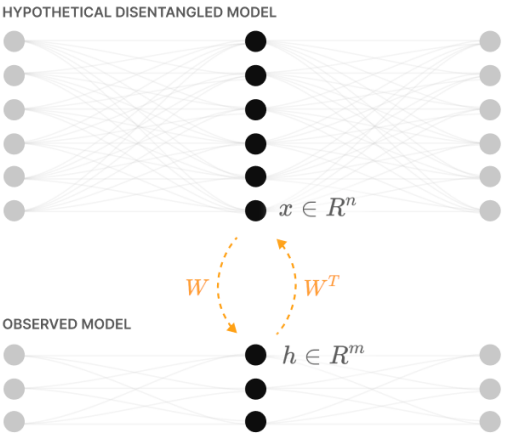
## Experiment Setup

Our goal is to explore whether a neural network can project a high dimensional vector $x \in R^n$ into a lower dimensional vector $h \in R^m$ and then recover it.[5]




Our first experiments will test the extent to which the idealized activations of an imagined larger model can be **stored** and **recovered** from a lower-dimensional space.

#### THE FEATURE VECTOR ($X$)

We begin by describing the high-dimensional vector $x$: the activations of our idealized, disentangled larger model. We call each element $x_i$ a "feature" because we're imagining features to be perfectly aligned with neurons in the hypothetical larger model. In a vision model, this might be a Gabor filter, a curve detector, or a floppy ear detector. In a language model, it might correspond to a token referring to a specific famous person, or a clause being a particular kind of description.

Since we don't have any ground truth for features, we need to create *synthetic data* for $x$ which simulates any important properties we believe features have from the perspective of modeling them. We make three major assumptions:

- **Feature Sparsity:** In the natural world, many features seem to be sparse in the sense that they only rarely occur. For example, in vision, most positions in an image don't contain a horizontal edge, or a curve, or a dog head [1]. In language, most tokens don't refer to Martin Luther King or aren't part of a clause describing music [2]. This idea goes back to classical work on vision and the statistics of natural images (see e.g. Olshausen, 1997, the section "Why Sparseness?" [24]). For this reason, we will choose a sparse distribution for our features.
- **More Features Than Neurons:** There are an enormous number of potentially useful features a model might represent.[6] This imbalance between features and neurons in real models seems like it must be a central tension in neural network representations.
- **Features Vary in Importance:** Not all features are equally useful to a given task. Some can reduce the loss more than others. For an ImageNet model, where classifying different species of dogs is a central task, a floppy ear detector might be one of the most important features it can have. In contrast, another feature might only very slightly improve performance.[7]

Concretely, our synthetic data is defined as follows: The input vectors $x$ are synthetic data intended to simulate the properties we believe the true underlying features of our task have. We consider each dimension $x_i$ to be a "feature". Each one has an associated sparsity $S_i$ and importance $I_i$. We let $x_i = 0$ with probability $S_i$, but is otherwise uniformly distributed between $[0, 1]$.[8] In practice, we focus on the case where all features have the same sparsity, $S_i = S$.