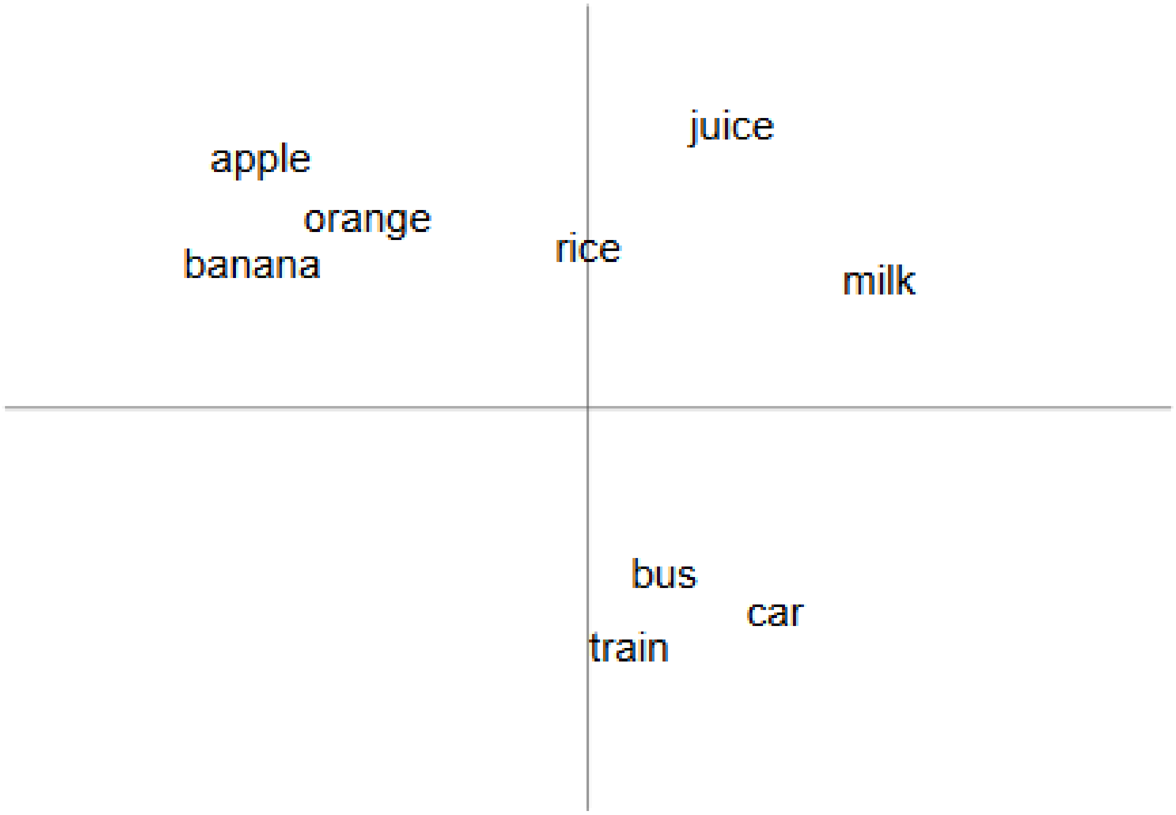
Embedding Techniques

Many slides from Girish K (Texas A&M) and Guy Golan (Israel)

“A word is known by the company it keeps”



Reference Materials

- Deep Learning for NLP by Richard Socher (http://cs224d.stanford.edu/)
- Tutorial and Visualization tool by Xin Rong (http://www-personal.umich.edu/~ronxin/pdf/w2vexp.pdf)
- Word2vec in Gensim by Radim Řehůřek (http://rare-technologies.com/deep-learning-with-word2vec-and-gensim/)

Word Representations

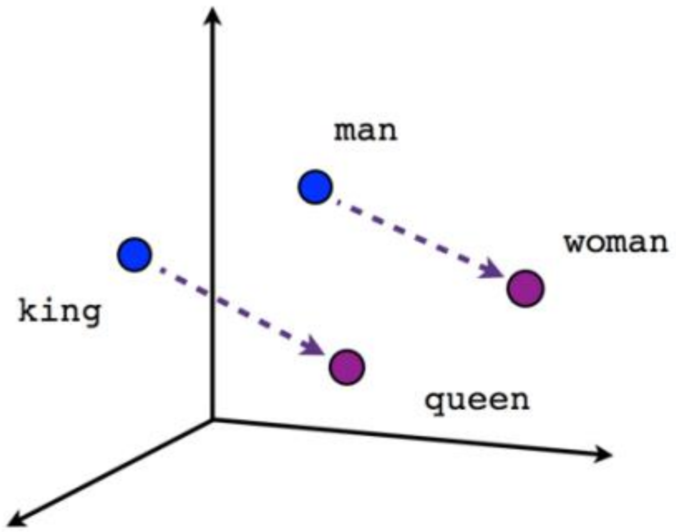
Traditional Method - Bag of Words Model	Word Embeddings
• Uses one hot encoding • Each word in the vocabulary is represented by one bit position in a HUGE vector. • For example, if we have a vocabulary of 10000 words, and “Hello” is the 4^{th} word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . 0 0 0 0 • Context information is not utilized	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, “Hello” might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

The Power of Word Vectors

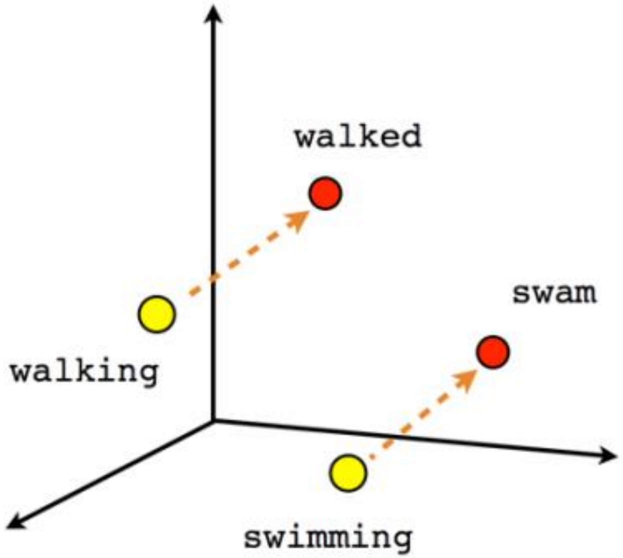
- They provide a fresh perspective to ***ALL*** problems in NLP, and not just solve one problem.

- They work in an unsupervised way: Not dependent on hand-labelled data.

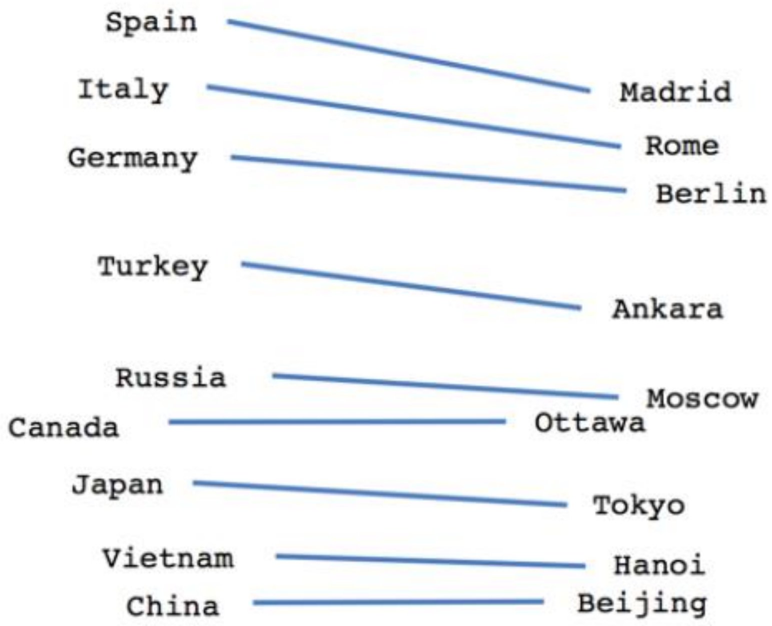
Examples



Male-Female



Verb tense



Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

So, how exactly does Word Embedding ‘solve all problems in NLP’?

Applications of Word Vectors

1. <u>Word Similarity</u>

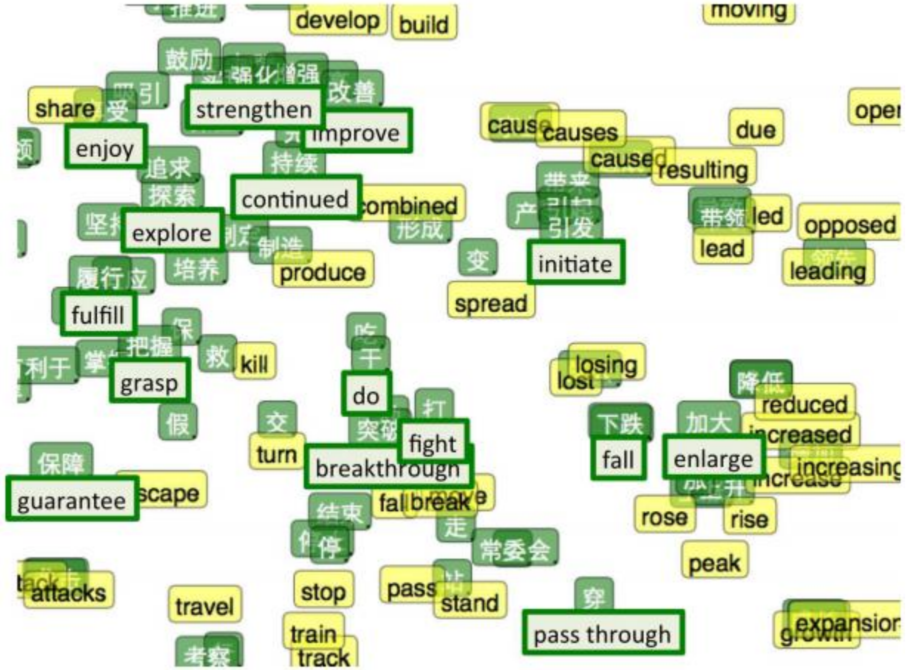
Classic Methods : Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries

- Easily identifies similar words and synonyms since they occur in similar contexts
- Stemming (thought -> think)
- Inflections, Tense forms
- *eg. Think, thought, ponder, pondering,*
- *eg. Plane, Aircraft, Flight*

Applications of Word Vectors

2. Machine Translation

Classic Methods: Rule-based machine translation, morphological transformation



Applications of Word Vectors

3. Part-of-Speech and Named Entity Recognition

Classic Methods: Sequential Models (MEMM , Conditional Random Fields), Logistic Regression

	POS WSJ (acc.)	NER CoNLL (F1)
State-of-the-art*	97.24	89.31
Supervised NN	**96.37**	**81.47**
Unsupervised pre-training followed by supervised NN**	**97.20**	**88.87**
+ hand-crafted features***	97.29	89.59

Applications of Word Vectors

4. Relation Extraction

Classic Methods : OpenIE, Linear programing models, Bootstrapping

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Applications of Word Vectors

5. Sentiment Analysis

Classic Methods : Naive Bayes, Random Forests/SVM

- Classifying sentences as positive and negative
- Building sentiment lexicons using seed sentiment sets
- No need for classifiers, we can just use cosine distances to compare unseen reviews to known reviews.

```
Enter word or sentence (EXIT to break): sad

Word: sad  Position in vocabulary: 4067

                                      Word       Cosine distance
------------------------------------------------------------------------
                                 saddening              0.727309
                                       Sad              0.661083
                                  saddened              0.660439
                              heartbreaking             0.657351
                              disheartening             0.650732
                              Meny_Friedman             0.648706
                     parishioner_Pat_Patello            0.647586
                                 saddens_me             0.640712
                                distressing             0.639909
                          reminders_bobbing             0.635772
                            Turkoman_Shiites            0.635577
                                    saddest             0.634551
                                unfortunate             0.627209
                                      sorry             0.619405
                                bittersweet             0.617521
                                     tragic             0.611279
                                  regretful             0.603472
```

Applications of Word Vectors

6. Co-reference Resolution

- Chaining entity mentions across multiple documents - can we find and unify the multiple contexts in which mentions occurs?

7. Clustering

- Words in the same class naturally occur in similar contexts, and this feature vector can directly be used with any conventional clustering algorithms (K-Means, agglomerative, etc). Human doesn’t have to waste time hand-picking useful word features to cluster on.

8. Semantic Analysis of Documents

- Build word distributions for various topics, etc.

Building these magical vectors . . .

- How do we actually build these super-intelligent vectors, that seem to have such magical powers?

- How to find a word's friends?

- We will discuss the most famous methods to build such lower-dimension vector representations for words based on their context
    1. Co-occurrence Matrix with SVD
    2. word2vec (*Google)*
    3. Global Vector Representations (GloVe) (*Stanford)*

Co-occurrence Matrix with Singular Value Decomposition

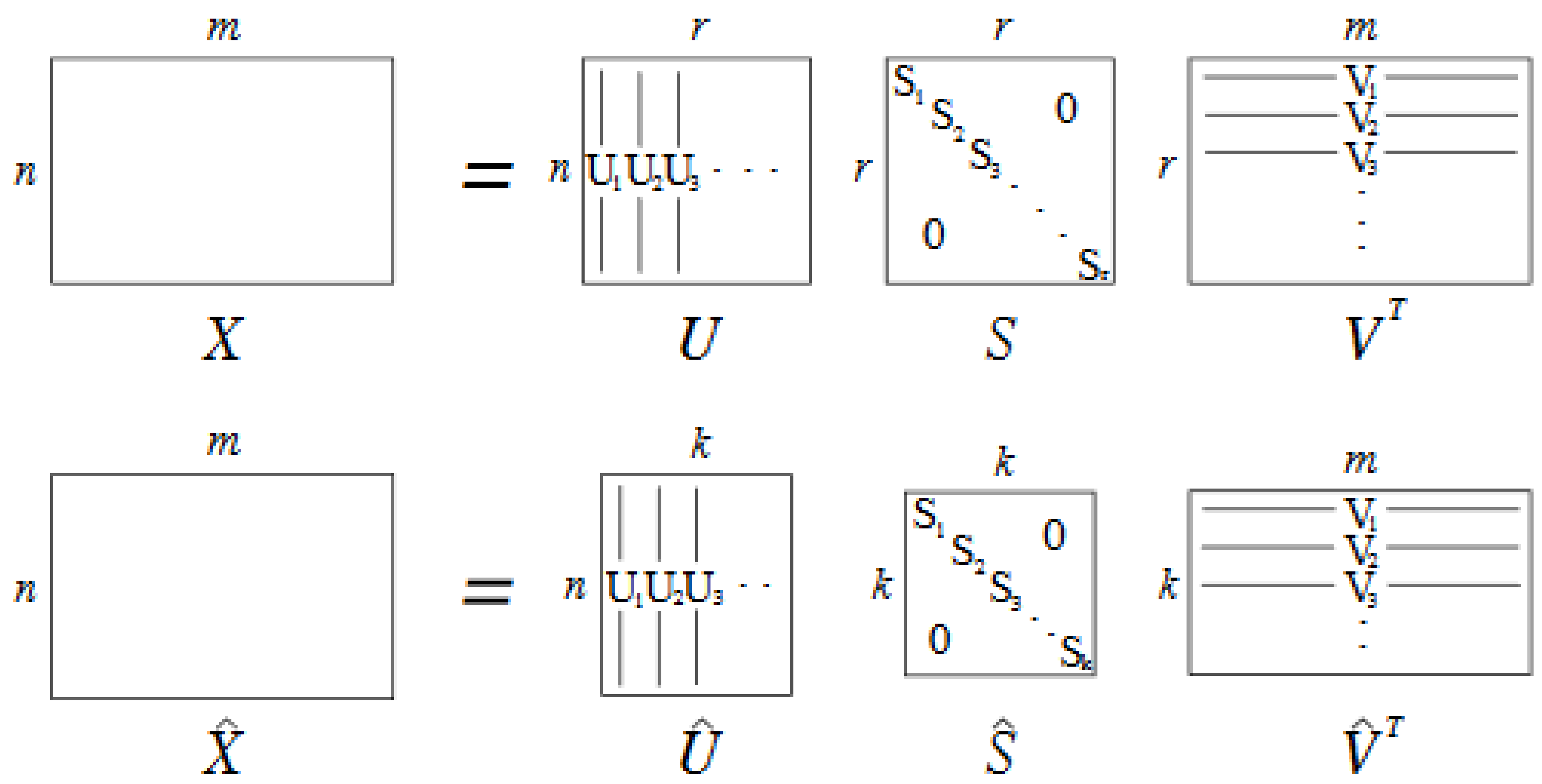
Building a co-occurrence matrix

Corpus = {"I like deep learning"
"I like NLP"
"I enjoy flying"}

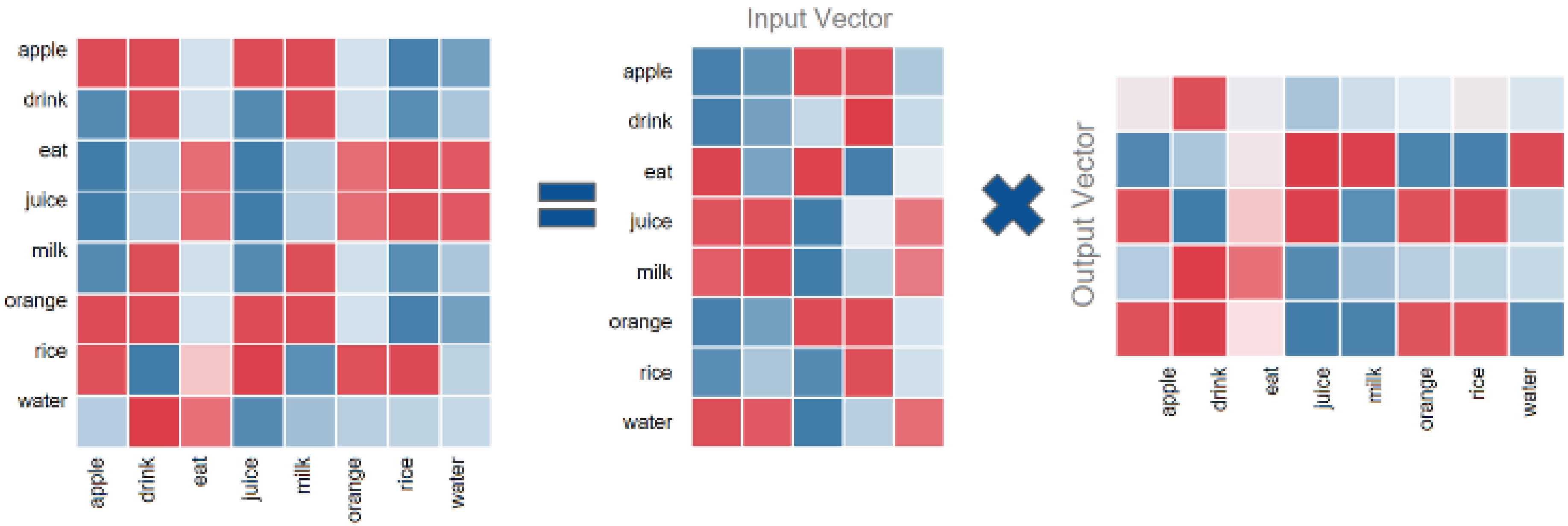
Context = previous word and next word

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

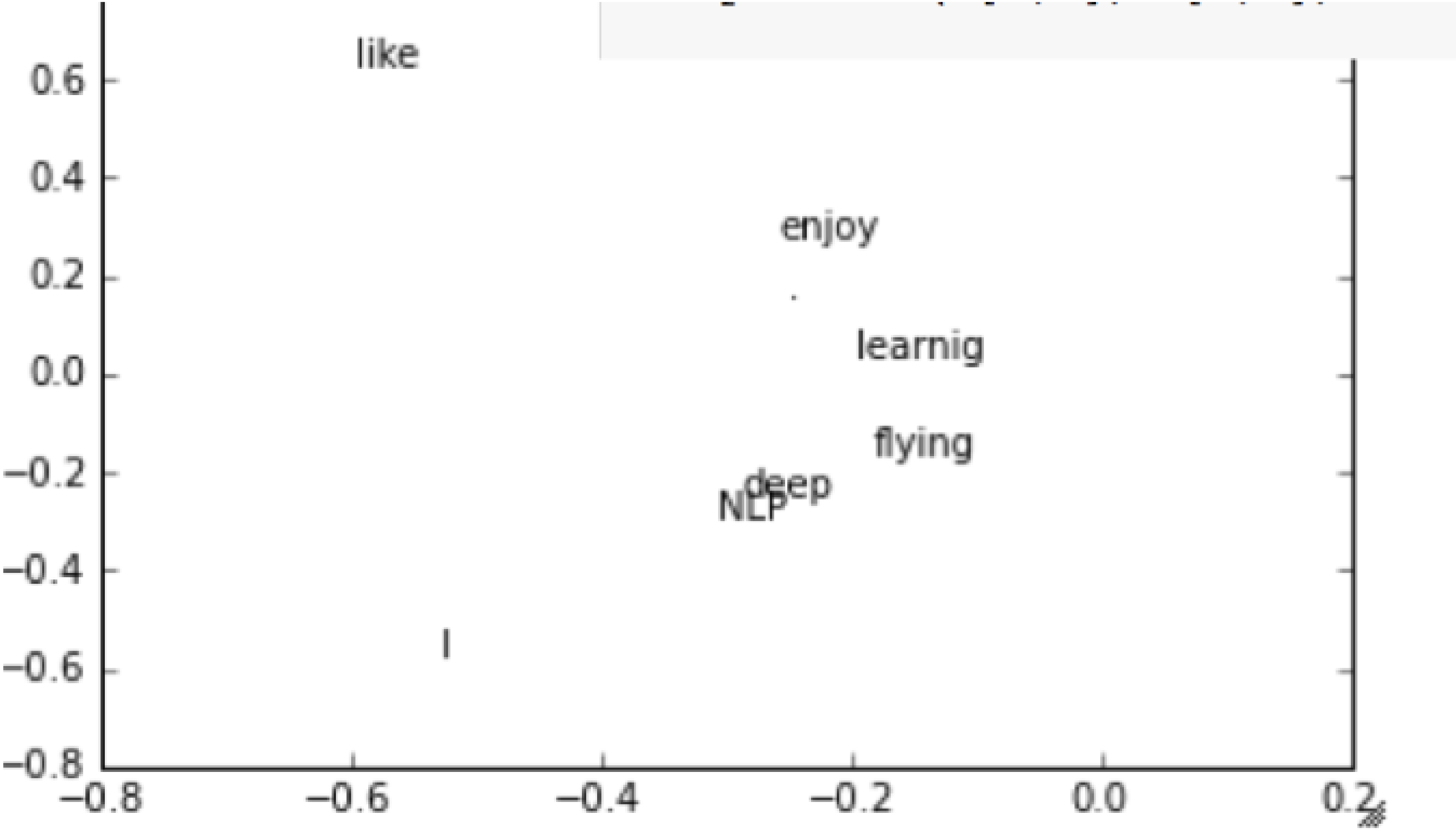
Dimension Reduction using Singular Value Decomposition



Singular Value Decomposition

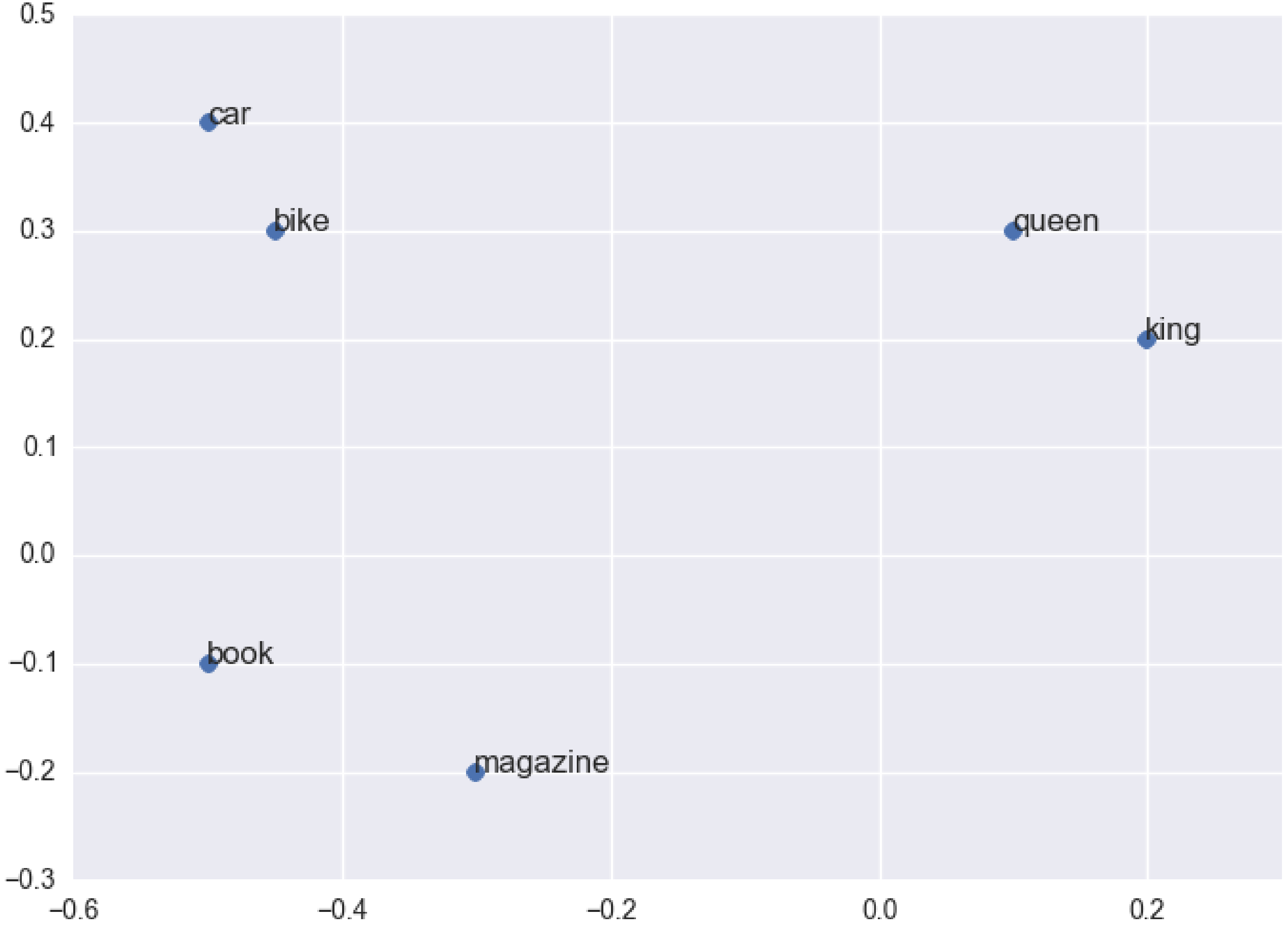


The problem with this method, is that we may end up with matrices having billions of rows and columns, which makes SVD computationally restrictive.



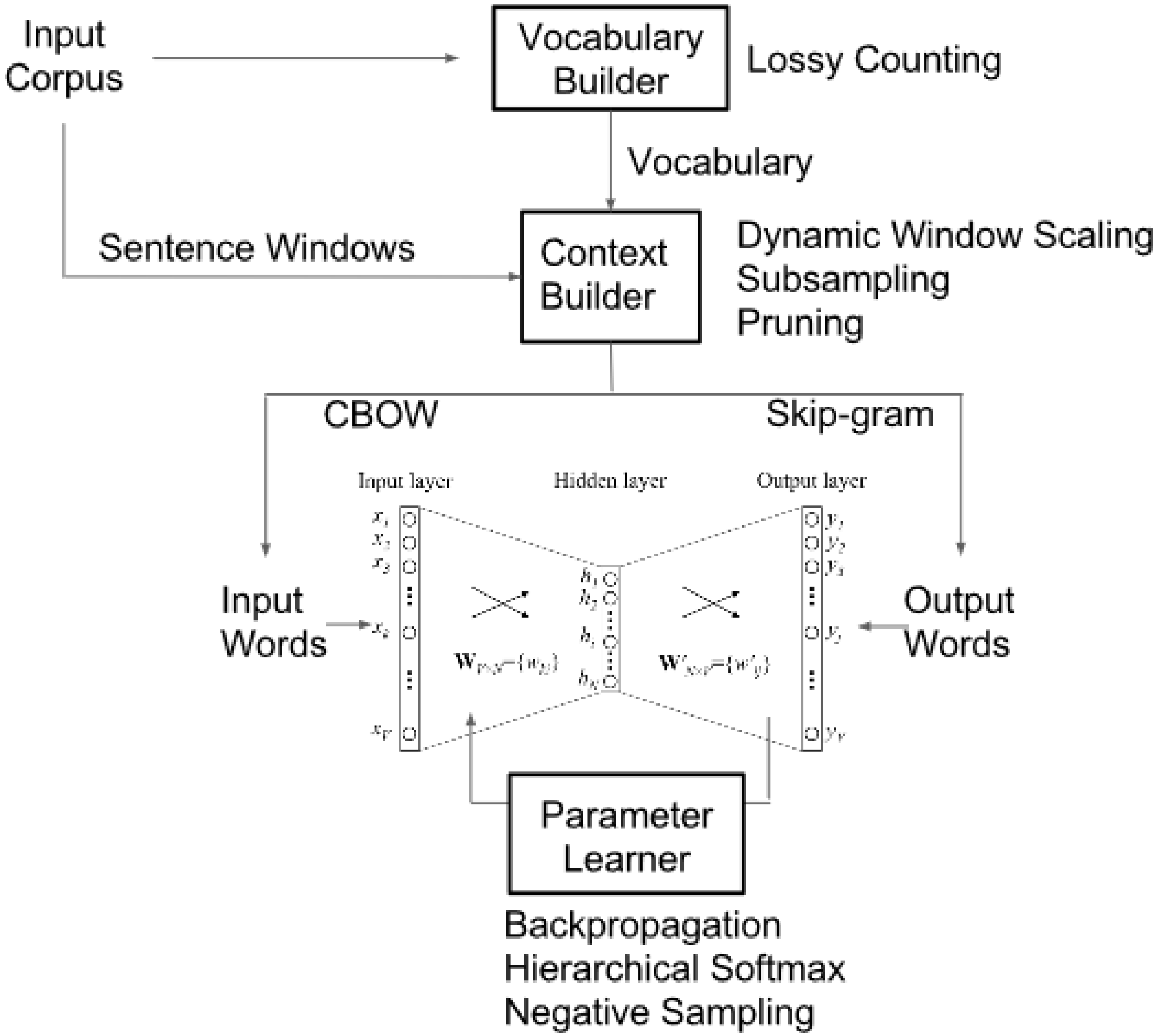
like
enjoy
learnig
flying
deep
NLP
I
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2

word2vec



car
bike
queen
king
book
magazine
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.6
−0.4
−0.2
0.0
0.2

Architecture



Context windows

- Context can be anything – a surrounding n-gram, a randomly sampled set of words from a fixed size window around the word

For example, assume context is defined as the word following a word.

i.e. $\boldsymbol{context(w_i) = w_{i+1}}$

Corpus : I ate the cat

Training Set : I|ate, ate|the , the|cat, cat|.

Training Data

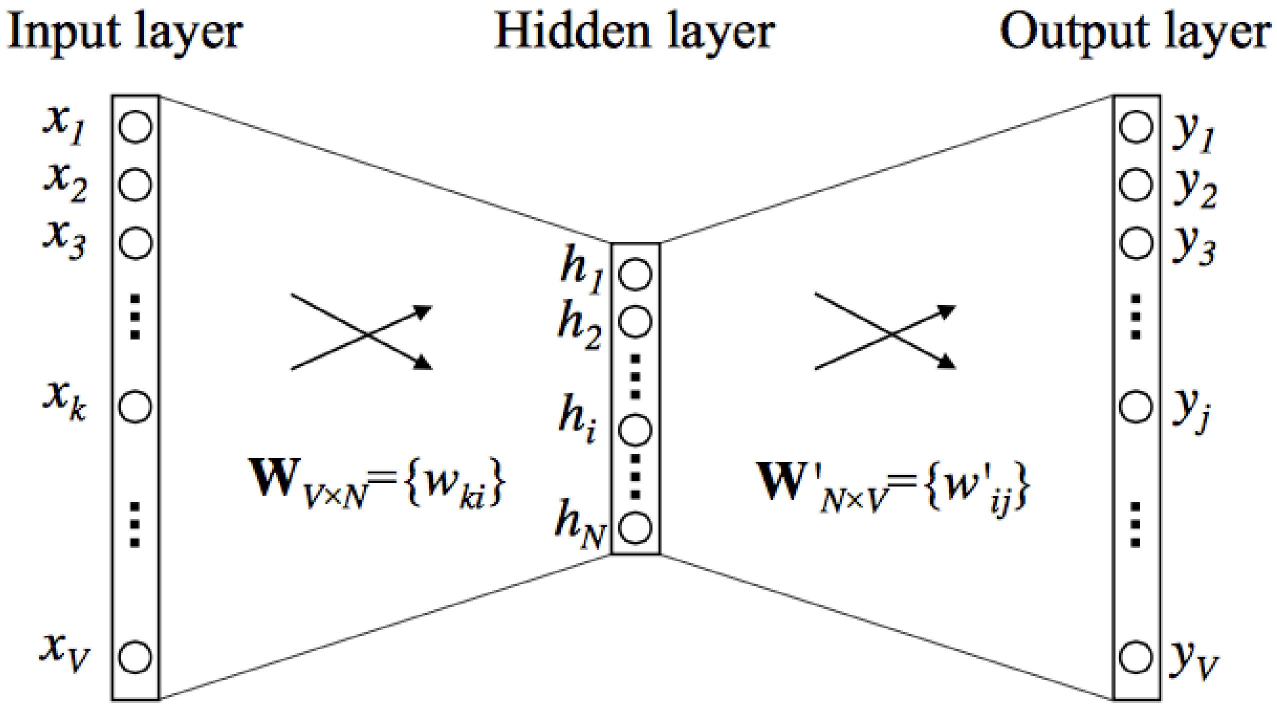
1. eat|apple
2. eat|orange
3. eat|rice
4. drink|juice
5. drink|milk
6. drink|water
7. orange|juice
8. apple|juice
9. rice|milk
10. milk|drink
11. water|drink
12. juice|drink

<u>Concept :</u>

1. Milk and Juice are drinks
2. Apples, Oranges and Rice can be eaten
3. Apples and Orange are also juices
4. Rice milk is a actually a type of milk!

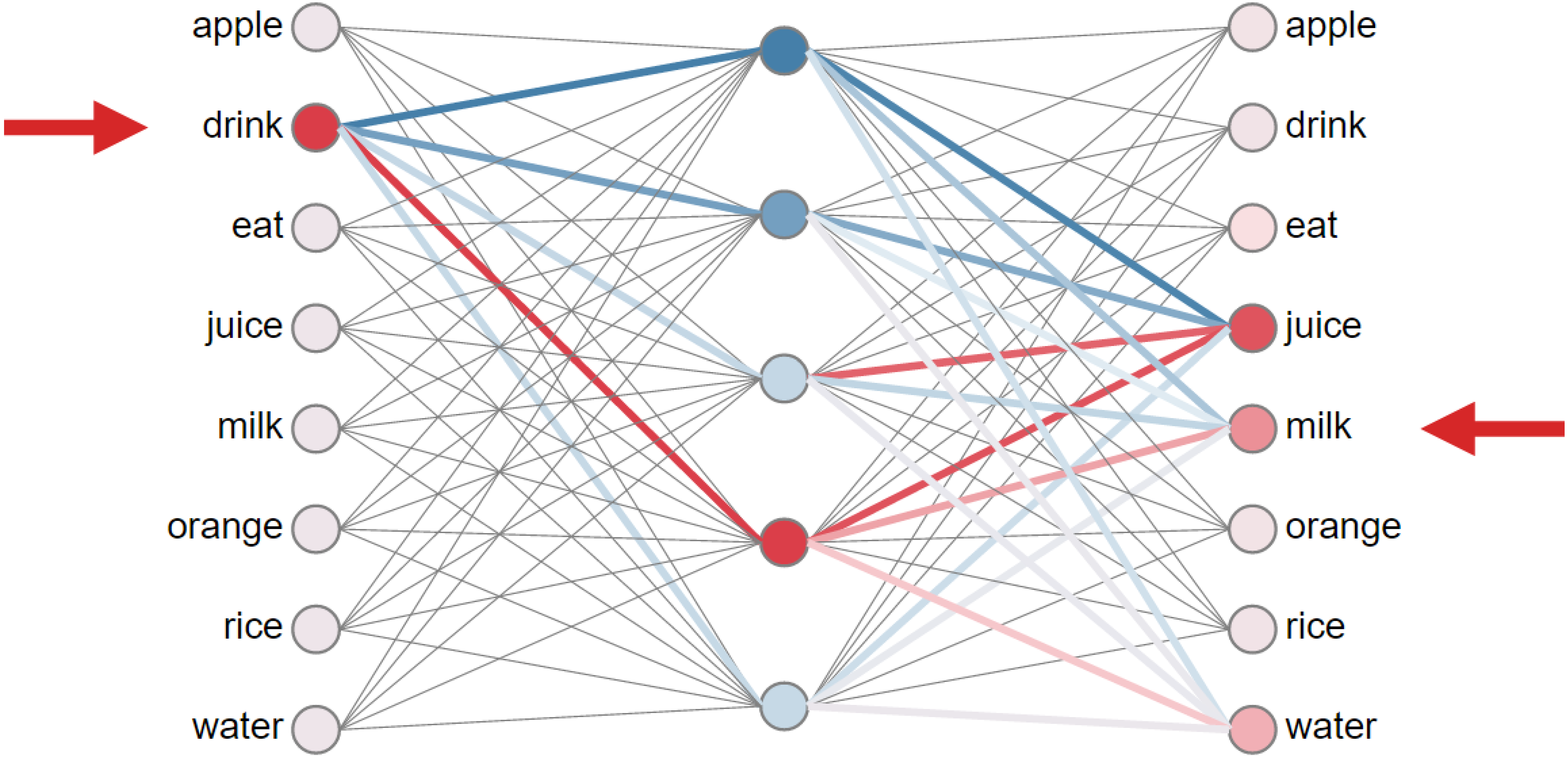
Intuitive Idea

- Use the middle layer of a DNN.



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$


apple
drink
eat
juice
milk
orange
rice
water
apple
drink
eat
juice
milk
orange
rice
water

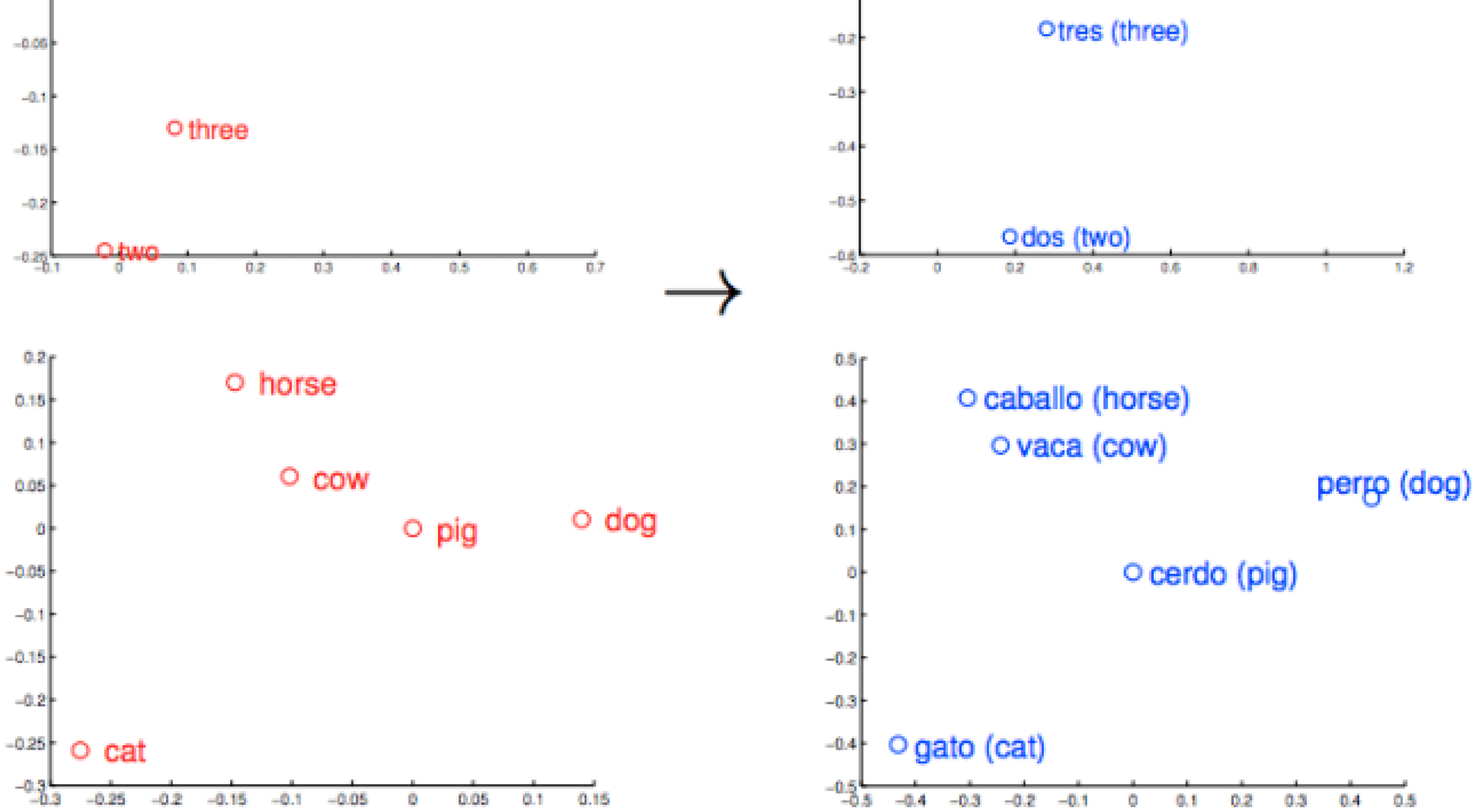
Some other buzzwords

- Known as neural embedding
- Often optimized using two methods
  1. Hierarchical Softmax
  2. Negative Sampling

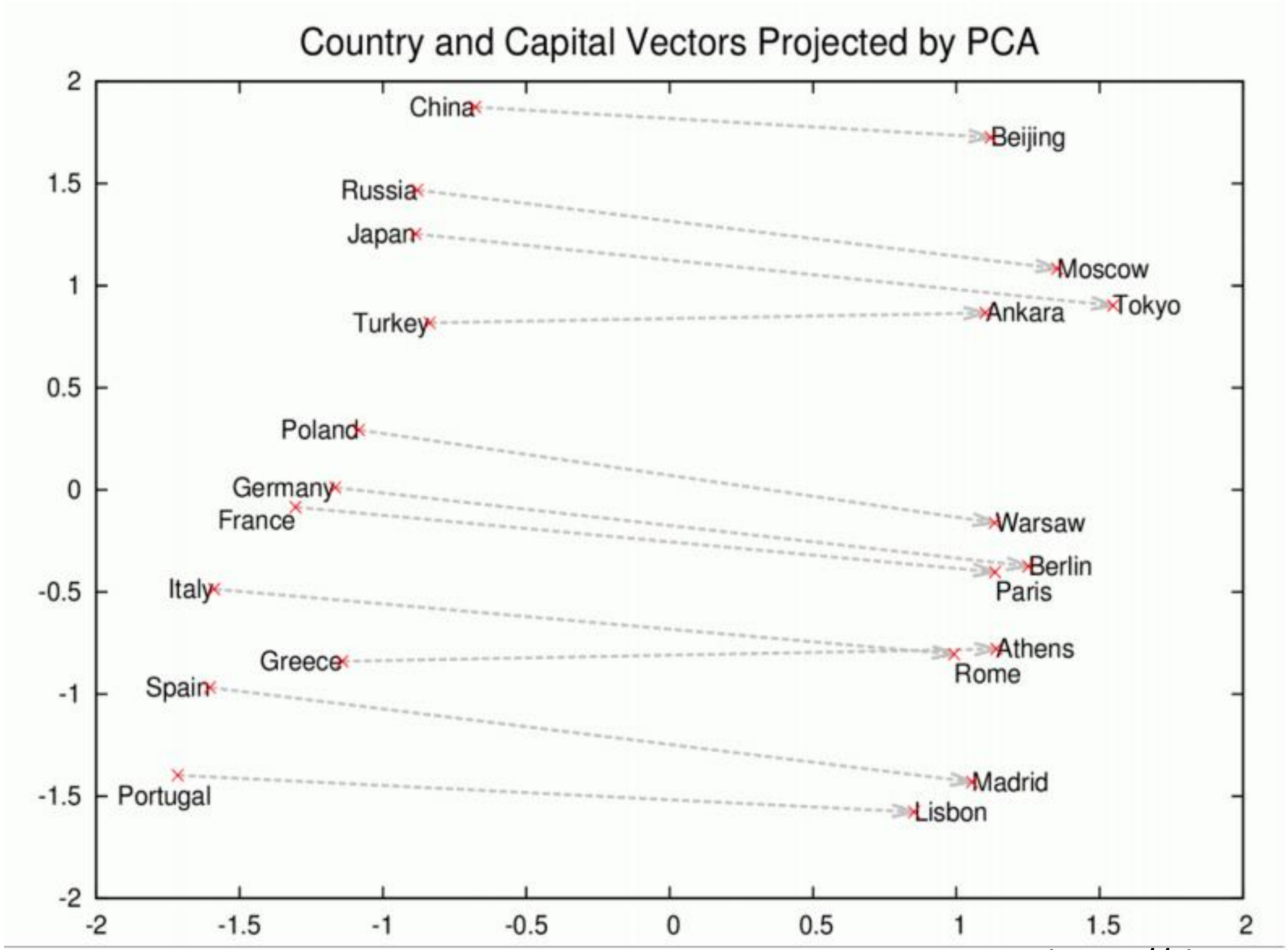
$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I} \right)$$

$$\log \sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^{\top} v_{w_I}) \right]$$

- CBOW(continuous bag-of-words) and Skip-gram based training
- Down Sampling of Frequent words
- Phrasal and paragraph vectors



https://skymind.ai/wiki/word2vec



https://skymind.ai/wiki/word2vec

Word2Vec Results

(Training with the Google news vocab)

- king:queen::man:[woman, Attempted abduction, teenager, girl] //Weird?
- China:Taiwan::Russia:[Ukraine, Moscow, Moldova, Armenia]
  //Two large countries and their small, estranged neighbors
- house:roof::castle:[dome, bell_tower, spire, crenellations, turrets]
- knee:leg::elbow:[forearm, arm, ulna_bone]

https://skymind.ai/wiki/word2vec

Word2Vec Results (Contd.)

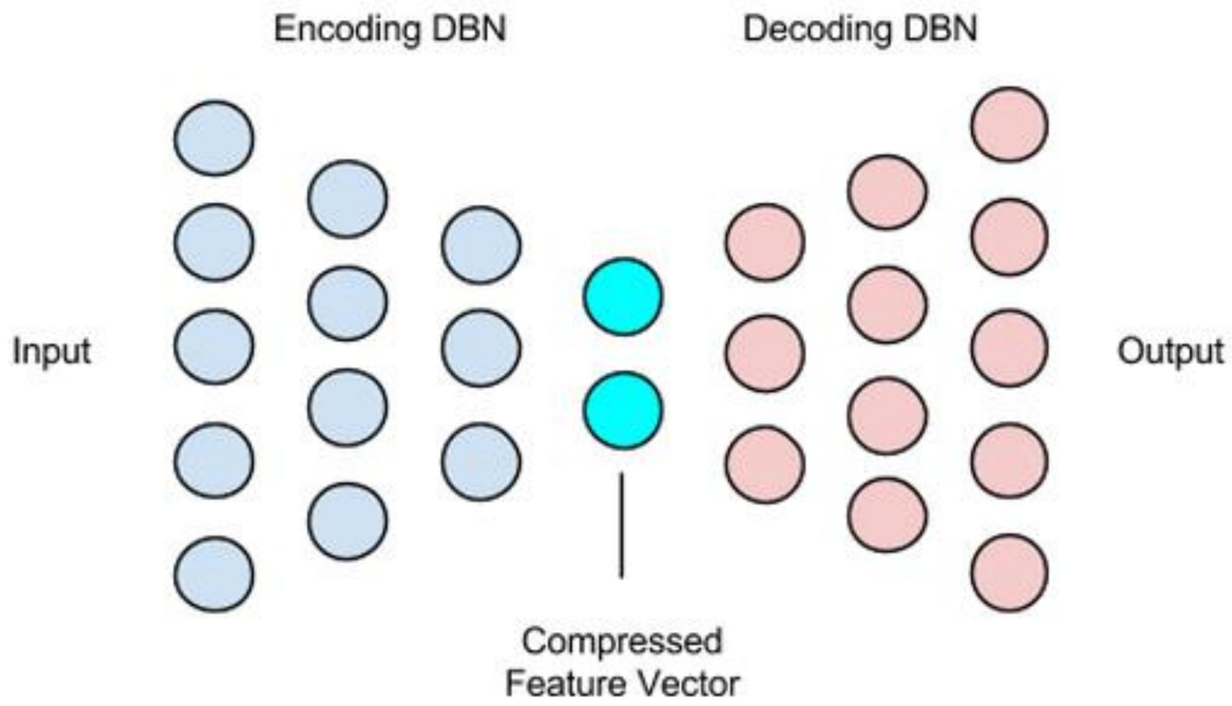
- Donald Trump:Republican::Barack Obama:[Democratic, GOP, Democrats, McCain]
  //It's interesting to note that, just as Obama and McCain were rivals,
  //so too, Word2vec thinks Trump has a rivalry with the idea Republican.

- monkey:human::dinosaur:[fossil, fossilized, Ice_Age_mammals, fossilization]
  //Humans are fossilized monkeys? Humans are what's left
  //over from monkeys? Humans are the species that beat monkeys
  //just as Ice Age mammals beat dinosaurs? Plausible.

https://skymind.ai/wiki/word2vec

Deep Autoencoders

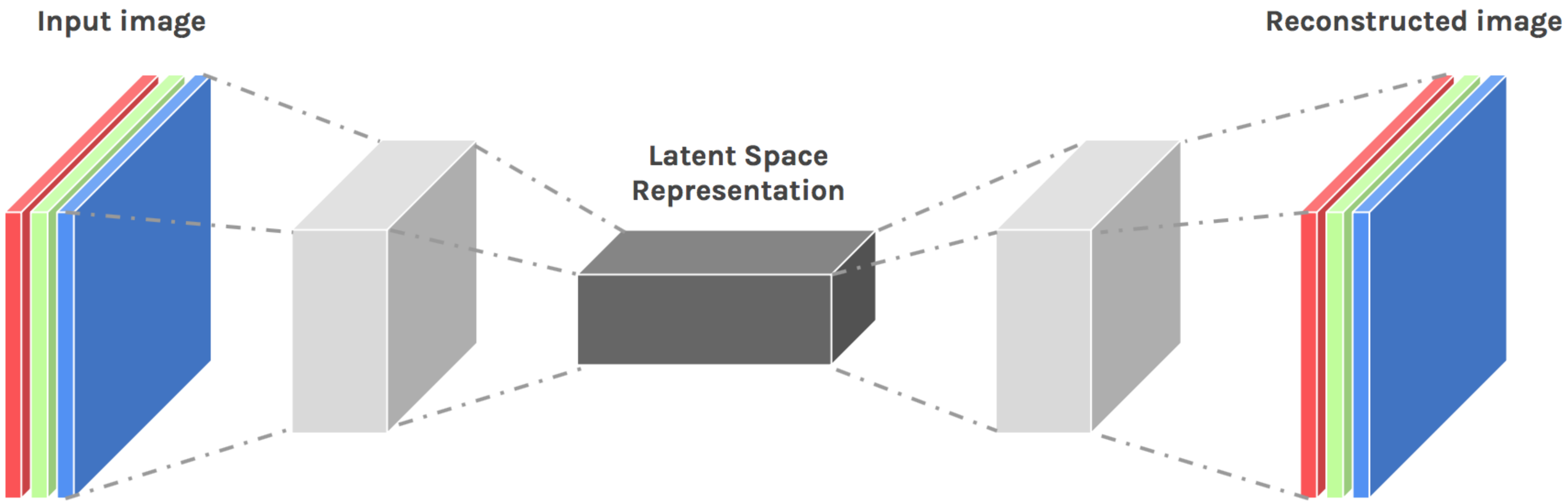
- Two (usually symmetrical) DNNs that represent encoding and decoding



Example

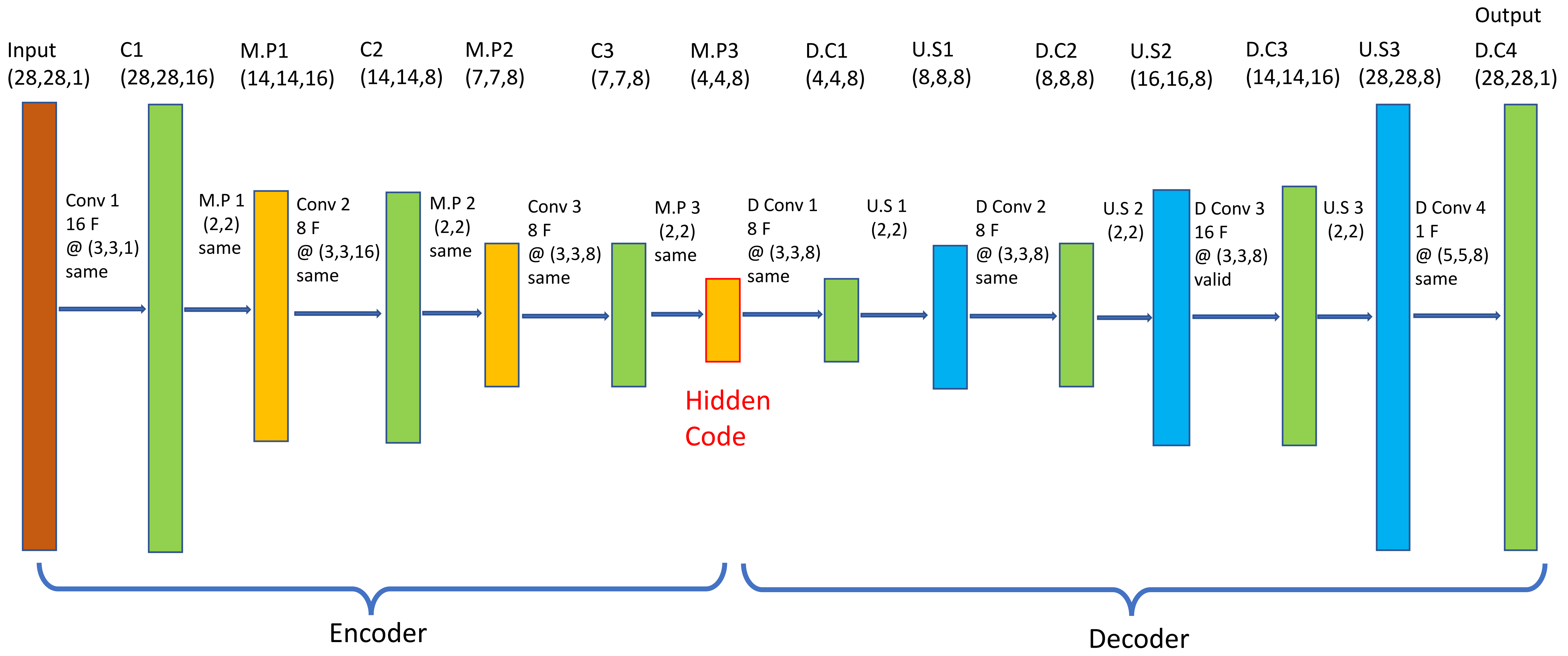
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html

Convolutional AE



* Input values are normalized
* All of the conv layers activation functions are relu except for the last conv which is sigm

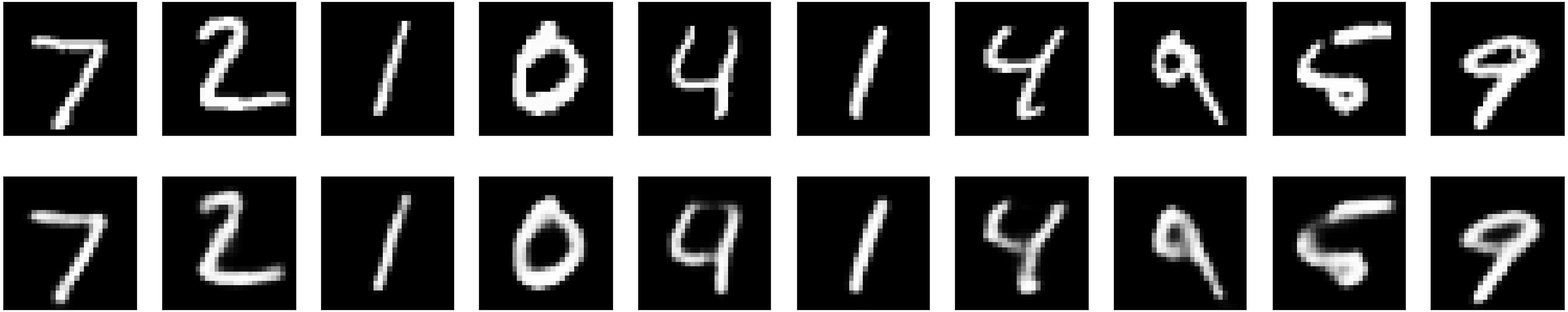
Convolutional AE



Convolutional AE – Keras example

Convolutional AE – Keras example results

- 50 epochs.
- 88% accuracy on validation set.



Regularization

Motivation:

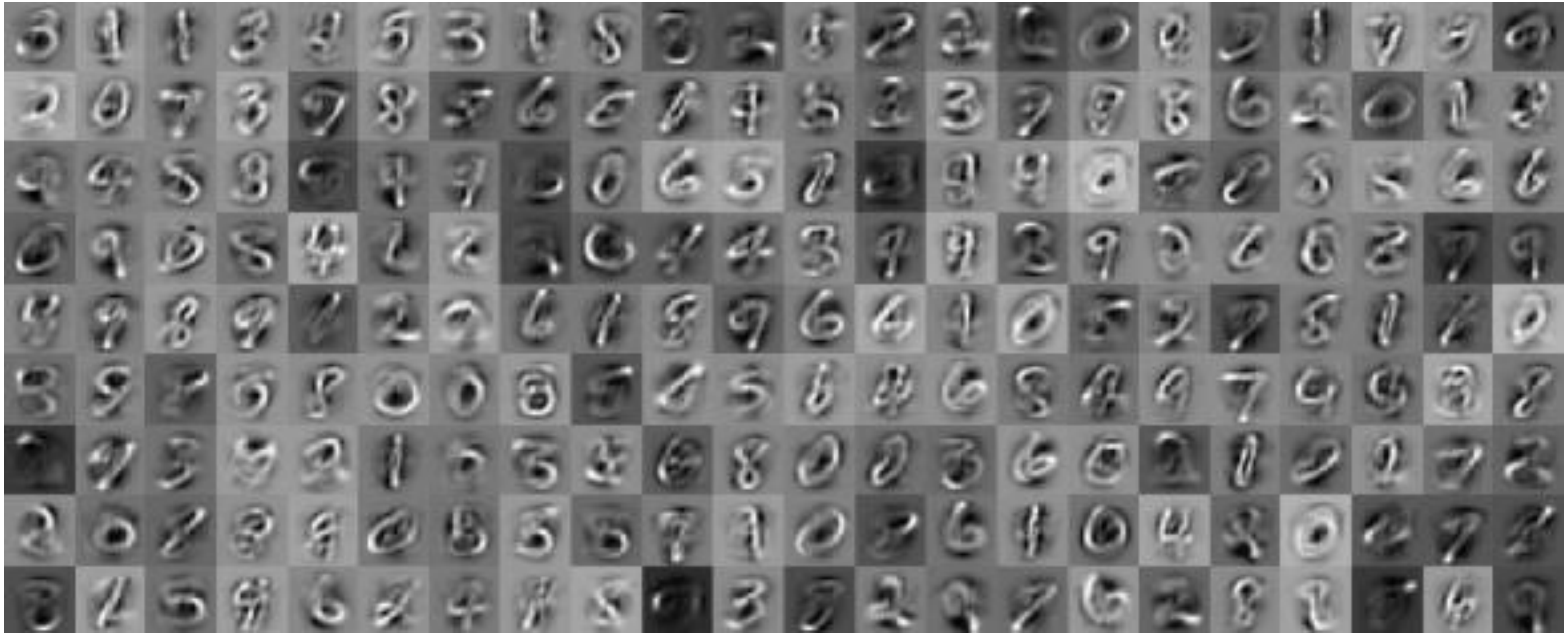
- We would like to learn meaningful features **without** altering the code's dimensions (Overcomplete or Undercomplete).

The solution: imposing other constraints on the network.

Sparsely Regulated Autoencoders

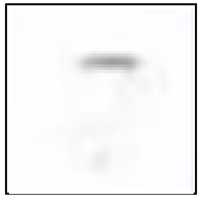
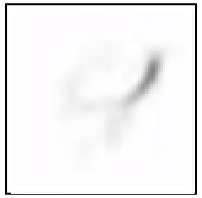
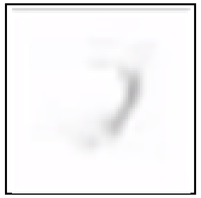
A bad example:

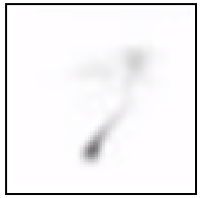
Activation Maps



Sparsely Regulated Autoencoders

- We want our learned features to be as **sparse** as possible.
- With sparse features we can generalize better.

 = 1*  + 1*  + 1*  + 1*  + 1* 

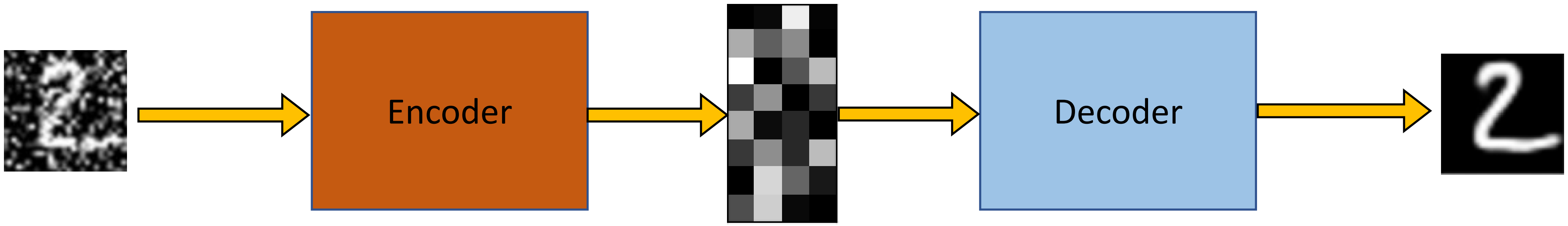
+ 1*  + 1*  + 0.8*  + 0.8* 

Denoising Autoencoders

<u>Intuition:</u>

- We still aim to encode the input and to NOT mimic the identity function.
- We try to undo the effect of *corruption* process stochastically applied to the input.

A more robust model



Noisy Input

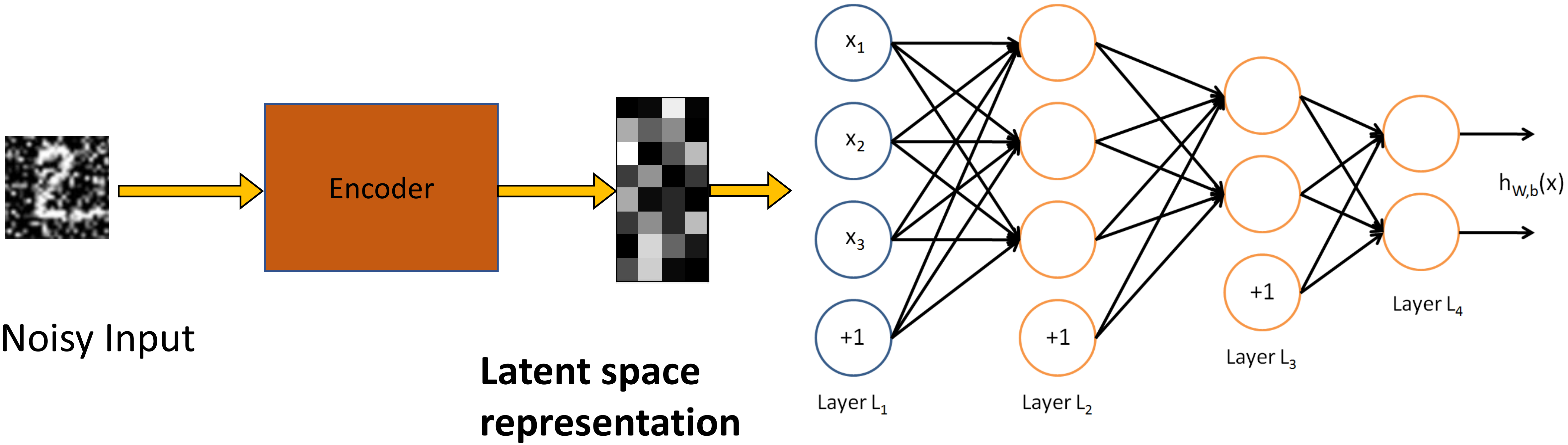
Latent space representation

Denoised Input

Denoising Autoencoders

Use Case:

- Extract robust representation for a NN classifier.



Denoising Autoencoders

Instead of trying to mimic the identity function by minimizing:

$$L\left(x, g\left(f(x)\right)\right)$$

where L is some loss function

A **DAE** instead minimizes:

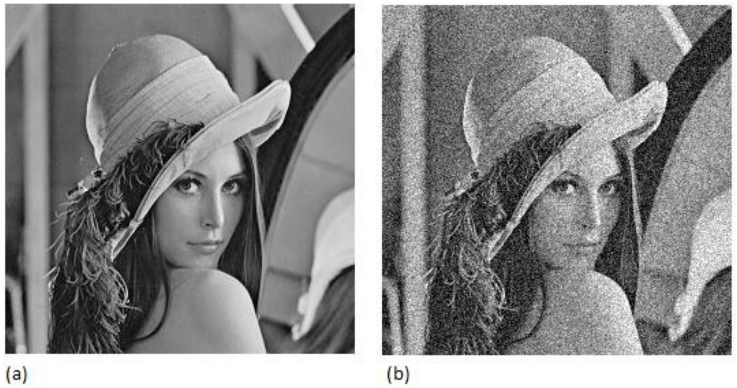
$$L\left(x, g\left(f(\tilde{x})\right)\right)$$

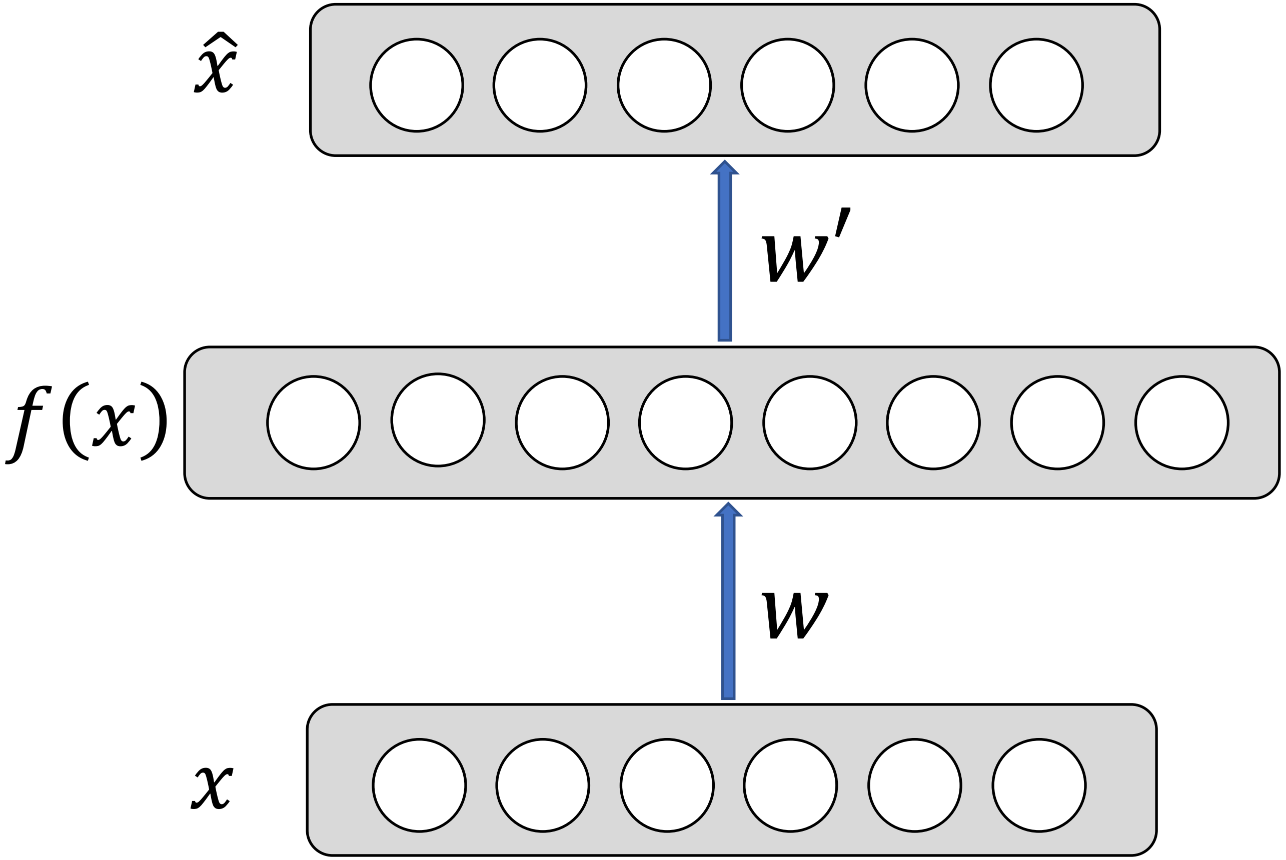
where $\tilde{x}$ is a copy of x that has been corrupted by some form of noise.

Denoising Autoencoders

Idea: A robust representation against noise:

- Random assignment of subset of inputs to 0, with probability v.
- Gaussian additive noise.





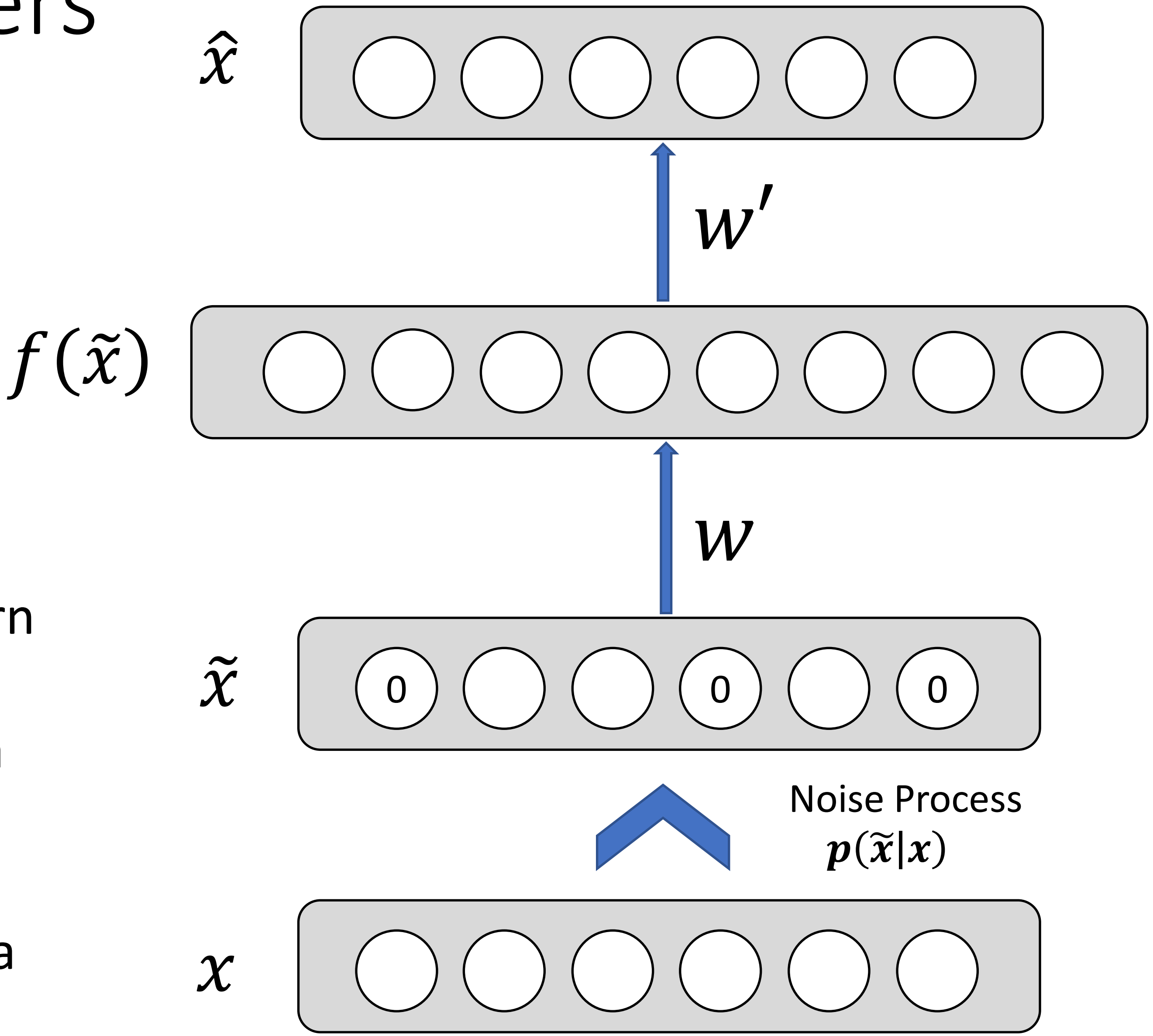
Denoising Autoencoders

- Reconstruction $\hat{x}$ computed from the corrupted input $\tilde{x}$.
- Loss function compares $\hat{x}$ reconstruction with the noiseless x.

❖ The autoencoder cannot fully trust each feature of x independently so it must learn the correlations of x's features.

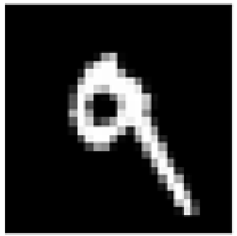
❖ Based on those relations we can predict a more 'not prune to changes' model.

➢ We are forcing the hidden layer to learn a generalized structure of the data.



Denoising Autoencoders - process

Taken some input x



Apply Noise

$\tilde{x}$

Denoising Autoencoders - process

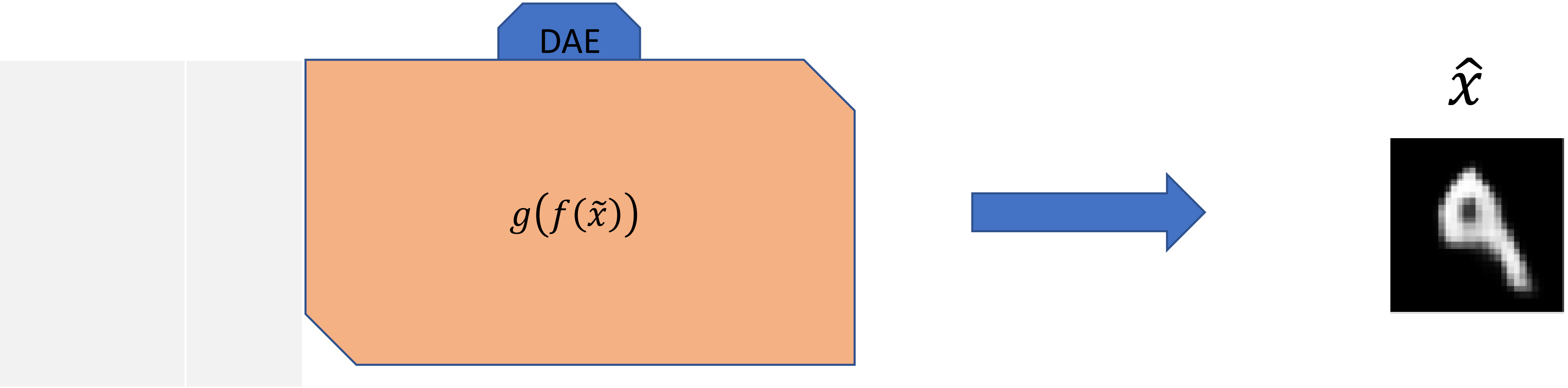
$\tilde{x}$

Encode And Decode

DAE

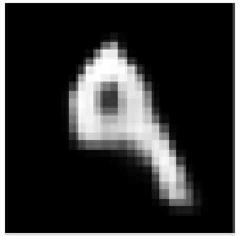
$g\left(f(\tilde{x})\right)$

Denoising Autoencoders - process



Denoising Autoencoders - process

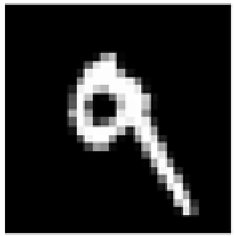
$\hat{x}$



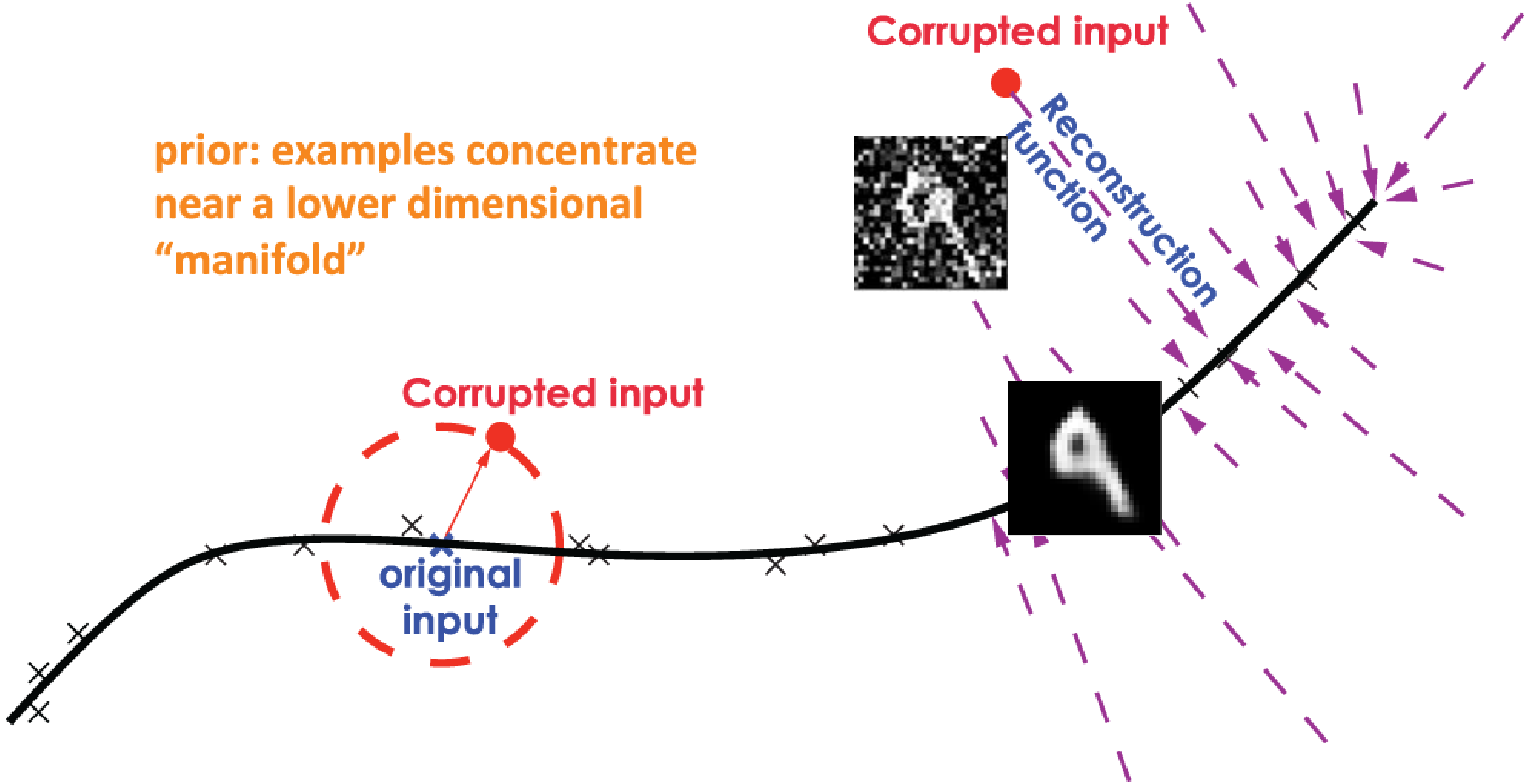
Compare

$=$

x

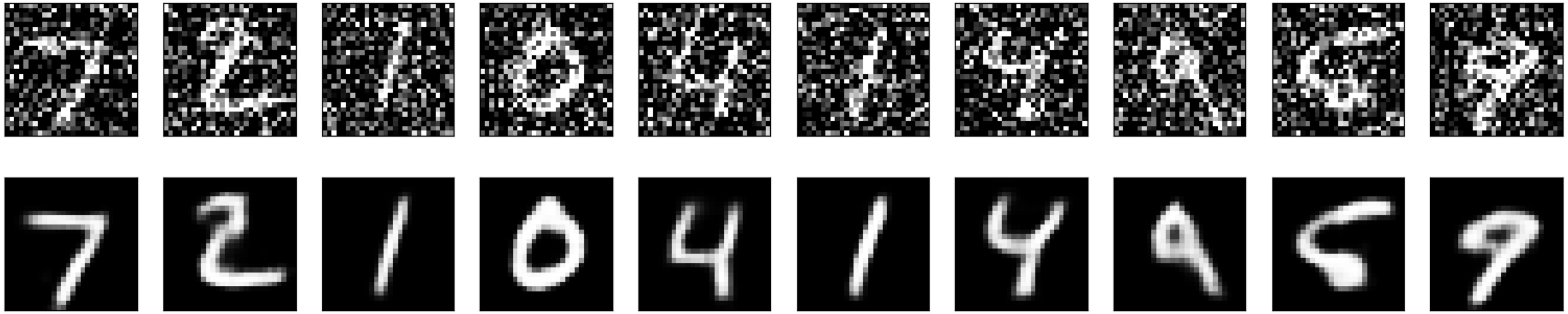


Denoising autoencoders



Denoising convolutional AE – keras

- 50 epochs.
- Noise factor 0.5
- 92% accuracy on validation set.



References

1. https://arxiv.org/pdf/1206.5538.pdf
2. http://www.deeplearningbook.org/contents/autoencoders.html
3. http://deeplearning.net/tutorial/dA.html
4. http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
5. http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders
6. http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf
7. https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663