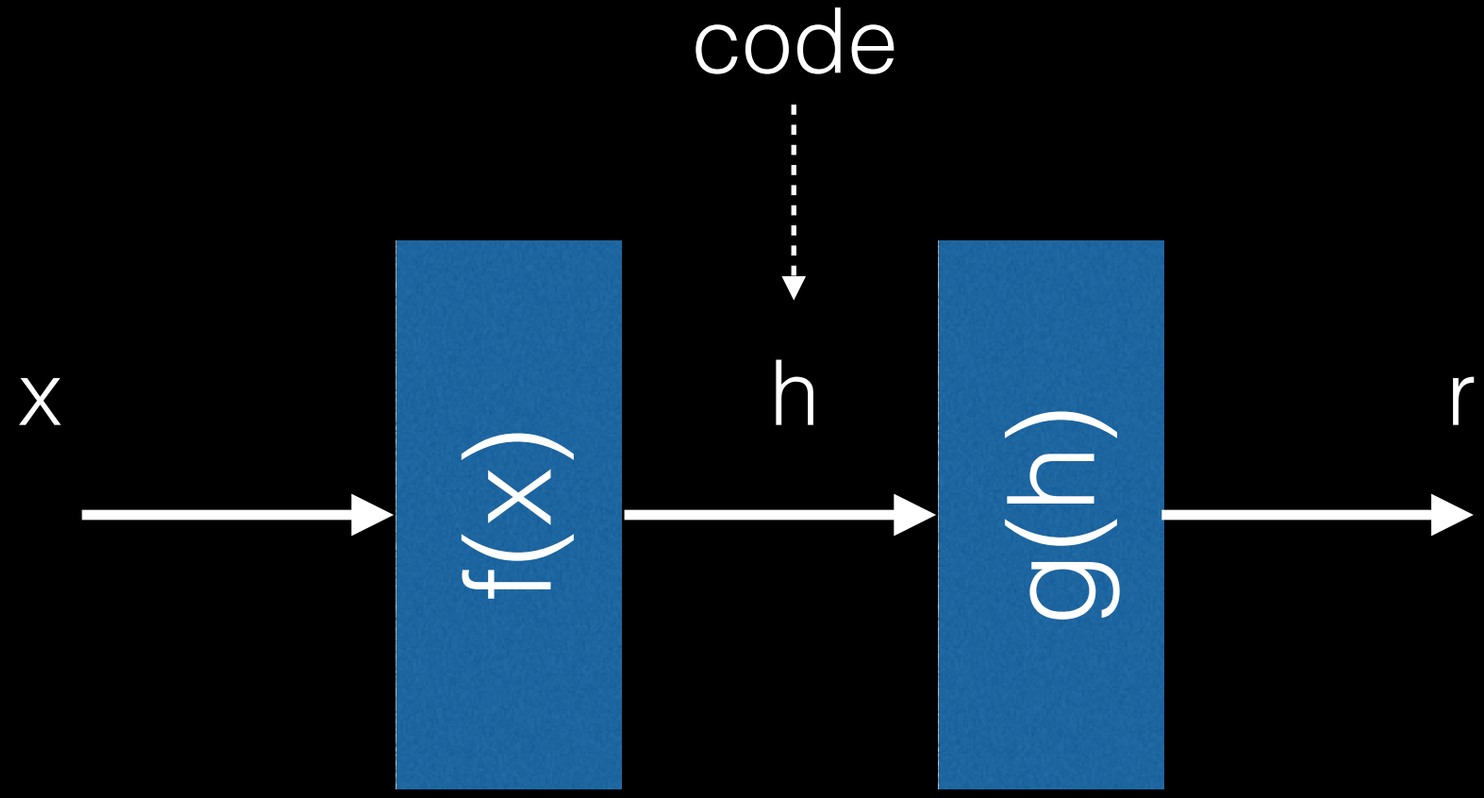
We hope the code h learns something fundamental about the input data x.

Autoencoder



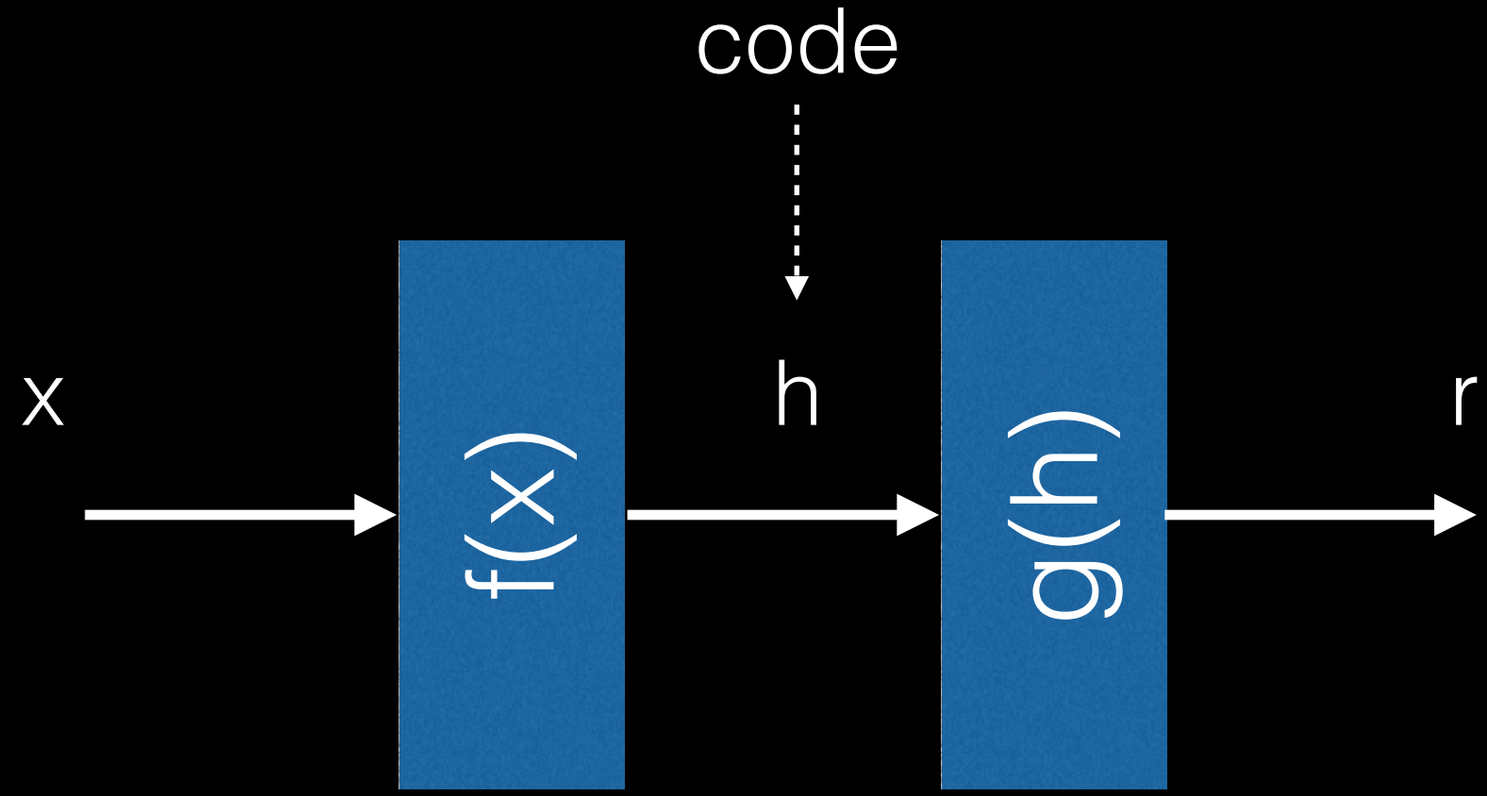
Then why not just make f(x) and g(h) equal to the identity?! This would minimize the loss.

Undercomplete Autoencoder



We can make the autoencoder **undercomplete** by forcing the code dimension h to be smaller than that of x.

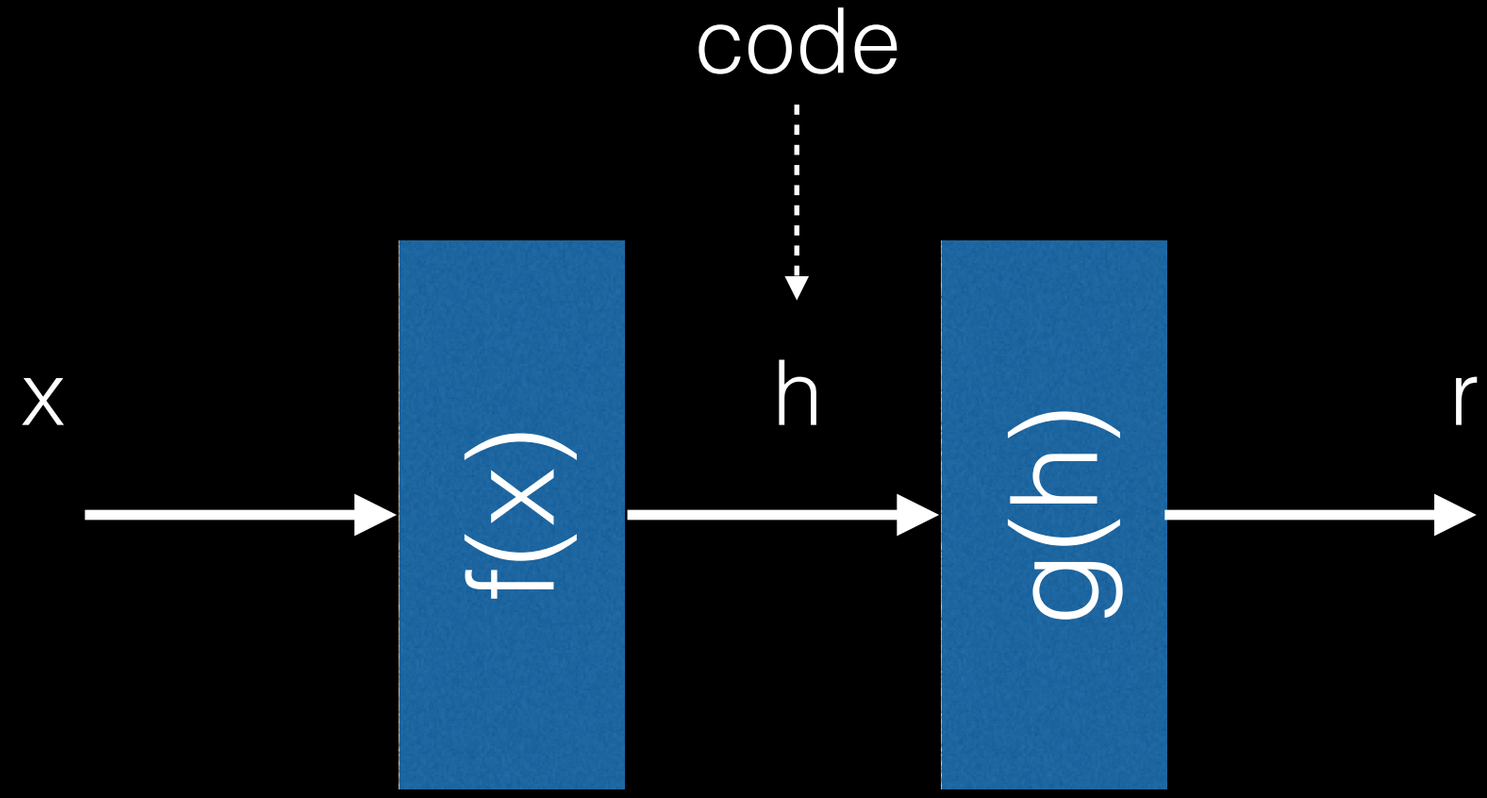
Undercomplete Autoencoder



If f and g are linear and the loss L is mean squared error, then this autoencoder learns the same dimensionality reducing subspaces as that learned by PCA.

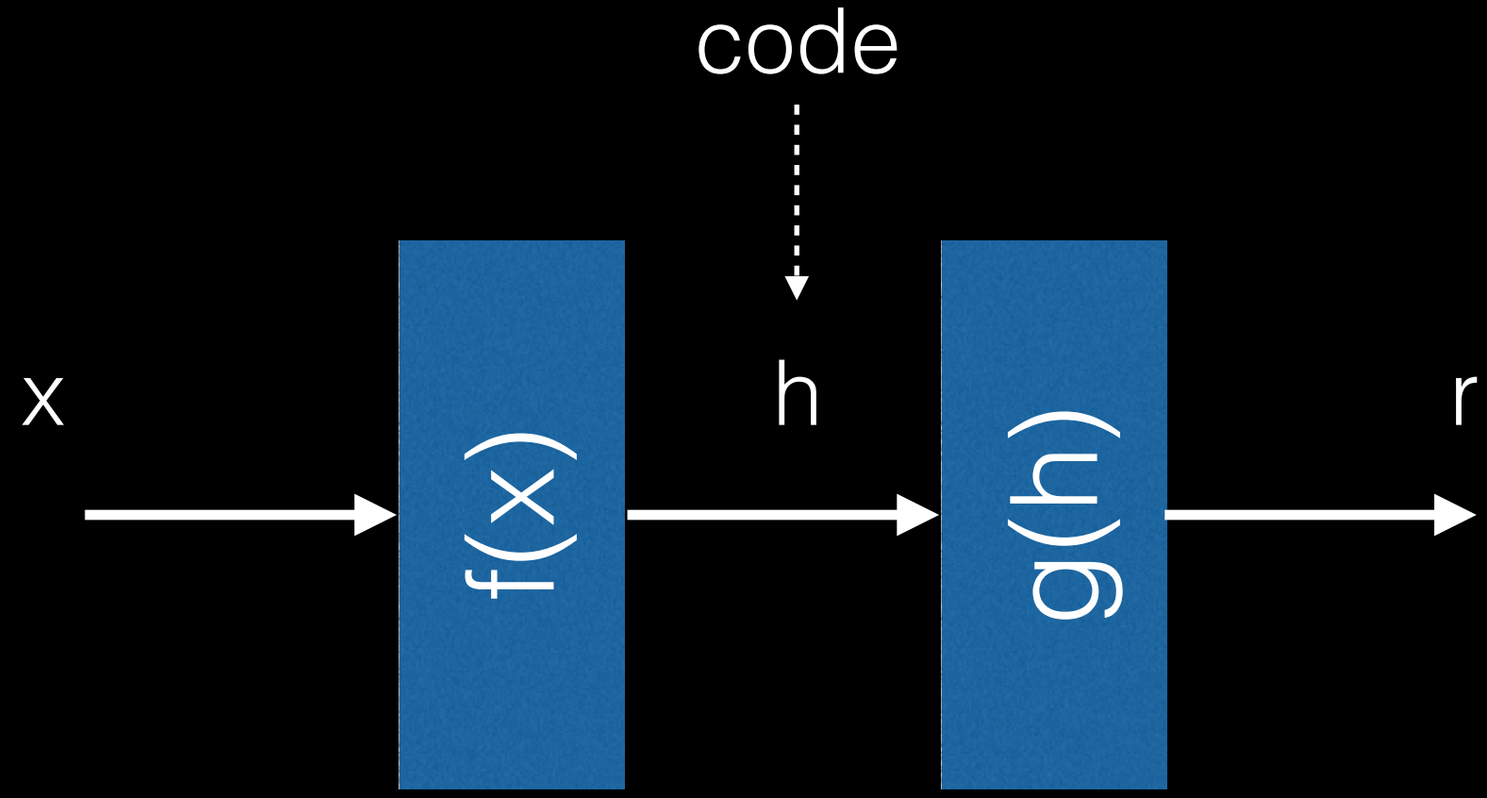
The hope is that we can force the network to learn a code h as a compact representation of the data, certainly beyond what would be possible with linear dimensionality reduction.

Regularized Autoencoder



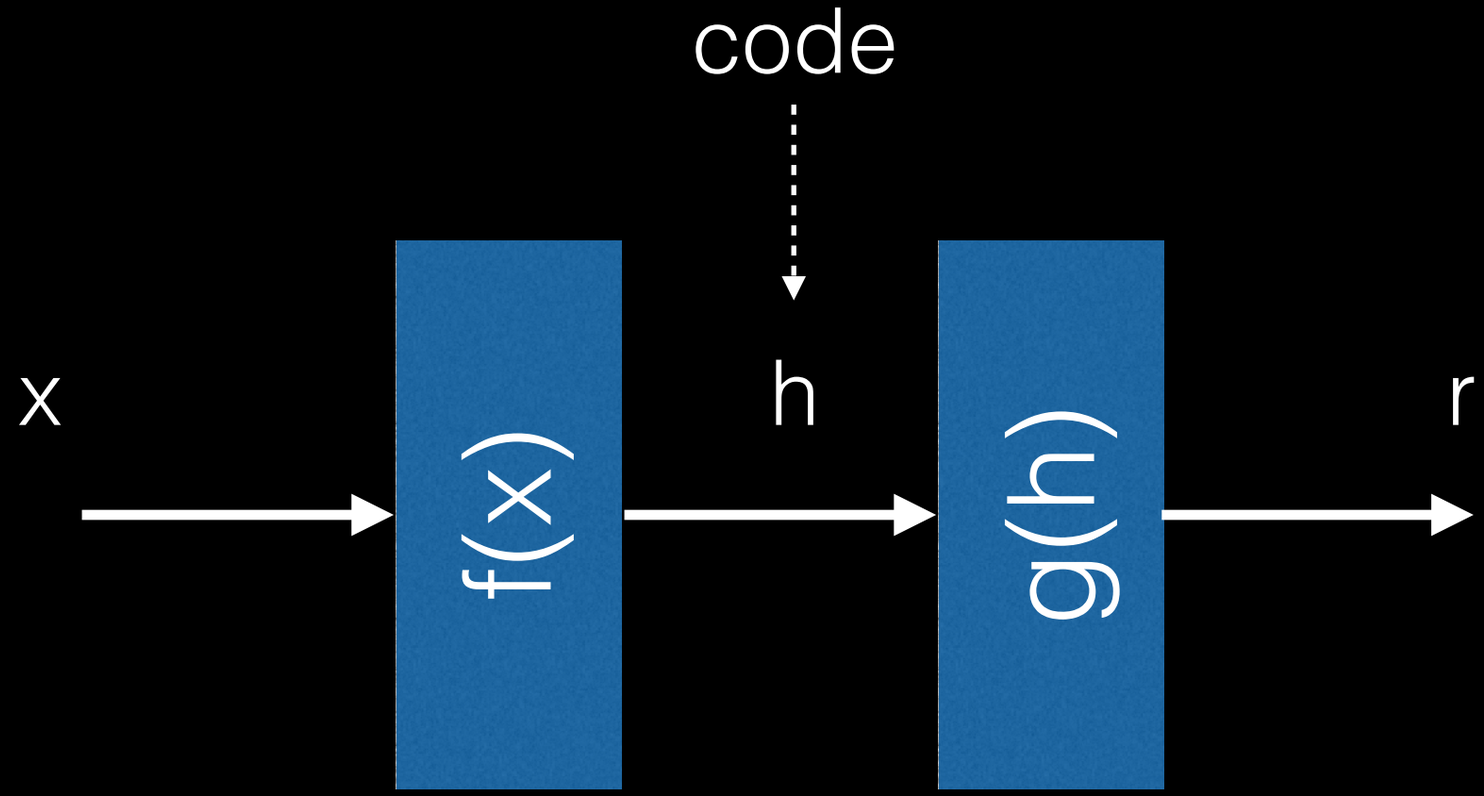
If the autoencoder **overcomplete** (dim h > dim x), then we can still get it to learn something useful, by adding **regularization**.

Regularized Autoencoder



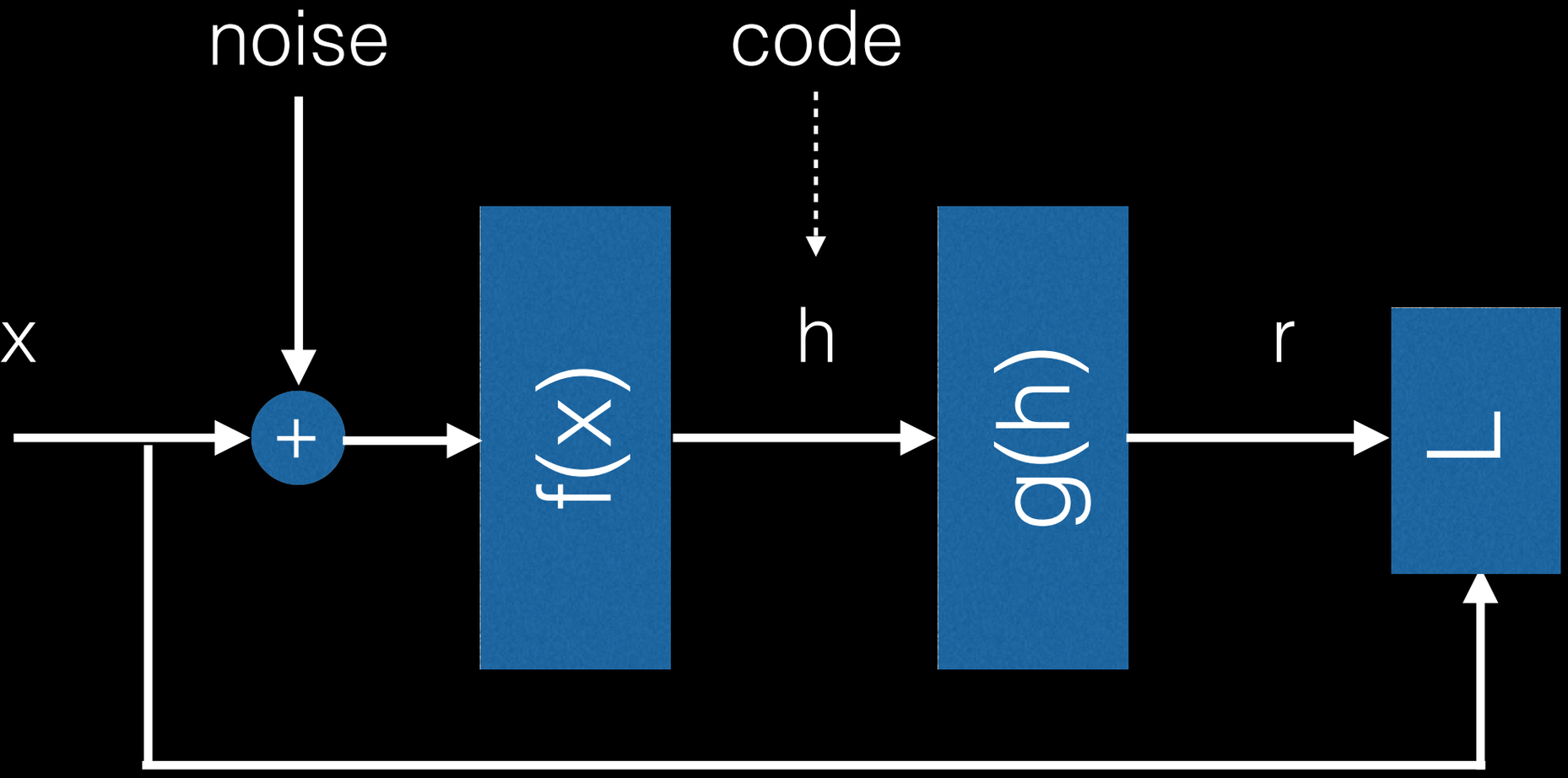
To do this, we can minimize L(x, g(f(x))) + R(h), where R(h) = $\lambda \sum_i |\mathrm{h}_i|$ is a L^1 regularizer used to enforce sparsity!

Regularized Autoencoder



Don’t forget that you are already an expert on sparsity!

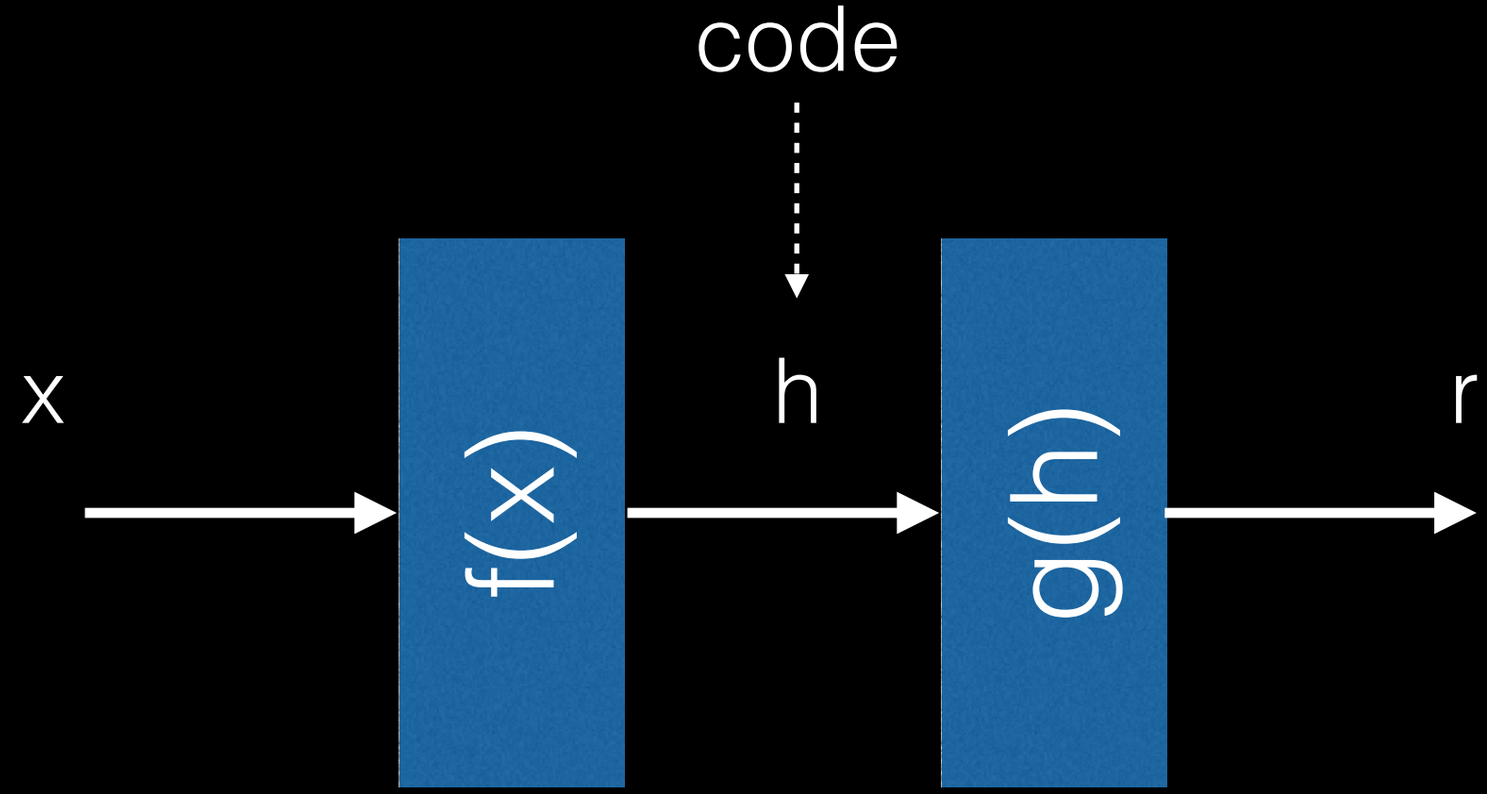
Denoising Autoencoder



Must remove noise to recreate the original noiseless signal. (Noise can be more than just additive.)

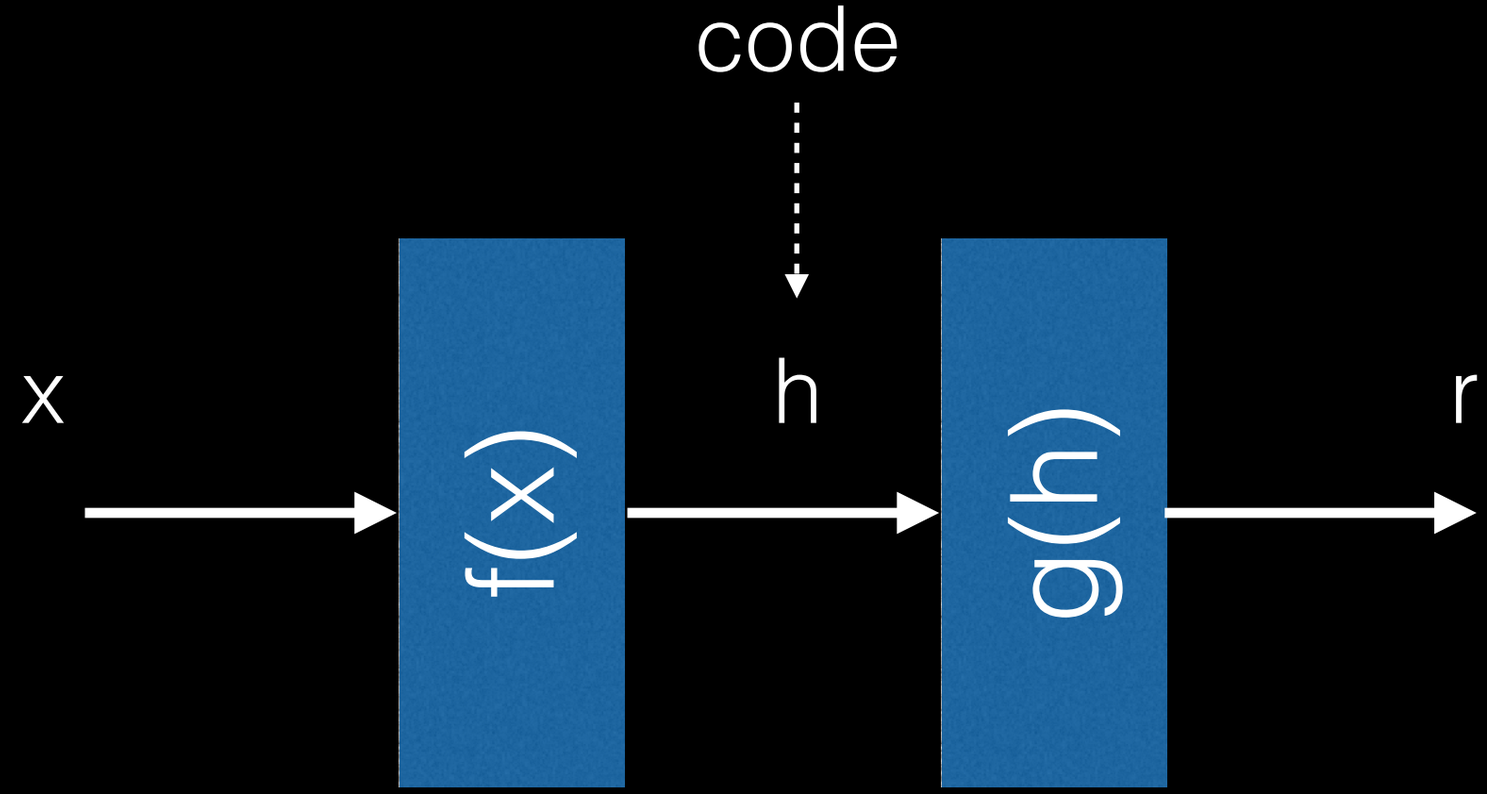
https://blog.keras.io/building-autoencoders-in-keras.html

Contractive Autoencoder (CAE)



We can chose the regularizer so that the code h doesn’t change much with small changes to the input x.

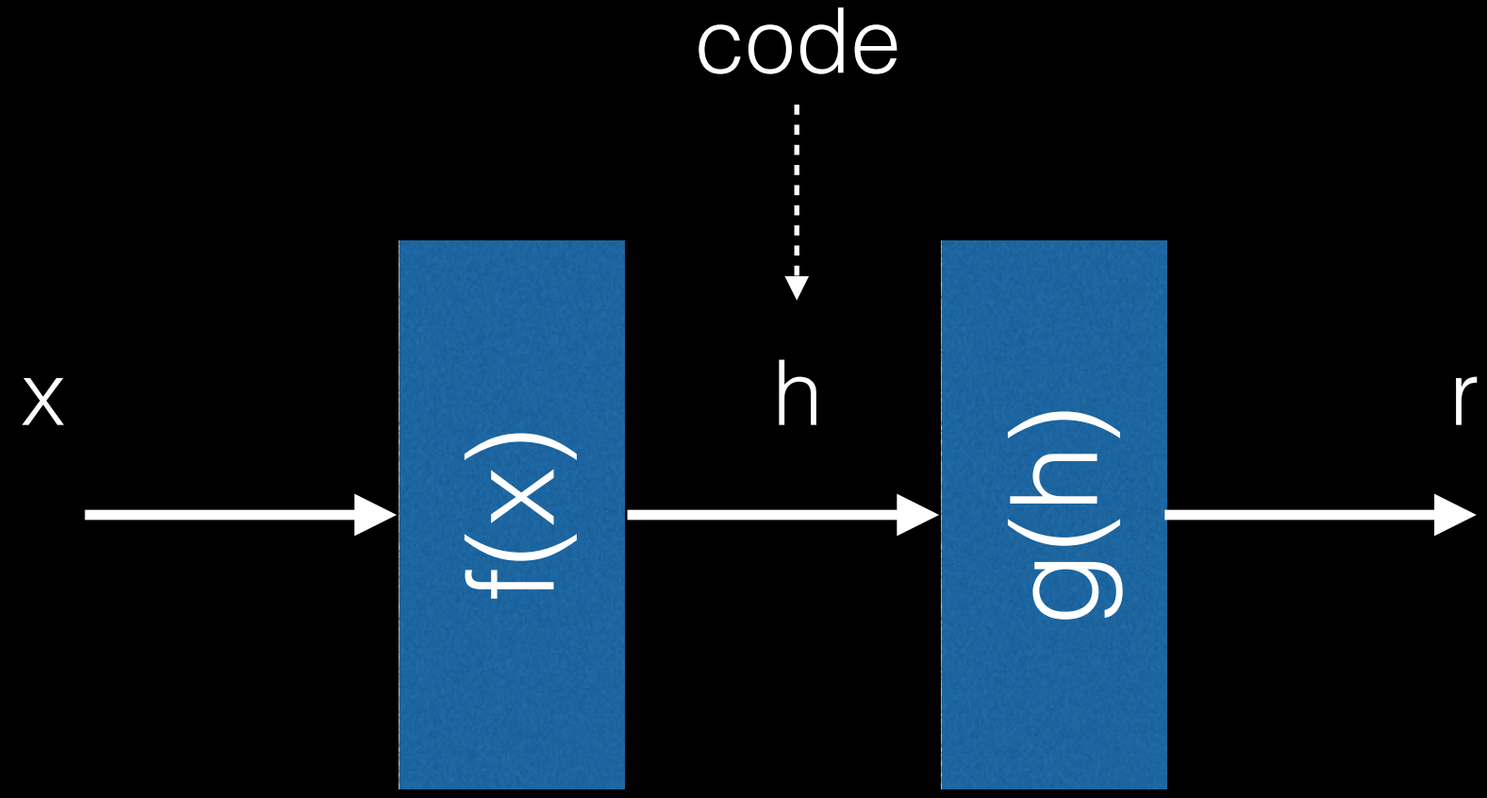
Contractive Autoencoder (CAE)



If we do this, then we force the code to keep similar inputs near each other once transformed into the code space h.

What should the R(h) look like to achieve this?

Contractive Autoencoder (CAE)



To do this, we can minimize L(x, g(f(x))) + R(h), where R(h) = $\lambda \sum_i ||\nabla_x \mathrm{h}_i||^2$.

So what is this Contractive Autoencoder doing?

Contractive Autoencoder

- CAE will try to find an code space h in which the inputs x are carefully arranged.
- So if two inputs x_1 and x_2 are close in the input space, they should be close or closer in the code space h.
- This is what the regularization is enforcing.
- So if the inputs have some natural ordering or layout, then the CAE should find this.
- This layout or ordering, along with the local coordinate system that parameterizes it is called a **manifold**.
- A mapping from an “ambient” input space x to code space h is called an embedding.

Contractive Autoencoder

- So how do we find this manifold structure from the CAE?
- Consider the Jacobian J of code h = f(x).
- The Jacobian matrix J evaluated at an input x should have most of its singular values < 1, since we have built the code space h to be contractive.
- But some the singular values of J will be > 1.
- And the directions of J corresponding to these singular values will specify our manifold.

What else can you do with autoencoders?

We have shown the autoencoders can reconstruct the inputs from undercomplete codes.

Maybe, these can be used to reconstruct the color channels in an image from only grayscale values!

Image Colorization [Zhang et al., 2016]

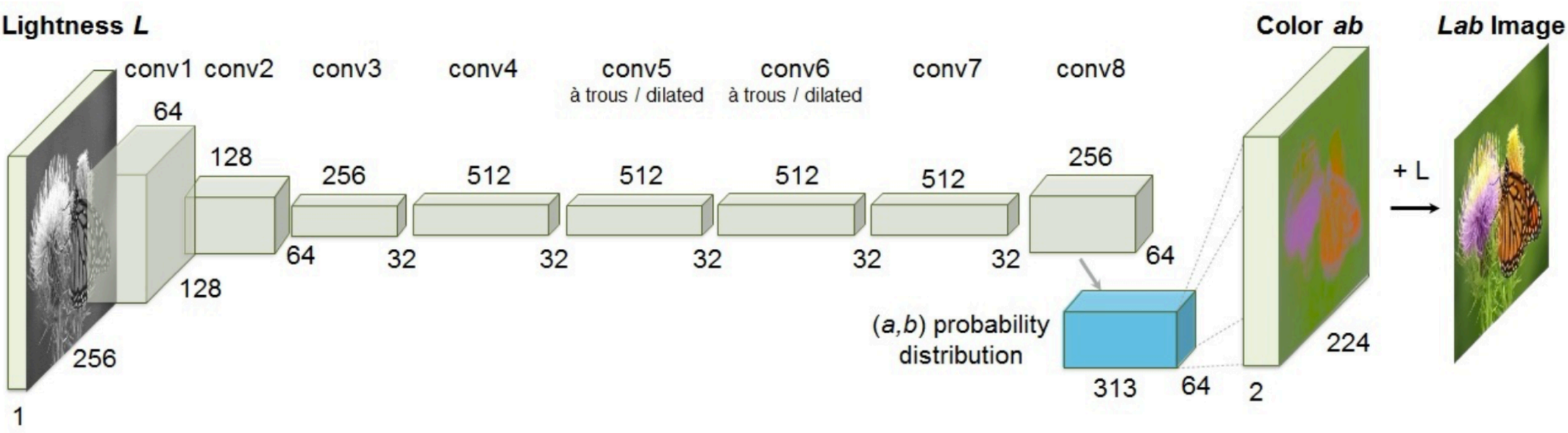


Image Colorization [Zhang et al., 2016]