CSC2535: 2013
Advanced Machine Learning
Lecture 8b

Image retrieval using multilayer neural networks

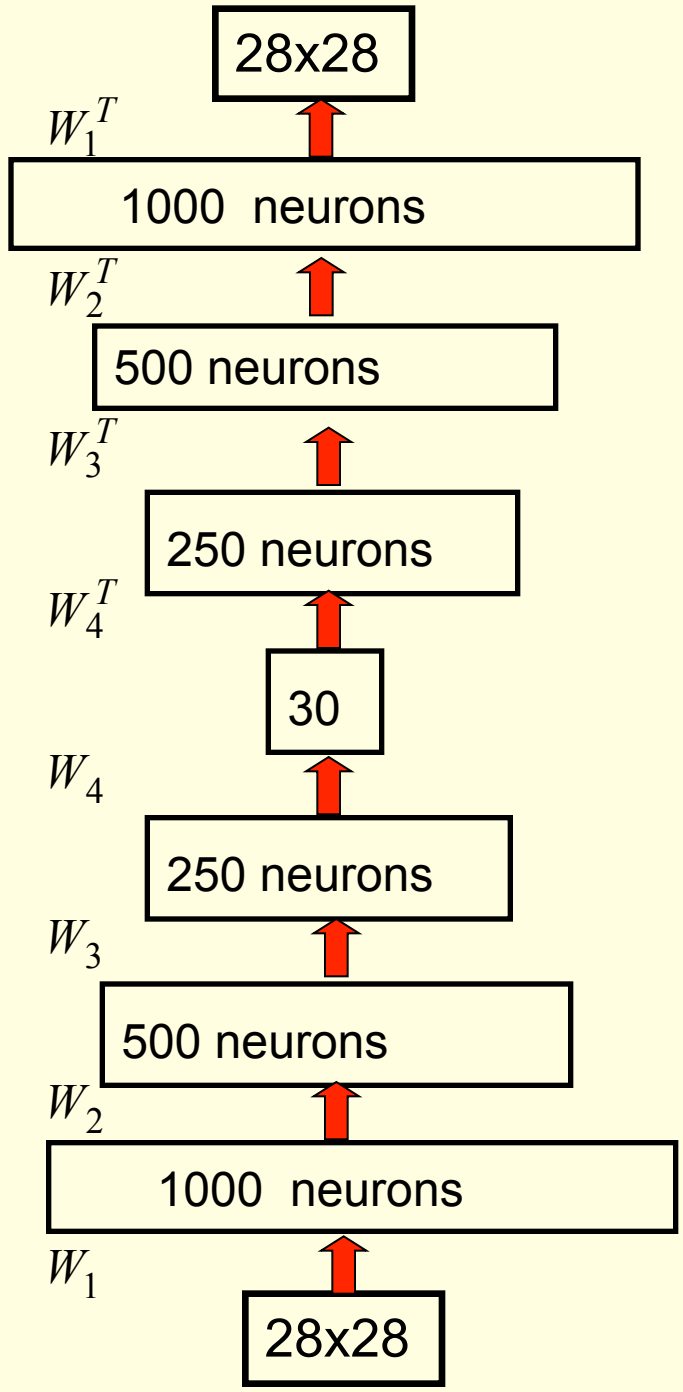
Geoffrey Hinton

Overview

- An efficient way to train a multilayer neural network to extract a low-dimensional representation.
- Document retrieval (published work with Russ Salakhutdinov)
  - How to model a bag of words with an RBM
  - How to learn binary codes
  - Semantic hashing: retrieval in no time
- Image retrieval (published work with Alex Krizhevsky)
  - How good are 256-bit codes for retrieval of small color images?
  - Ways to use the speed of semantic hashing for much higher-quality image retrieval (work in progress).

Deep Autoencoders
(with Ruslan Salakhutdinov)

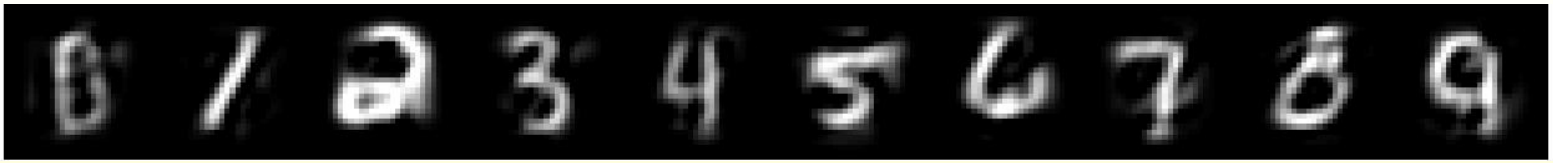
- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them:
  - First train a stack of 4 RBM's
  - Then "unroll" them.
  - Then fine-tune with backprop.



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

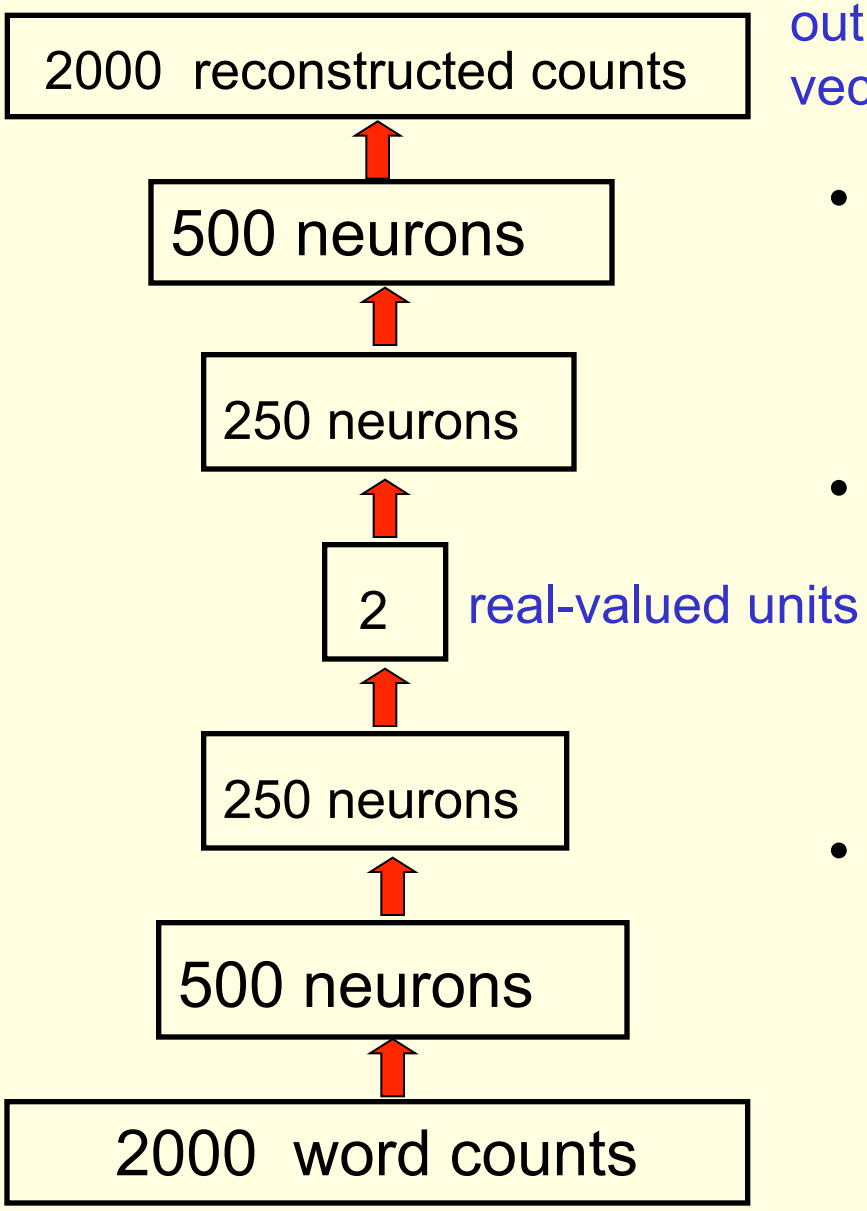
real data

30-D deep auto



30-D PCA

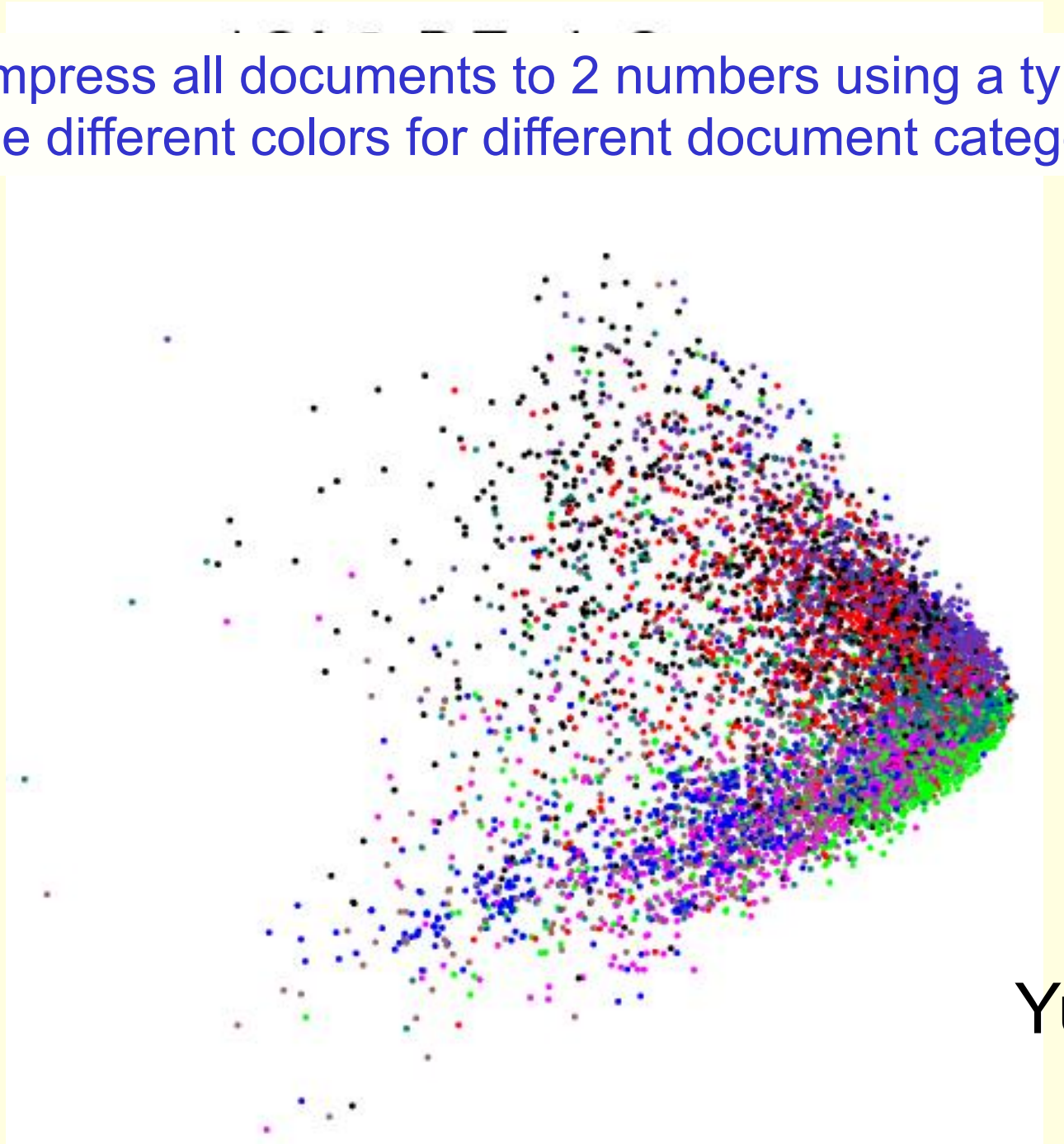
Compressing a document count vector to 2 numbers



- We train the autoencoder to reproduce its input vector as its output
- This forces it to compress as much information as possible into the 2 real numbers in the central bottleneck.
- These 2 numbers are then a good way to visualize documents.

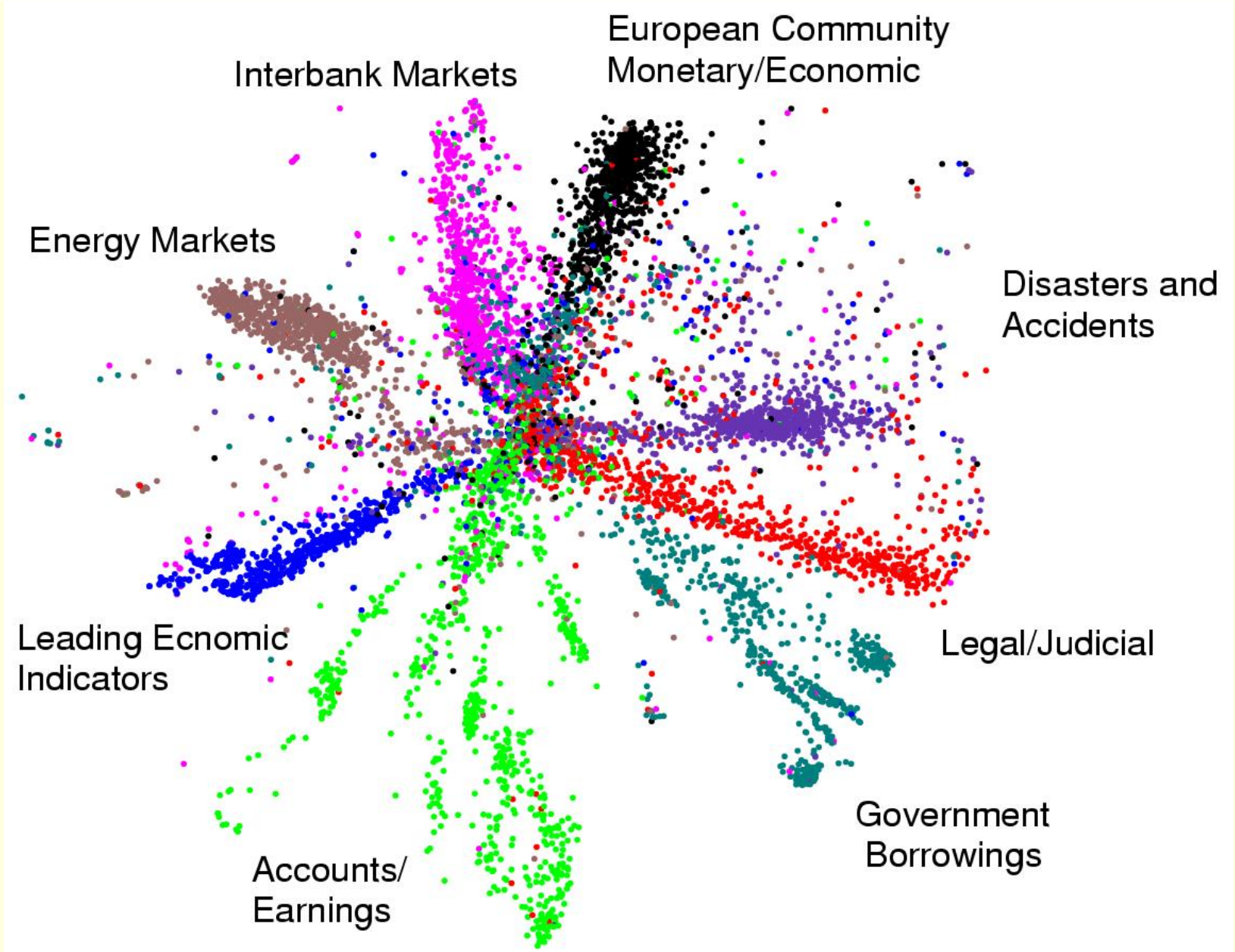
We need a special type of RBM to model counts

First compress all documents to 2 numbers using a type of PCA
Then use different colors for different document categories



Yuk!

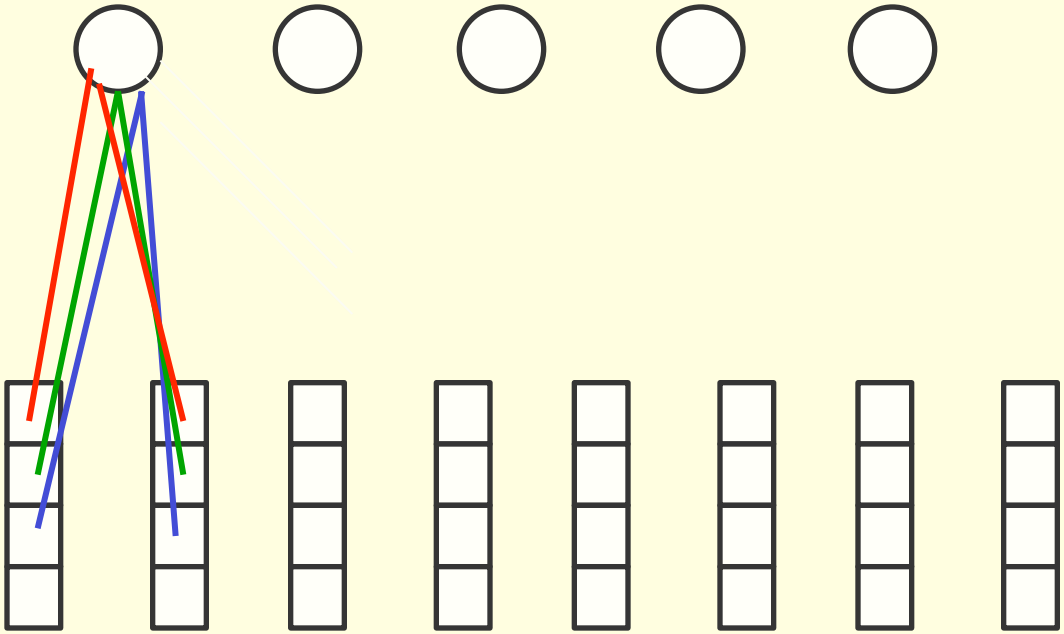
First compress all documents to 2 numbers.
Then use different colors for different document categories



The replicated softmax model: How to modify an RBM to model word count vectors

- Modification 1: Keep the binary hidden units but use "softmax" visible units that represent 1-of-N
- Modification 2: Make each hidden unit use the same weights for all the visible softmax units.
- Modification 3: Use as many softmax visible units as there are non-stop words in the document.
  - So its actually a family of different-sized RBMs that share weights. Its not a single generative model.
- Modification 4: Multiply each hidden bias by the number of words in the document (not done in our earlier work)
- The replicated softmax model is much better at modeling bags of words than LDA topic models (in NIPS 2009)

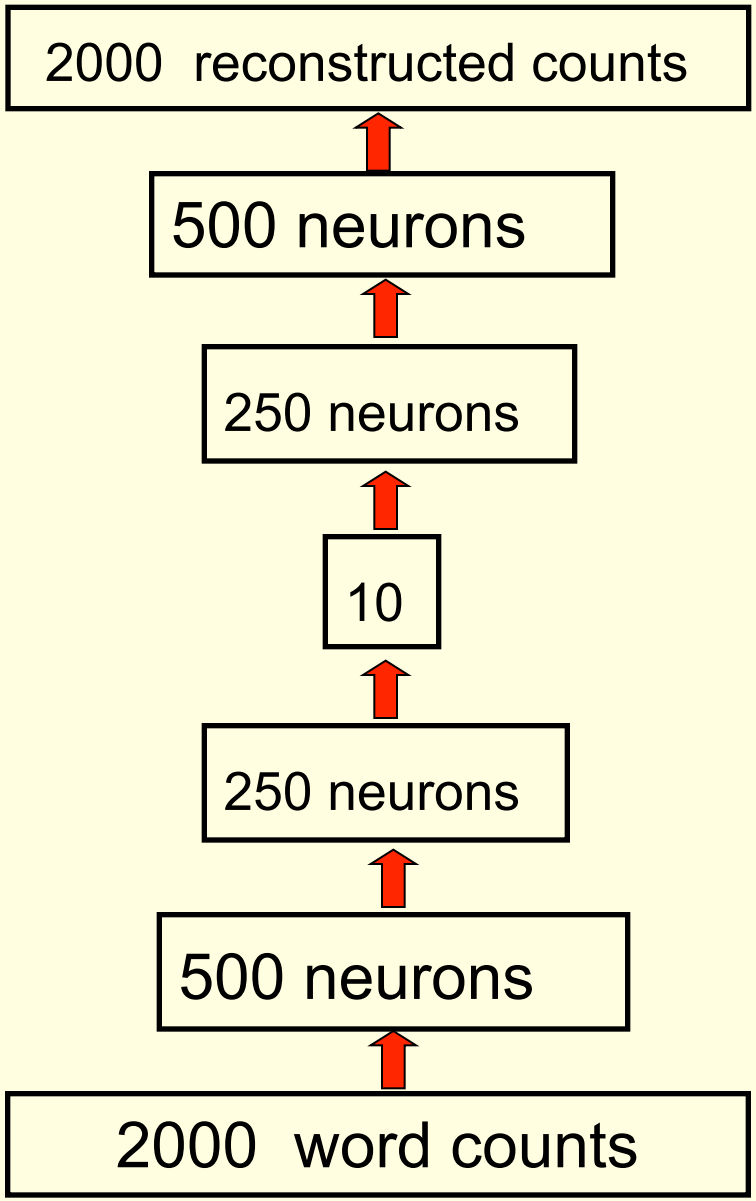
The replicated softmax model



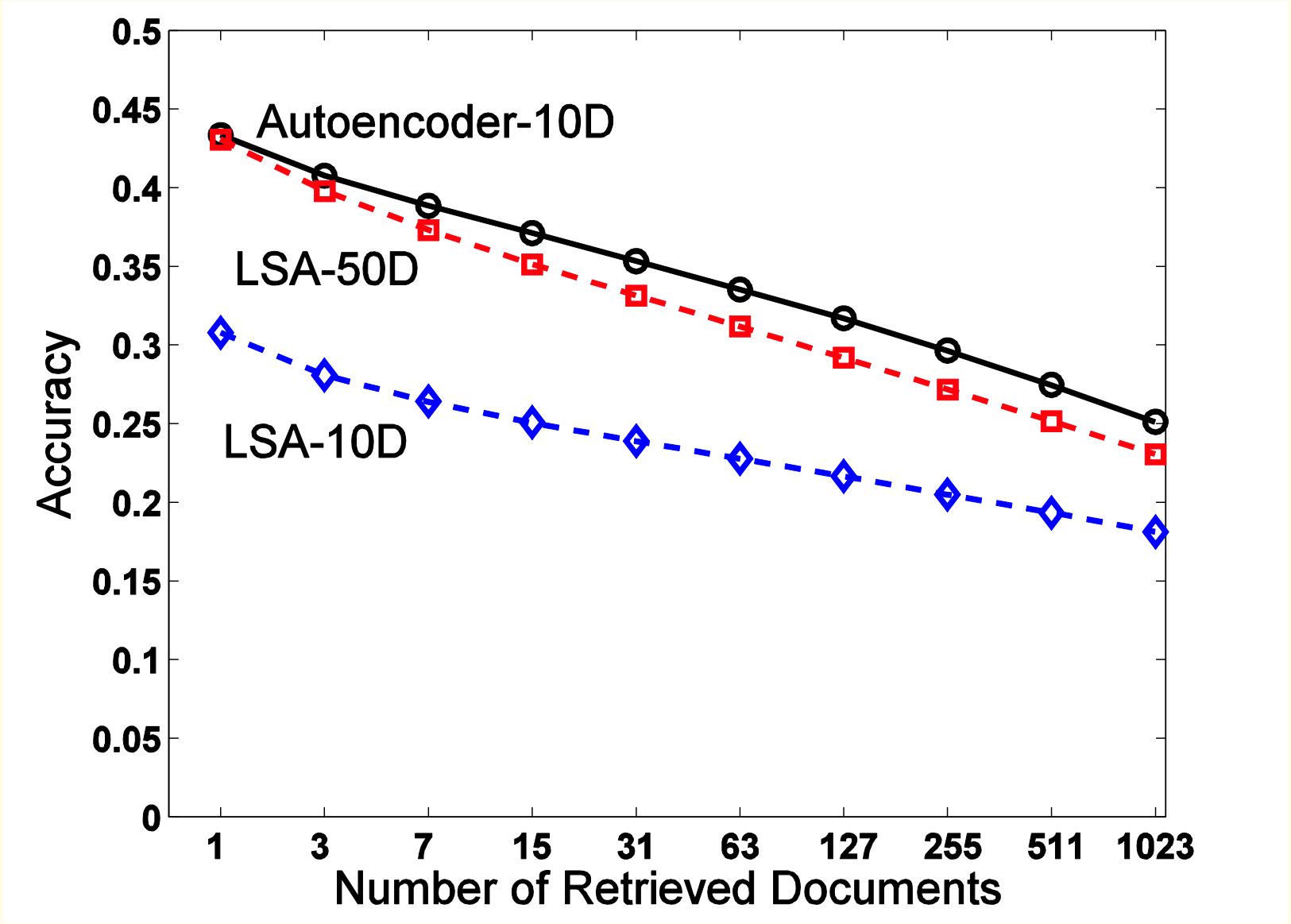
All the models in this family have 5 hidden units. This model is for 8-word documents.

Finding real-valued codes for retrieval

- Train an auto-encoder using 10 real-valued units in the code layer.
- Compare with Latent Semantic Analysis that uses PCA on the transformed count vector
- Non-linear codes are much better.

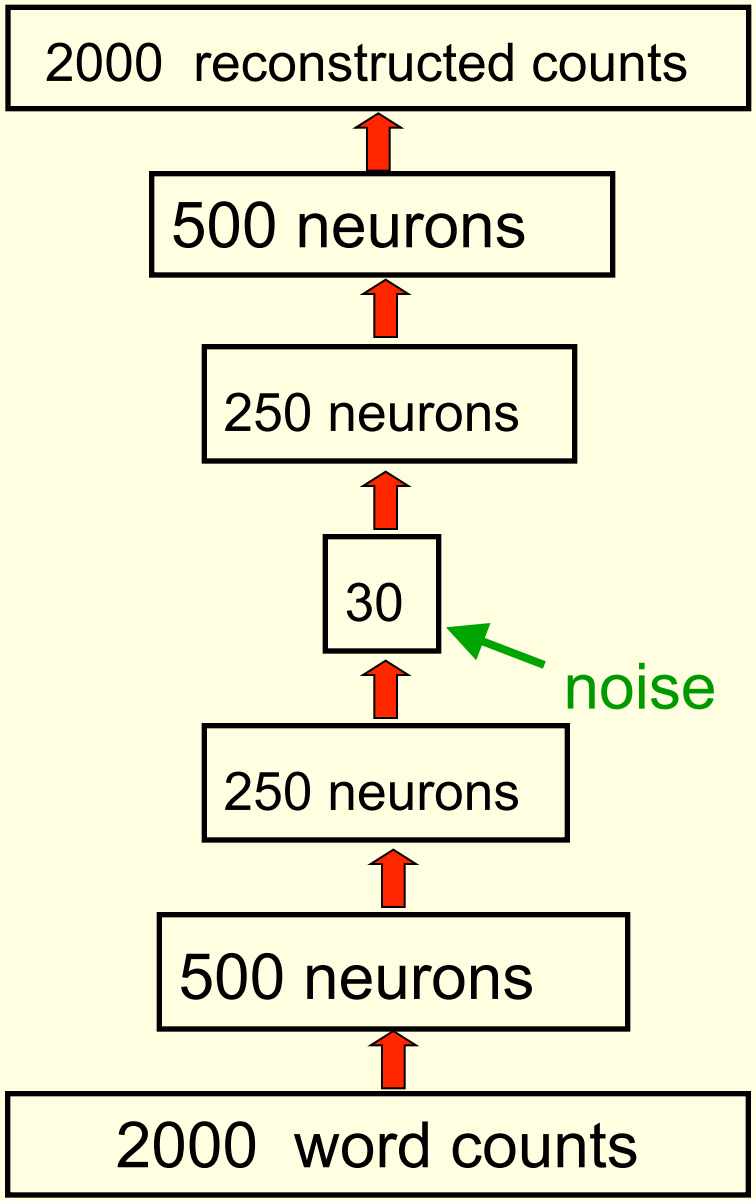


Retrieval performance on 400,000 Reuters business news stories

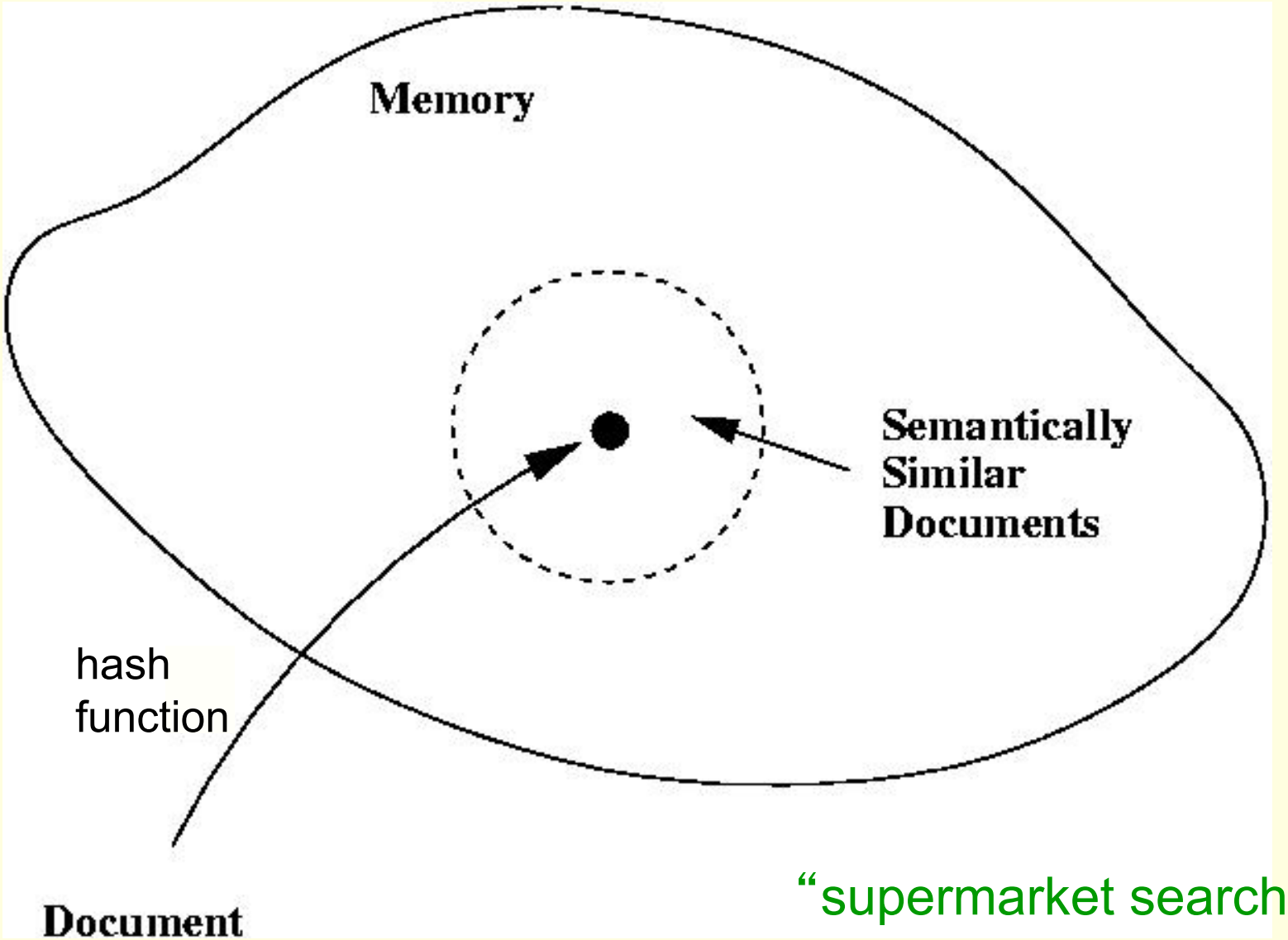


Finding binary codes for documents

- 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" vector for each training case is fixed. So we still get a deterministic gradient.
  - 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.



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



"supermarket search"

Another view of semantic hashing

- Fast retrieval methods typically work by intersecting stored lists that are associated with cues extracted from the query.
- Computers have special hardware that can intersect 32 very long lists in one instruction.
  - Each bit in a 32-bit binary code specifies a list of half the addresses in the memory.
- Semantic hashing uses machine learning to map the retrieval problem onto the type of list intersection the computer is good at.

How good is a shortlist found this way?

- Russ has only implemented it for a million documents with 20-bit codes --- but what could possibly go wrong?
  - A 20-D hypercube allows us to capture enough of the similarity structure of our document set.
- The shortlist found using binary codes actually improves the precision-recall curves of TF-IDF.
  - Locality sensitive hashing (the fastest other method) is much slower and has worse precision-recall curves.

Semantic hashing for image retrieval

- Currently, 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.
- 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 easy to store and match

A two-stage method

- First, use semantic hashing with 30-bit binary codes to get a long “shortlist” of promising images.
- Then use 256-bit binary codes to do a serial search for good matches.
  - This only requires a few words of storage per image and the serial search can be done using fast bit-operations.
- But how good are the 256-bit binary codes?
  - Do they find images that we think are similar?

Some depressing competition

- Inspired by the speed of semantic hashing, Weiss, Fergus and Torralba (NIPS 2008) used a very fast spectral method to assign binary codes to images.
  - This eliminates the long learning times required by deep autoencoders.
- They claimed that their spectral method gave better retrieval results than training a deep auto-encoder using RBM's.
  - But they could not get RBM's to work well for extracting features from RGB pixels so they started from 384 GIST features.
  - This is too much dimensionality reduction too soon.

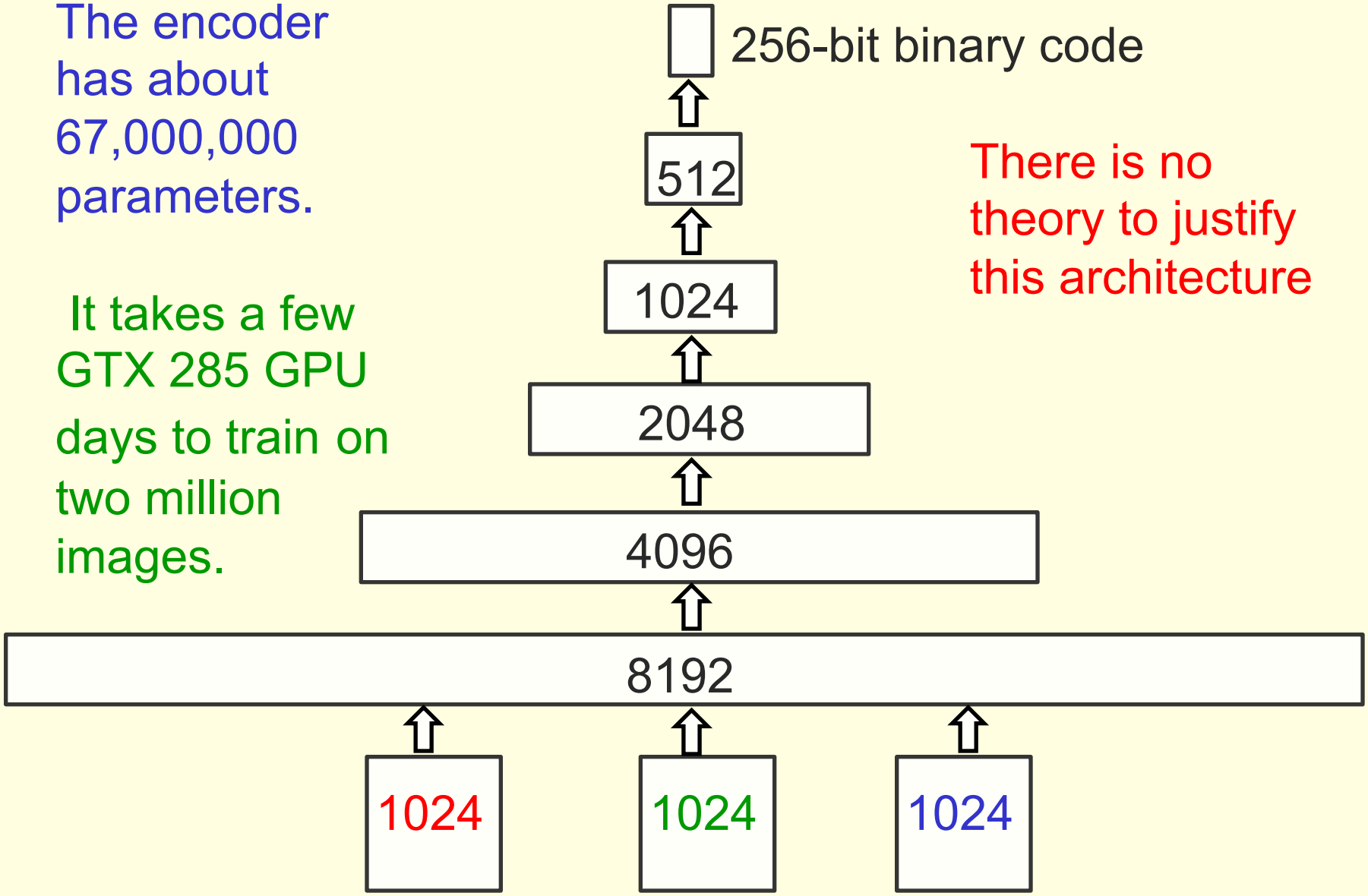
A comparison of deep auto-encoders and the spectral method using 256-bit codes
(Alex Krizhevsky)

- Train auto-encoders “properly”
  - Use Gaussian visible units with fixed variance. Do not add noise to the reconstructions.
  - Use a cluster machine or a big GPU board.
  - Use a lot of hidden units in the early layers.
- Then compare with the spectral method
  - The spectral method has no free parameters.
- Also compare with Euclidean match in pixel space

Krizhevsky's deep autoencoder

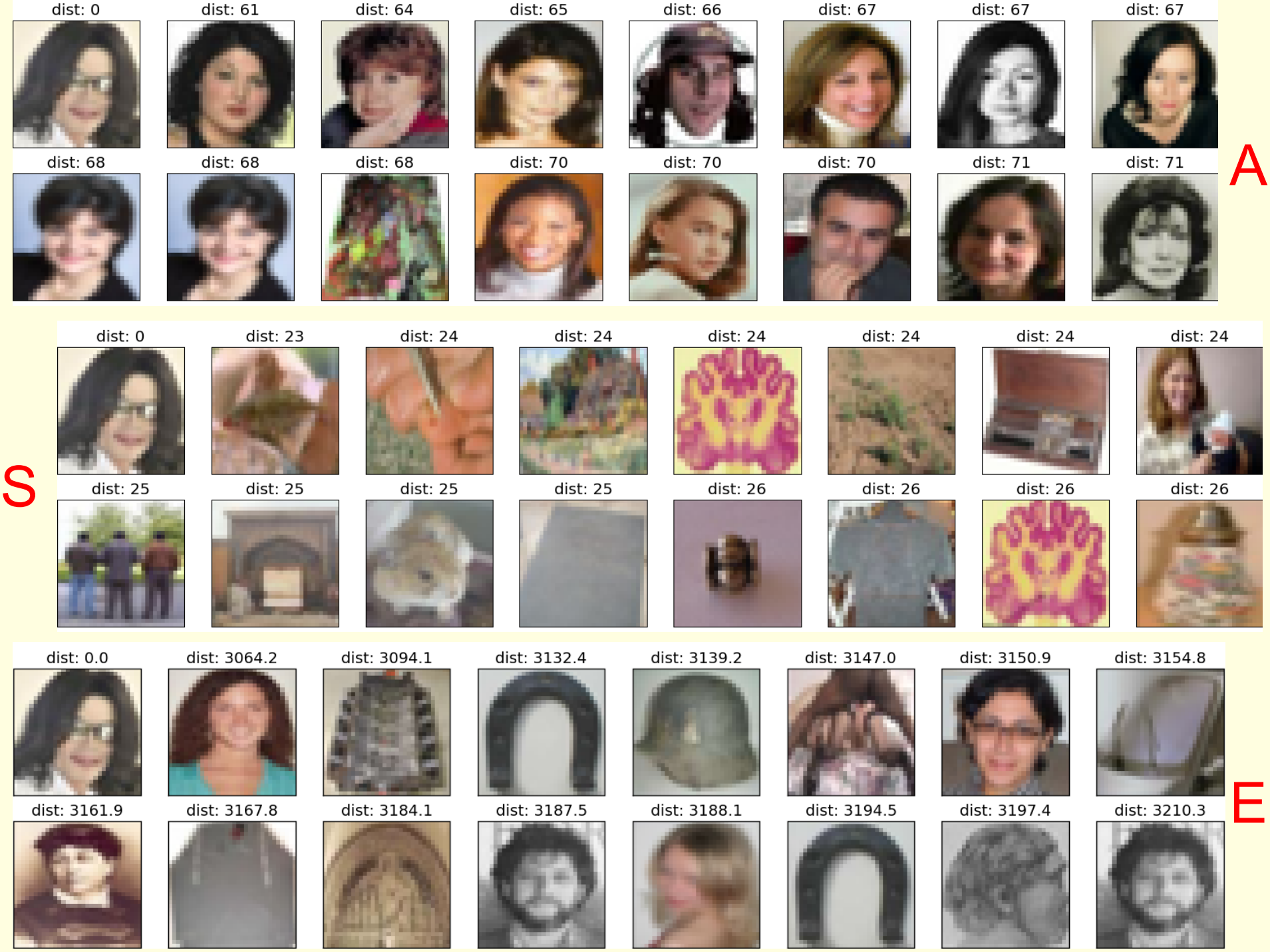
The encoder has about 67,000,000 parameters.

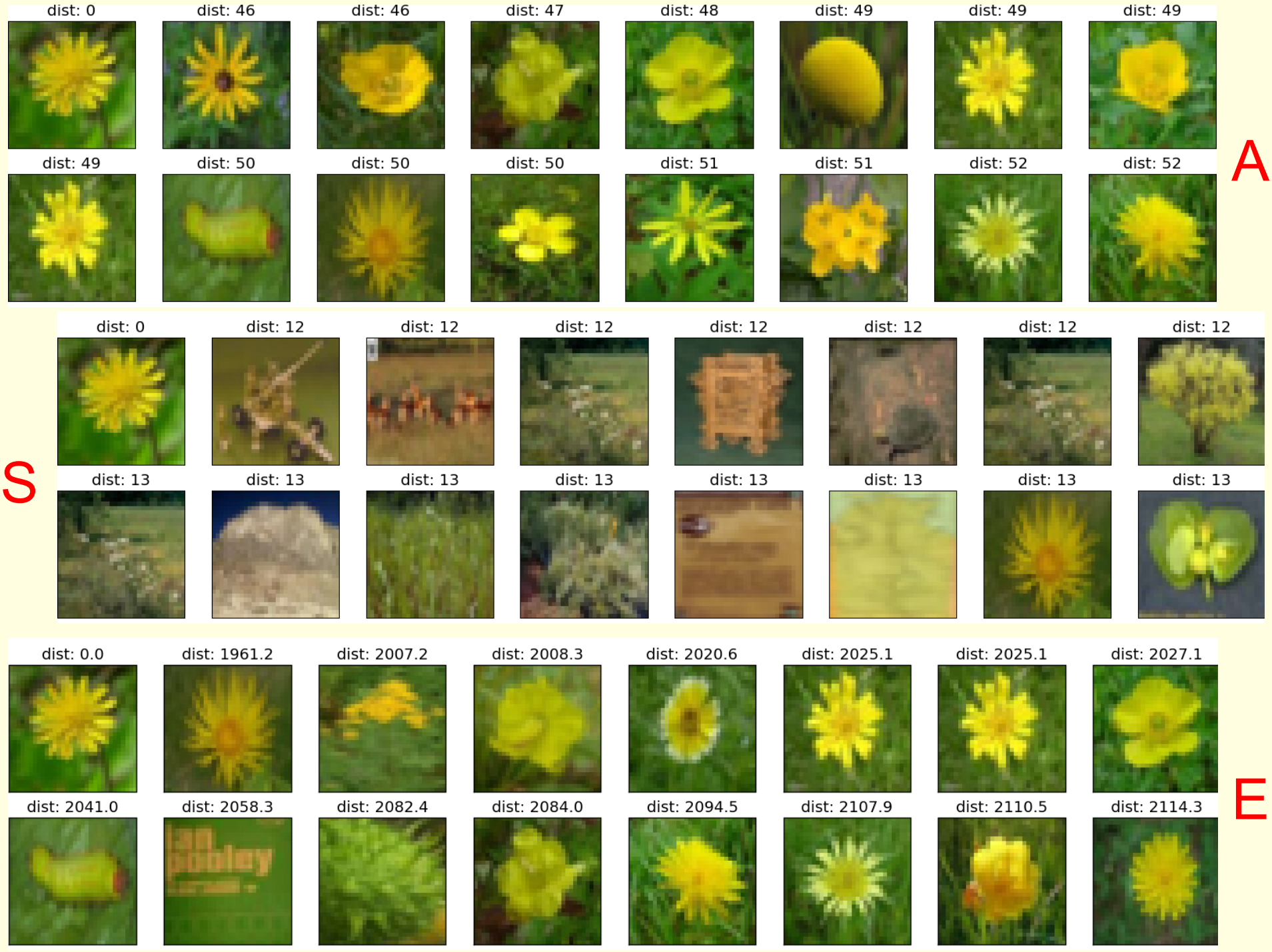
It takes a few GTX 285 GPU days to train on two million images.



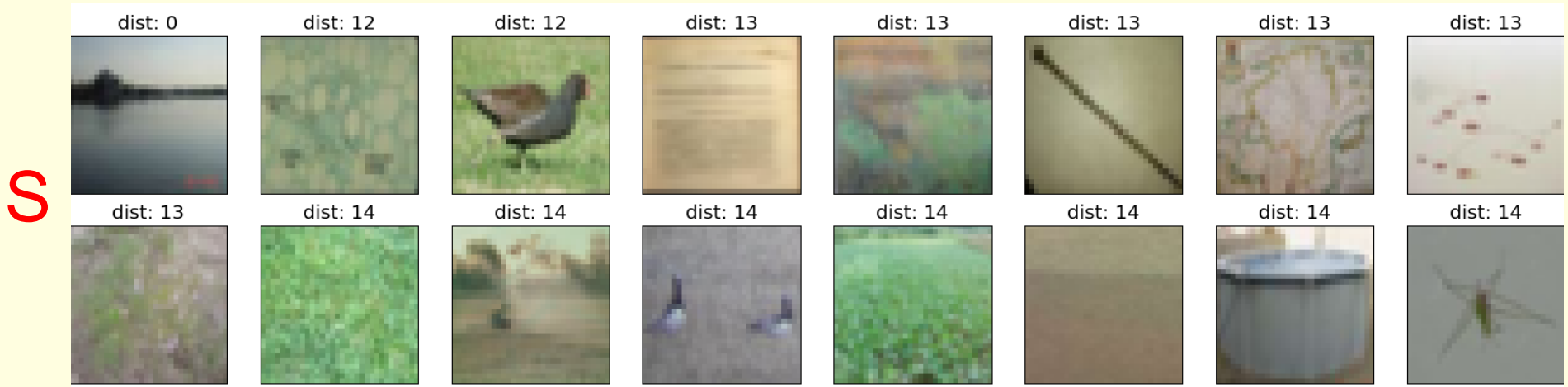
There is no theory to justify this architecture


dist: 0
dist: 51
dist: 52
dist: 53
dist: 55
dist: 55
dist: 57
dist: 58
dist: 58
dist: 59
dist: 59
dist: 59
dist: 59
dist: 60
dist: 60
dist: 60
A
dist: 0
dist: 20
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
dist: 22
dist: 22
dist: 22
dist: 22
dist: 22
dist: 22
dist: 23
dist: 23
dist: 23
S
dist: 0.0
dist: 2480.0
dist: 2558.3
dist: 2678.7
dist: 2757.0
dist: 2806.3
dist: 2807.3
dist: 2831.2
dist: 2846.5
dist: 2894.5
dist: 2906.0
dist: 2914.8
dist: 2917.1
dist: 2917.1
dist: 2924.8
dist: 2943.4
E

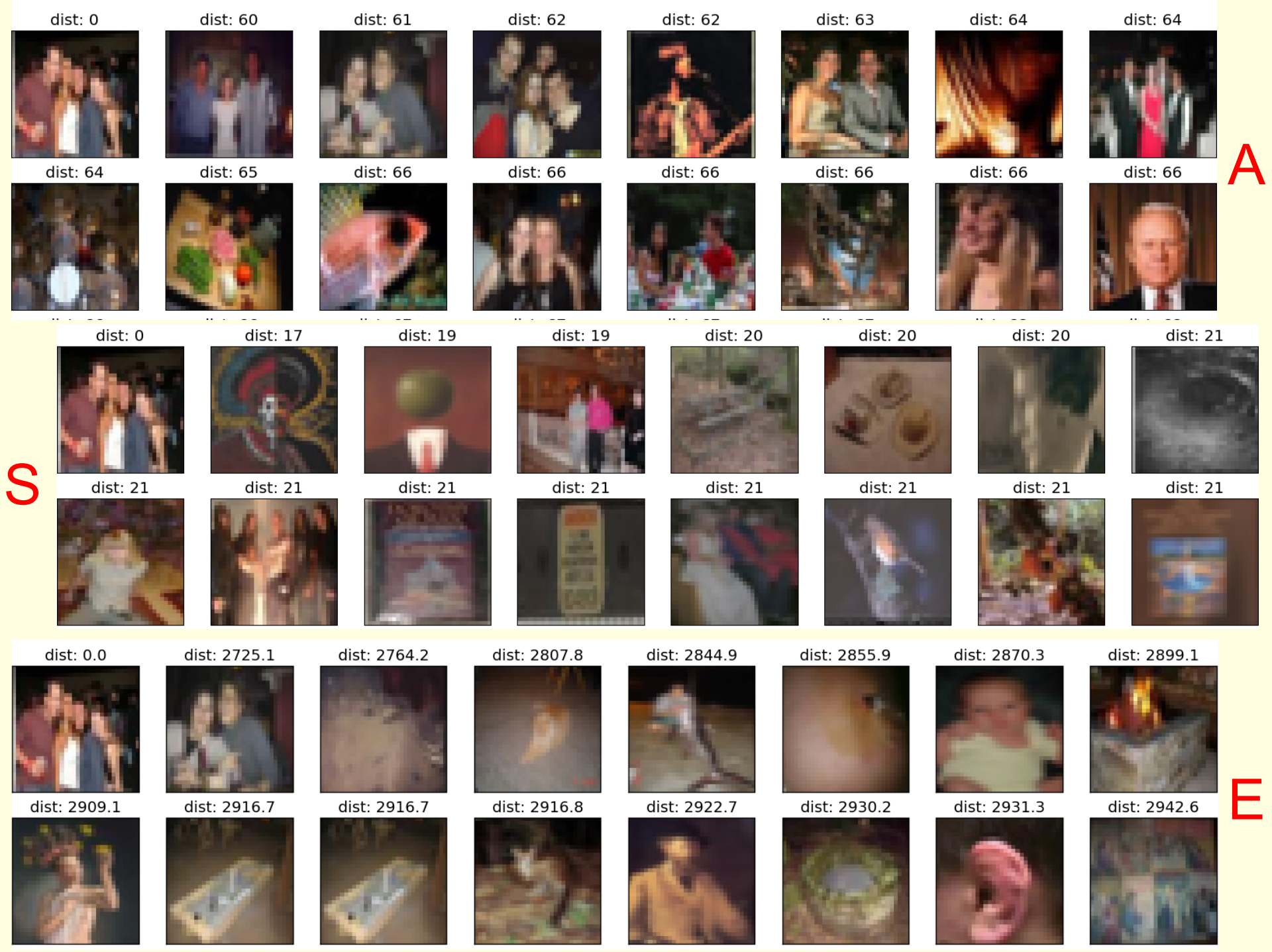

dist: 0
dist: 61
dist: 64
dist: 65
dist: 66
dist: 67
dist: 67
dist: 67
dist: 68
dist: 68
dist: 68
dist: 70
dist: 70
dist: 70
dist: 71
dist: 71
A
dist: 0
dist: 23
dist: 24
dist: 24
dist: 24
dist: 24
dist: 24
dist: 24
S
dist: 25
dist: 25
dist: 25
dist: 25
dist: 26
dist: 26
dist: 26
dist: 26
dist: 0.0
dist: 3064.2
dist: 3094.1
dist: 3132.4
dist: 3139.2
dist: 3147.0
dist: 3150.9
dist: 3154.8
dist: 3161.9
dist: 3167.8
dist: 3184.1
dist: 3187.5
dist: 3188.1
dist: 3194.5
dist: 3197.4
dist: 3210.3
E


dist: 0
dist: 46
dist: 46
dist: 47
dist: 48
dist: 49
dist: 49
dist: 49
A
dist: 49
dist: 50
dist: 50
dist: 50
dist: 51
dist: 51
dist: 52
dist: 52
S
dist: 0
dist: 12
dist: 12
dist: 12
dist: 12
dist: 12
dist: 12
dist: 12
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
E
dist: 0.0
dist: 1961.2
dist: 2007.2
dist: 2008.3
dist: 2020.6
dist: 2025.1
dist: 2025.1
dist: 2027.1
dist: 2041.0
dist: 2058.3
dist: 2082.4
dist: 2084.0
dist: 2094.5
dist: 2107.9
dist: 2110.5
dist: 2114.3


dist: 0
dist: 41
dist: 45
dist: 45
dist: 45
dist: 45
dist: 49
dist: 50
dist: 50
dist: 51
dist: 51
dist: 52
dist: 52
dist: 52
dist: 52
dist: 52
A


dist: 0
dist: 12
dist: 12
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
dist: 13
dist: 14
dist: 14
dist: 14
dist: 14
dist: 14
dist: 14
dist: 14
S


dist: 0.0
dist: 1617.0
dist: 1624.4
dist: 1732.4
dist: 1795.7
dist: 1903.2
dist: 1924.8
dist: 1928.4
dist: 1935.2
dist: 1939.5
dist: 1951.5
dist: 1960.2
dist: 1967.7
dist: 1975.8
dist: 1981.1
dist: 1991.1
E


dist: 0
dist: 60
dist: 61
dist: 62
dist: 62
dist: 63
dist: 64
dist: 64
A
dist: 64
dist: 65
dist: 66
dist: 66
dist: 66
dist: 66
dist: 66
dist: 66
S
dist: 0
dist: 17
dist: 19
dist: 19
dist: 20
dist: 20
dist: 20
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
dist: 21
E
dist: 0.0
dist: 2725.1
dist: 2764.2
dist: 2807.8
dist: 2844.9
dist: 2855.9
dist: 2870.3
dist: 2899.1
dist: 2909.1
dist: 2916.7
dist: 2916.7
dist: 2916.8
dist: 2922.7
dist: 2930.2
dist: 2931.3
dist: 2942.6

The next step

- Implement the semantic hashing stage for images.
- Check that a long shortlist still contains many good matches.
  - It works OK for documents, but they are very different from images.
  - Losing some recall may be OK. People don't miss what they don't know about.

An obvious extension

- Use a multimedia auto-encoder that represents captions and images in a single code.
  - The captions should help it extract more meaningful image features such as "contains an animal" or "indoor image"
- RBM's already work much better than standard LDA topic models for modeling bags of words.
  - So the multimedia auto-encoder should be
    + a win (for images)
    + a win (for captions)
    + a win (for the interaction during training)

A less obvious extension

- Semantic hashing gives incredibly fast retrieval but its hard to go much beyond 32 bits.
- We can afford to use semantic hashing several times with variations of the query and merge the shortlists
  - Its easy to enumerate the hamming ball around a query image address in ascending address order, so merging is linear time.
- Apply many transformations to the query image to get transformation independent retrieval.
  - Image translations are an obvious candidate.

Summary

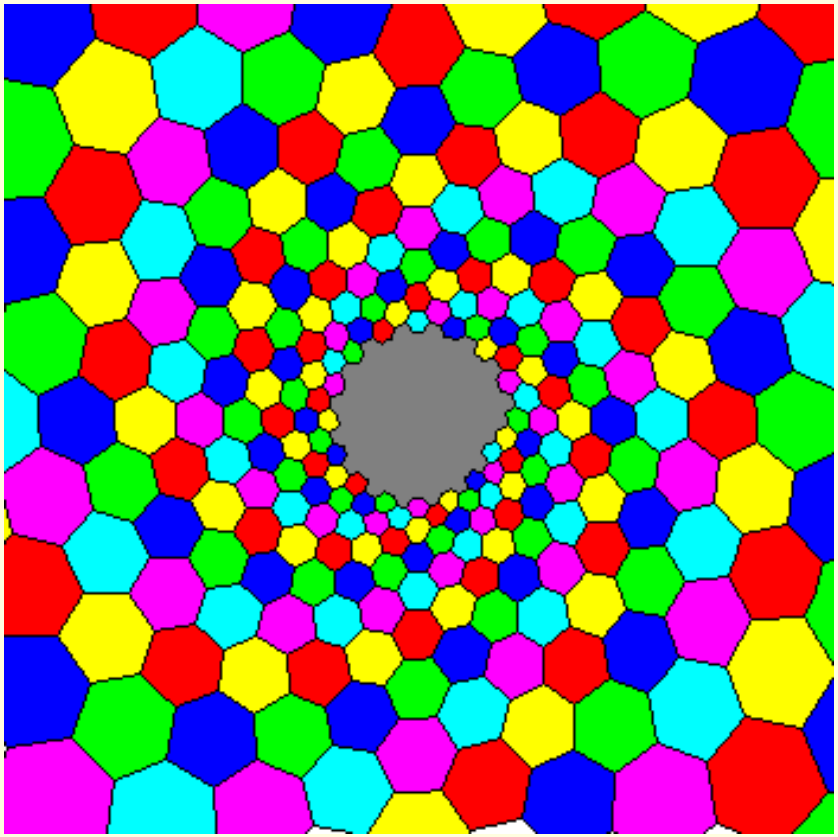
- Restricted Boltzmann Machines provide an efficient way to learn a layer of features without any supervision.
  - Many layers of representation can be learned by treating the hidden states of one RBM as the data for the next.
- This allows us to learn very deep nets that extract short binary codes for unlabeled images or documents.
  - Using 32-bit codes as addresses allows us to get approximate matches at the speed of hashing.
- Semantic hashing is fast enough to allow many retrieval cycles for a single query image.
  - So we can try multiple transformations of the query.

A more interesting extension

- Computer vision uses images of uniform resolution.
  - Multi-resolution images still keep all the high-resolution pixels.
- Even on 32x32 images, people use a lot of eye movements to attend to different parts of the image.
  - Human vision copes with big translations by moving the fixation point.
  - It only samples a tiny fraction of the image at high resolution. The “post-retinal’’ image has resolution that falls off rapidly outside the fovea.
  - With less “neurons” intelligent sampling becomes even more important.

How to perceive a big picture with a small brain

- Even a human brain cannot afford high-resolution everywhere.
  - By limiting the input we make it possible to use many layers of dense features intelligently.
- For fine discrimination that requires high-resolution in several different places we must integrate over several fixations.



A much better “retina”.

A more human metric for image similarity

- Two images are similar if fixating at point X in one image and point Y in the other image gives similar post-retinal images.
- So use semantic hashing on post-retinal images.
  - The address space is used for post-retinal images and each address points to the whole image that the post-retinal image came from.
  - So we can accumulate similarity over multiple fixations.
- The whole image addresses found after each fixation have to be sorted to allow merging ☹

Starting from a better input representation

- First learn a good model for object recognition that can deal wit multiple objects in the same image.
- Then use the outputs of the last hidden layer as the inputs to a deep autoencoder.
- This should work really well.
  - Euclidean distance on the activities n the last hidden layer already works extremely well.

cue

Euclidean nearest neighbors using the 4096 activities in the last hidden layer