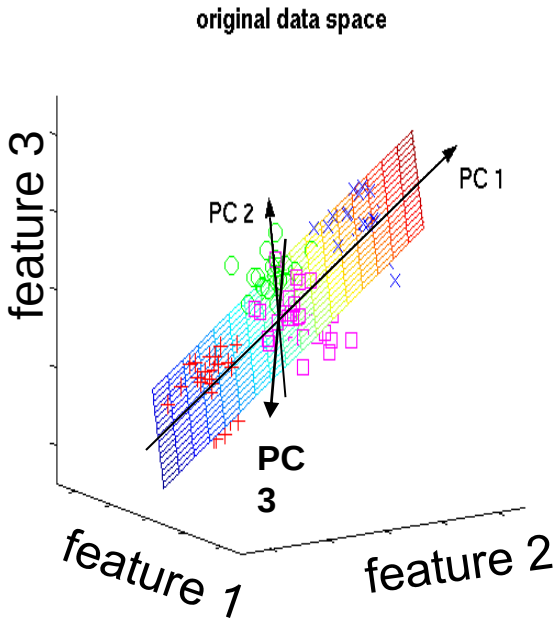
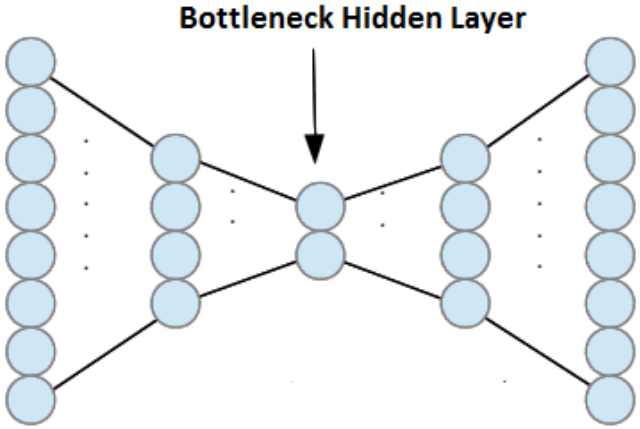
Neural Networks for unsupervised learning
From Principal Components Analysis
to Autoencoders to semantic hashing





Beate Sick

Many slides are taken form Hinton's great lecture on NN:
https://www.coursera.org/course/neuralnets

What is semantic hashing?

Semantics

From Wikipedia, the free encyclopedia

Semantics (from Ancient Greek: σημαντικός *sēmantikós*, "significant")[1][2] is the study of meaning. It focuses on the relation between *signifiers*, like words, phrases, signs, and symbols, and what they stand for; their denotation. Linguistic

Semantic hashing in my understanding:

Procedure that allows to find in an efficient way documents or images that are similar in their “meaning”

Dimensionality Reduction

Zurich University of Applied Sciences

zh aw

Input data may have thousands or millions of dimensions!

- images have many thousands of pixels
- text data has many letters or words
- sensors collect thousands of signals

Are all those dimensions (letters, pixels) necessary?

Goal: Represent data with fewer dimensions!

- Work with fewer features: get rid of p>>n situation
- Better visualization: our imagination does not go beyond 3D
- Discover “intrinsic dimensionality” of data: learn low-dim information representation
- Get a compact hash tag for each high dimensional data point
- Get a measure for abnormality by quantifying the reconstruction error

Pioneer work

Zurich University of Applied Sciences

zh aw

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

Artificial Neural Networks – fully connected or CNN

Fully connected:

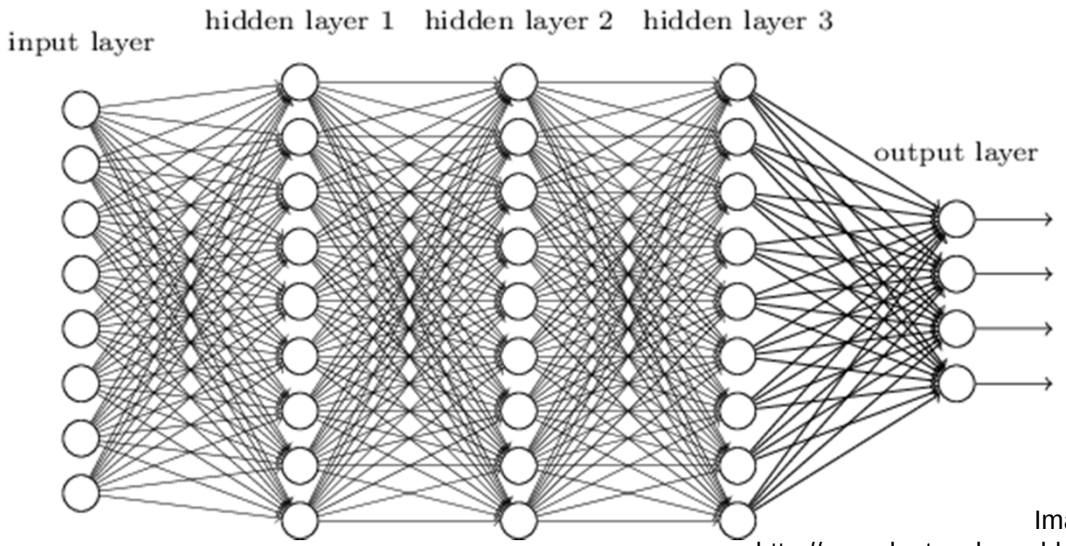


Image:
http://neuralnetworksanddeeplearning.com/chap6.html

CNN:

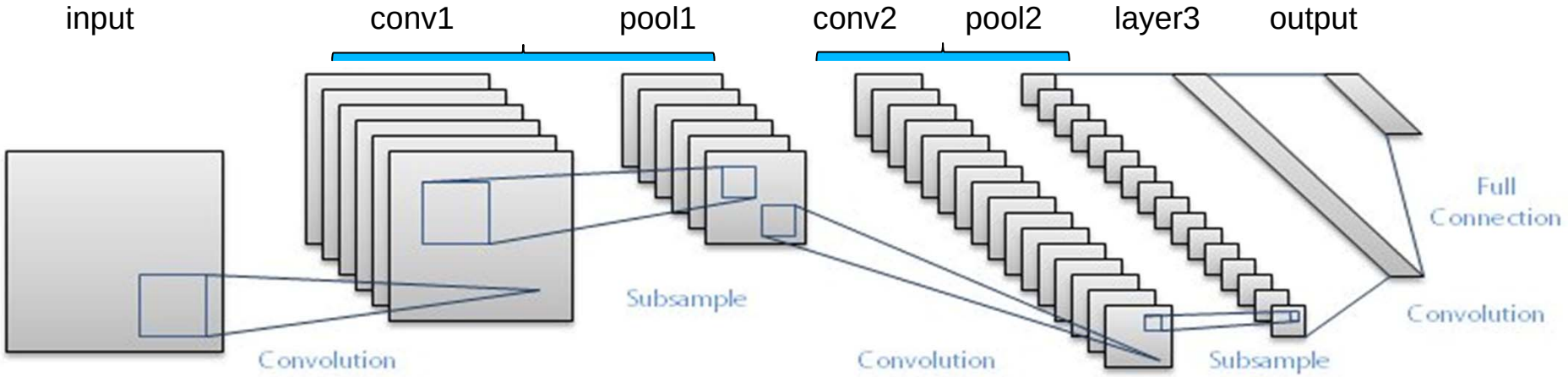


Image: Master Thesis Christopher Mitchell http://www.cemetech.net/projects/ee/scouter/thesis.pdf

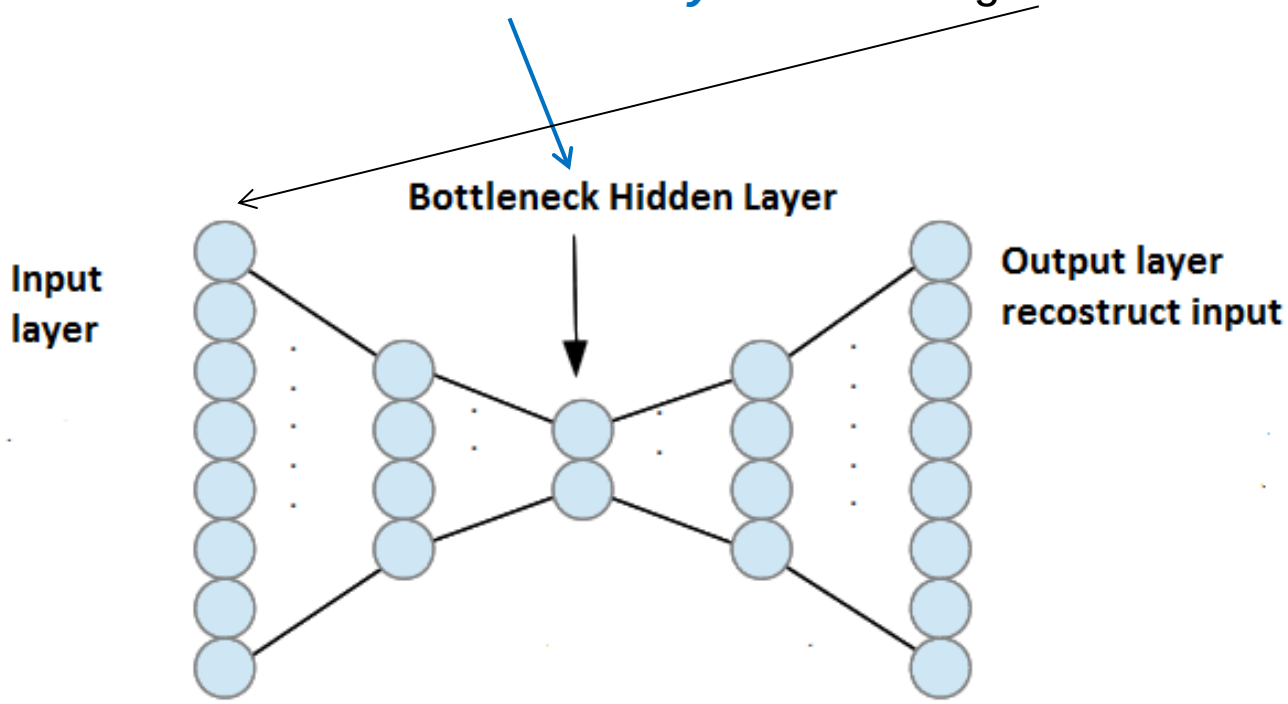
Dimension reduction via Autoencoder (AE)

Train AE using stochastic gradient backpropagation

Zurich University of Applied Sciences

zh aw

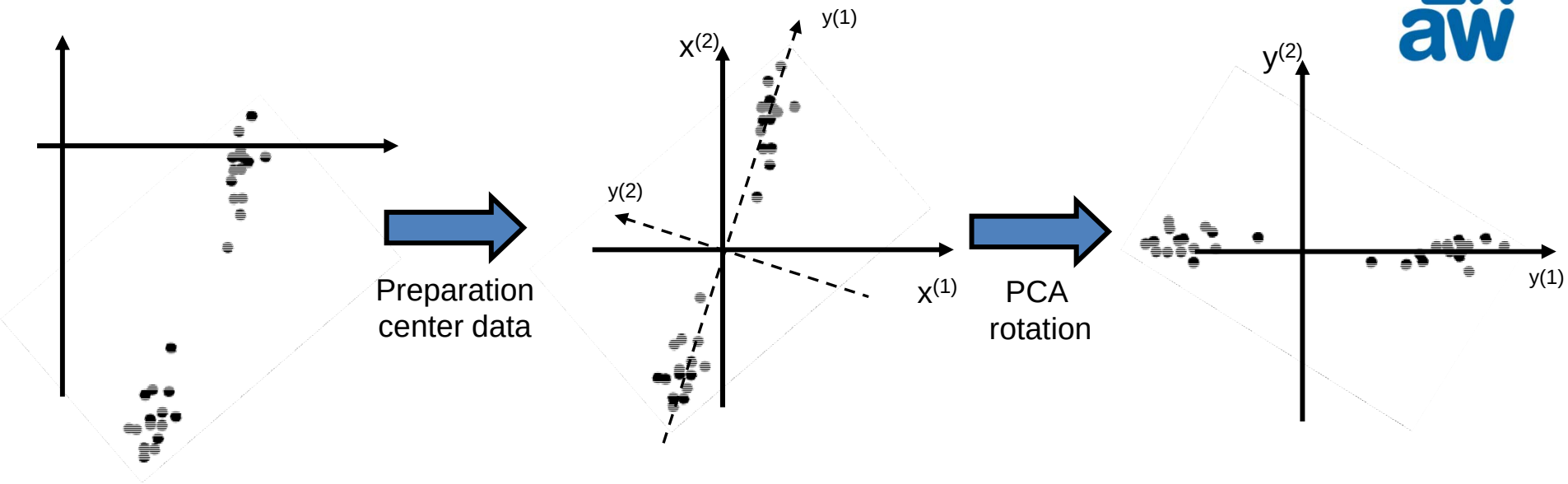
Rather than picking a subset of the features, we can ***construct k<n new features y*** from existing n features.



The AE network gets the optimization task to reconstruct the input which requires that essential information is represented in only few dimensions corresponding to the number neurons in the bottleneck.

Image from: http://codingplayground.blogspot.ch/2015/10/what-are-autoencoders-and-stacked.html

Dimension reduction by PCA



PCA is a rotation of the coordinate system -> no information is lost

The first principal component (PC_1 or Y_1) explains most of the variance, the second the second most,…. All PC components explains the whole variance.

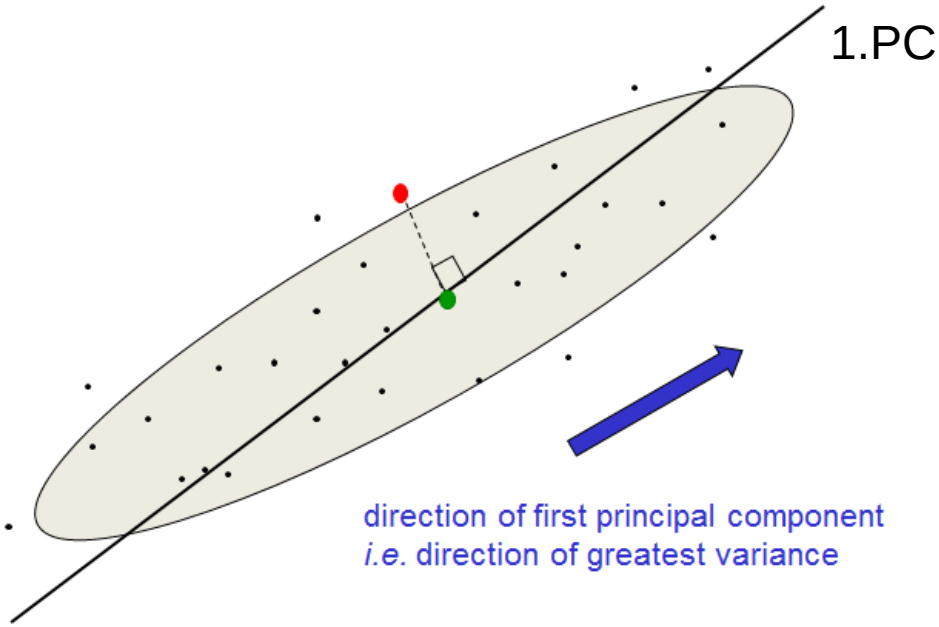
Dimension reduction is done by dropping higher PCs.

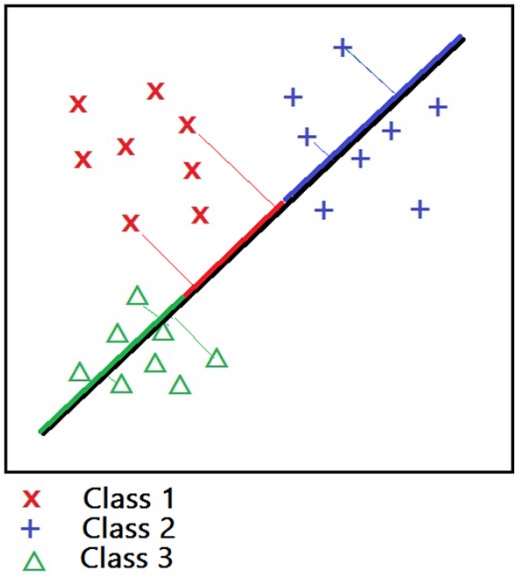
Each principle component can be represented as linear combination of the original features and the coefficients a_{ij} in the linear combination are called “loadings”.

linear combination:

$$Y_k = a_{1k} X_1 + a_{2k} X_2$$

A picture of PCA starting from 2D -> 1D representation Reconstruction error





The red point is represented in 1D by the green point – the position orthogonal to the 1.PC (low-dim hyperplane) cannot be reconstructed, instead the mean of (over all data) of this orthogonal direction is taken.

The reconstruction of the red point has an reconstruction-error equal to the squared distance between red and green points.

Construction of PCs in PCA

PCA Rotation can be achieved by multiplying **X** with an orthogonal rotation matrix **A**

$$\underset{=}{Y}_{(nxp)} = \underset{=}{X}_{(nxp)} \cdot \underset{=}{A}_{(pxp)} \Leftrightarrow \underset{=}{X}_{(nxp)} = \underset{=}{Y}_{(nxp)} \cdot \underset{=}{A}^{T}_{(pxp)}$$

The columns $\underline{a}_k$ of **A** are given by the normalized Eigenvector $\underline{a}$ of the the Covariance-matrix **Sx**

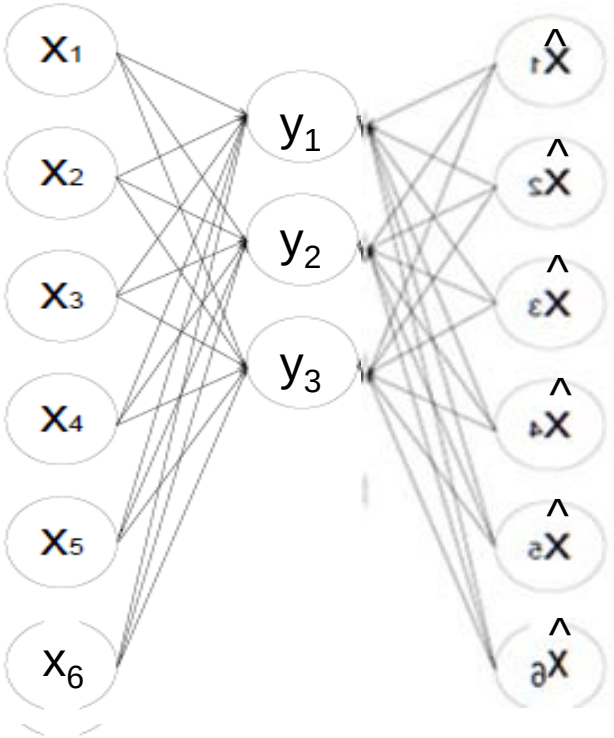
Reconstruct **X** with only k<p PCs:

$$\hat{\underset{=}{X}}_{(nxp)} = \left(\underset{=}{Y}_{(nxk)}, \underset{=}{0}_{(nxp-k)}\right) \cdot \underset{=}{A}^{T}_{(pxp)}$$

PCA minimizes reconstruction error
over all available m data points

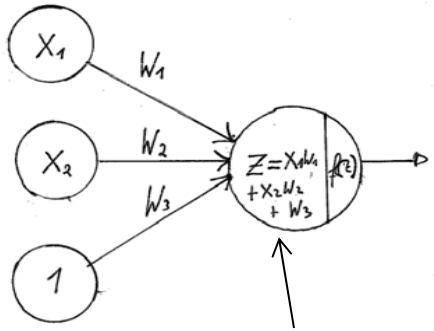
$$\sum_{i=1}^{m} || \hat{x}^{(i)} - x^{(i)} ||^2 = \sum_{i=1}^{m} || \hat{\underset{=}{X}}_{i\bullet} - \underset{=}{X}_{i\bullet} ||^2$$

$$y_k = \sum_{j=1}^{6} a_{jk} x_j$$



The first k PCs define k-dim hyperplane to which sum of squared projection errors is minimal

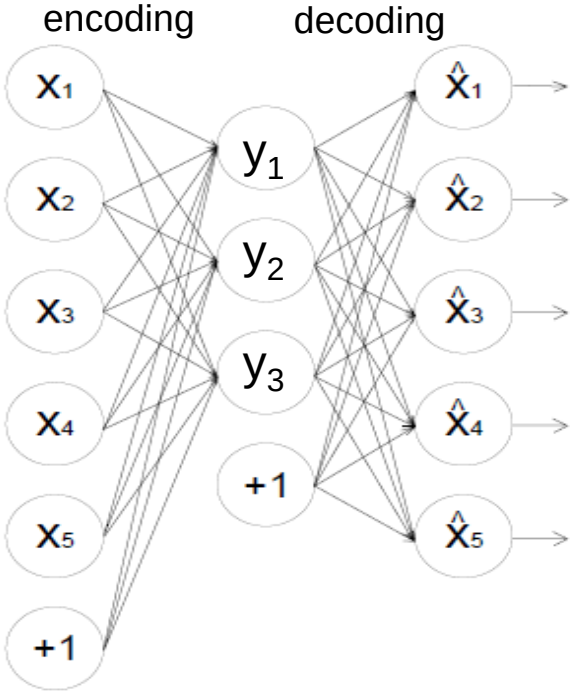
Dimensionality Reduction by an Autoencoder (AE)



$$y = f(z)$$

$$z = x_1 w_1 + x_2 w_2 + w_3$$

$$\underset{(3)}{\underline{y}} = \underset{(3)}{\underline{b}} + \underset{(3\text{x}5)}{\underline{\underline{W}}} \cdot \underset{(5)}{\underline{x}} \qquad \underset{(5)}{\underline{\hat{x}}} = \underset{(5)}{\underline{b}}' + \underset{(5\text{x}3)}{\underline{\underline{W}}^T} \cdot \underset{(3)}{\underline{y}}$$



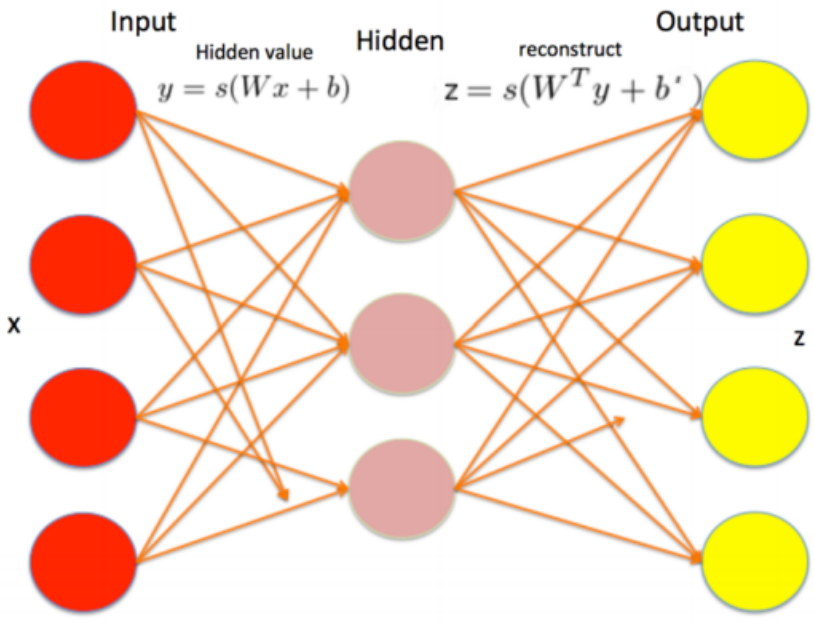
reconstruction-error

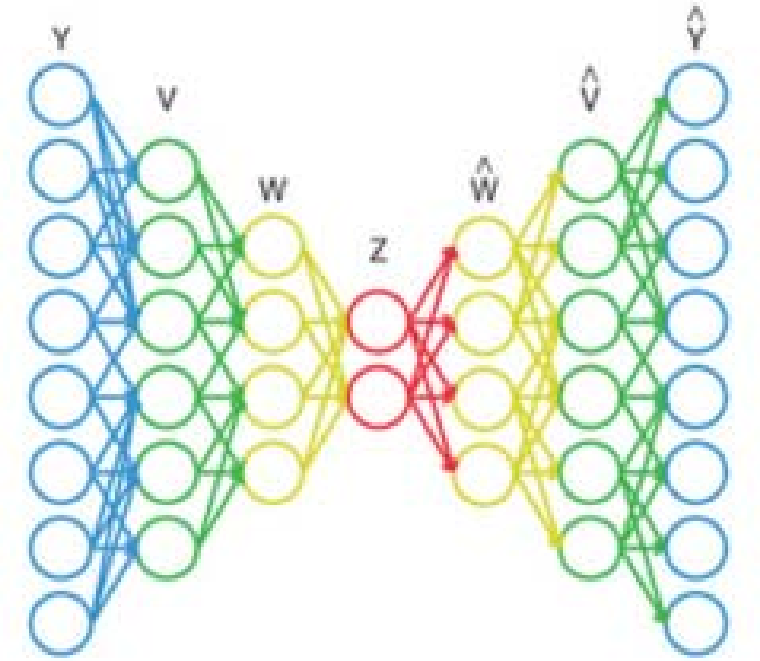
$$= \sum_{i=1}^{m} \| \underset{(\text{n})}{\underline{\hat{x}}}^{(i)} - \underset{(n)}{\underline{x}}^{(i)} \|^2$$

$$= \sum_{i=1}^{m} \| \underset{(\text{n})}{\underline{b}}' + \underset{(\text{n}\,xk)}{\underline{\underline{W}}^T} \left(\underset{(k)}{\underline{b}} + \underset{(kxn)}{\underline{\underline{W}}} \cdot \underset{(n)}{\underline{x}}^{(i)} \right) - \underset{(n)}{\underline{x}}^{(i)} \|^2$$

The k linear units of an AE with 1 hidden layer span the same hyperplane as 1to k-PCs

Shallow and deep Autoencoder (AE)

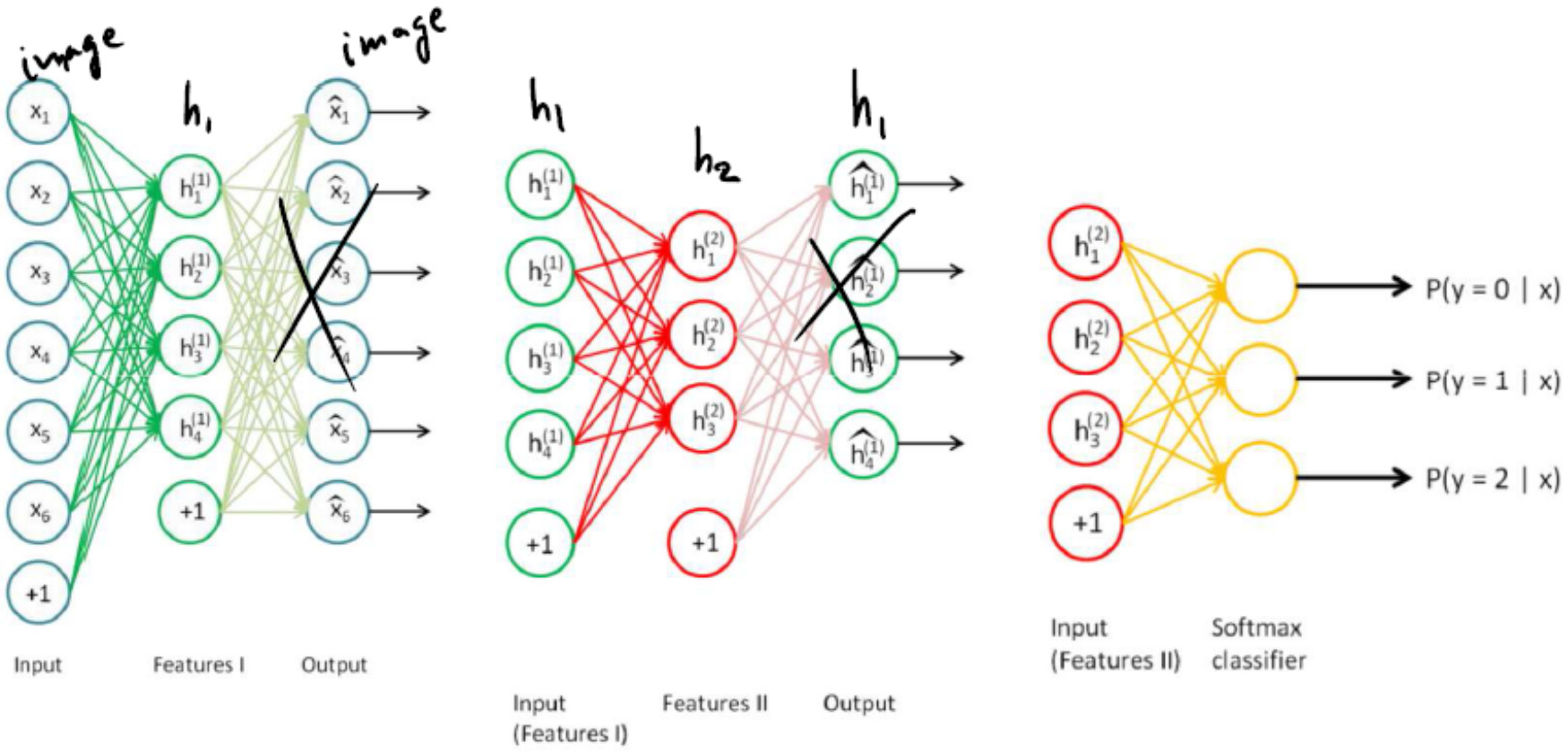




AE allow to generalize PCA by using non-linear units or more hidden layers:

- Defining curved subspaces
- Defining non-linear, hierarchical, complex features
- Defining localized, distributed features (feature maps in convolutional architectures)

Deep AE are often hard to train ->
Use greedy layer-wise training of deep AE



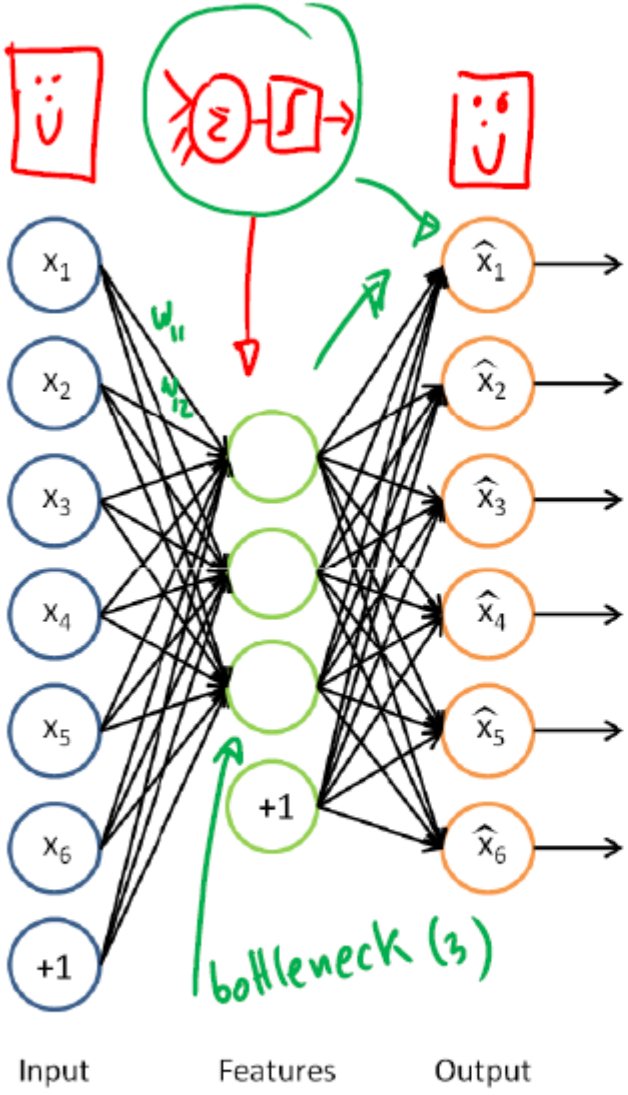
To train our neural network, we will use the cost function:

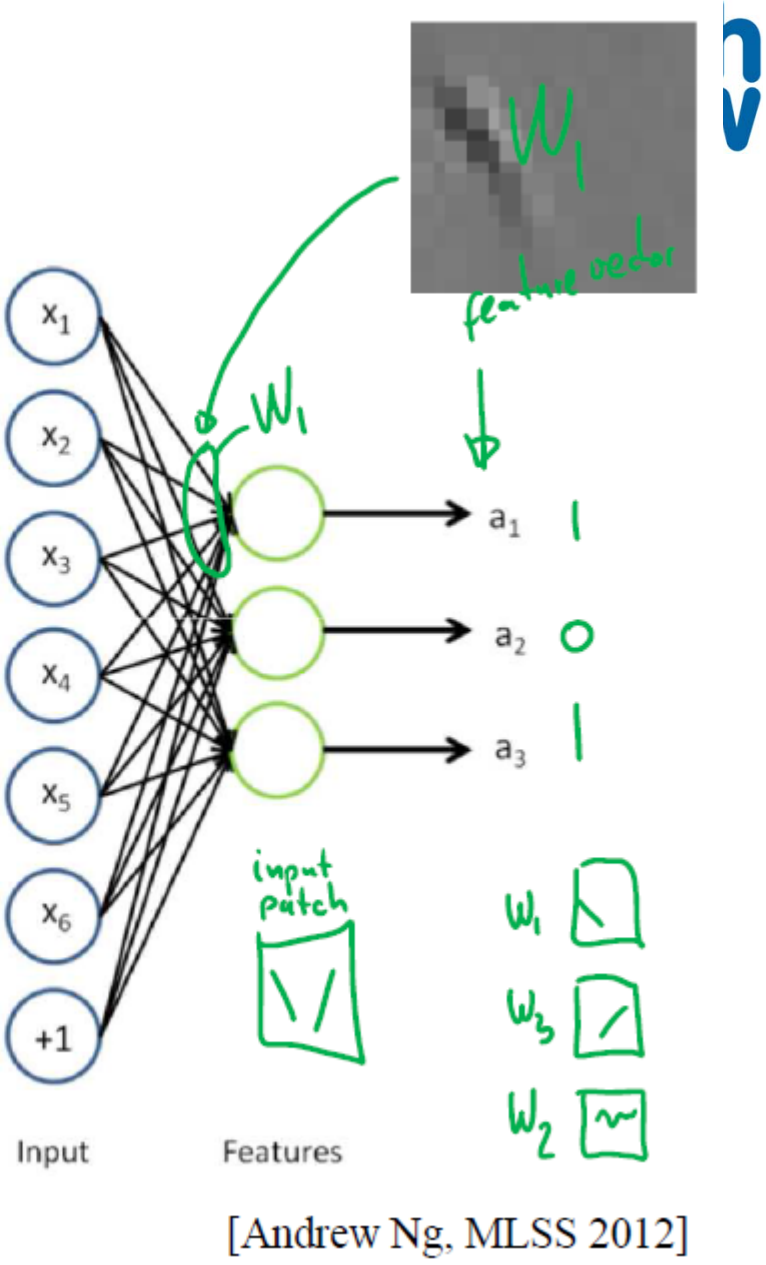
$$J(W,b;x,y) = \frac{1}{2}\left(\|h_{W,b}(x) - y\|^2\right) - \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

Weight decay term
penalty, regularization

[Andrew Ng, MLSS 2012]

AE can be used for feature construction

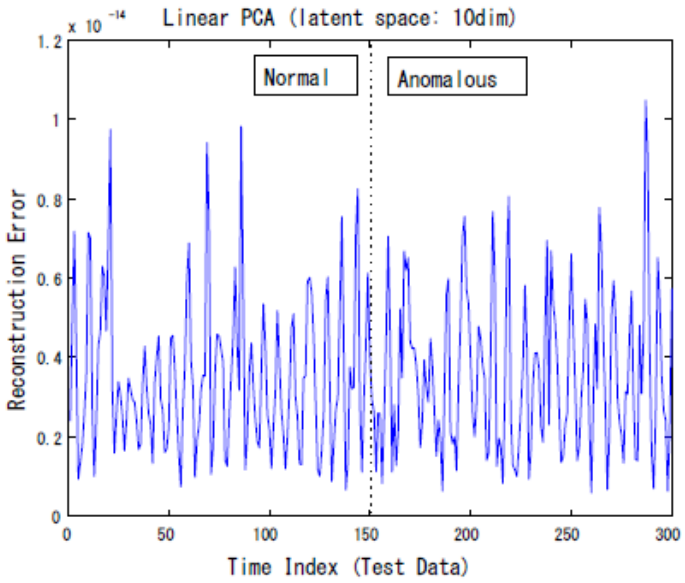


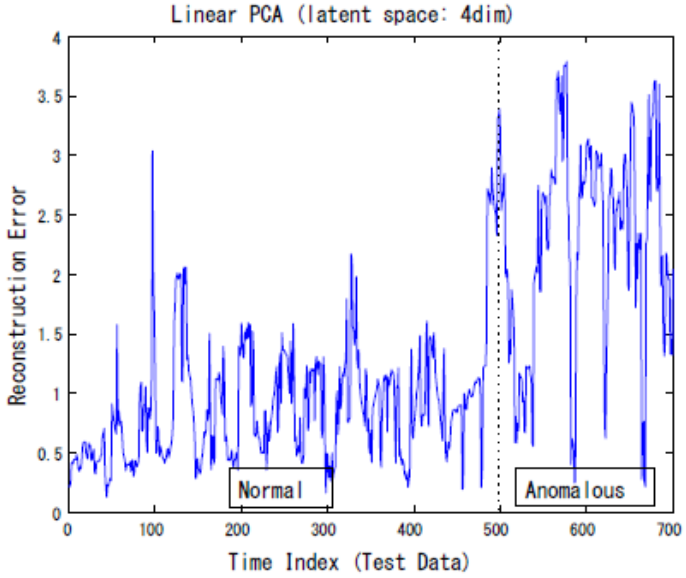


[Andrew Ng, MLSS 2012]

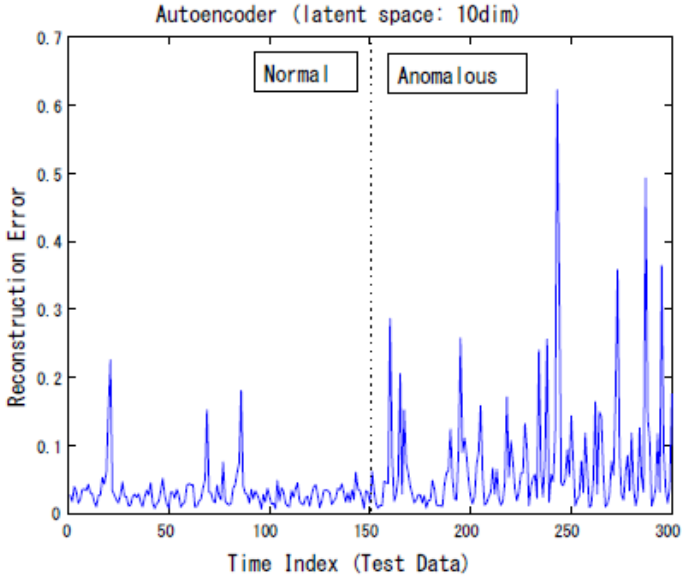
R package: autoencoder

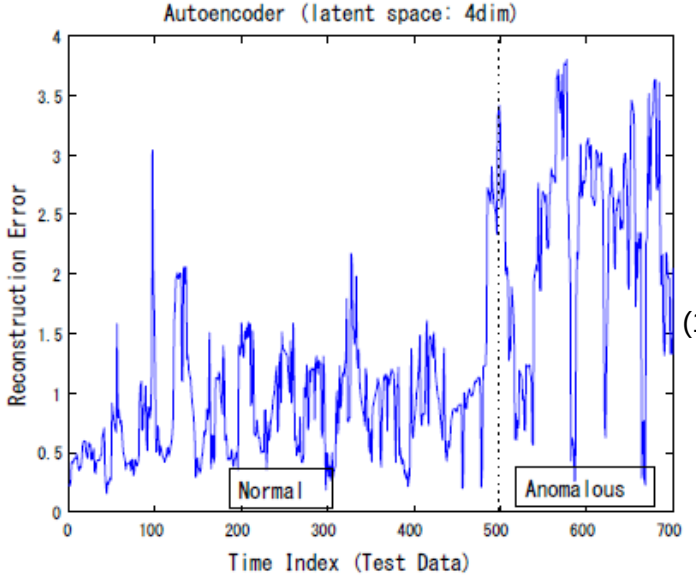
Use reconstruction error for abnormality detection compare PCA and AE based methods





PCA





AE
(1 non-linear hidden layer)

Mayu Sakurada
2015

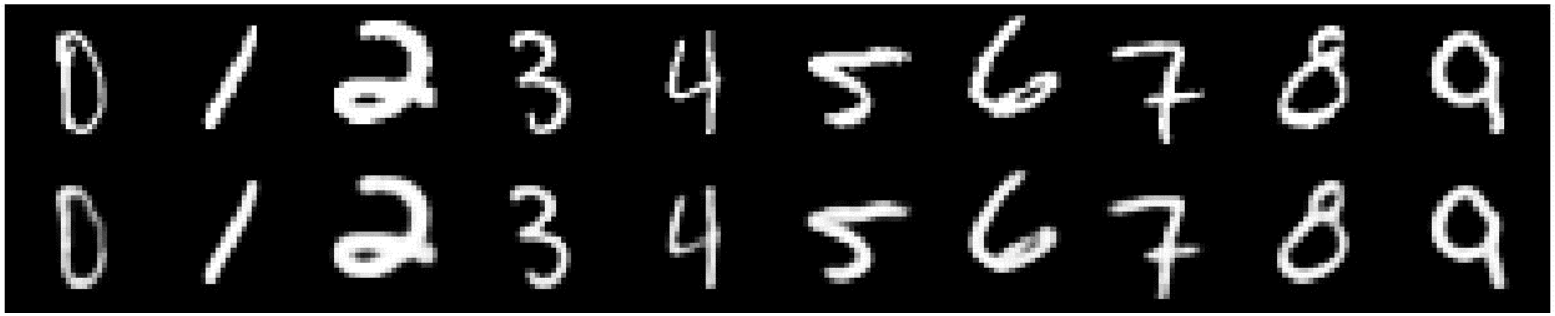
25 components of simulated Lorenz system

17 continuous sensor data from a satellite

A comparison of methods for compressing digit images to 30 real numbers

Zurich University of Applied Sciences

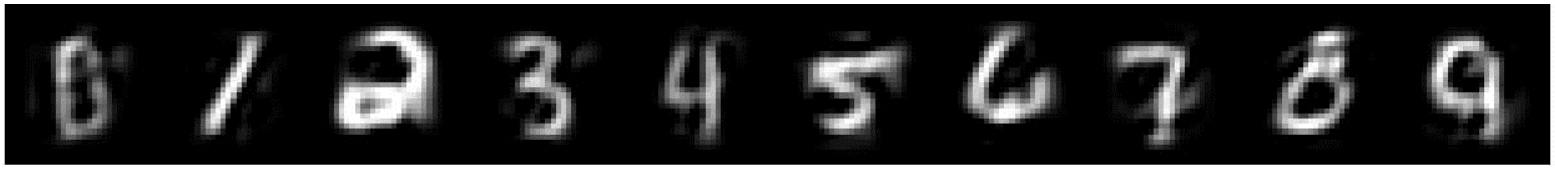
zh aw



real data

30-D deep auto

30-D PCA



Slide taken from Hinton lectures

How to find documents that are similar to a query document

Convert each document into a “bag of words”.

This is word count vector ignoring order.

Ignore stop words (like “the” or “over”) but still has a length of >2000

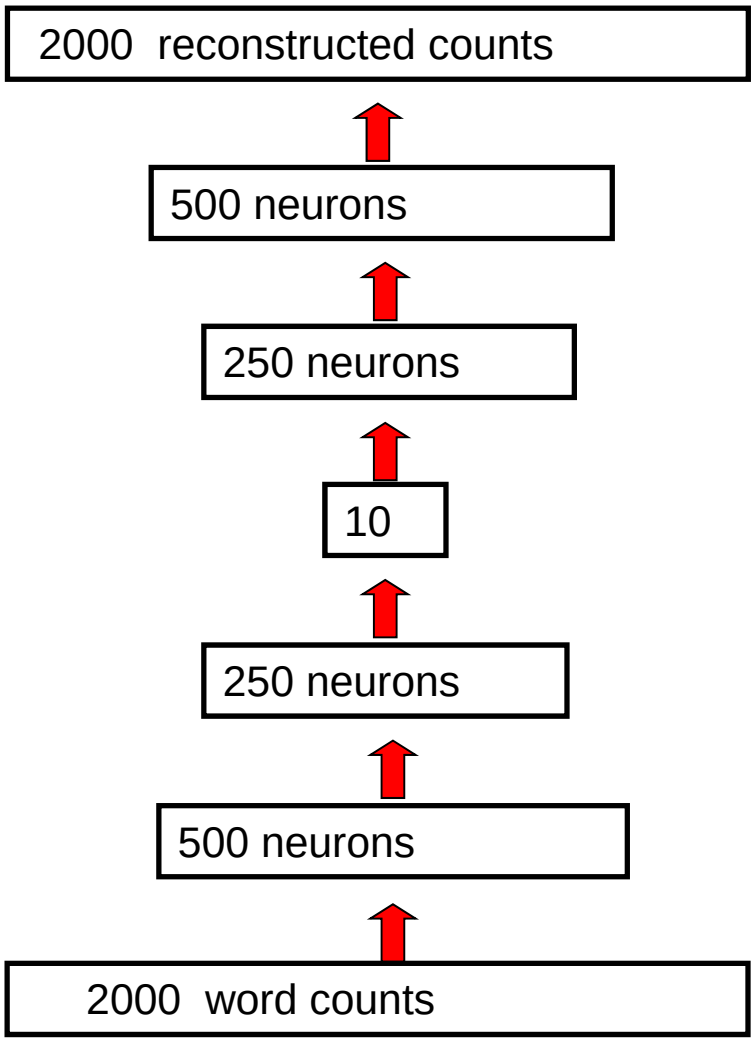
We could compare word count vectors of the query document and millions of other documents, but this is too slow.

Approach: Reduce word count vectors to a much smaller vector that still contains most of the information about the content of the document.

0	fish
0	cheese
2	vector
2	count
0	school
2	query
1	reduce
1	bag
0	pulpit
0	iraq
2	word

Slide taken from Hinton lectures

How to compress the count vector



We train the neural network to reproduce its input vector as its output

This forces it to compress as much information as possible into the 10 numbers in the central bottleneck.

These 10 numbers are then a good way to compare documents.

Slide taken from Hinton lectures

The non-linearity used for reconstructing bags of words

Zurich University of Applied Sciences

zh aw

Divide the counts in a bag of words vector by N, where N is the total number of non-stop words in the document.

The resulting probability vector gives the probability of getting a particular word if we pick a non-stop word at random from the document.

At the output of the autoencoder, we use a softmax.

The probability vector defines the desired outputs of the softmax.

We treat the word counts as probabilities, but we make the visible to hidden weights N times bigger than the hidden to hidden weights take into account that we have N observations from the probability distribution.

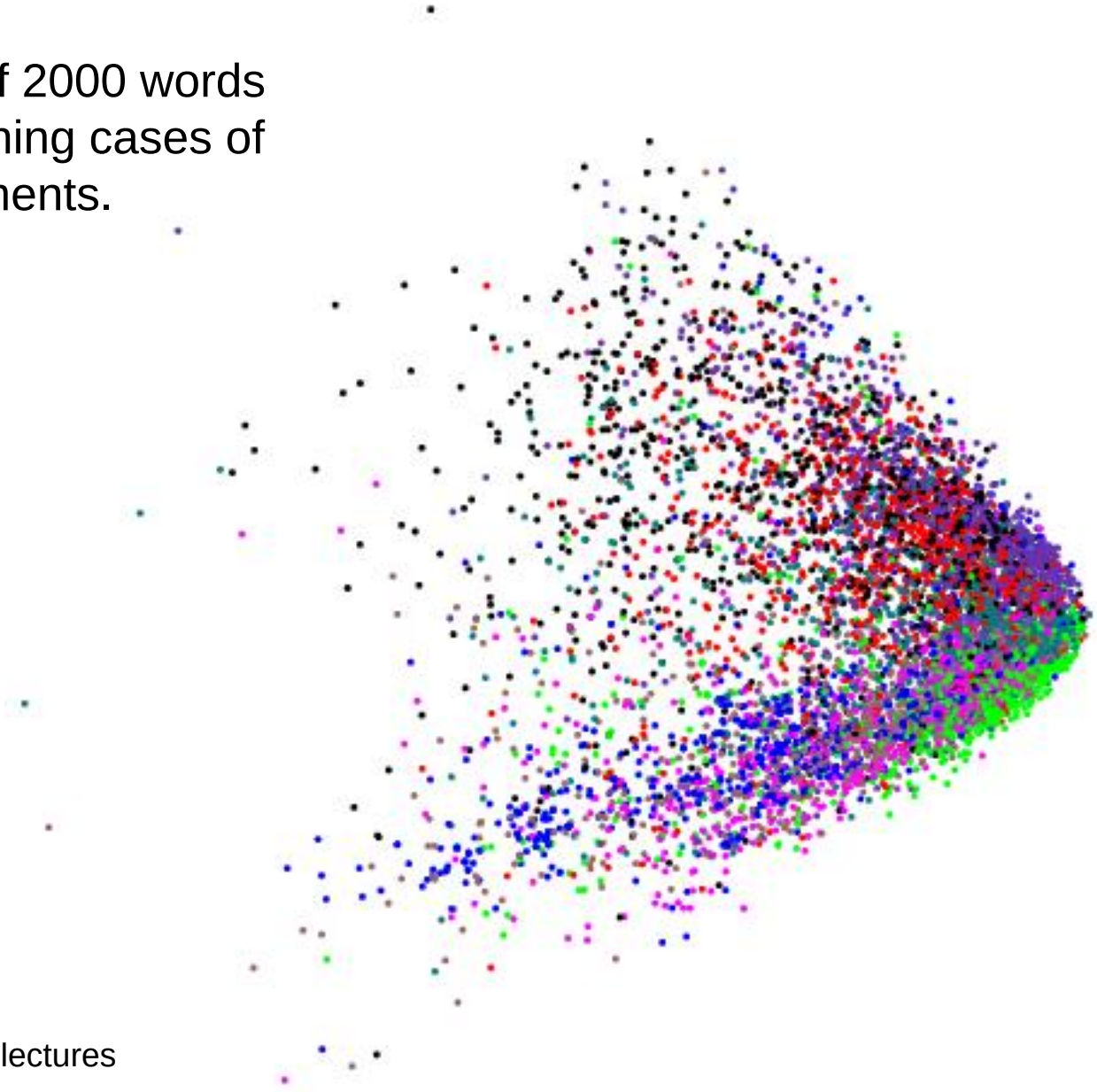
Slide taken from Hinton lectures

First compress all documents to 2 numbers using PCA
Then use different colors for different categories.

Zurich University of Applied Sciences

zh aw

Train on bags of 2000 words for 400,000 training cases of business documents.

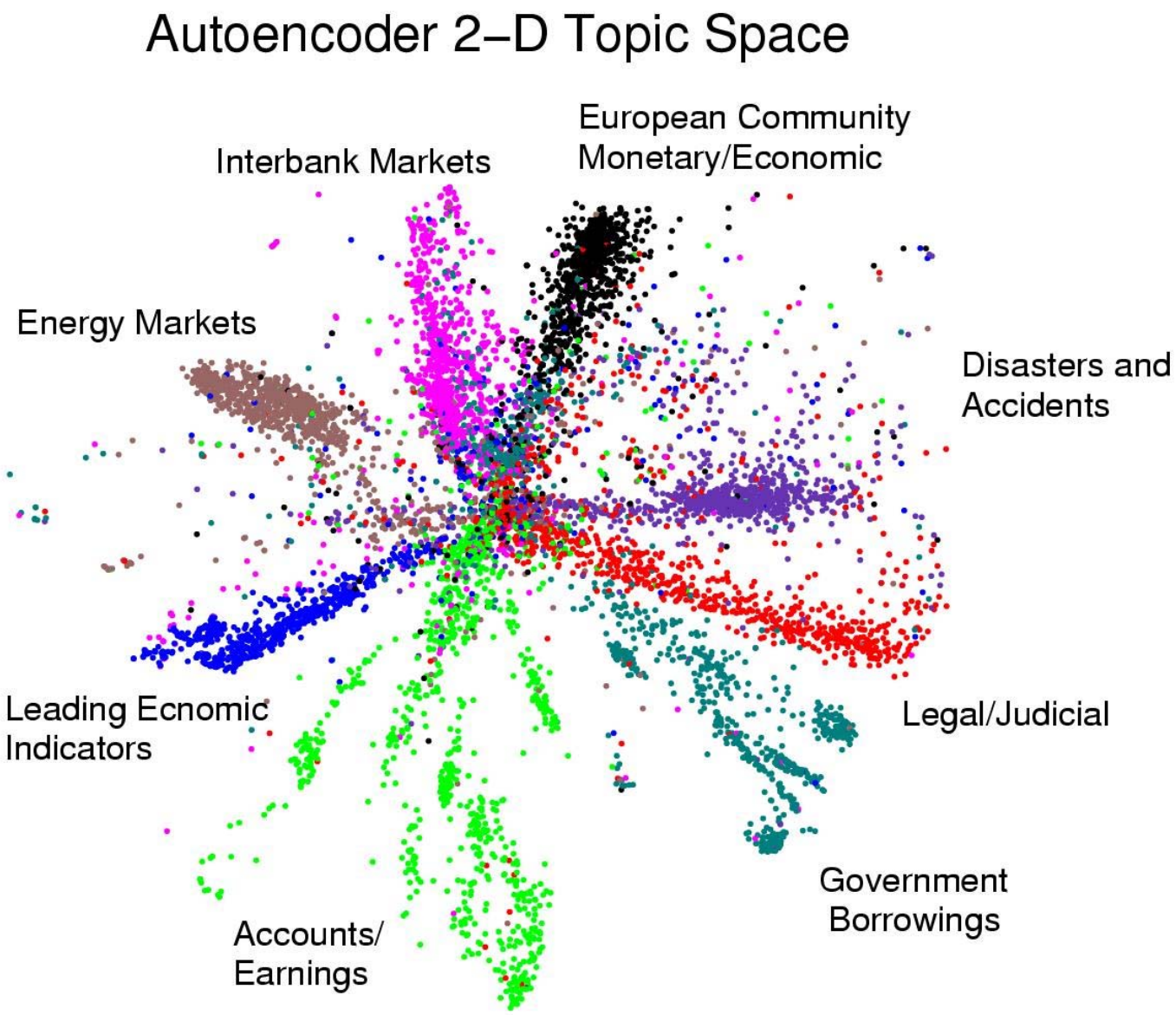


Slide taken from Hinton lectures

First compress all documents to 2 numbers using deep auto. Then use different colors for different document categories

Zurich University of Applied Sciences

zh aw



Slide taken from Hinton lectures

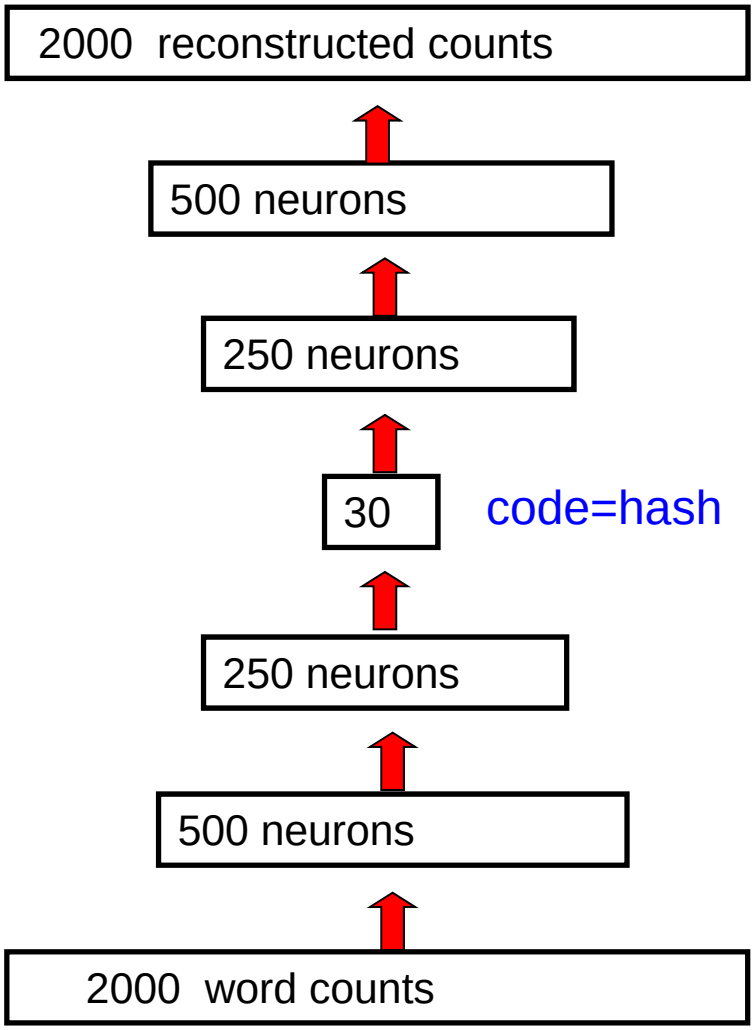
Finding binary codes for documents

Train an auto-encoder using 30 logistic units for the code layer.

During the fine-tuning stage, add noise to the inputs to the code units. The noise forces their activities to become bimodal in order to resist the effects of the noise.

Then we simply threshold the activities of the 30 code units to get a binary code.

Krizhevsky discovered later that its easier to just use binary stochastic units in the code layer during training.



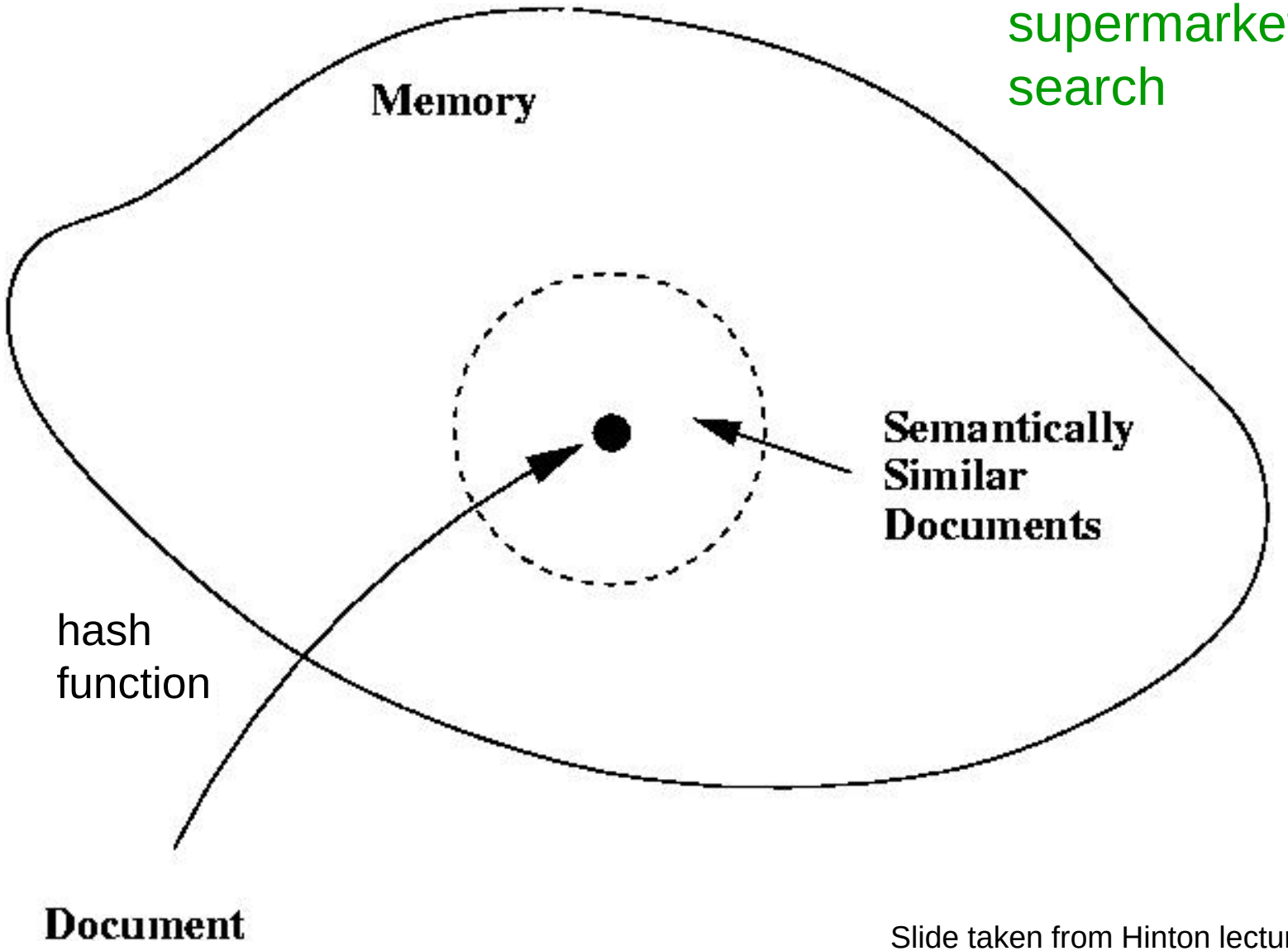
Slide taken from Hinton lectures

Using a deep autoencoder as a hash-function for finding approximate matches

Zurich University of Applied Sciences

zh aw

supermarket search



Slide taken from Hinton lectures

Binary codes for image retrieval

Image retrieval is typically done by using the captions.
Why not use the images too?

Pixels are not like words:
individual pixels do not tell us much about the content.

Extracting object classes from images is hard (now it is possible!)

Maybe we should extract a real-valued vector that has information about the content? Matching real-valued vectors in a big database is slow and requires a lot of storage.

Short binary codes are very easy to store and match.

Slide taken from Hinton lectures

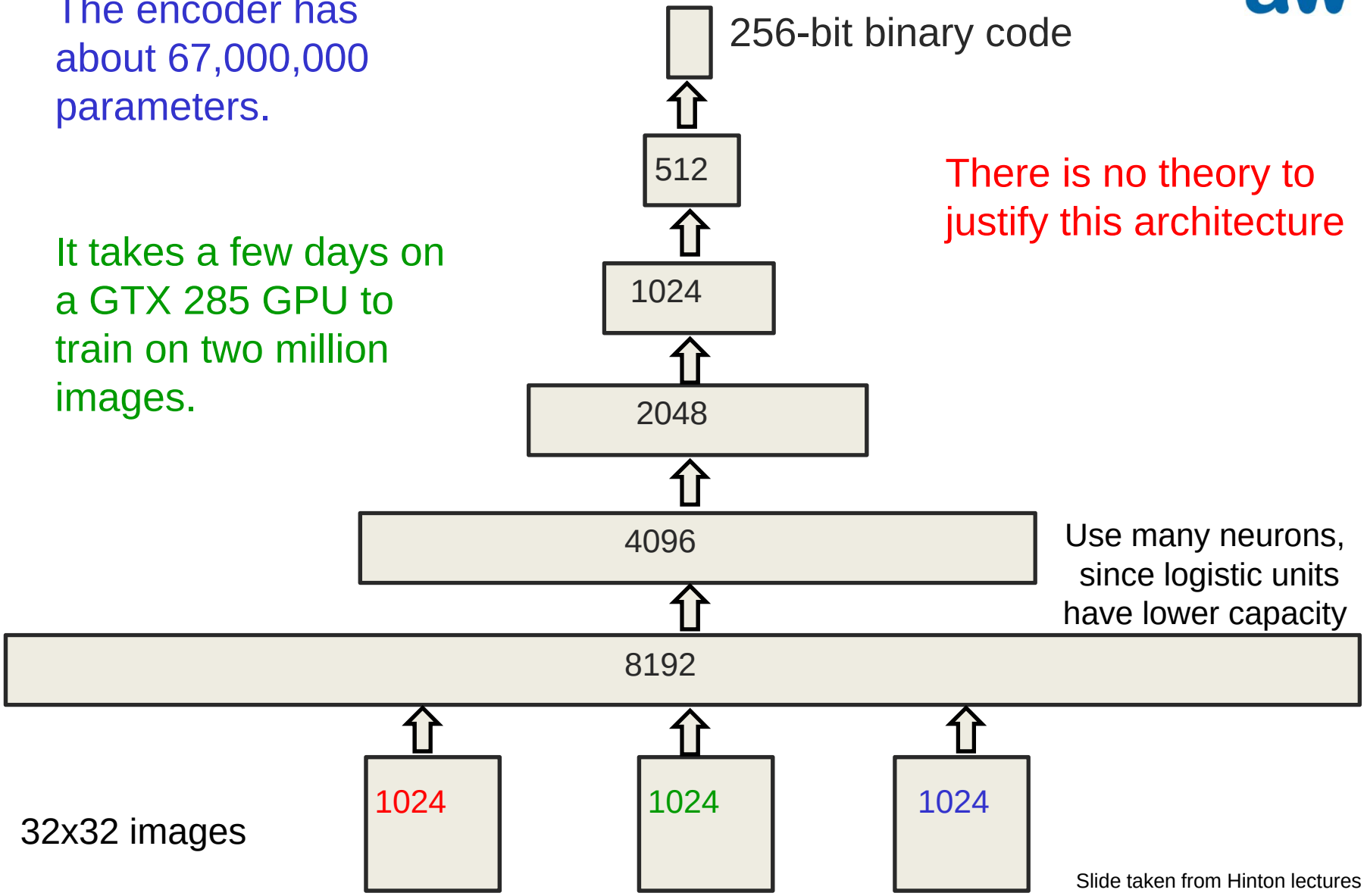
Krizhevsky's deep autoencoder

Zurich University of Applied Sciences

zh aw

The encoder has about 67,000,000 parameters.

It takes a few days on a GTX 285 GPU to train on two million images.



There is no theory to justify this architecture

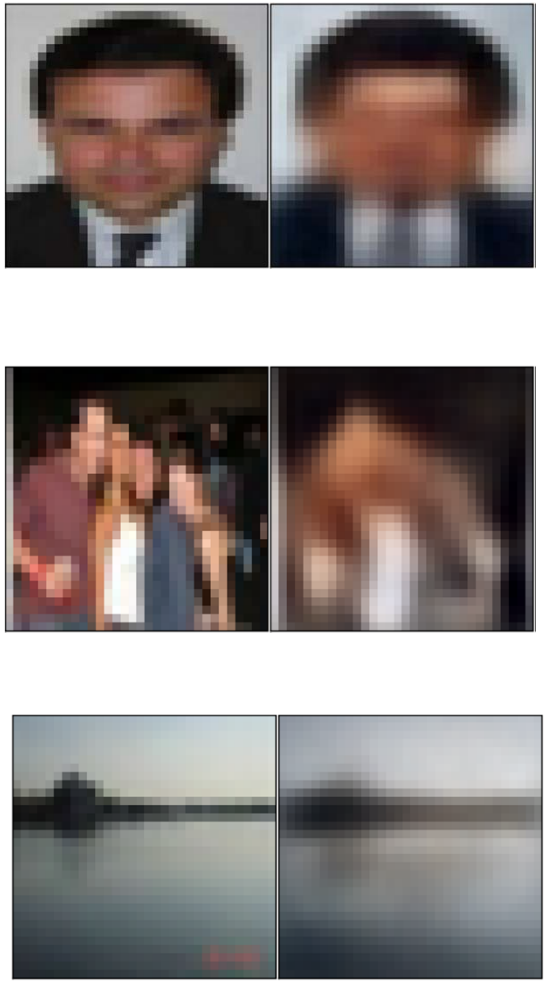
Use many neurons, since logistic units have lower capacity

Slide taken from Hinton lectures

Reconstructions of 32x32 color images from 256-bit codes

Zurich University of Applied Sciences

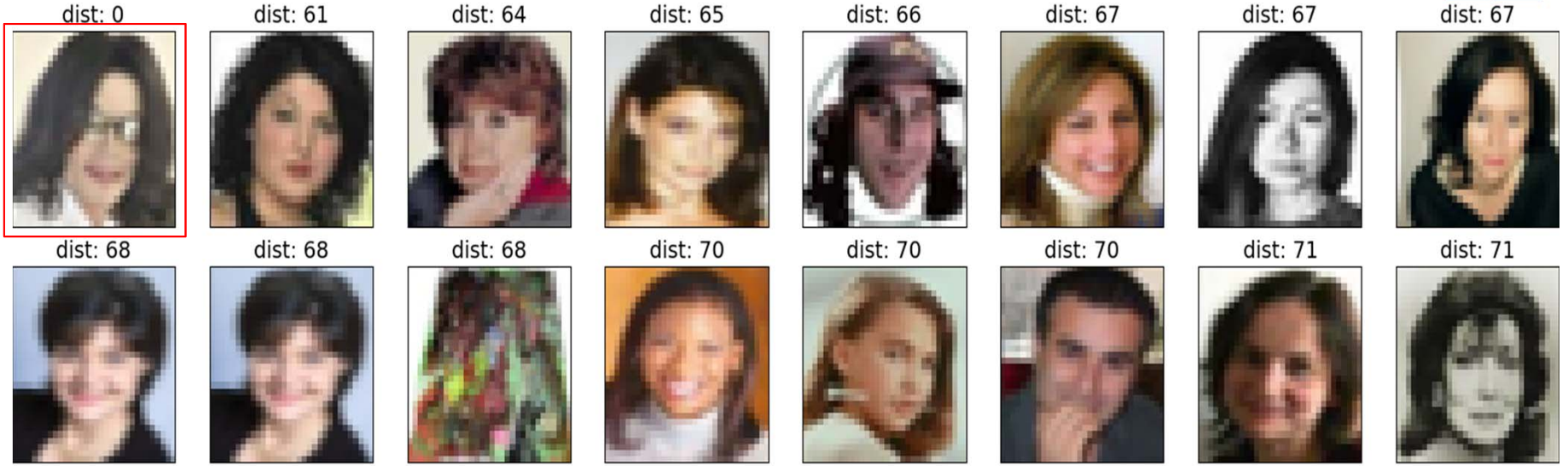
zh aw



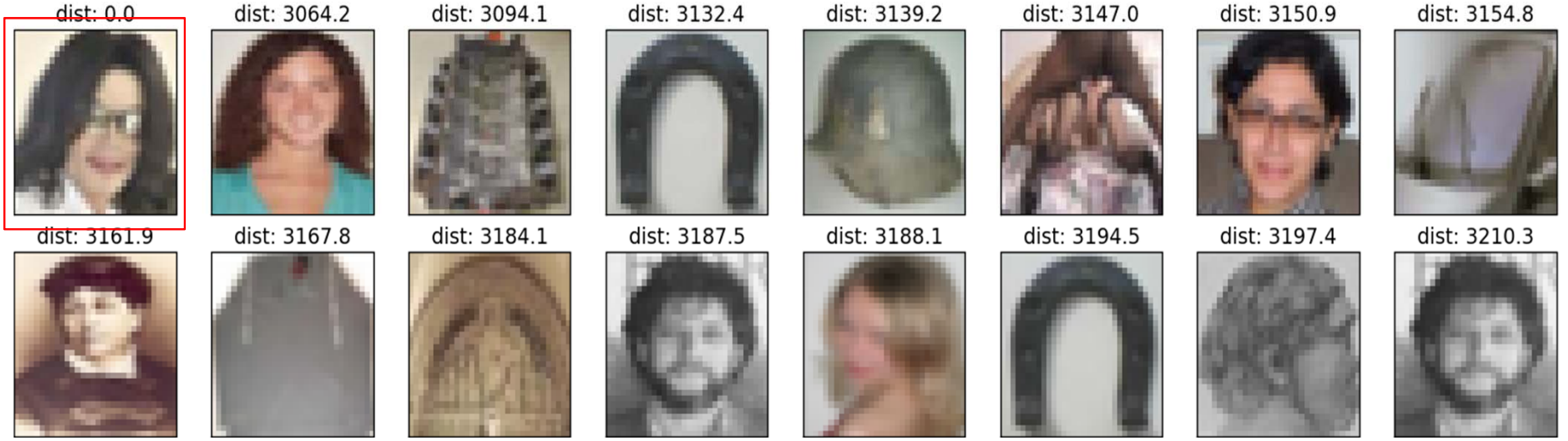
Slide taken from Hinton lectures

Zurich University of Applied Sciences

retrieved using 256 bit codes



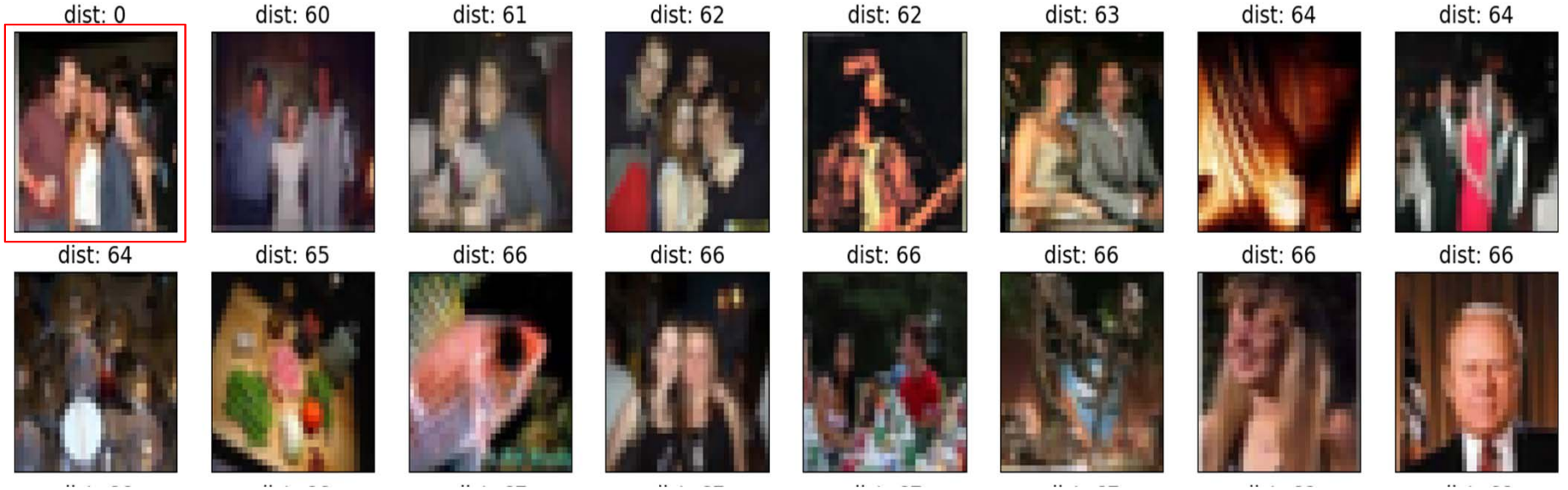
retrieved using Euclidean distance in pixel intensity space



Slide taken from Hinton lectures

retrieved using 256 bit codes

Zurich University of Applied Sciences



retrieved using Euclidean distance in pixel intensity space

Slide taken from Hinton lectures

How to make image retrieval more sensitive to objects and less sensitive to pixels

Zurich University of Applied Sciences

zh aw

First train a big net to recognize lots of different types of objects in real images.

Use net that won the ImageNet competition

Then use the activity vector in the last hidden layer as the representation of the image. Use e.g. the Euclidian distance between the activity vectors in the last hidden layer.

This should be a much better representation to match than the pixel intensities.

Slide taken from Hinton lectures

Zurich University of Applied Sciences

zh aw

Leftmost column is the search image.

Other columns are the images that have the most similar feature activities in the last hidden layer.

Slide taken from Hinton lectures