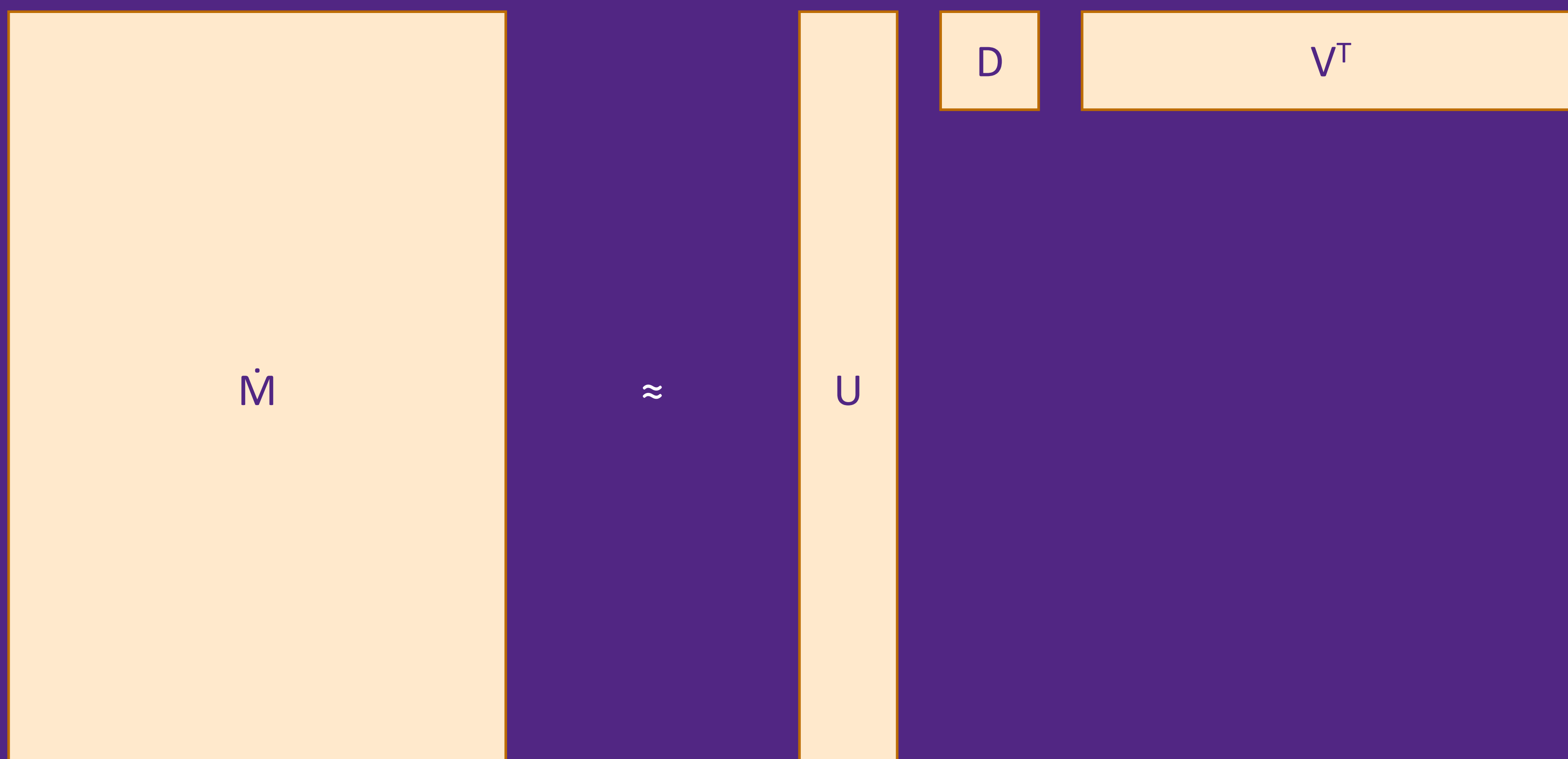
SVD/LSA as a Neural Network

Singular Value Decomposition

$$M_{m\times n} \approx \dot{M}_{m\times n} = U_{m\times p}\, D_{p\times p}\, {V^T}_{p\times n}$$



Rows of V represent documents

- $(\Sigma^{-1}{}_{p \times p}\, U^{T}{}_{p \times m})\dot{M}_{m \times n} = V^{T}{}_{p \times n}$

- Each element of column *j* of V^T is a weighted sum of a column of $\dot{M}$

- In NN terms:
  - one-layer network
  - *m* inputs, *p* outputs
  - fully-connected
  - linear transfer function (no ReLU or anything)
  - weights are $(\Sigma^{-1}{}_{p \times p}\, U^{T}{}_{p \times m})$

Rows of V represent documents

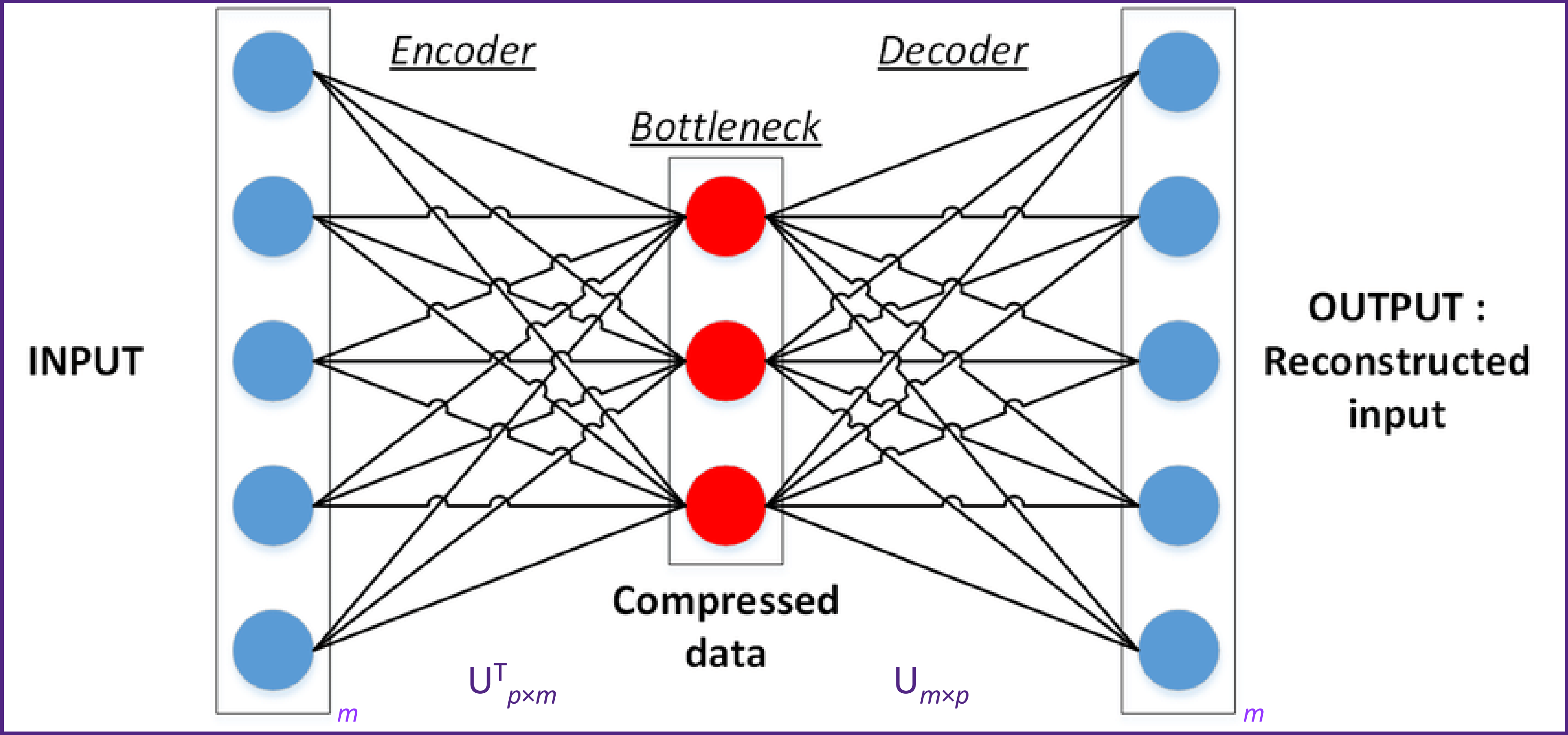
- $U_{m \times p} \Sigma_{p \times p} V^T{}_{p \times n} = \dot{M}_{m \times n}$

- Each element of column *j* of $\dot{M}$ is a weighted sum of a column of V^T

- In NN terms:
  - one-layer network
  - *p* inputs, *m* outputs
  - fully-connected
  - linear transfer function (no ReLU or anything)
  - weights are $(U_{m \times p} \Sigma_{p \times p})$ (I'll call them $\tilde{U}_{m \times p}$)

“LSA” Autoencoder

$$\dot{M}_{\cdot,j} = \tilde{U}_{m\times p}\, \tilde{U}^{T}_{p\times m}\, \dot{M}_{\cdot,j}$$



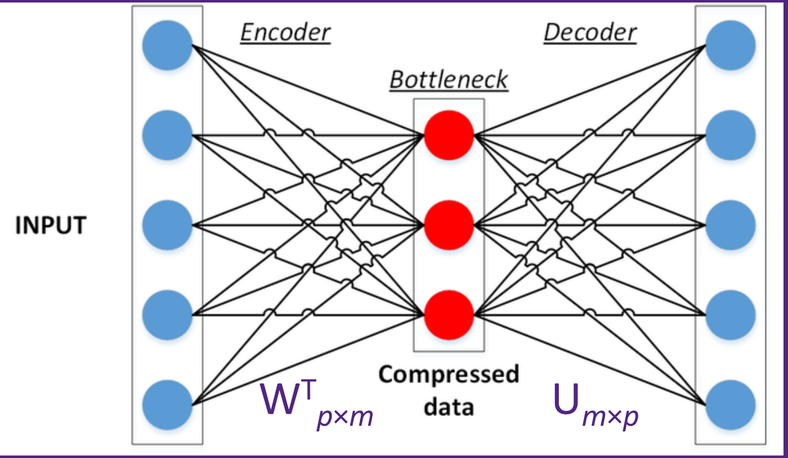
Question: - Are there any issues? (using m neurons in hidden units) A) Yes B) No

SVD/LSA Versus Autoencoder

- "auto"-encoder meaning "self"-encoder
  - Learns smaller representation for each input (column, document) vector
- Learned weights $\tilde{U}_{m \times p}$ give row (word) representations
- The entries in $\tilde{U}$ will not exactly match the ones you would get from SVD depending on how you train, but they span the same space.
- They're not "ordered" in terms of importance.
- Full discussion here (optional):
  - https://arxiv.org/pdf/1804.10253.pdf

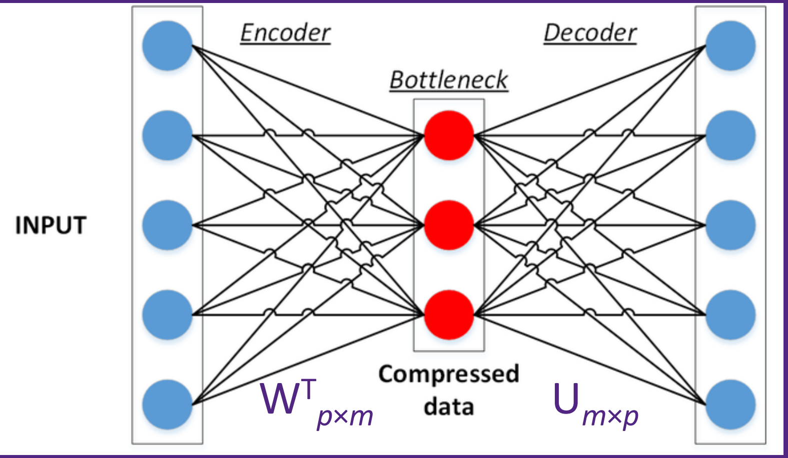
word2vec

- *Simplified from Mikolov, Tomas; et al. "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781 [Optional]*
- NN-based word representation learner
  - m input units (one per word)
  - p hidden units (dimension of representation)
  - m output units (one per word)
- **Output not same as input**
- $\varsigma(U_{m\times p}W^{T}_{p\times m}\mathbf{x}_{m\times 1}) = \mathbf{y}_{m\times 1}$



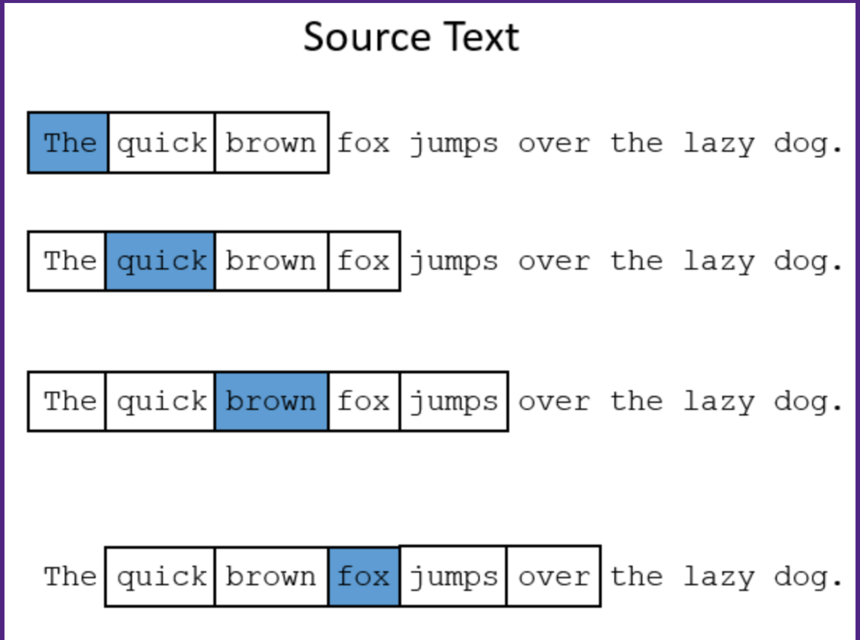
word2vec

- $\boldsymbol{\varsigma}(U_{m\times p}W^{T}_{p\times m}\mathbf{x}_{m\times 1}) = \mathbf{y}_{m\times 1}$
- **x** and **y** both represent a word or collection of words
- Matrix $U_{m\times p}$ that we learn will hold our word representations



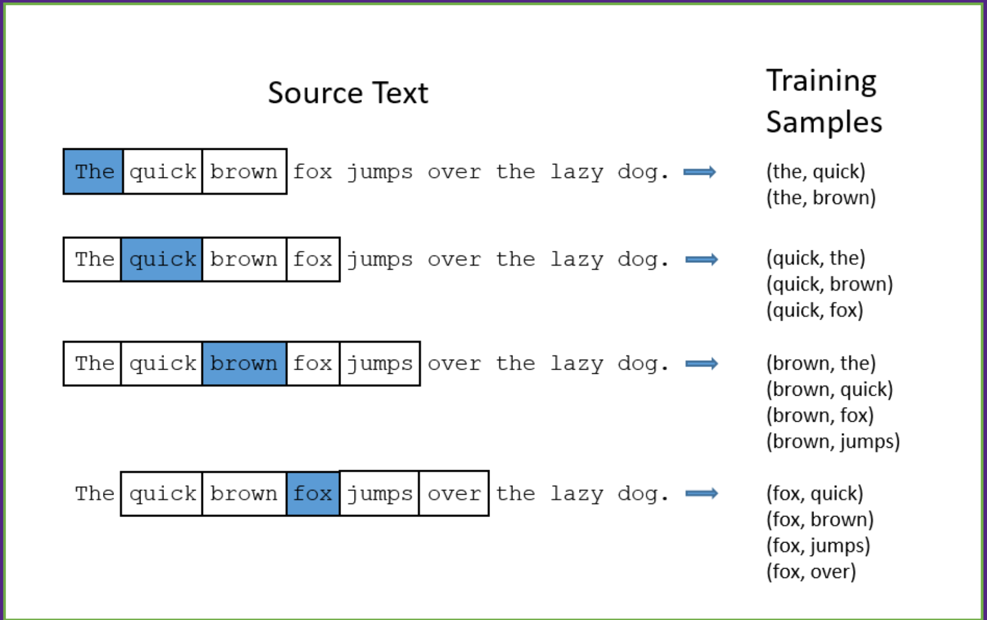
Choosing **x** and **y**: "Continuous Bag of Words"

- CBoW
  - Vector **y** (target) is one-hot encoding of a word
  - Vector **x** (input) is average of one-hot encodings of "context words"
  - Input-output pair created for every position of a "sliding window"



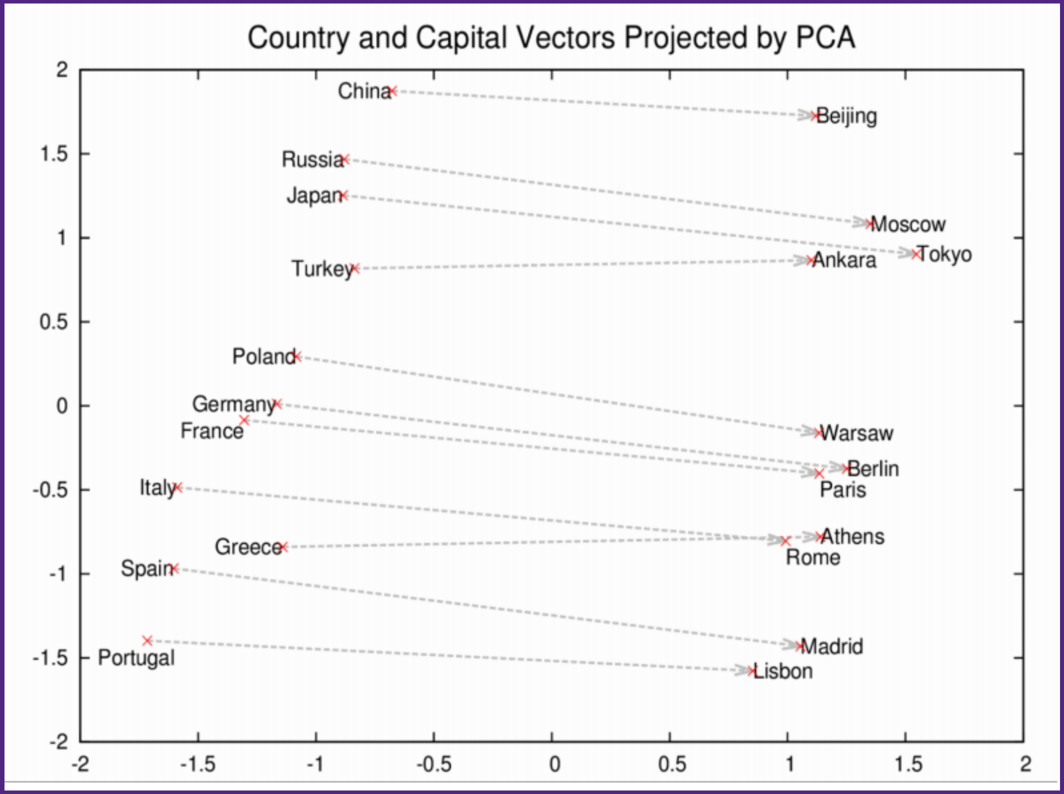
Choosing **x** and **y**: "Skip-Gram"

- Vector **x** (input) is one-hot encoding of a word
- Vector **y** (target) is one-hot encoding of a "nearby" word
- Multiple training vectors per sliding window position



"Analogy Task"

- "France is to Paris as Germany is to ________?"
- Paris – France + Germany = ???



Structured Representations from Neural Nets – In General

word2vec idea

- Purpose of word2vec was never to “predict an output”
- What matters is the representation created as a “side-effect” of training

Intermediate Representations

- Same intermediate representation → same output
- Advantageous for NN to map different inputs that should give same output to similar representation

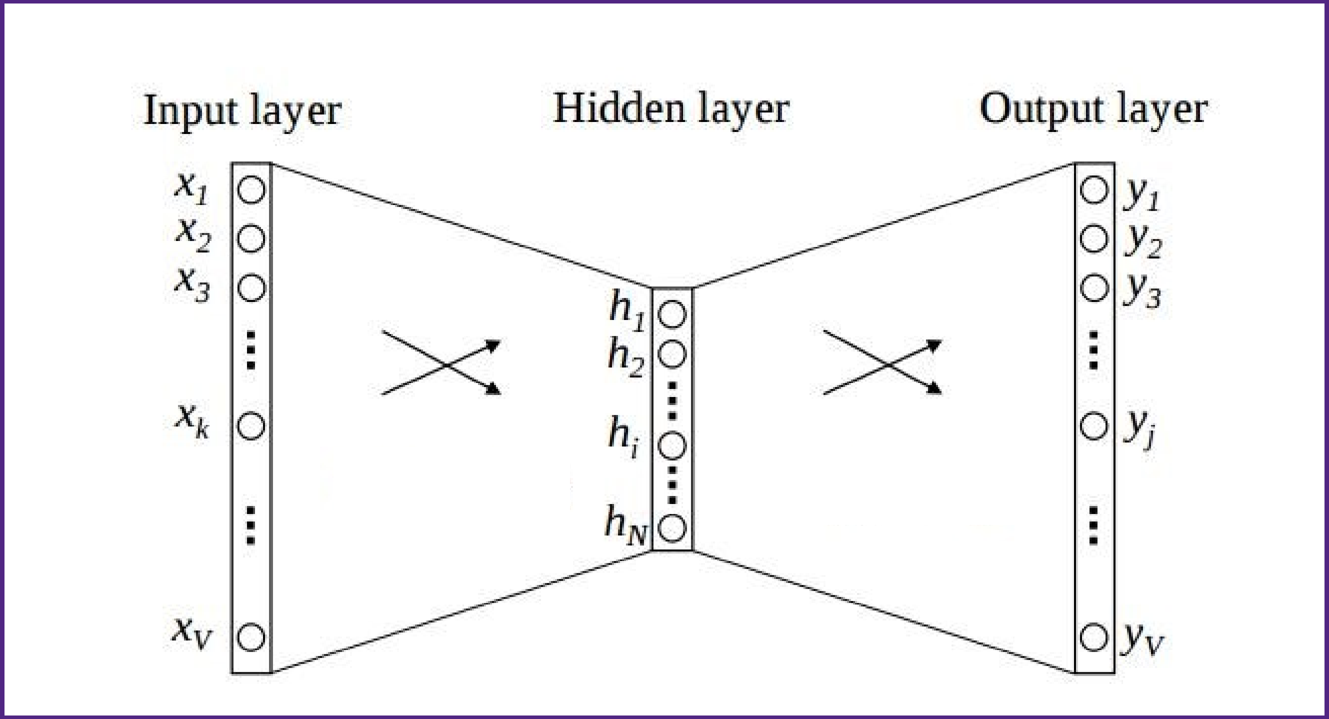
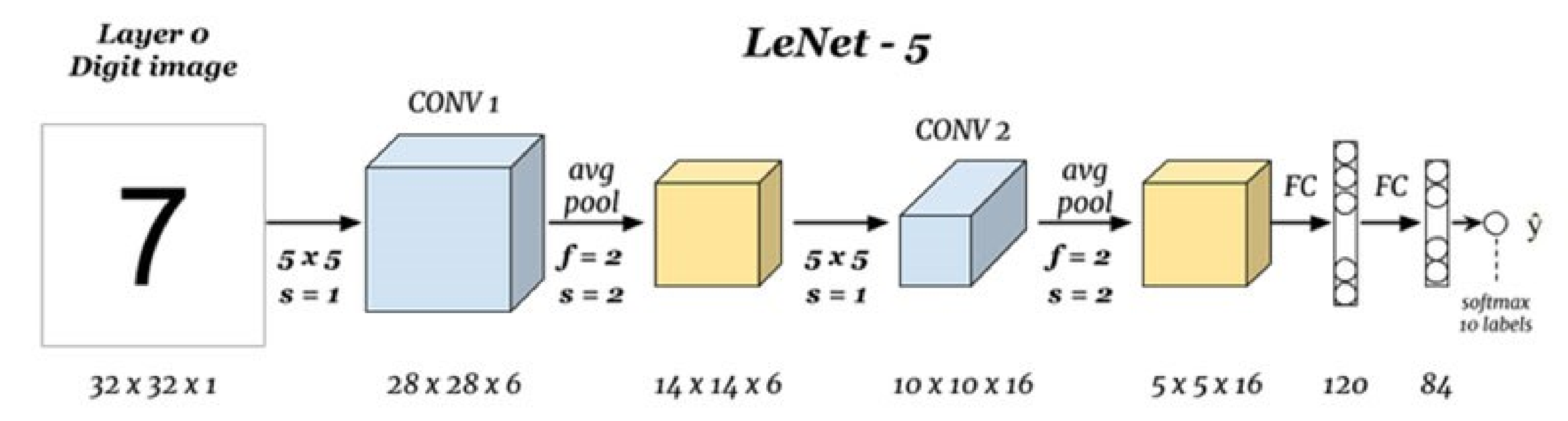


Image Recognition



1024 inputs

Designing Your Own Representations

- Is there a labelling task that can define a useful notion of similarity for you?
- Even if you can't learn that task "100%" you might still learn structure!
  - Recall: Skip Gram model trains on (brown, the), (brown, quick), (brown, fox), (brown, jumps) – impossible to produce a single "right answer" for input "brown"!
- Train a NN that has at least one layer that "compresses" the input.
- Output of that hidden layer, or the hidden layer weights, become your representation

Summary

- We can learn representations that capture relational structure
- LSA/SVD Does this
- LSA/SVD is similar to Autoencoding neural networks
- Other learning tasks and architectures learn different structures
  - CBoW and Skip Grams for word2vec
  - Classification for images
- You can design your own!