DEEP LEARNING IN AUTOMATIC SPEECH RECOGNITION

AUTOMATIC SPEECH RECOGNITION

What is ASR?

- Automatic Speech Recognition
- Input: Speech
- Output: Text

Why ASR?

- Natural:
  - Requires no special training
- Flexible:
  - Leaves hands and eyes free
- Efficient:
  - Has high data rate
- Economical:
  - Can be transmitted/received inexpensively

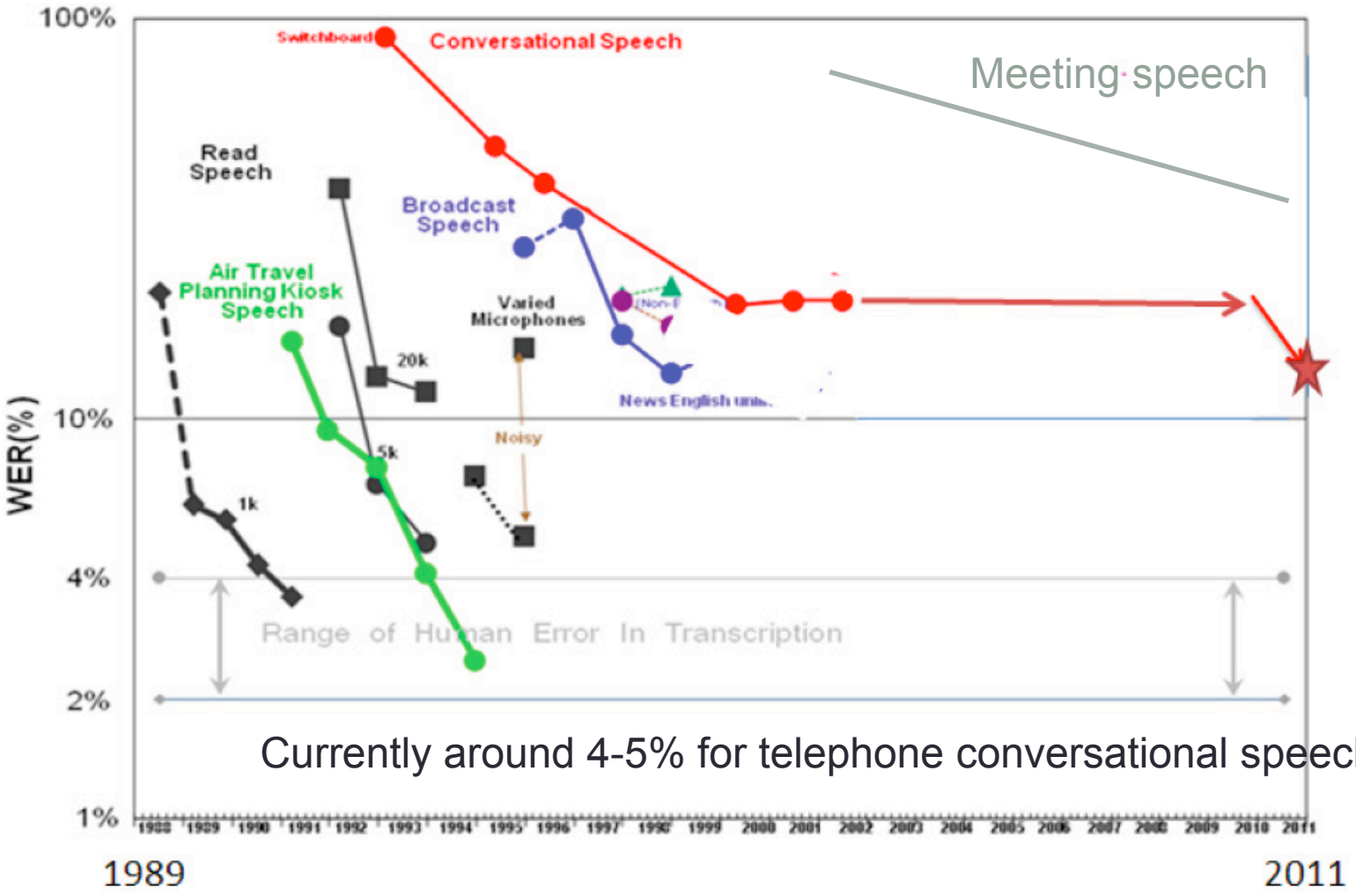
ASR performance

- Word error rate (WER) = Insertion error + substitution error + deletion error
- Word Accuracy = 1 – WER

- I ate lunch with Peter yesterday.
- I ate two lunches with Peter ___.
- INS SUB DEL

- $\text{WER} = \frac{1 + 1 + 1}{6} = 50\%$

ASR performance



Currently around 4-5% for telephone conversational speech

Types of ASR (by size)

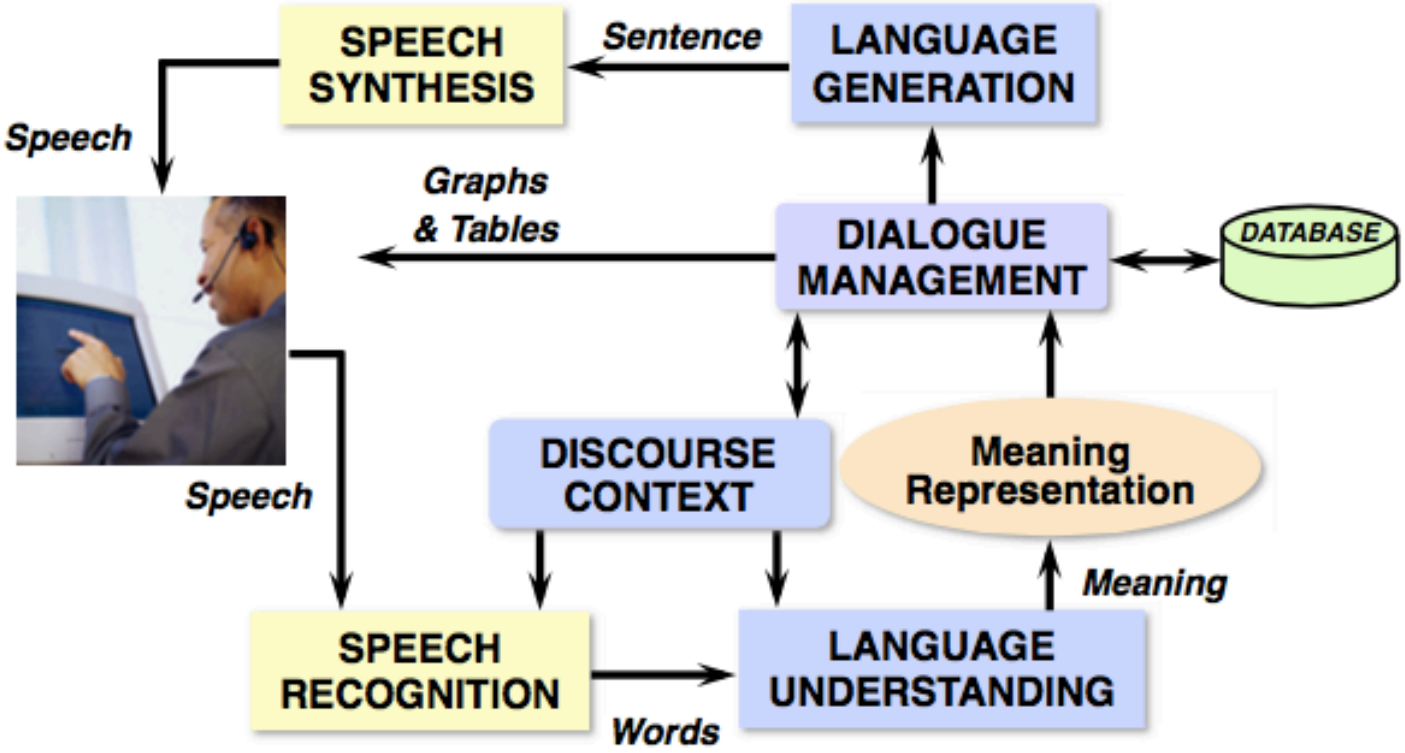
- Wake word (hot word)
  - Okay google, hey Alexa
  - Low power, continuously listening
- Low vocabulary (<1000 number of words)
  - Smart home
- Large vocabulary (>1000 number of words)
  - Voice search
  - Dictation

Types of ASR

- Close talking - Far field
- Speaker dependent – Speaker independent
- Isolated word – continuous speech
- Read speech - Conversational speech
- Small – large vocabulary

Other issues

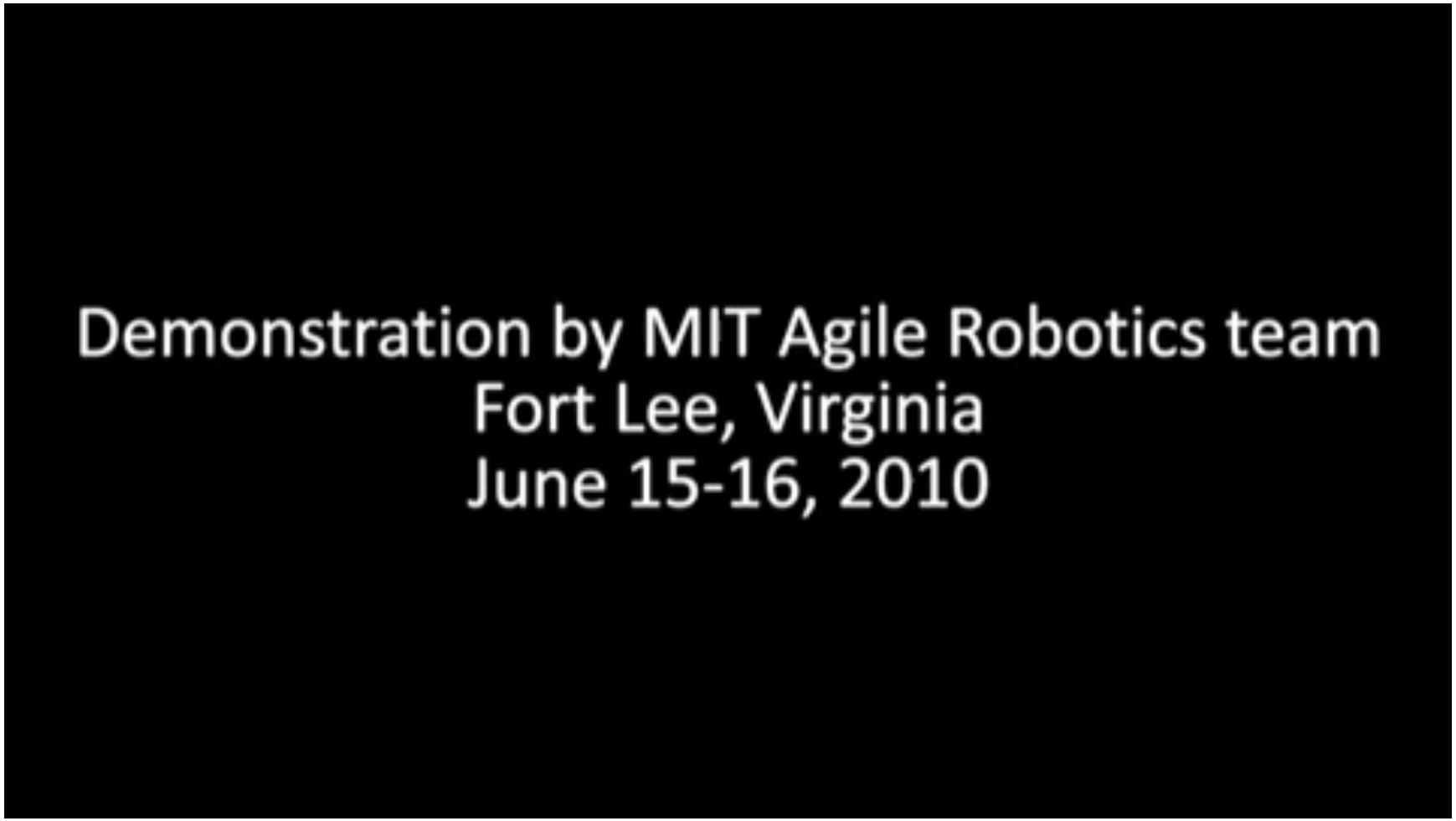
- Accents
- Background noise
- Dialogues (ASR meets Natural Language Processing)



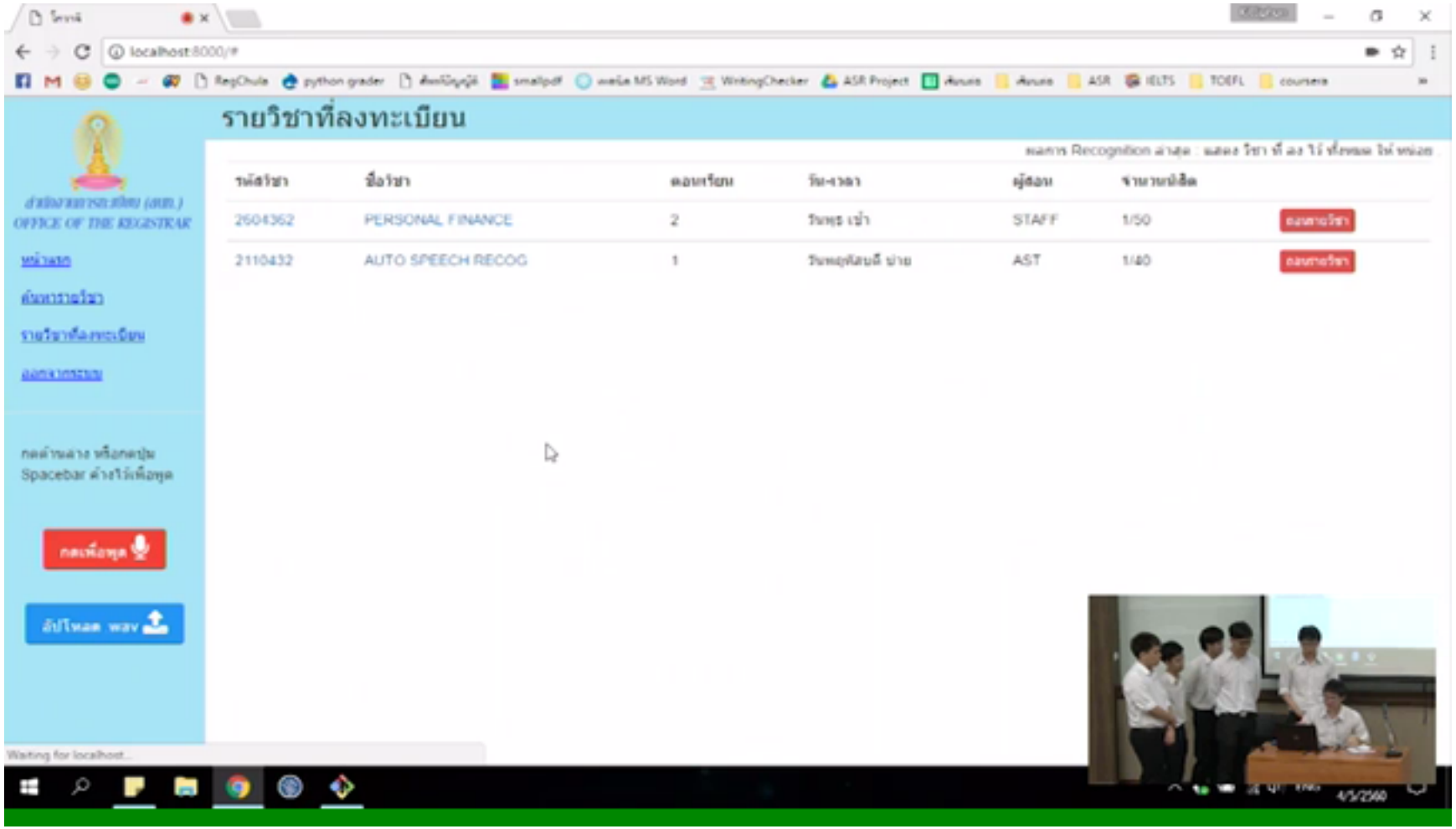
Speech related tasks

- Speaker recognition/verification
  - Who's speaking?
- Speaker diarization
  - Who speaks when?
- Language identification
- Emotion recognition
- Text-to-speech (TTS)
- Speech enhancement

ASR in robotics



ASR in applications



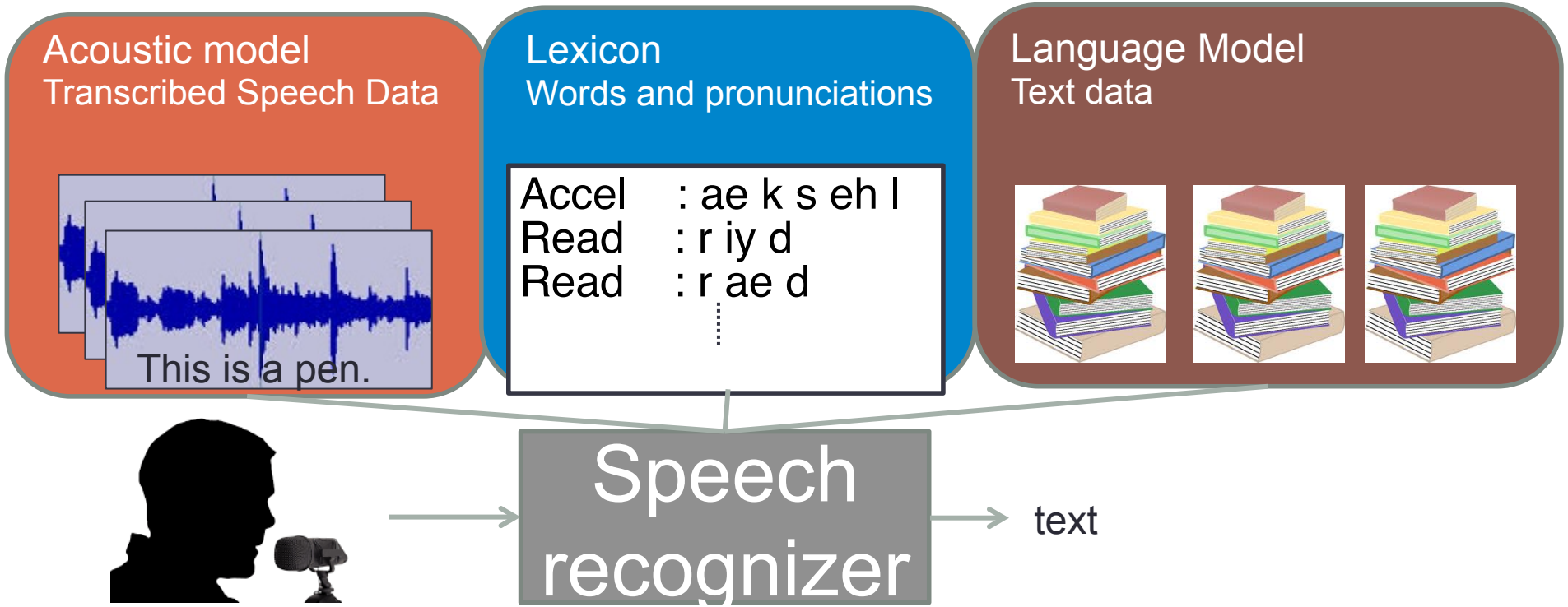
Speech applications

- Diagnosis
  - Depression, autism, …
- Monitoring
  - Team coordination, customer satisfaction
  - Keyword spotting
- Information retrieval and search
- Speech2speech translation

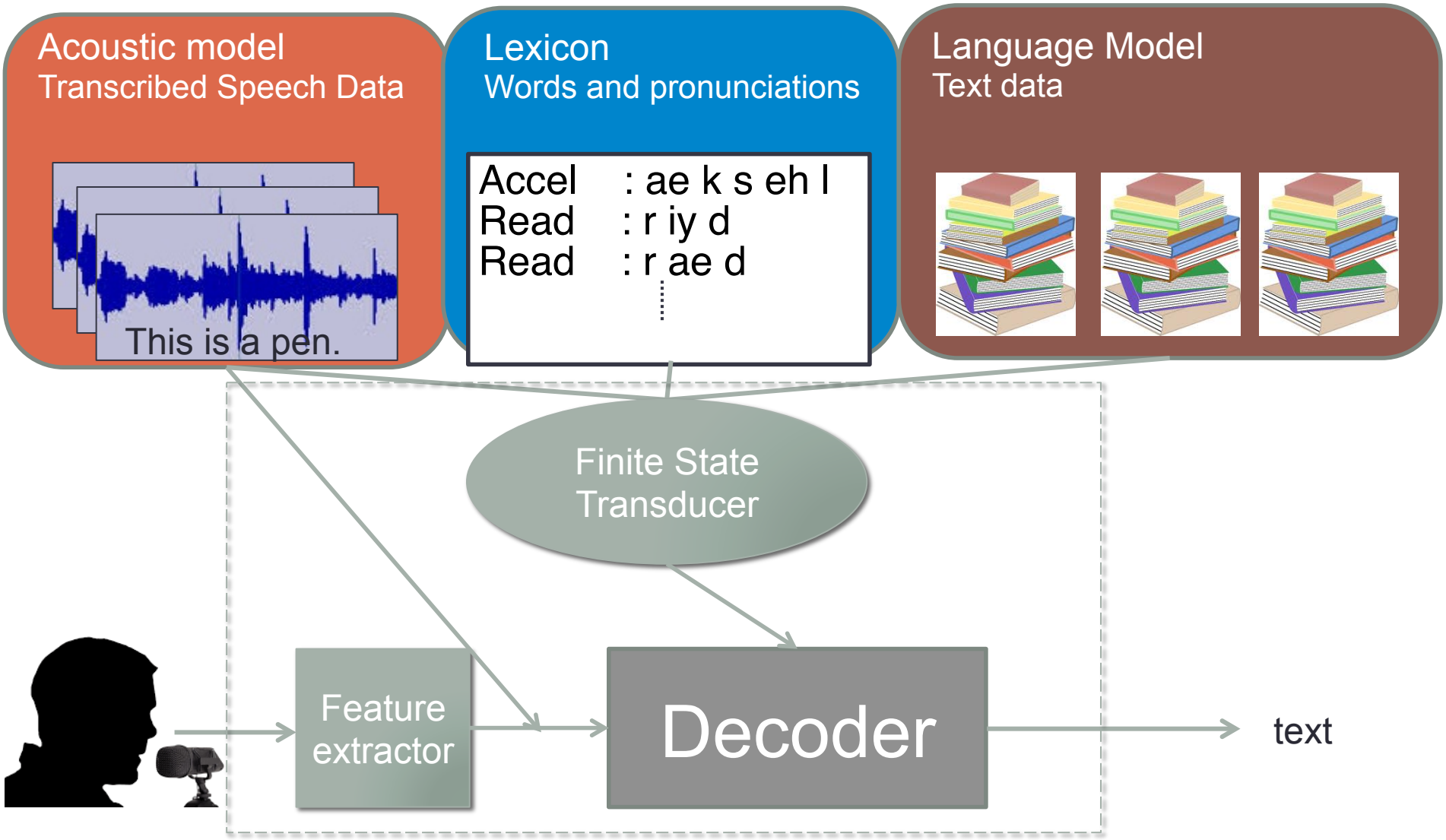
The ASR equation

$$= argmax_{W,L} P(X \mid L) P(L \mid W) P(W)$$

X - waveform, L - pronunciation, W - words

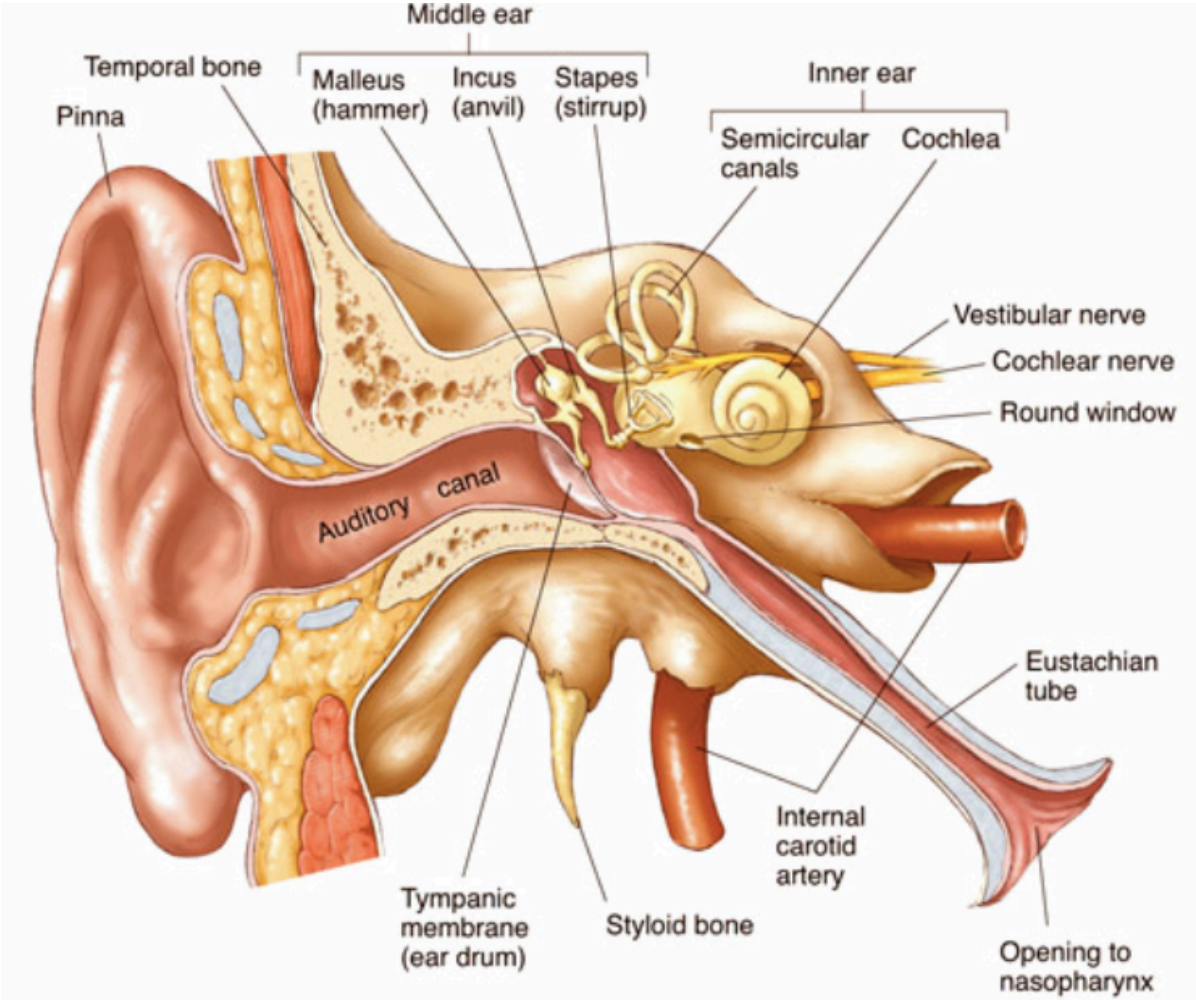


Inside the recognizer

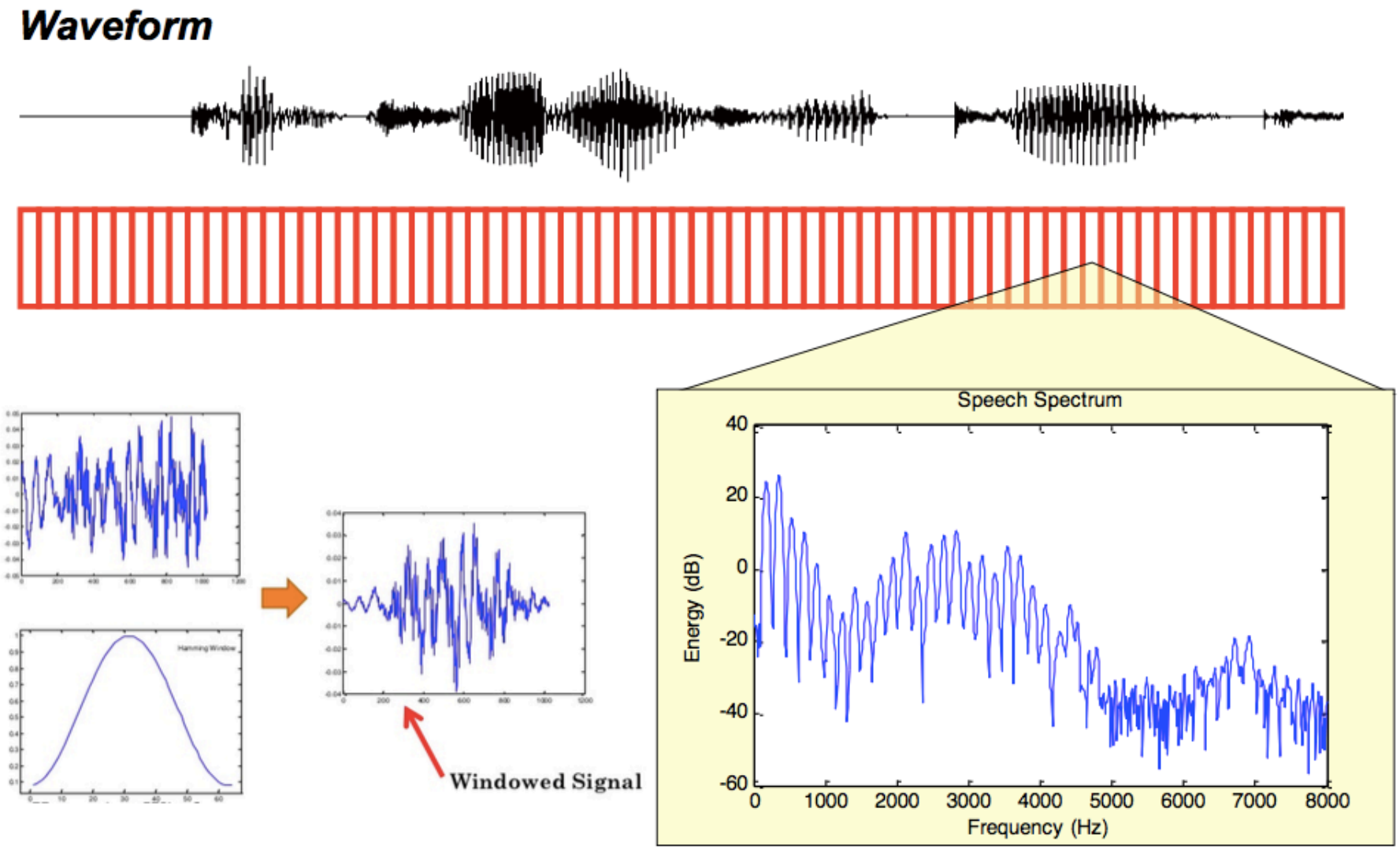


Feature extractor

- Motivation: copy the human ear

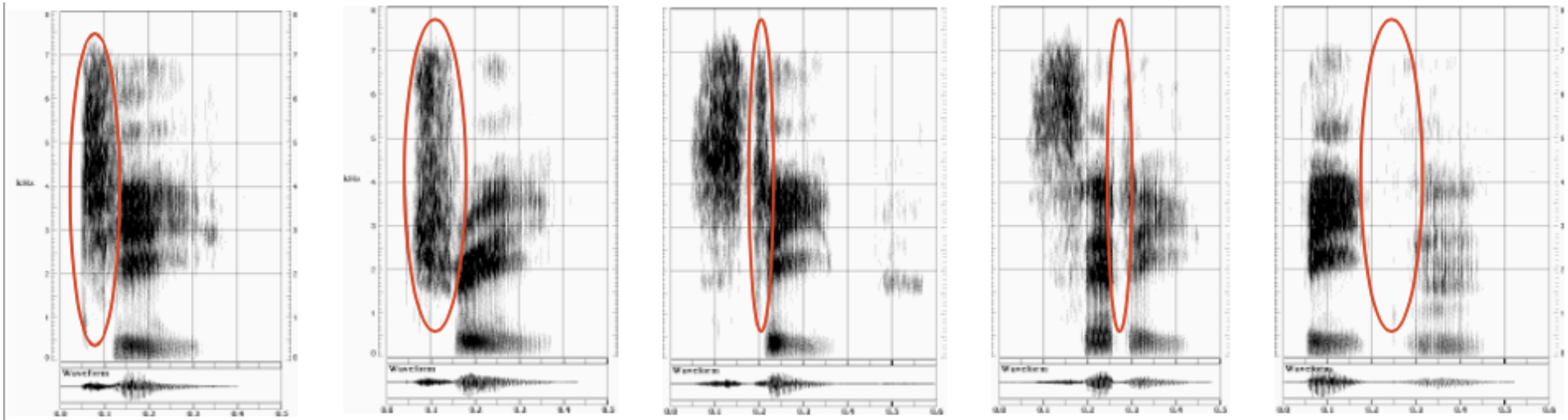


Feature extractor

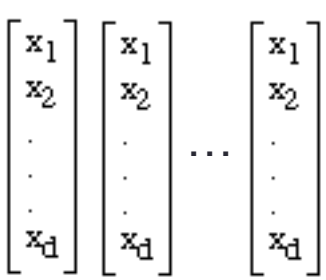


Feature extractor

T in various contexts



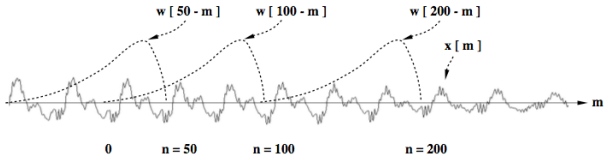
MFCC (Mel Frequency Cepstral Coefficient)



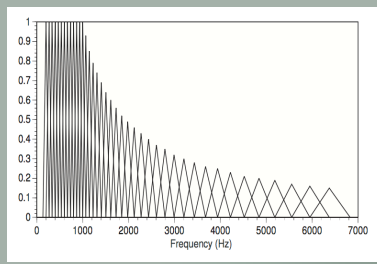
Select lower coefficients

DCT

log



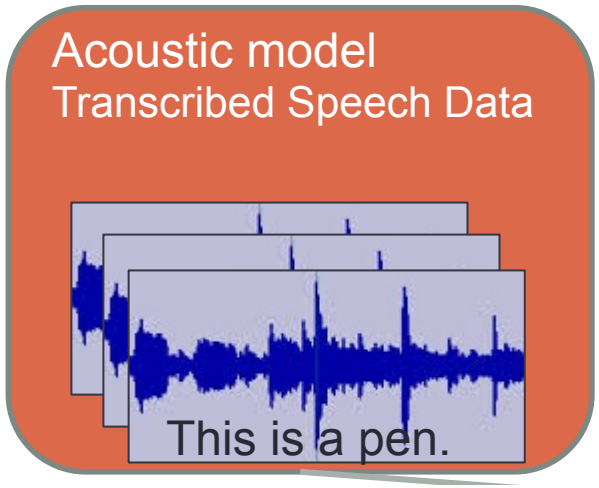
|STFT|

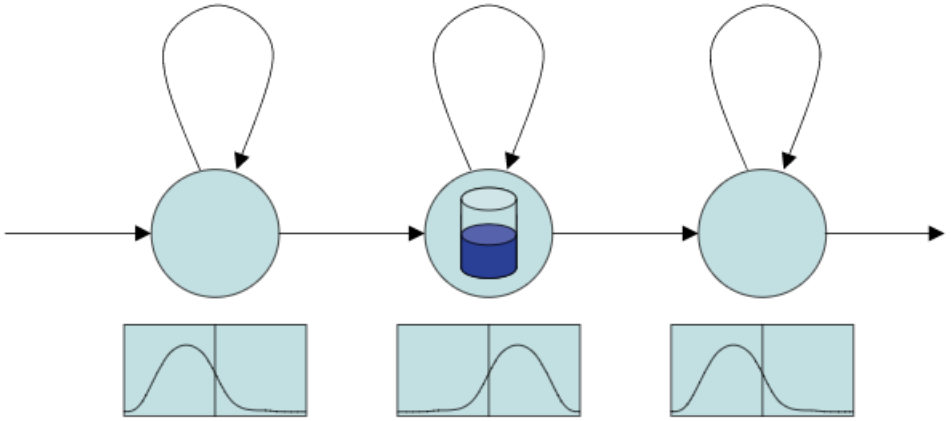


Acoustic model (AM)

- Models the sound of the language (phonemes)
- P(input MFCC frame | Phoneme)
  - P(X | า) vs P(X | ะ)

- Maps: sound to phoneme
- Typically use Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs)

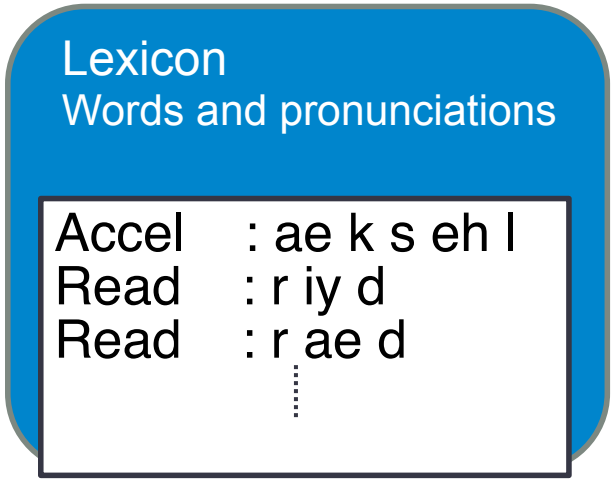




Lexicon

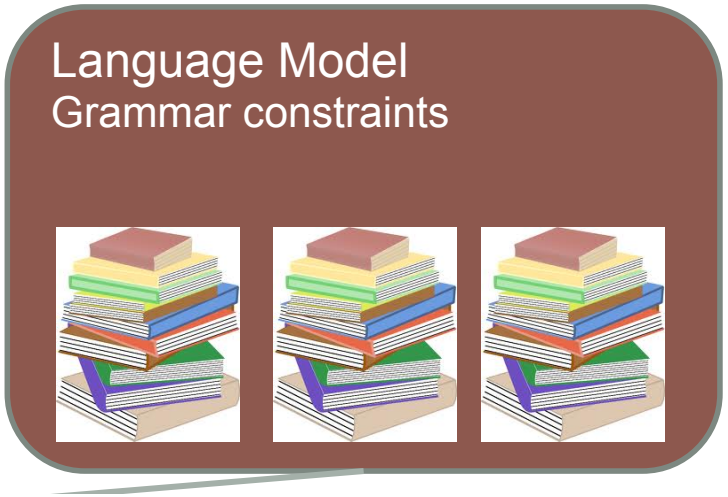
- Models the pronunciation of words in the language
- What pronunciations is possible
  - การ: ก า น
  - กบฏ: ก ะ บ โ-ะ ด

- Maps: phoneme to word



Language model (LM)

- Models the grammar or allowed commands
- Which word should follow which word
- $P(w_n \mid w_1,\ldots w_{n-1})$
  - P(apple | “I eat an”)
- Maps: word to sentence
- Typically use n-grams

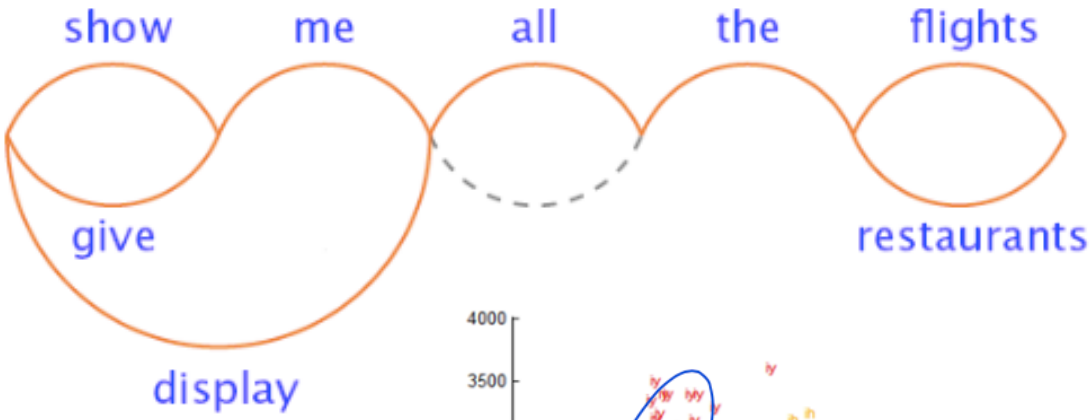


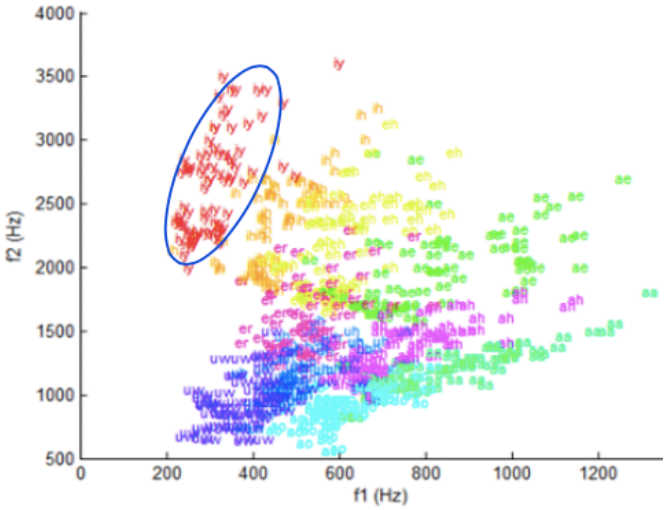
Search

- AM, Leixicon, and LM are constraints in a search problem

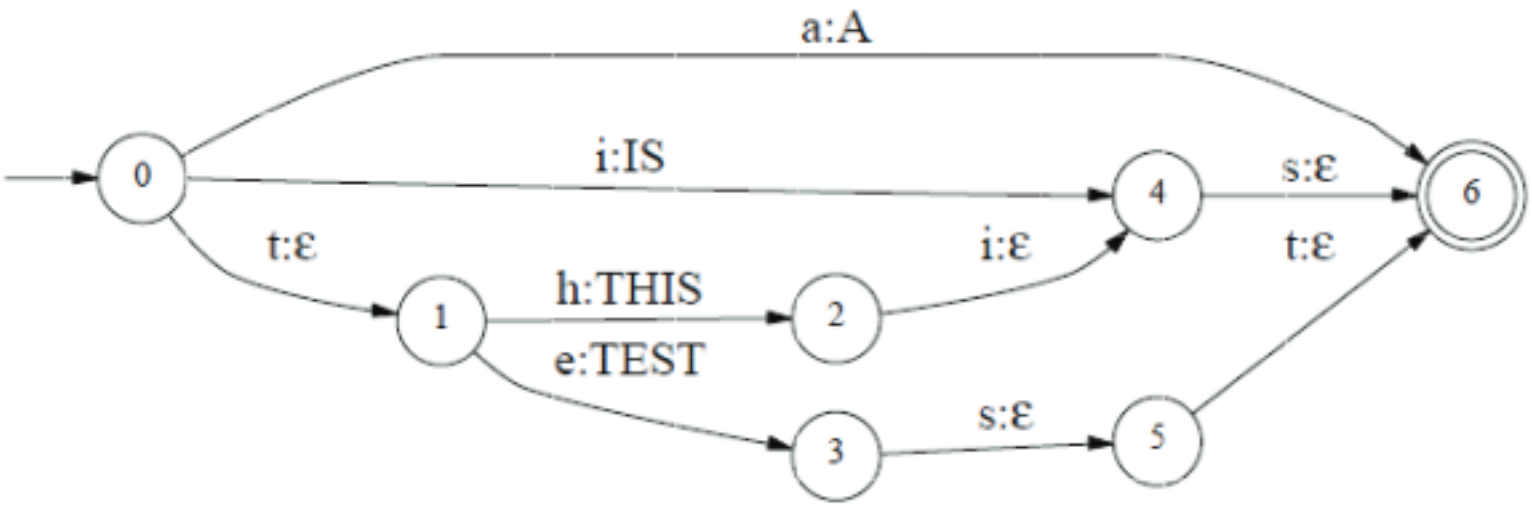
$$w^* = argmax_{W,L} P(X \mid L) P(L \mid W) P(W)$$

a	(ax | ey)
...	
beach	b iy ch
...	
nice	n (iy | ay) s
...	
recognize	r eh k ax gd n ay z
...	
speech	s p- iy ch
...	
stata	s t- (ey | aa) tf ax
...	
tomato	t ax m (ey | aa) tf ow
...	
wreck	r eh kd
...	



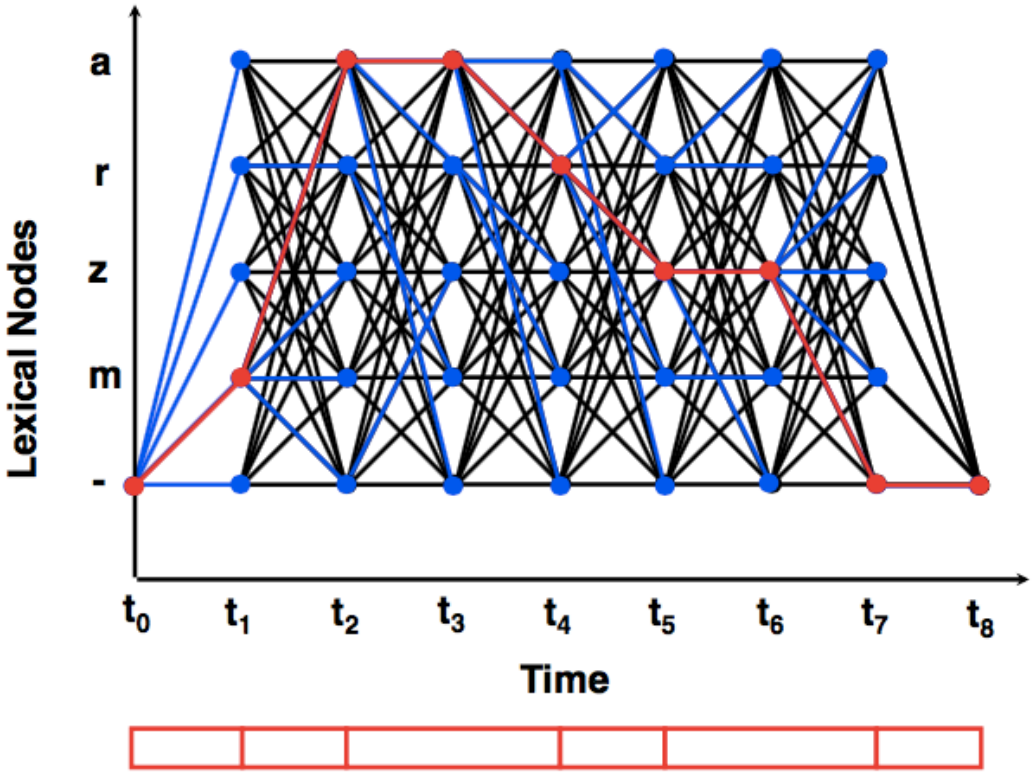


Decoding graph

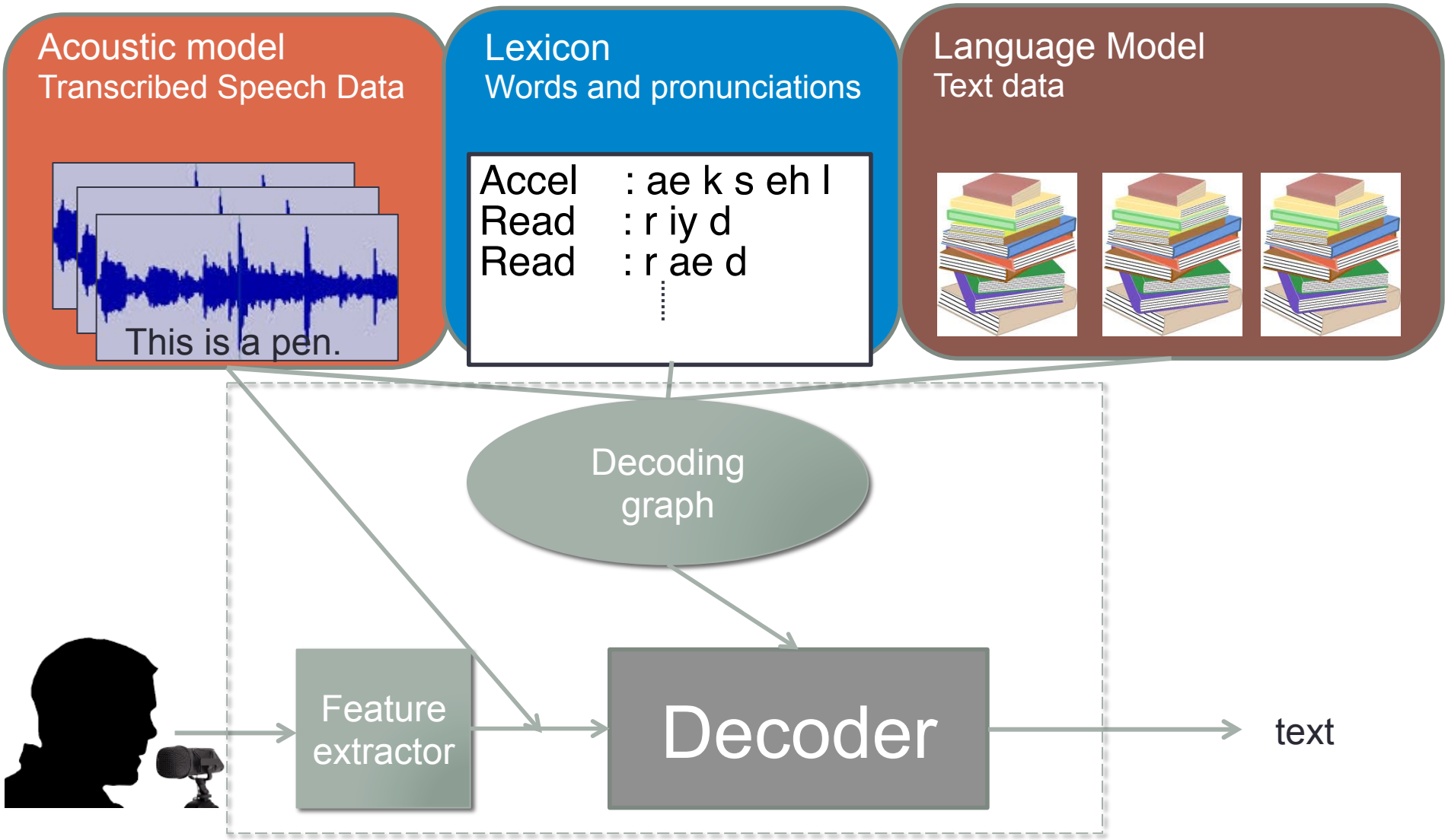


Viterbi search

- In order to speed up the search, dynamic programming is used – Viterbi algorithm



Inside the recognizer



NEURAL NETWORKS

DNNs (Deep Neural Networks)

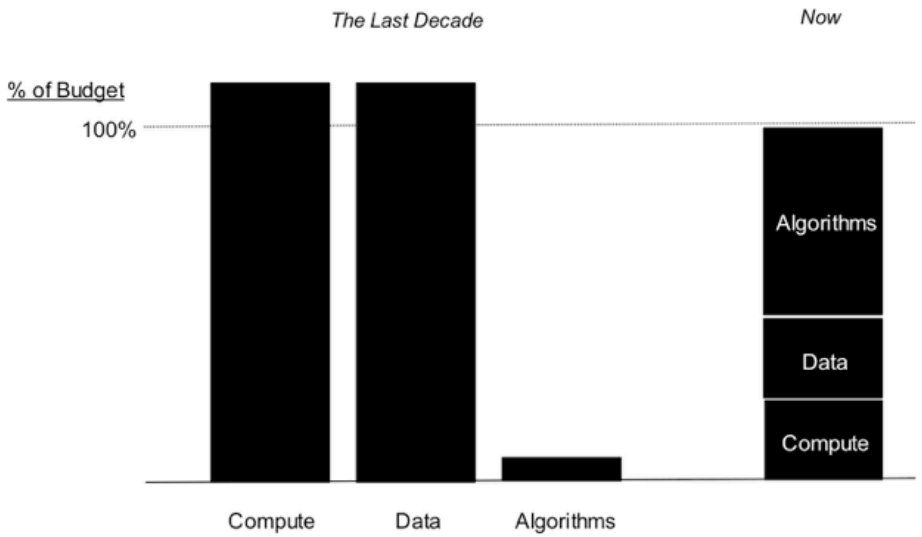
- Why deep learning?
- Greatly improved performance in ASR and other tasks (Computer Vision, Robotics, Machine Translation, NLP, etc.)
- Surpassed human performance in many tasks

Task	Previous state-of-the-art	Deep learning (2012)	Deep learning (2017)
TIMIT	24.4%	20.0%	17.0%
Switchboard	23.6%	16.1%	5.5%
Google voice search	16.0%	12.3%	4.9%

Geoffrey Hinton, Deep Neural Networks for Acoustic Modeling in Speech Recognition, 2012

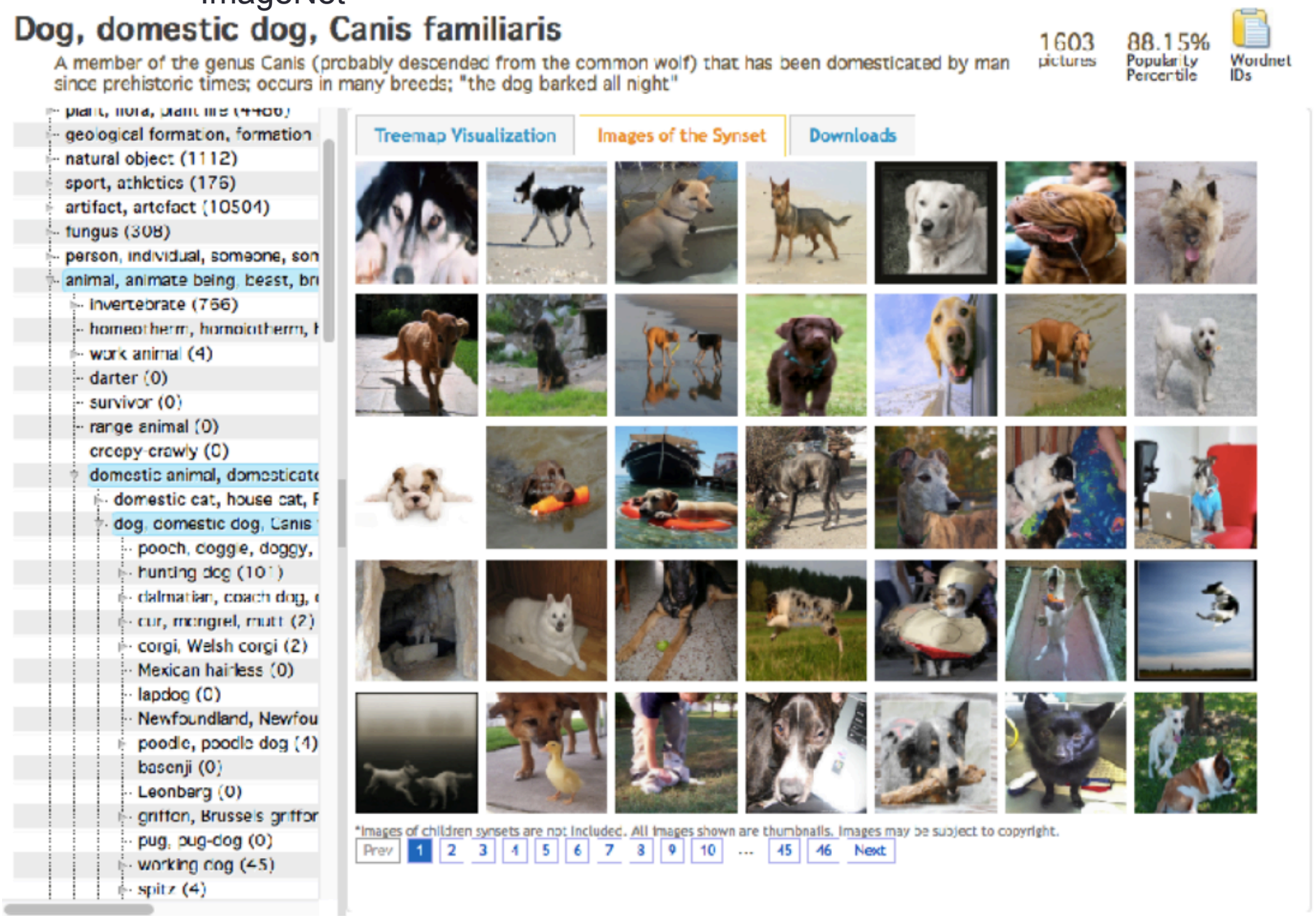
Why now

- Neural Networks has been around since 1990s
- Big data – DNN can take advantage of large amounts of data better than other models
- GPU – Enable training bigger models possible
- Deep – Easier to avoid bad local minima when the model is large



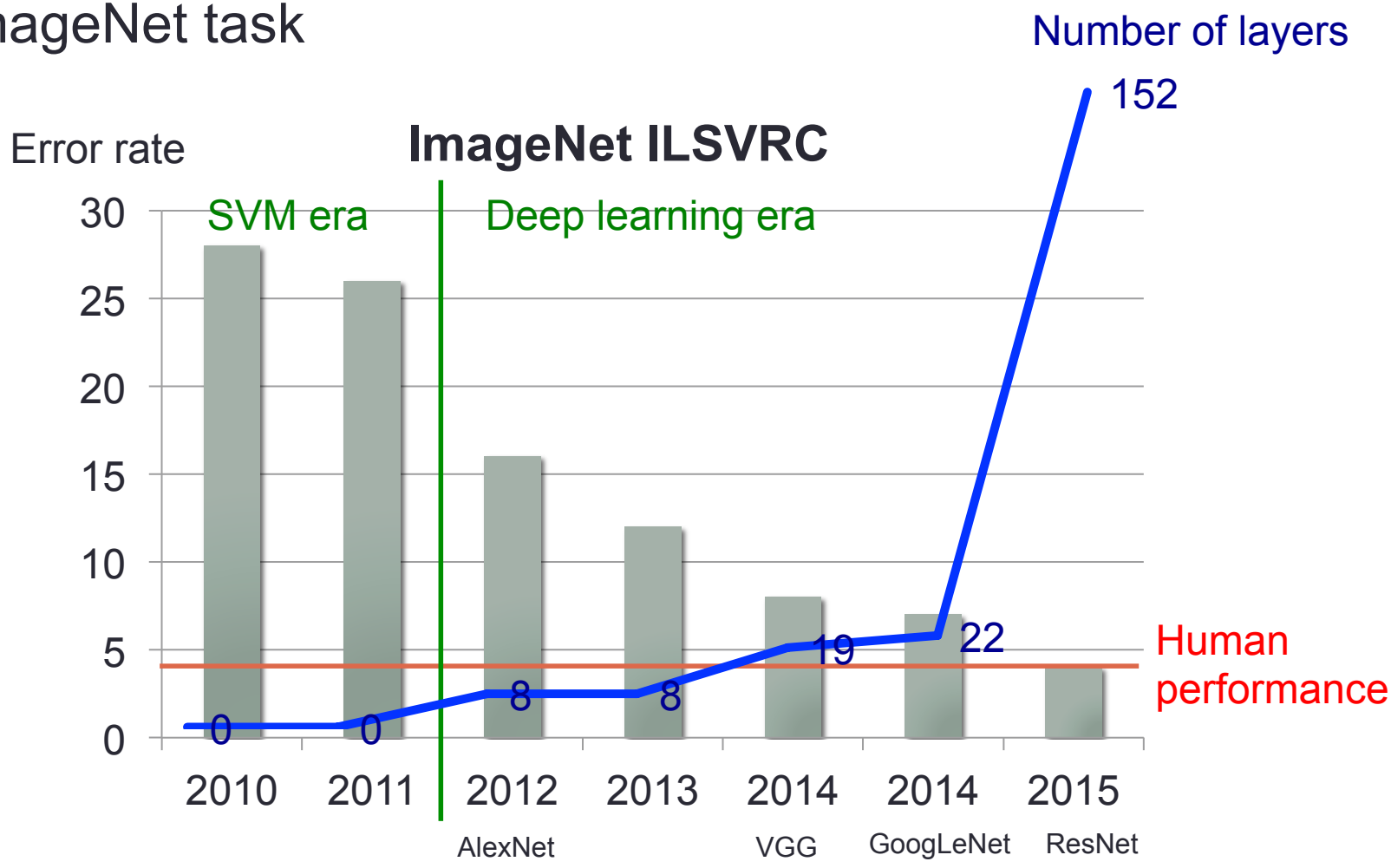
http://www.kdnuggets.com/2017/06/practical-guide-machine-learning-understand-differentiate-apply.html

ImageNet


Dog, domestic dog, Canis familiaris
A member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds; "the dog barked all night"
1603 pictures
88.15% Popularity Percentile
Wordnet IDs
geological formation, formation
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
fungus (308)
person, individual, someone, son
animal, animate being, beast, br
invertebrate (756)
work animal (4)
darter (0)
survivor (0)
range animal (0)
creepy-crawly (0)
domestic animal, domesticate
dog, domestic dog, Canis
pooch, doggie, doggy,
hunting dog (10)
corgi, Welsh corgi (2)
Mexican hairless (0)
lapdog (0)
Newfoundland, Newfou
poodle, poodle dog (4)
basenji (0)
Leonberg (0)
pug, pug-dog (0)
working dog (45)
spitz (4)
Treemap Visualization
Images of the Synset
Downloads
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.
Prev 1 2 3 4 5 6 7 8 9 10 ... 45 46 Next

Wider and deeper networks

- ImageNet task



Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

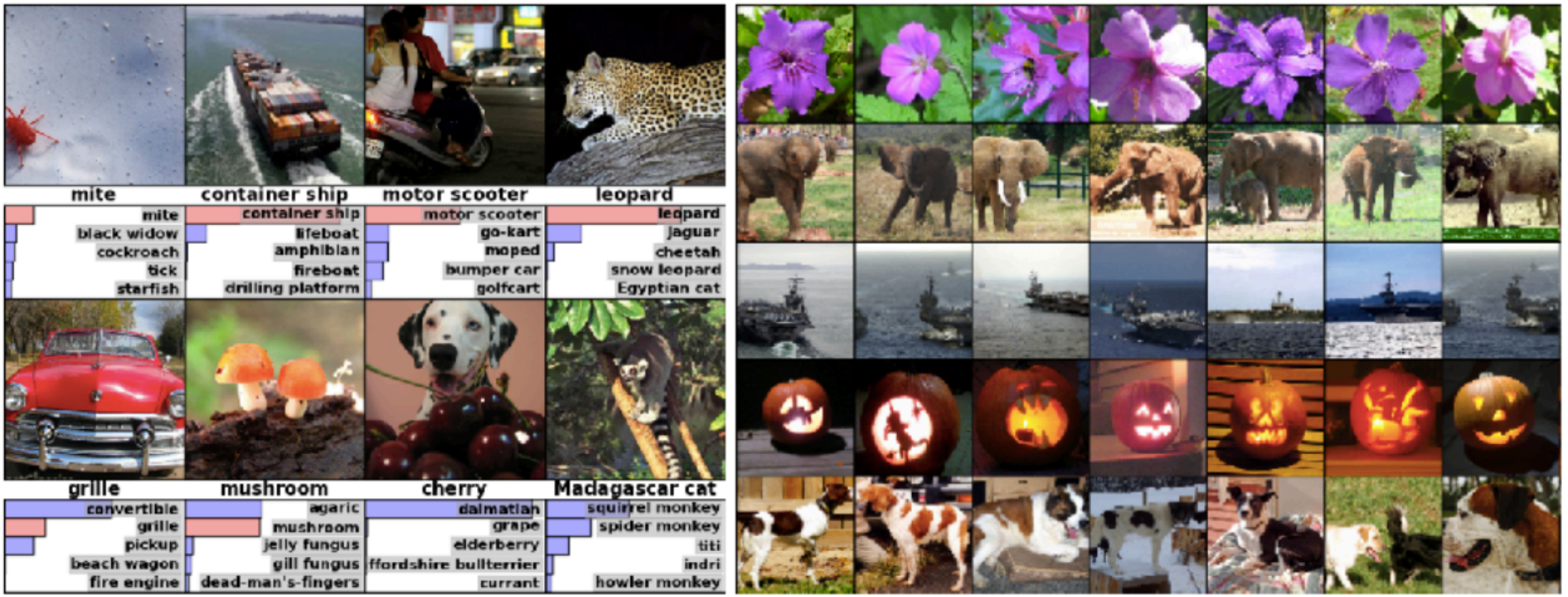
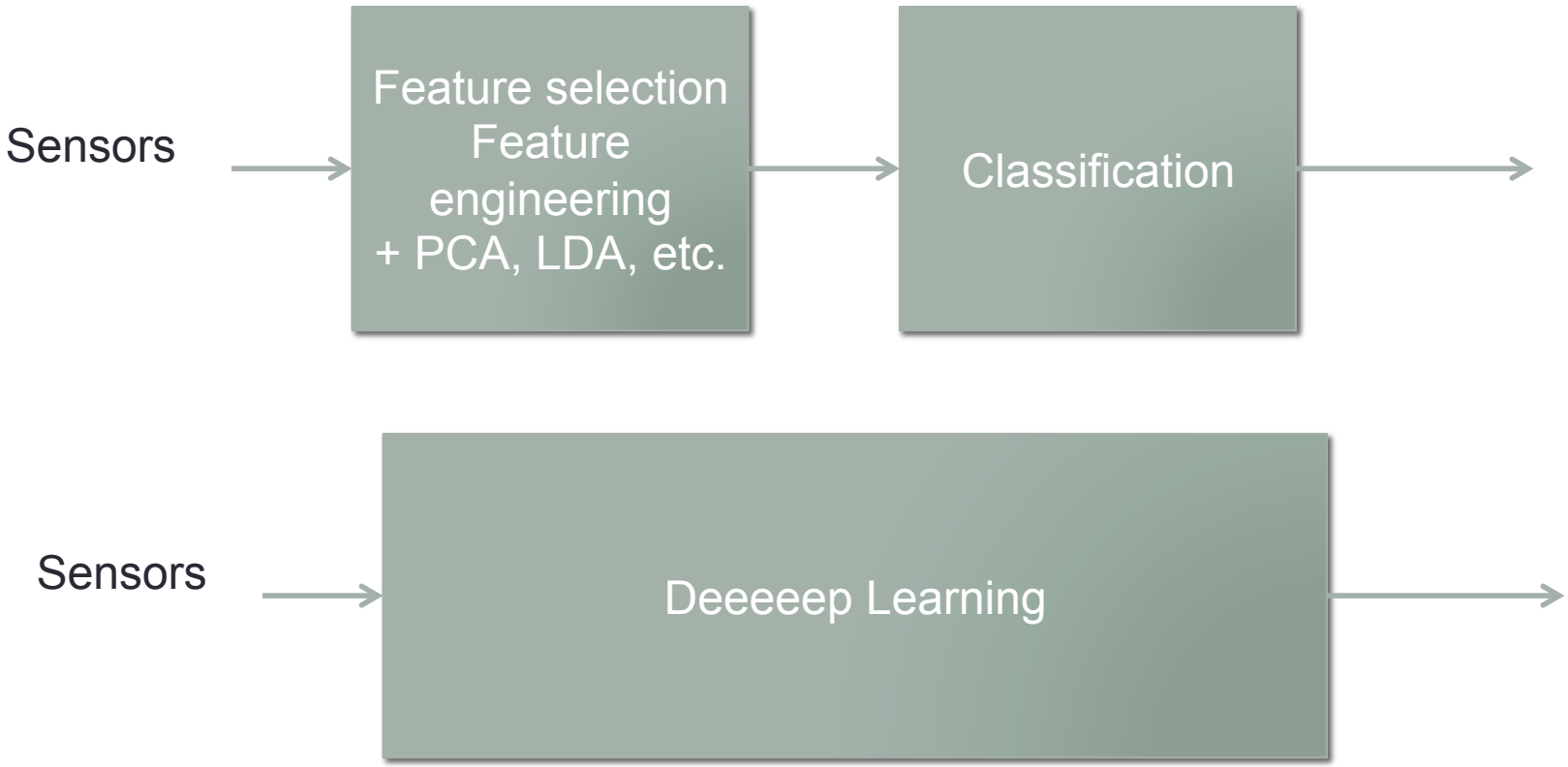


Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

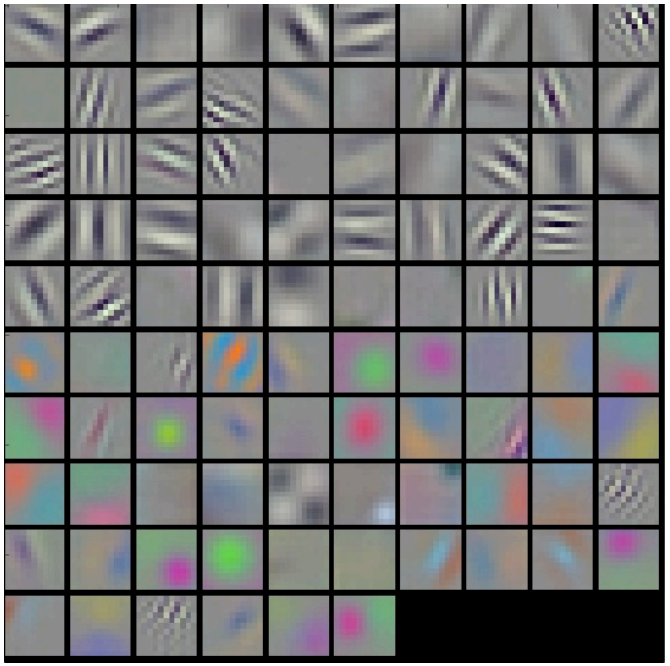
Why is deep learning good

- Traditional machine learning approaches need feature engineering
  - Features are based on human understanding of the phenomena
  - Have some simplifying assumptions
- Human knowledge captures known knowns, and Known unknowns NOT unknown unknowns
- **Deep learning combines features engineering and modeling into one single optimization problem**

Traditional VS Deep learning



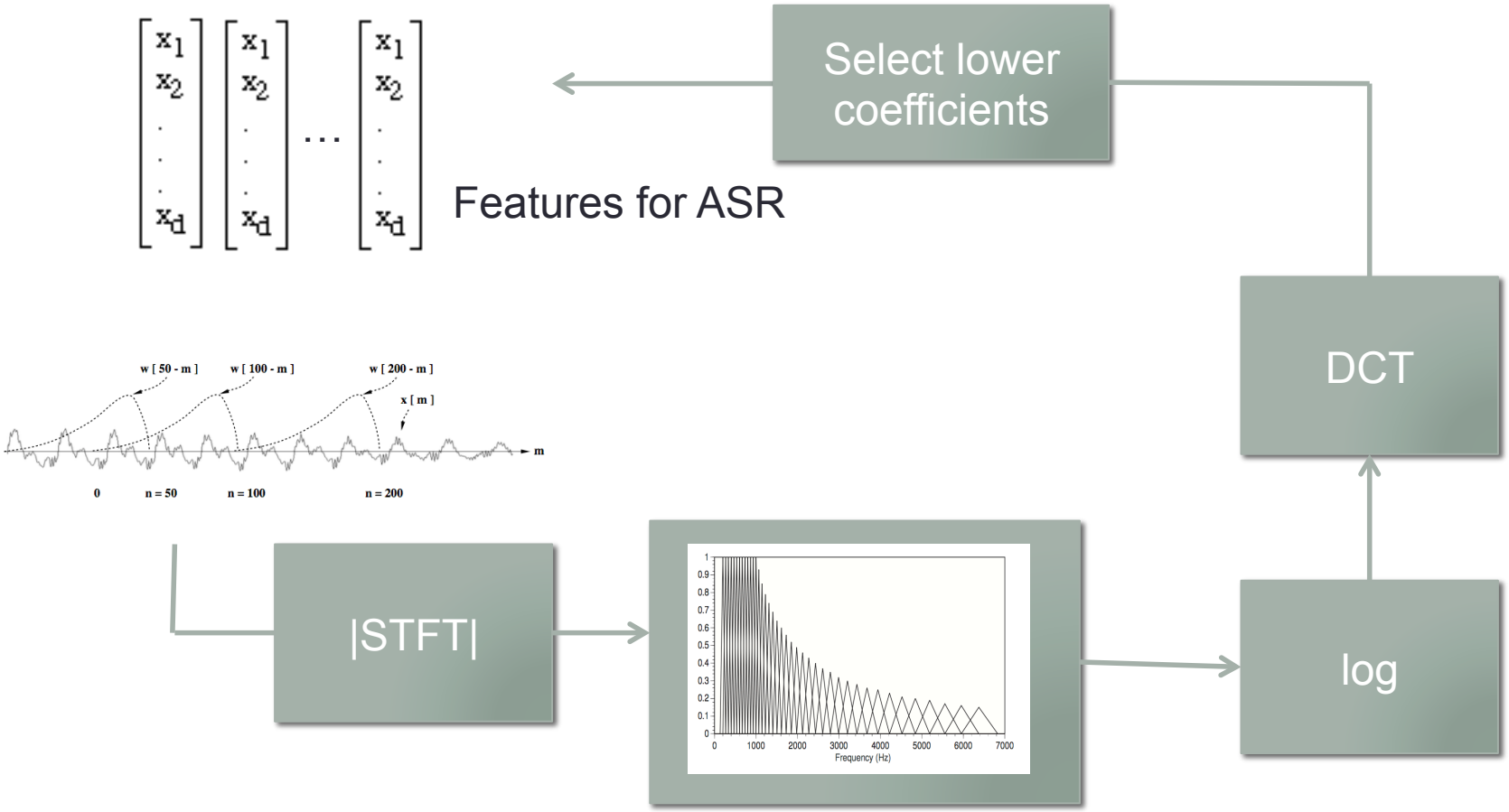
Learning representations



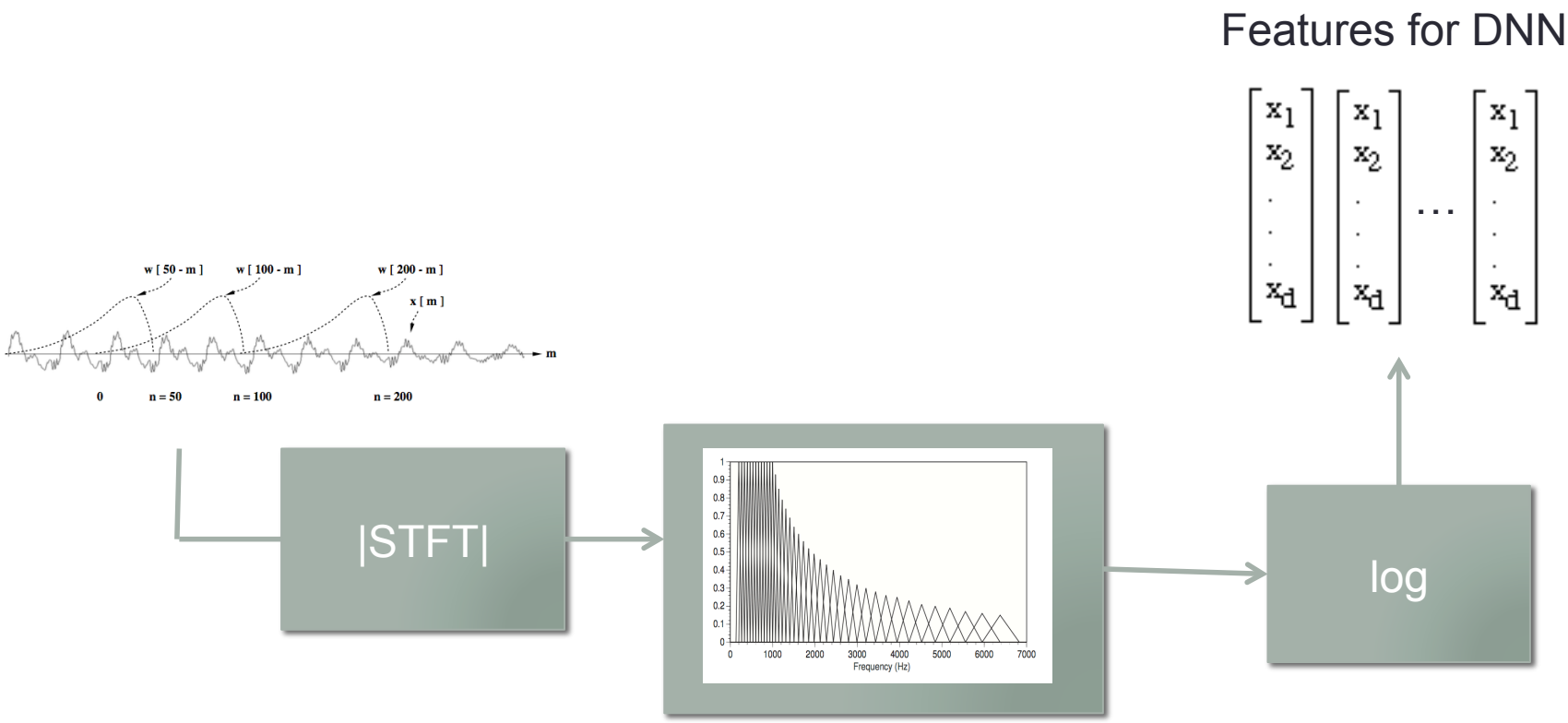
Features learned from a CNN for a vision task.
Inputs are now raw pixels instead of descriptors.

http://cs231n.github.io/understanding-cnn/

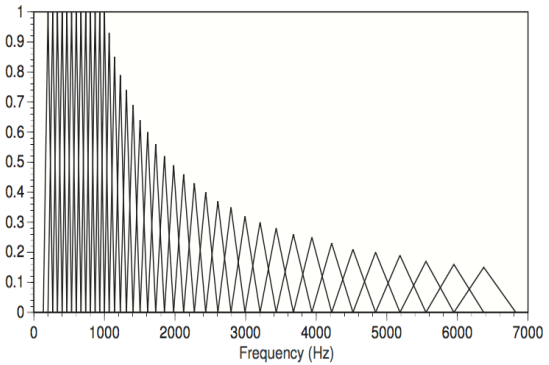
MFCC (Mel Frequency Cepstral Coefficient)



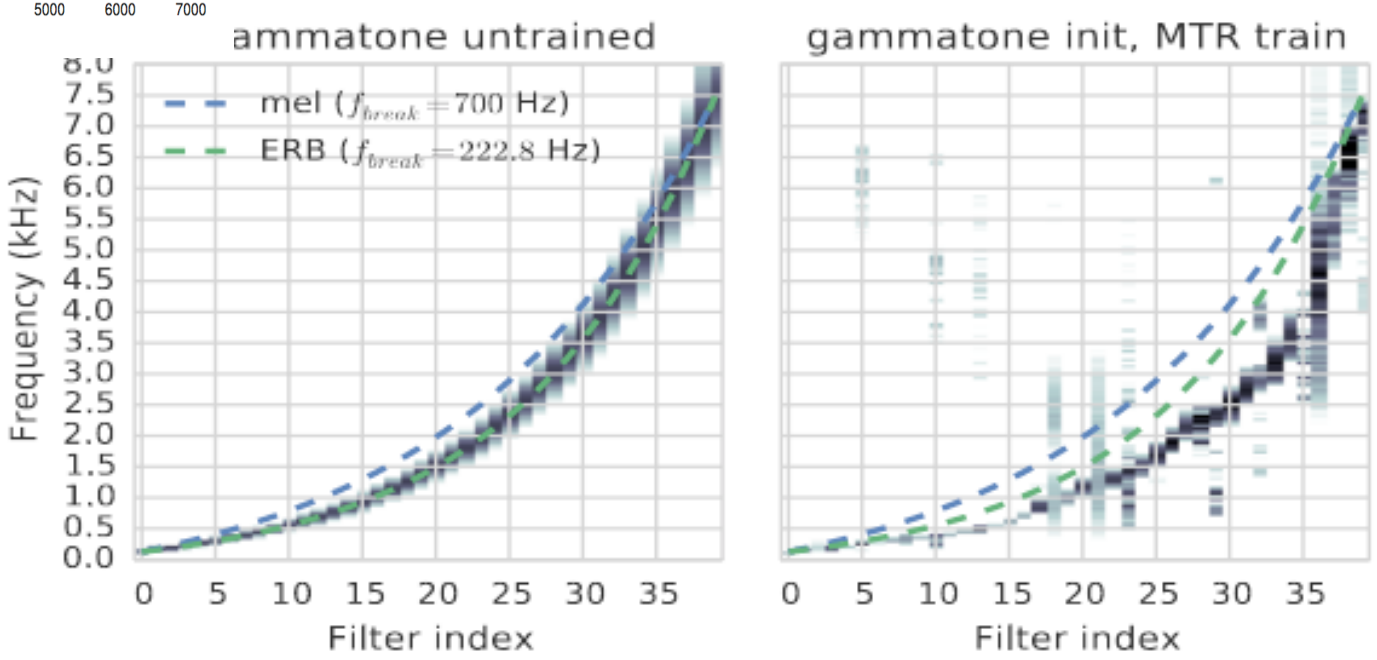
MFCC (Mel Frequency Cepstral Coefficient)



Learning the Mel filters



Filter type	WER
DNN+Random	16.4
DNN+Gammatone (untrained)	16.4
DNN+Gammatone (trained)	16.2



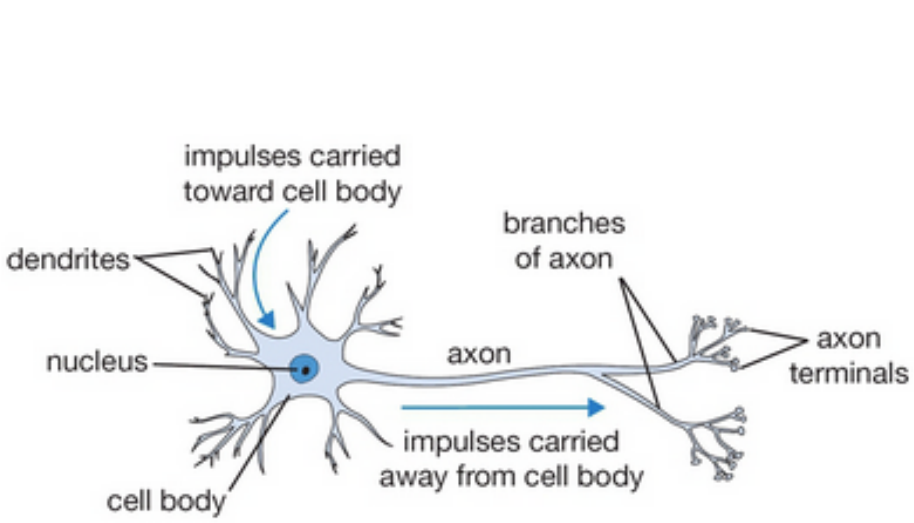
Learning the Speech Front-end with Raw Waveform CLDNNs, 2015

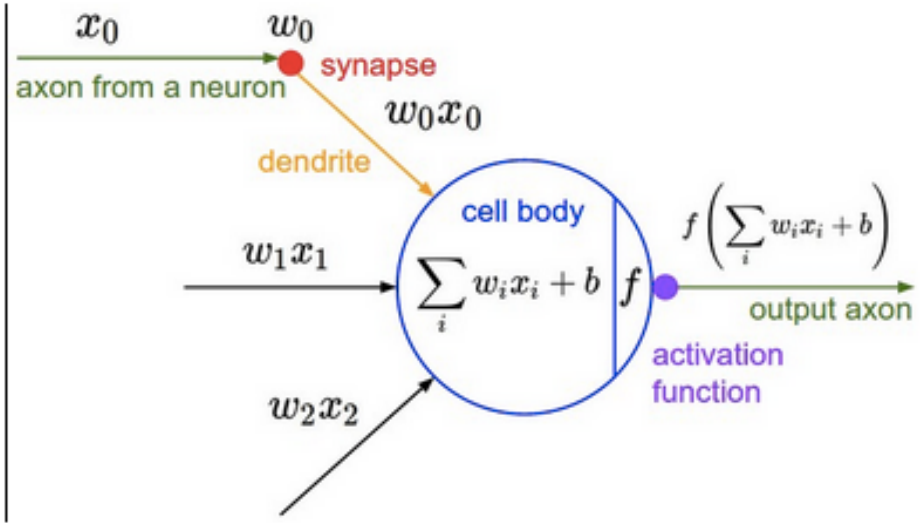
Neural networks

- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function
  - SGD and backprop
- CNN, RNN, LSTM, GRU

Fully connected networks

- Many names: feed forward networks or deep neural networks or multilayer perceptron or artificial neural networks
- Composed of multiple neurons

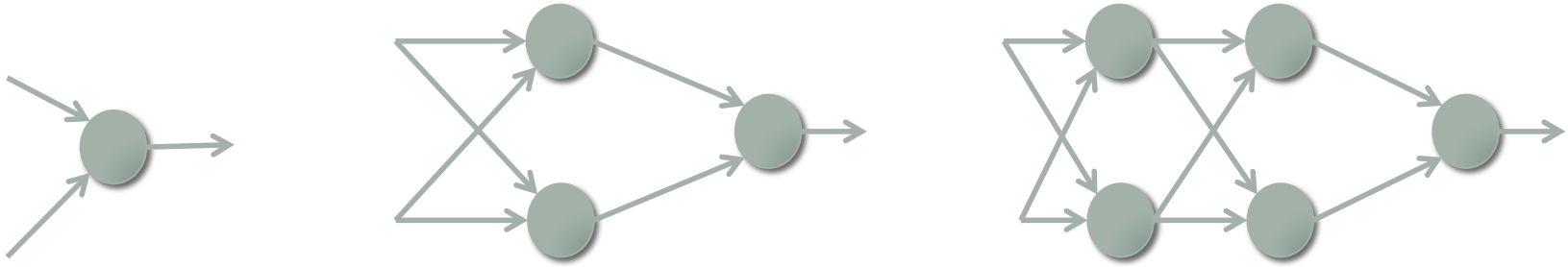


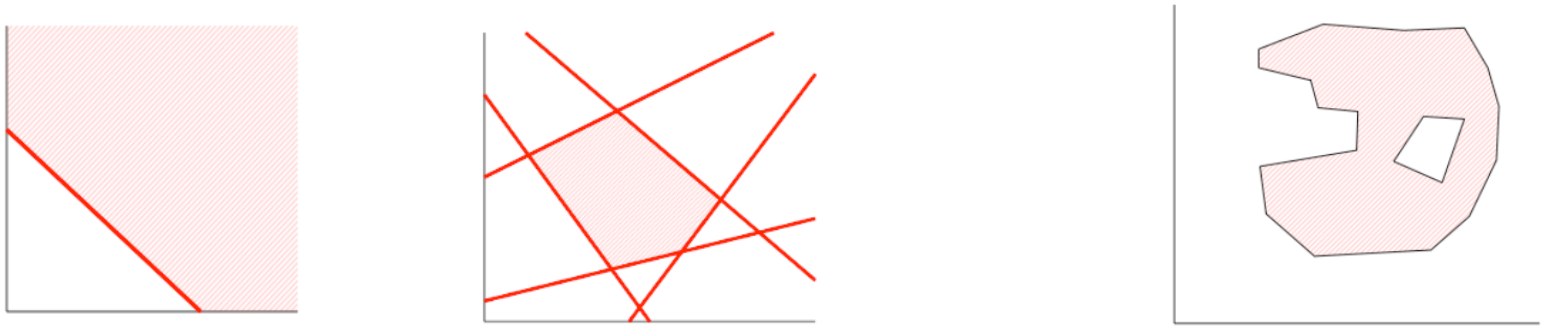


http://cs231n.github.io/neural-networks-1/

Combining neurons

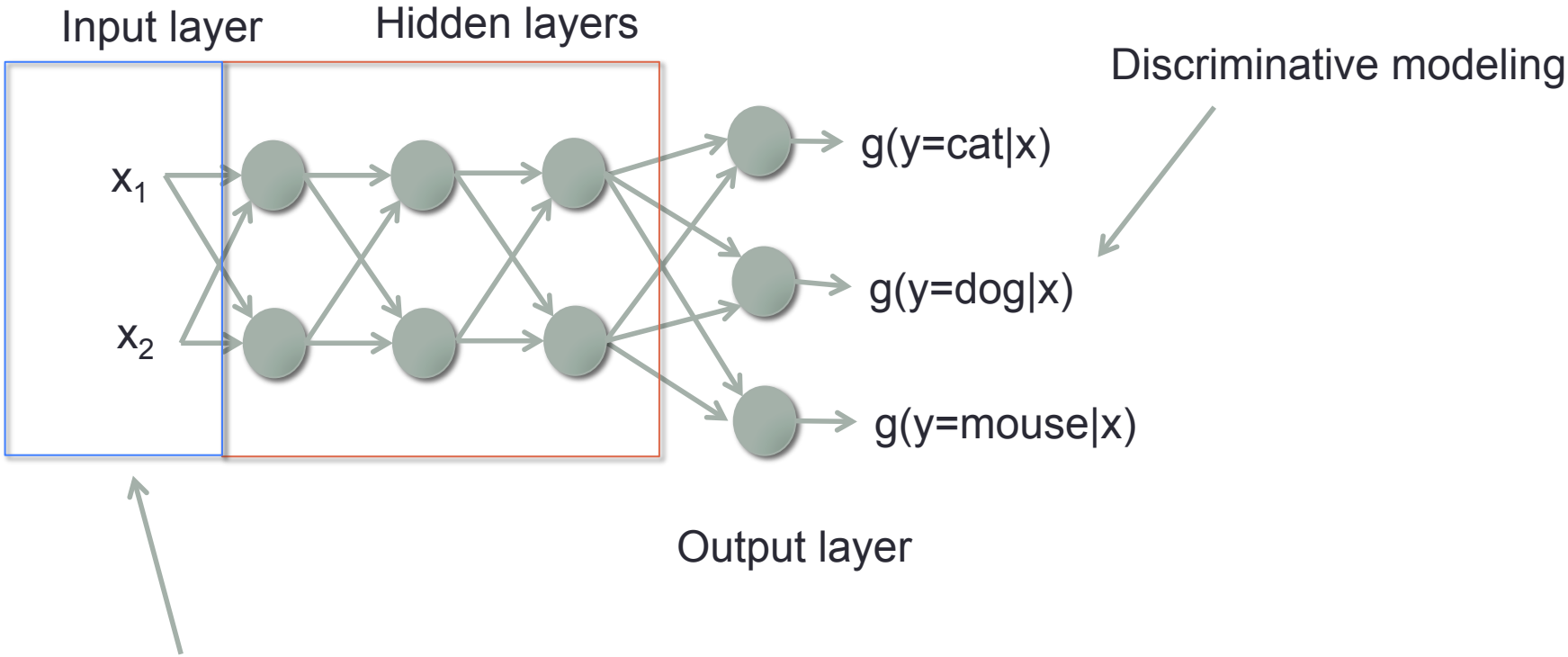
- Each neuron splits the feature space with a hyperplane
- Stacking neuron creates more complicated decision boundaries
- More powerful but prone to overfitting





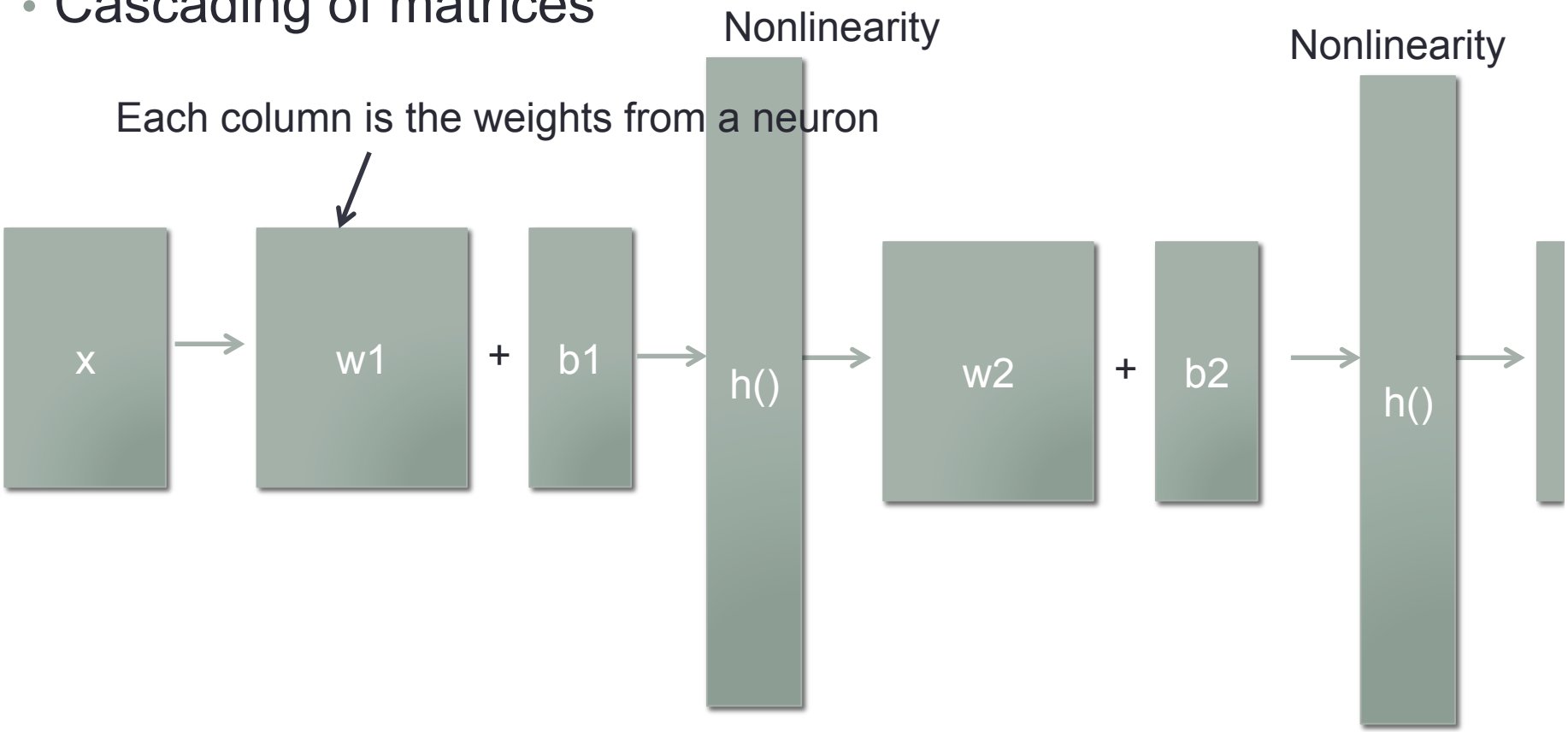
Terminology

Deep in Deep neural networks means many hidden layers



More linear algebra

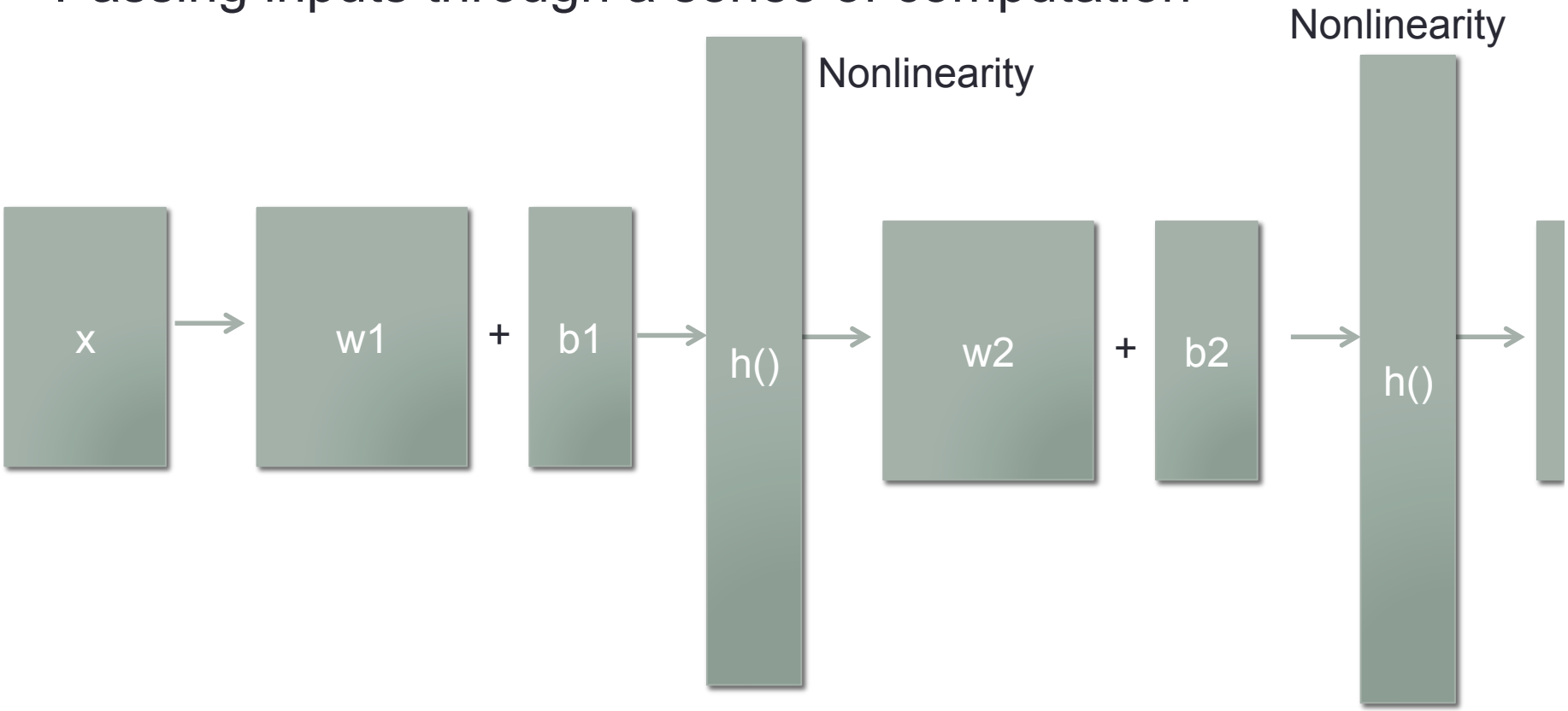
- Cascading of matrices



$h(W_2^T\, h(W_1^T X + \mathbf{b_1}) + \mathbf{b_2})$

Computation graph

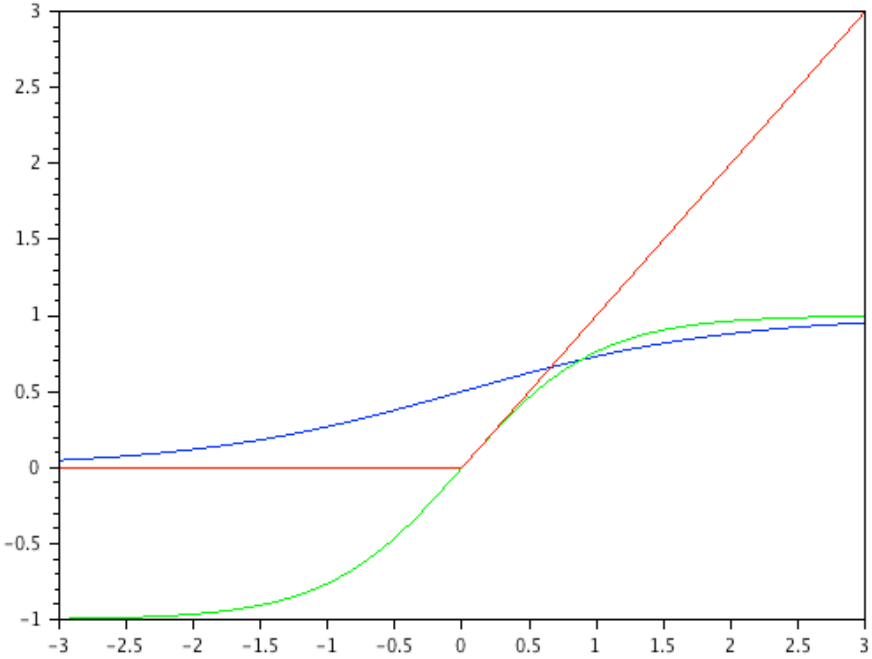
- Passing inputs through a series of computation



$h(W_2^T\, h(W_1^T X + \mathbf{b_1}) + \mathbf{b_2})$

Non-linearity

- The Non-linearity is important in order to stack neurons
  - If F is linear, a multi layered network can be collapsed as a single layer (by just multiplying weights together)
- Sigmoid or logistic function
- tanh
- Rectified Linear Unit (ReLU)
- Most popular is ReLU and its variants (Fast to train, and more stable)



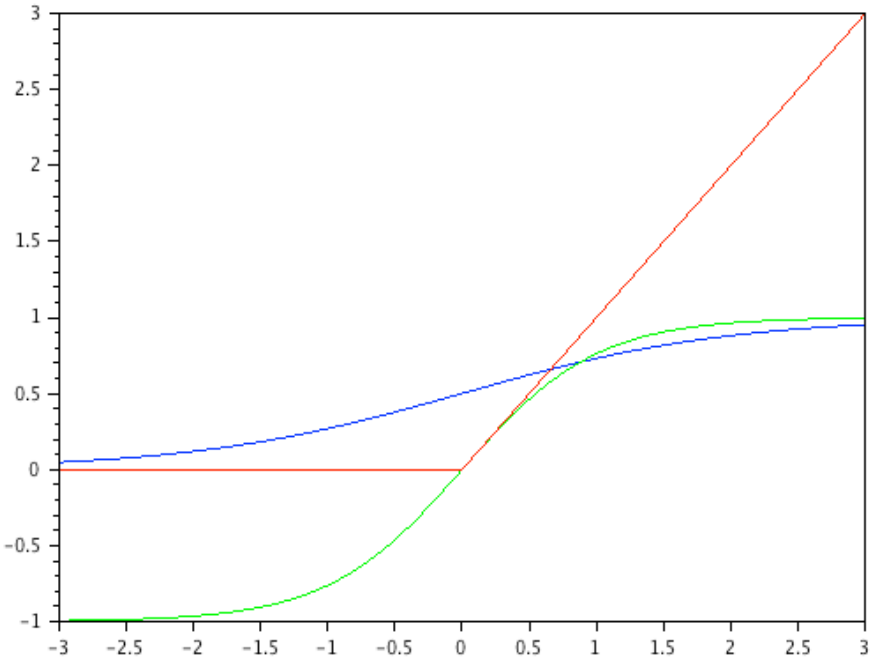
Non-linearity

- Sigmoid $\frac{1}{1+e^{-x}}$
- tanh

$$tanh(x)$$

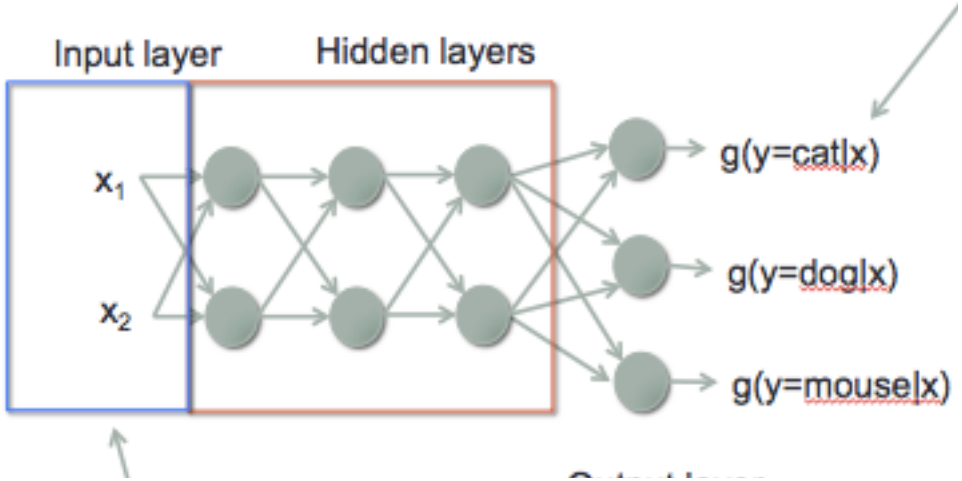
- Rectified Linear Unit (ReLU)

$$max(0, x)$$



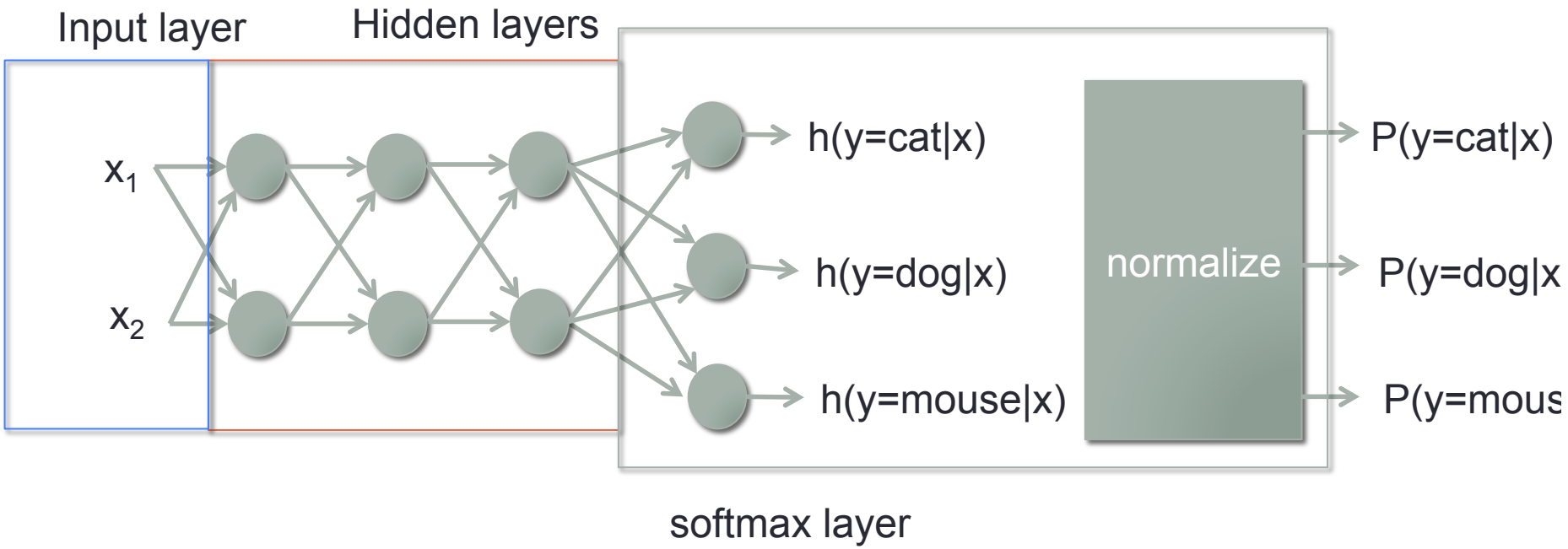
Output layer – Softmax layer

- We usually wants the output to mimic a probability function (0<=P<=1,sums to 1)
- Current setup has no such constraint
- The current output should have highest value for the correct class.
  - Value can be positive or negative number
- Takes the exponent
- Add a normalization



Softmax layer

$$P(y=j|x) = \frac{e^{h(y=j|x)}}{\sum_y e^{h(y|x)}}$$

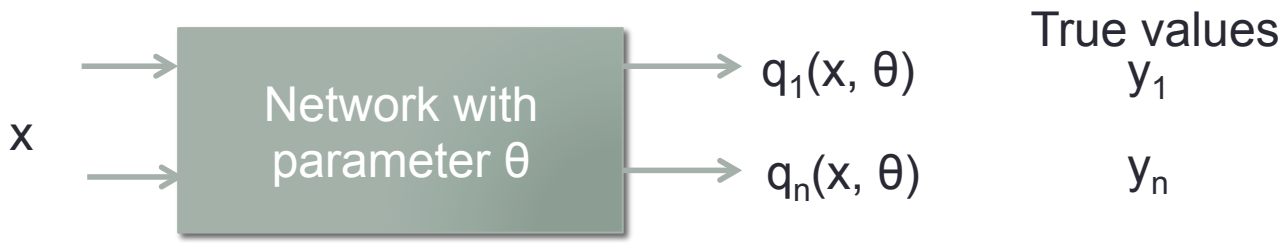


Neural networks

- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function
  - SGD and backprop
- CNN, RNN, LSTM, GRU

Objective function (Loss function)

- Can be any function that summarizes the performance into a single number
- Two main loss functions
  - Cross entropy
  - Sum of square errors



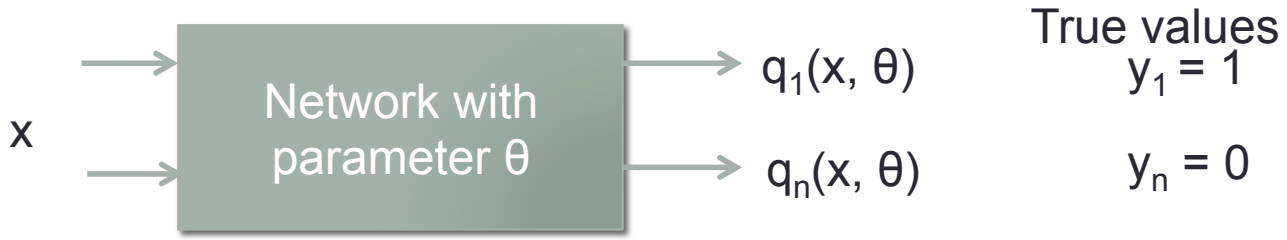
Cross entropy loss

- Used for softmax outputs (probabilities), or classification tasks

$$L = -\Sigma_n y_n log q_n(x, \theta)$$

- Where y_n is 1 if data x comes from class n
  0 otherwise

- L only has the term from the correct class
- L is non negative with highest value when the output matches the true values, a “loss” function

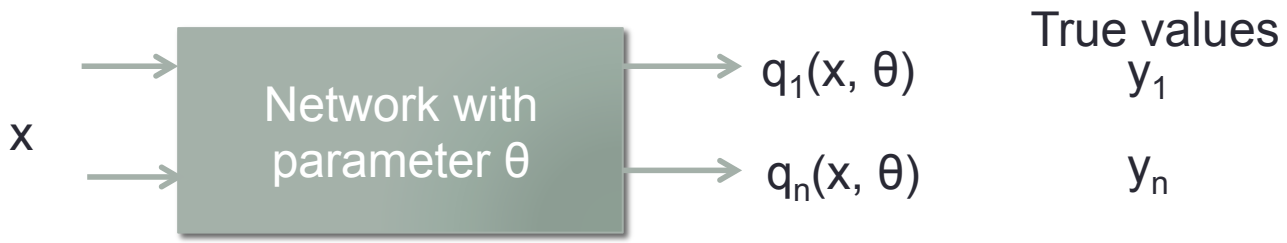


Sum of square errors

- Used for any real valued outputs such as regression

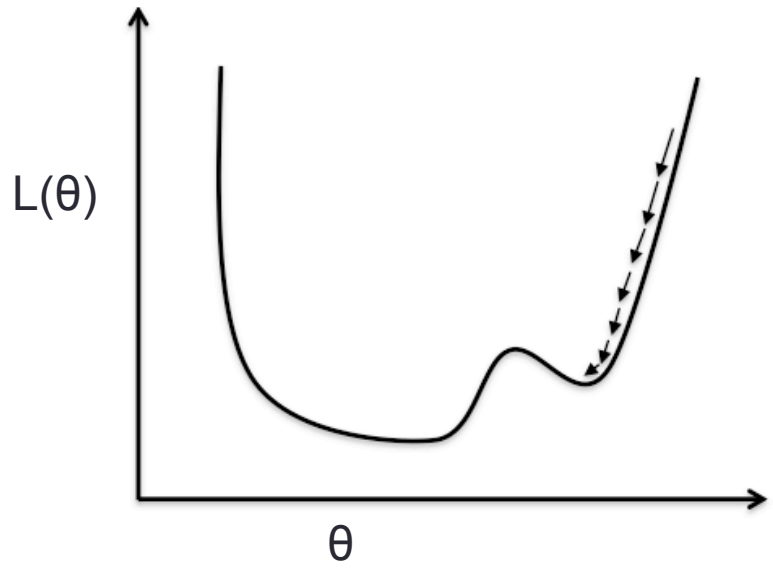
$$L = \frac{1}{2}\Sigma_n(y_n - q_n(x,\theta))^2$$

- Non negative, the better the lower the loss



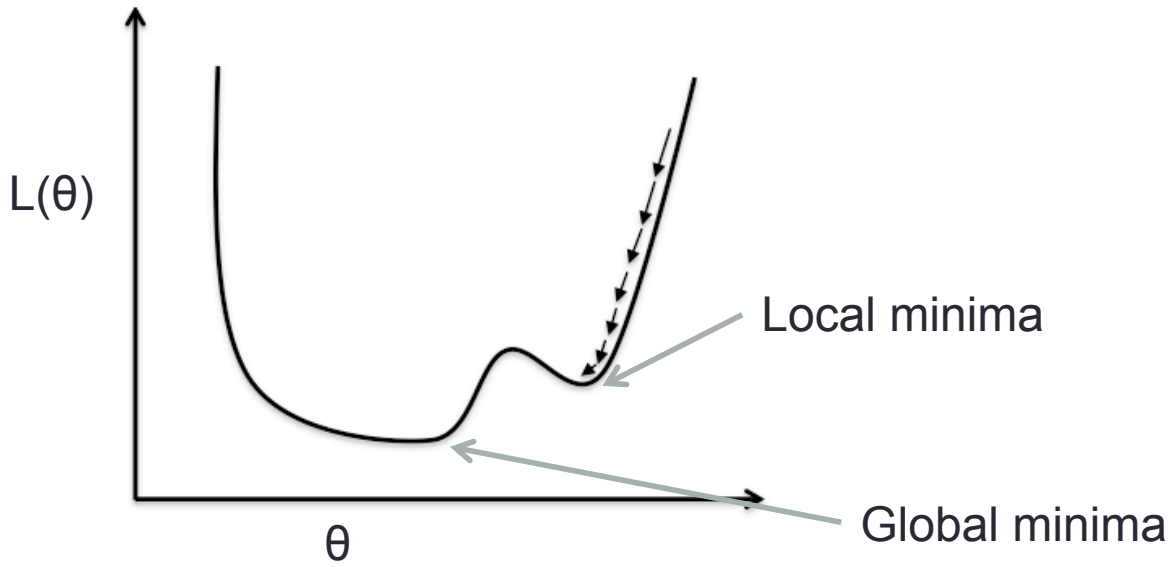
Minimization using gradient descent

- We want to minimize L with respect to θ (weights and biases)
  - Differentiate with respect to θ
  - Gradients passes through the network by Back Propagation



Deep vs Shallow

- If deep, you can avoid most local minima



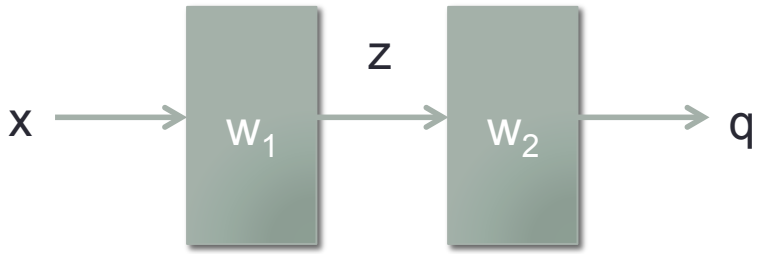
Kenji Kawaguchi,Deep Learning without Poor Local Minima, 2016 https://arxiv.org/abs/1605.07110

Differentiating a neural network model

- We want to minimize loss by gradient descent
- A model is very complex and have many layers! How do we differentiate this!!?

Back propagation

- To update the network, we need to compute the direction (gradient) that minimizes the loss function
  1. Compute the loss function of a mini-batch (forward pass)
  2. Compute the gradient (partial derivative) with respect to each parameter (backward pass)
- Key: Use chain rule



True value: y

$z = w_1 x \qquad q = w_2 z$

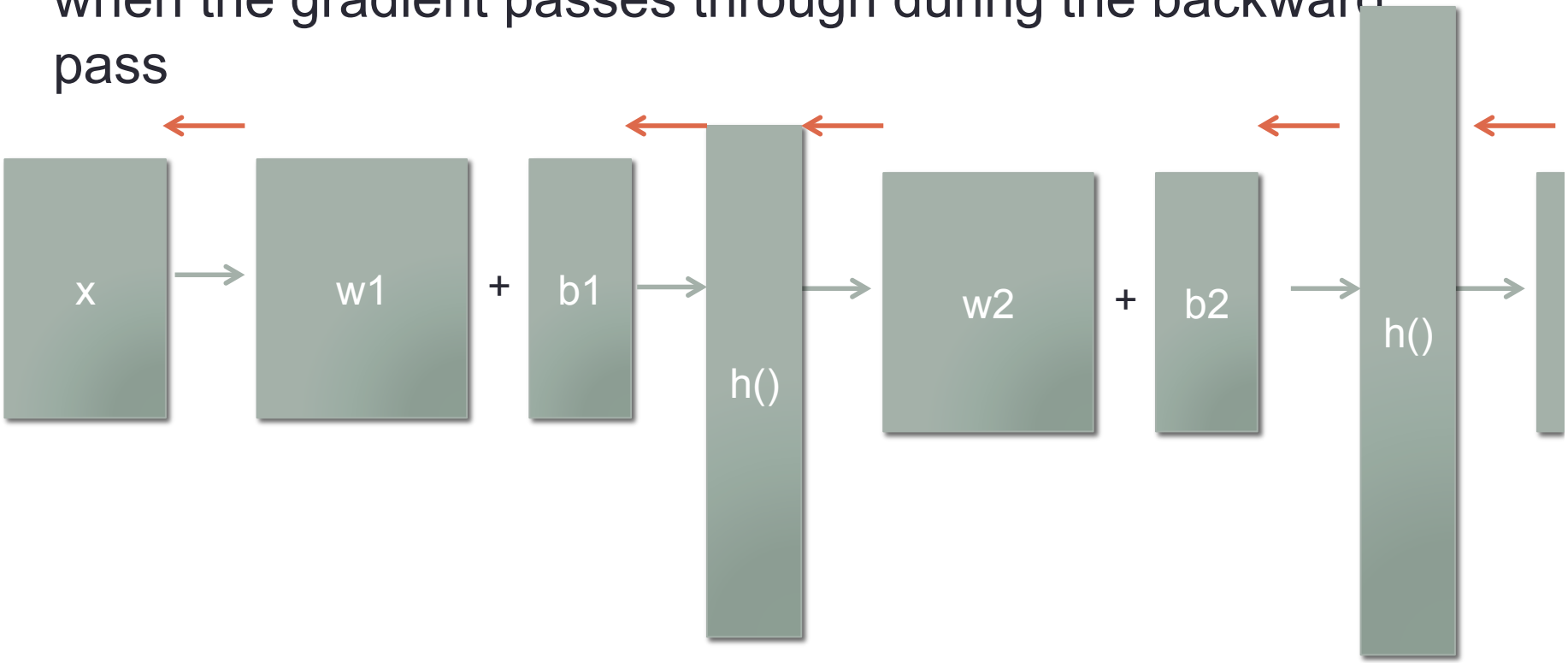
$L = (y-q)^2$

$$\frac{dL}{dw_2} = \frac{dL}{dq}\frac{dq}{dw_2}$$
$$= [2(y-q)(-1)][z]$$

$$\frac{dL}{dw_1} = \frac{dL}{dq}\frac{dq}{dz}\frac{dz}{dw_1}$$
$$= [2(y-q)(-1)][w_2][x]$$

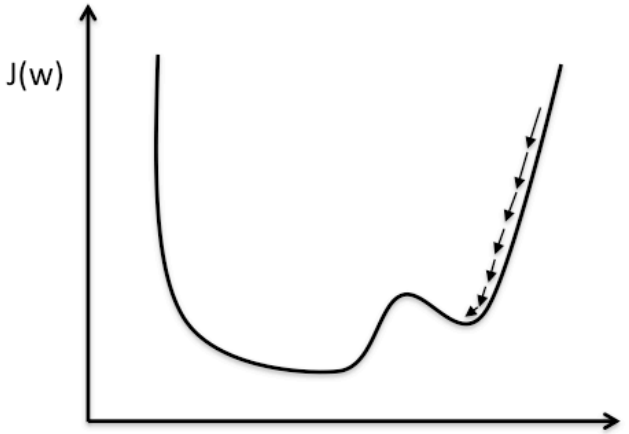
Backprop and computation graph

- We can also define what happens to a computing graph when the gradient passes through during the backward pass



This lets us to build any neural networks without having to redo all the derivation as long as we define a forward and backward computation for the block.

Initialization



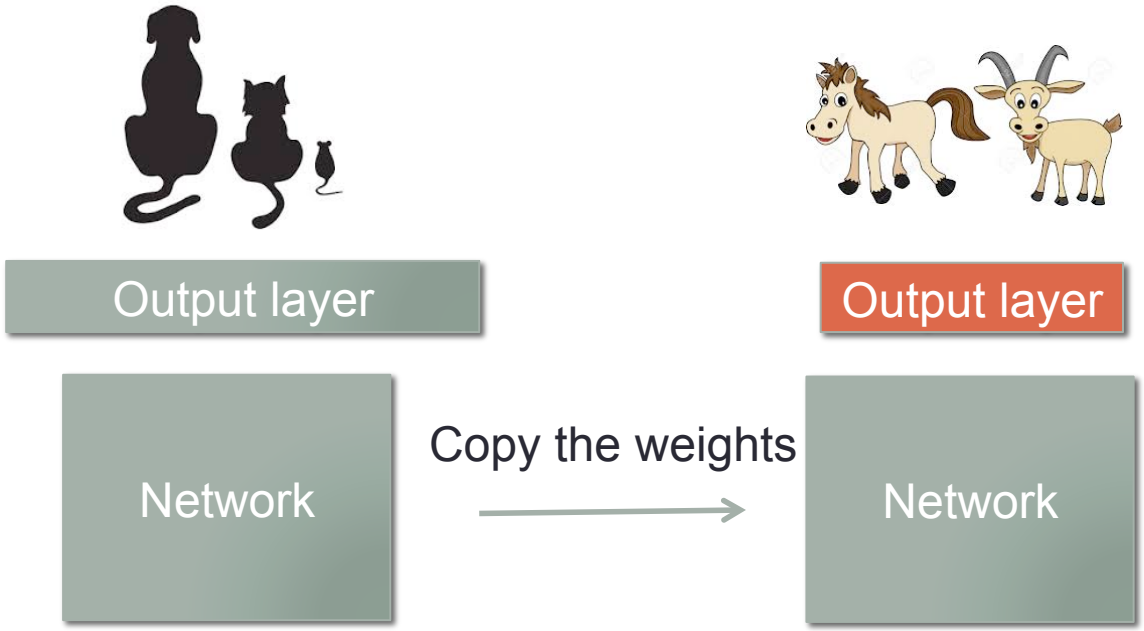
- The starting point of your descent
- Important due to local minimas
- Not as important with large networks AND big data

- Now usually initialized randomly
- Or use transfer learning

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010

Transfer learning

- Initialize weights by using parameters from another network
- Neural networks learn features!
- Features should be useful in related task



Neural networks

- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function
  - SGD and backprop
- CNN, RNN, LSTM, GRU

Convolutional Neural Networks (CNNs)

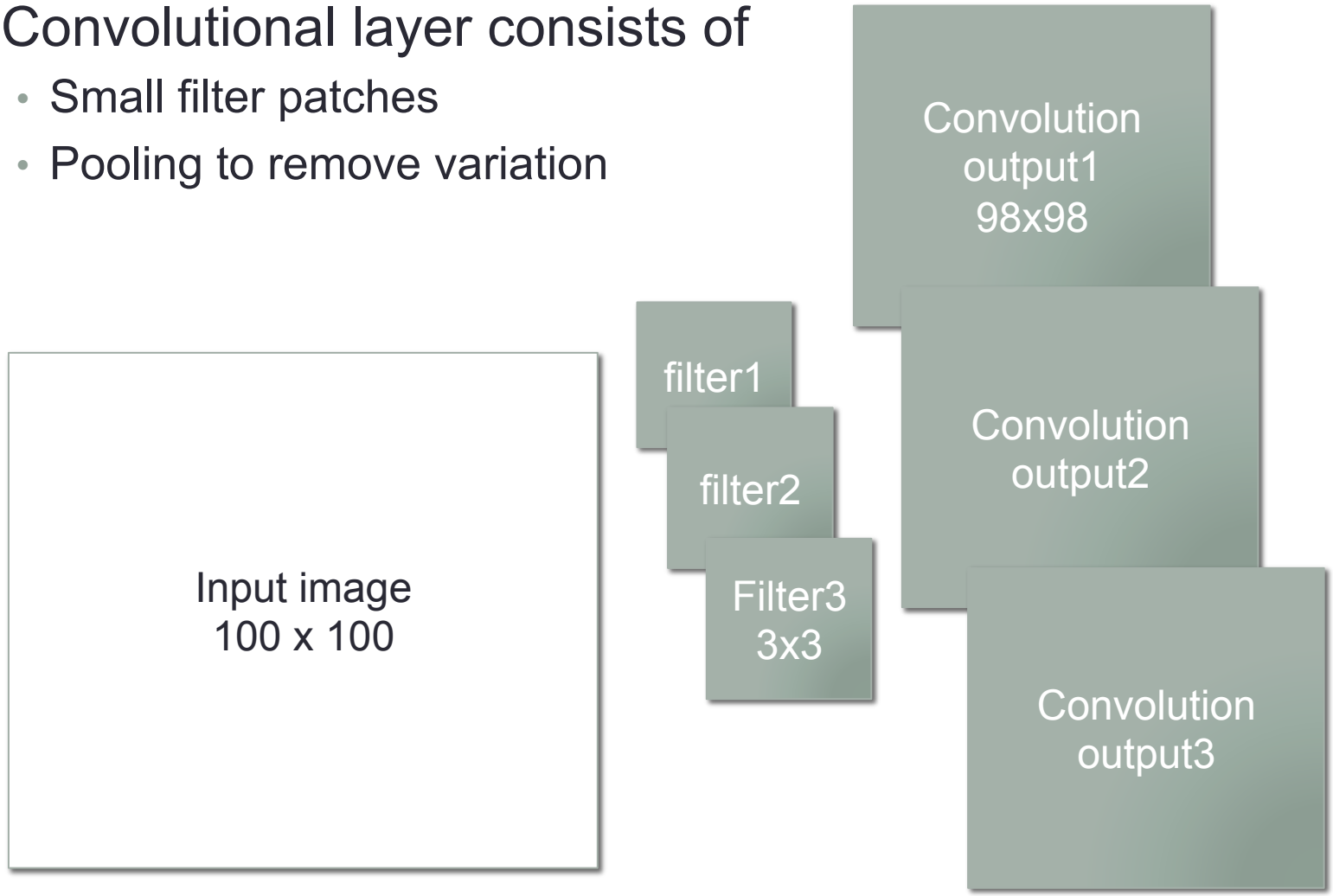
- Consider an image of a cat. DNNs need different neurons to learn every possible location a cat can be

- Can we use the same parameters to learn that a cat exists regardless of location?

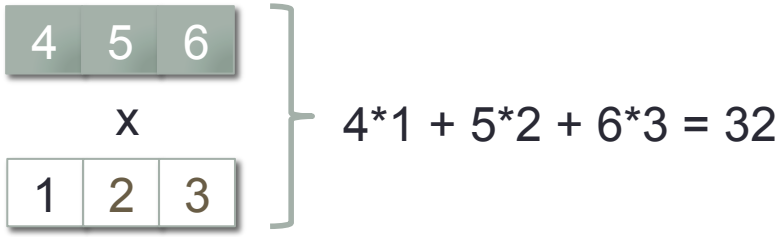
- 2 parts: convolutional layer and pooling layer

Convolutional filters

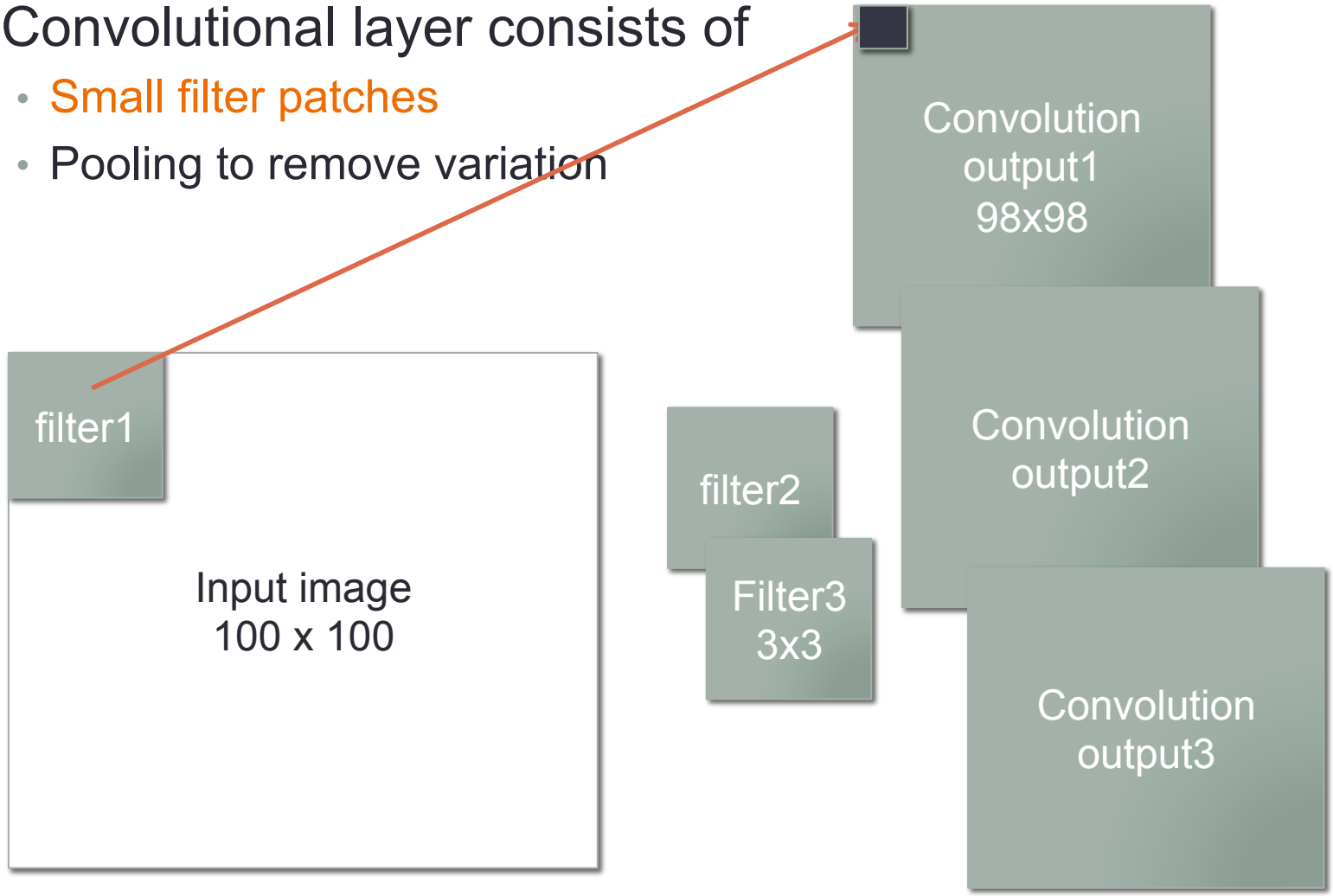
- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



Convolutional filters

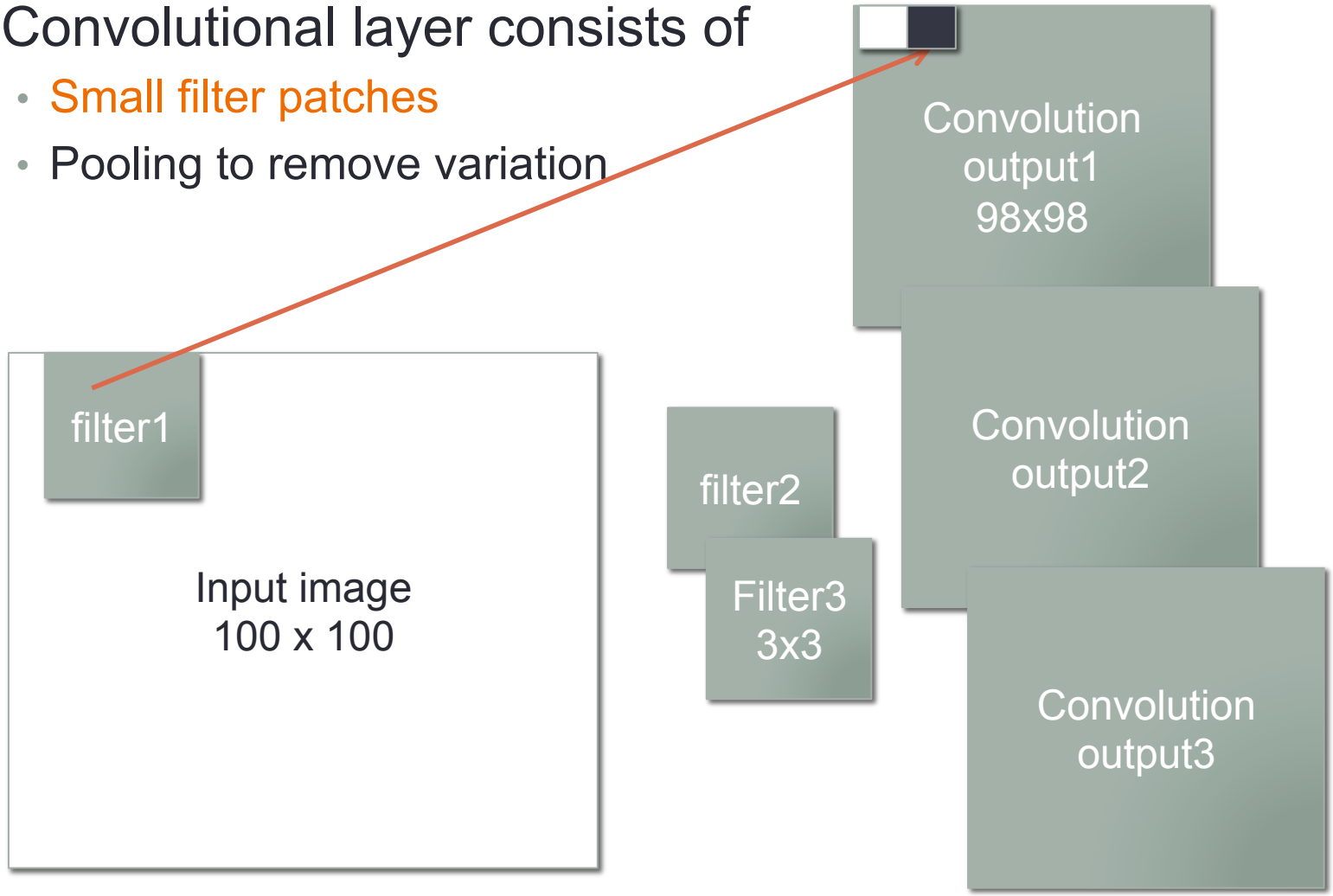


- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



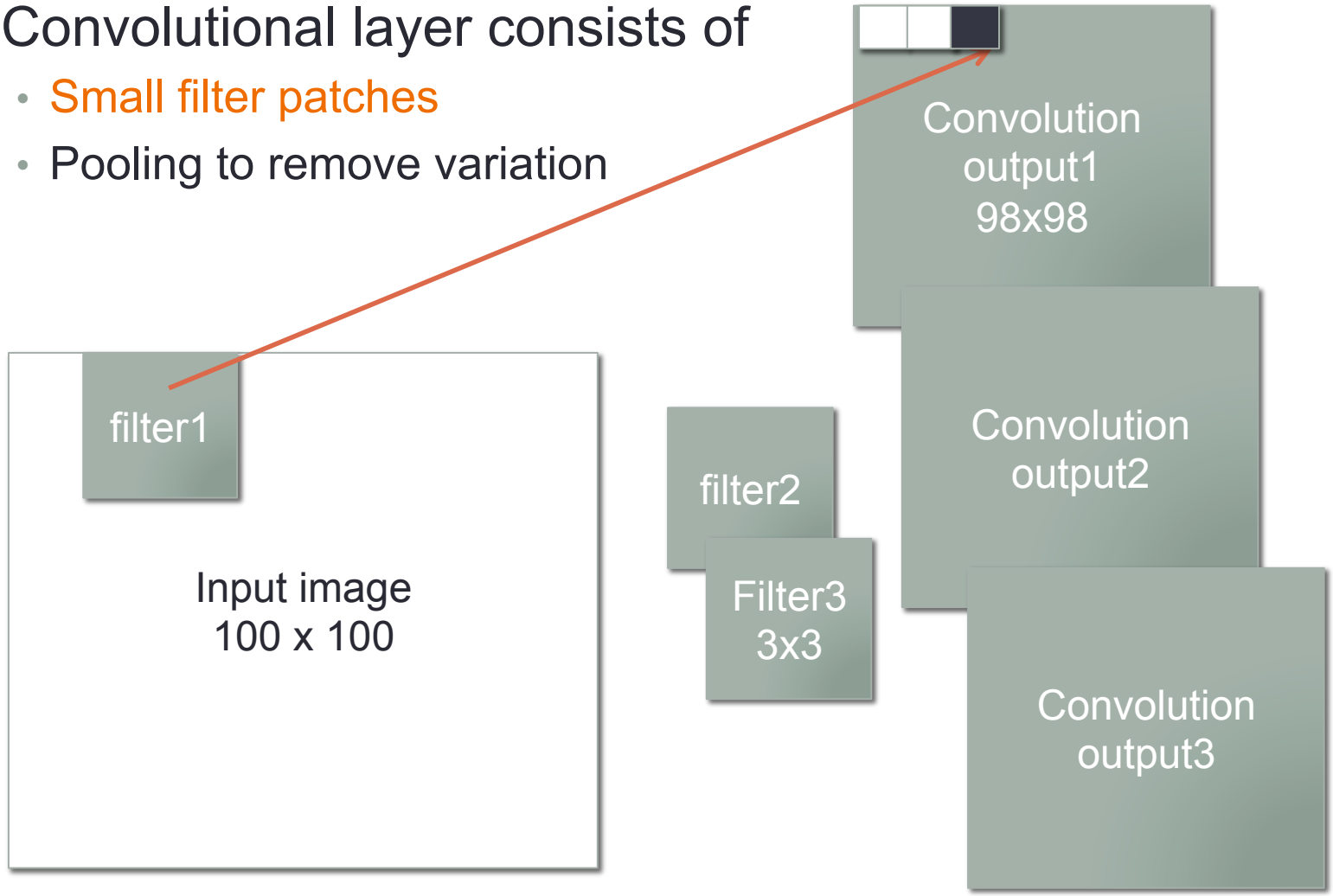
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



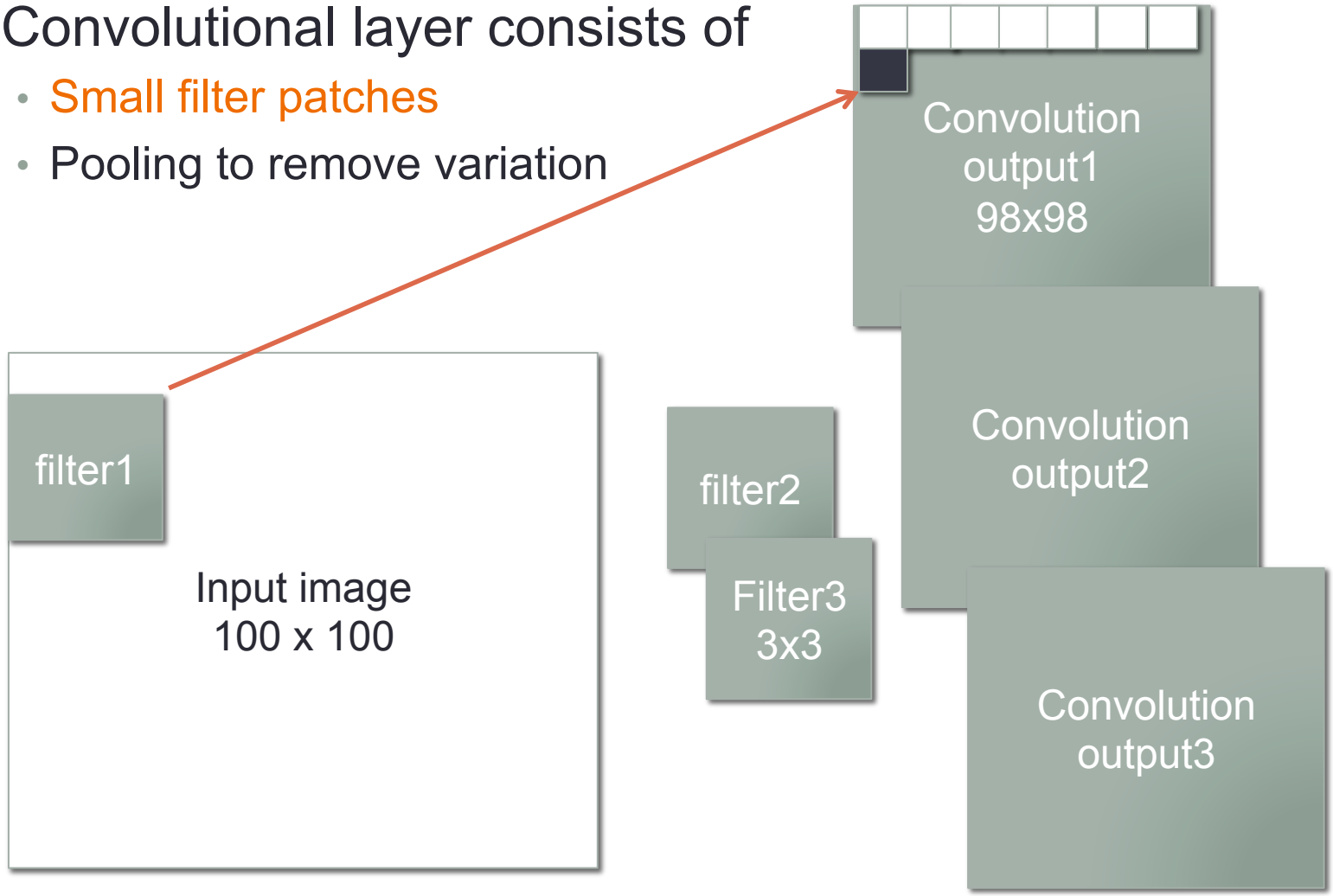
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



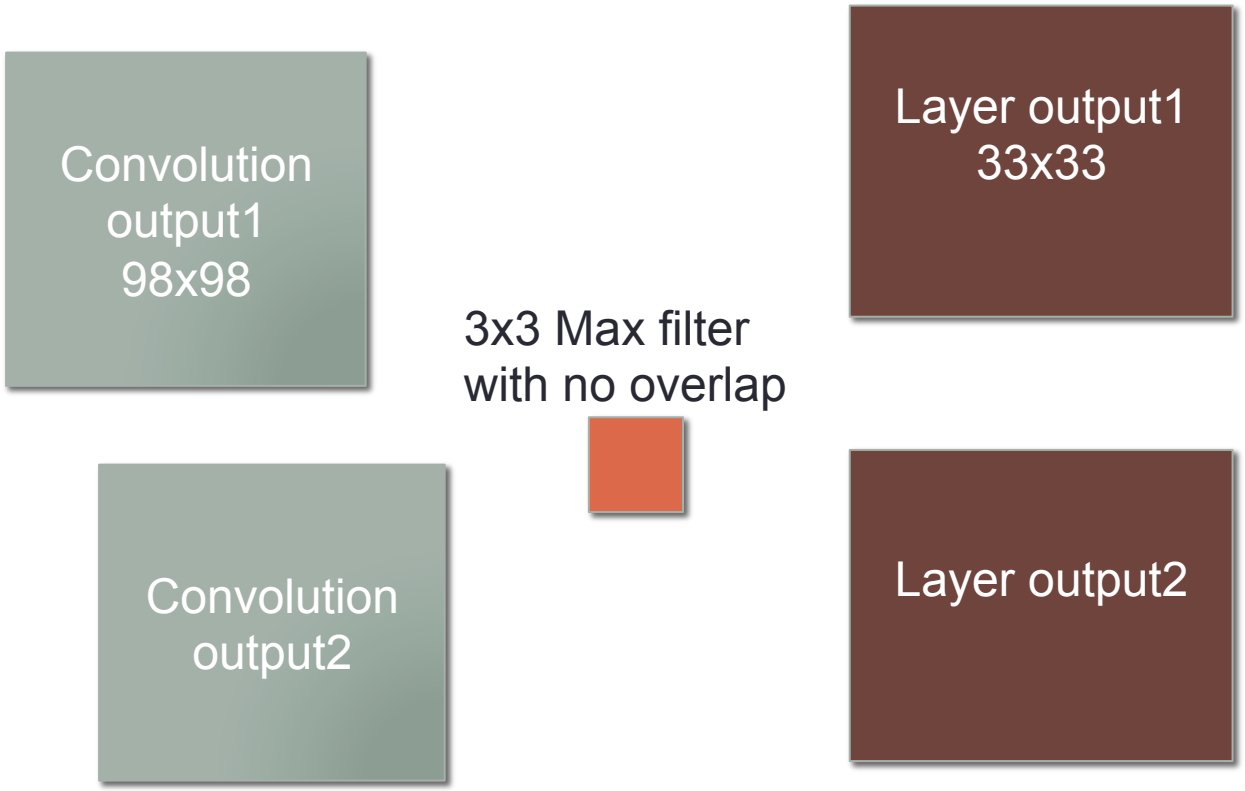
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



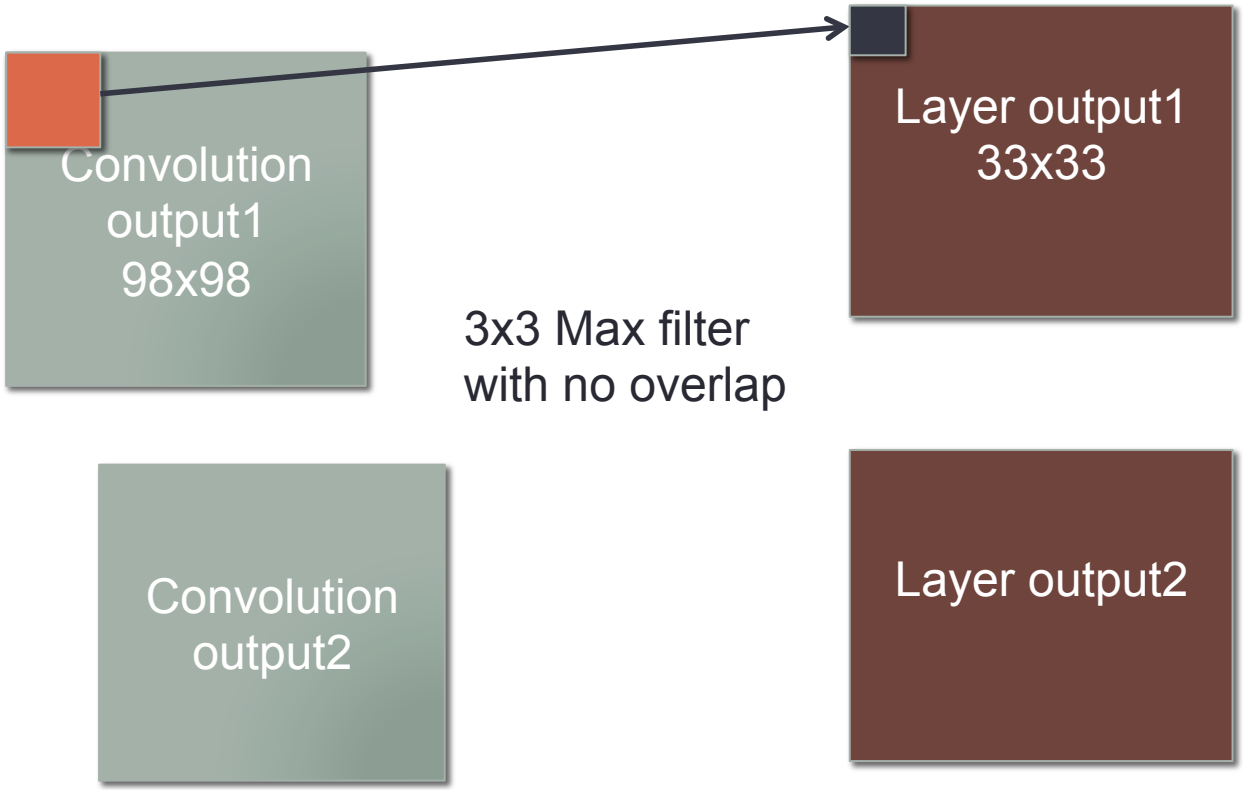
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



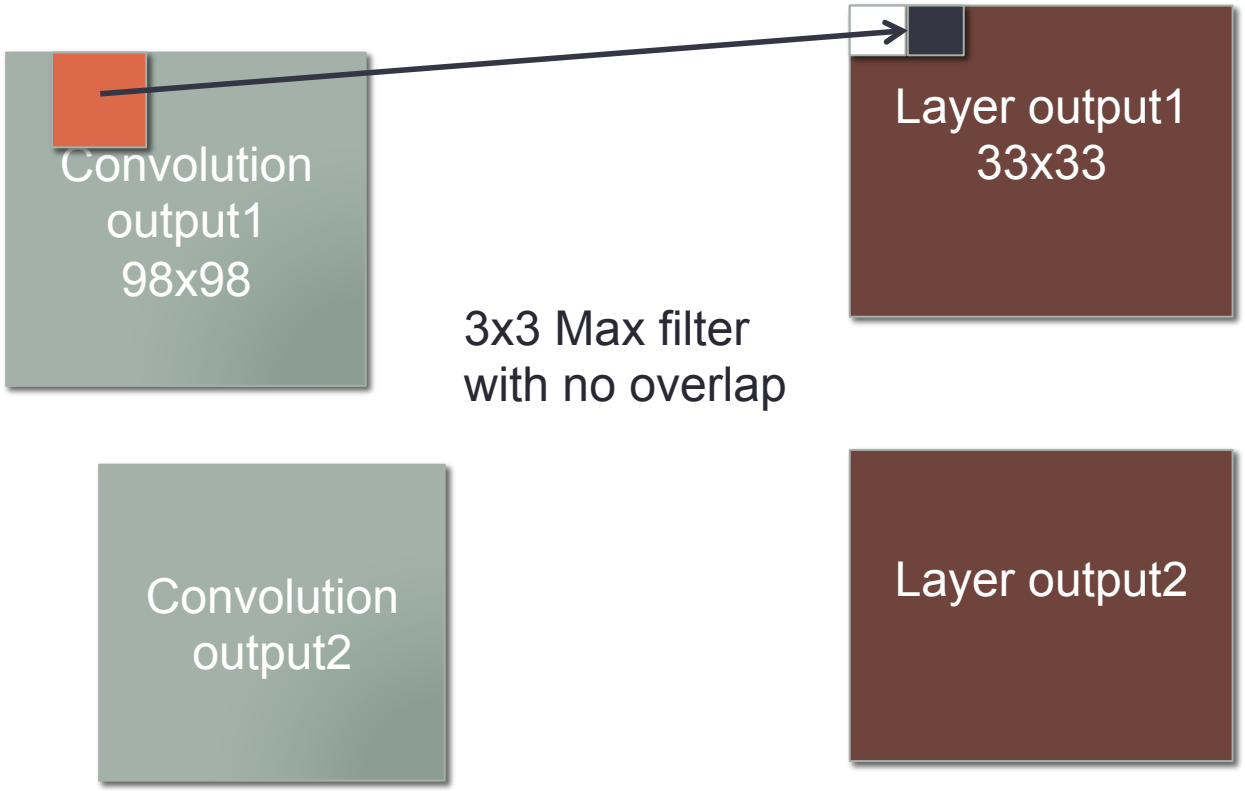
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



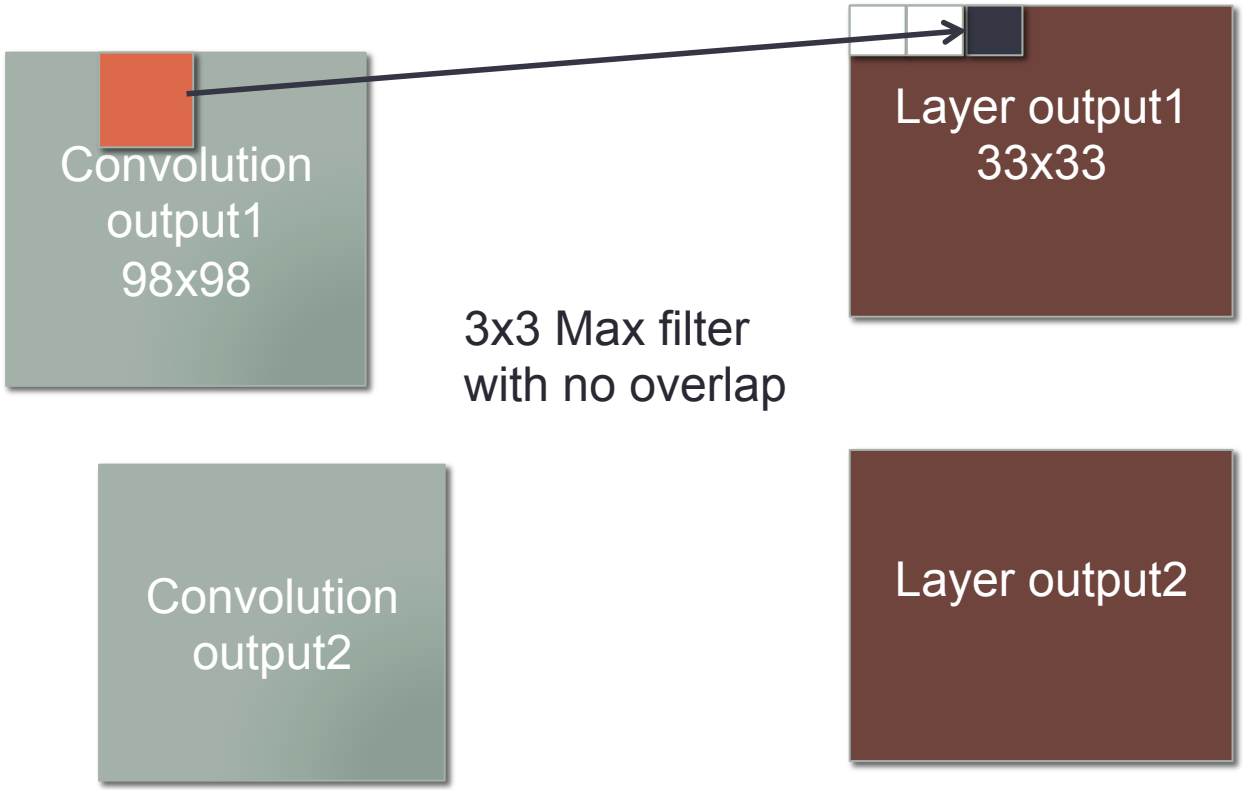
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



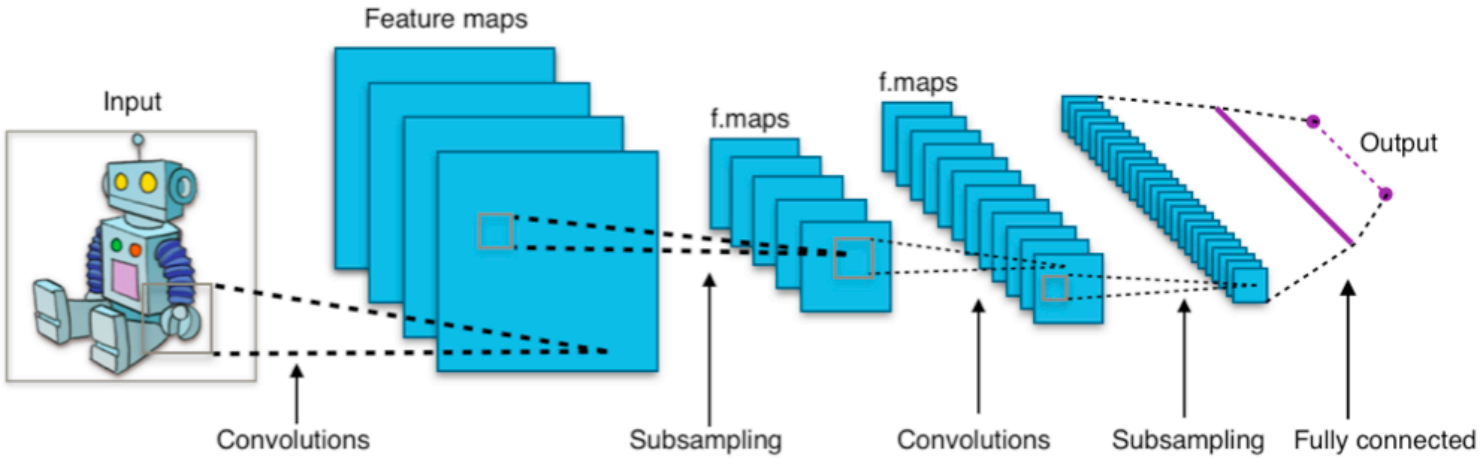
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



CNN overview

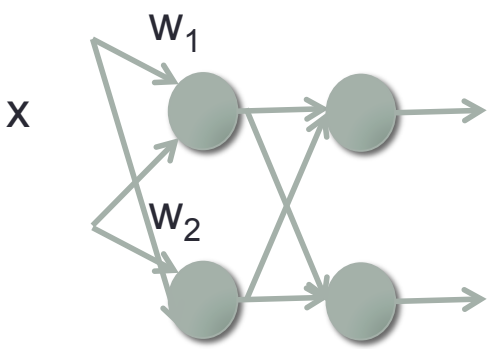
- Filter size, number of filters, filter shifts, and pooling rate are all parameters
- Usually followed by a fully connected network at the end
  - CNN is good at learning low level features
  - DNN combines the features into high level features and classify



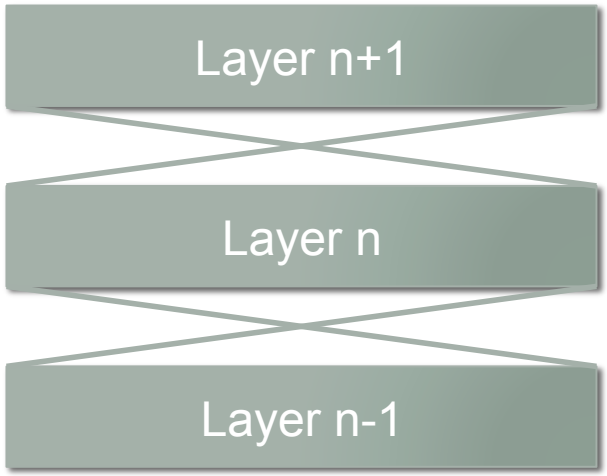
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

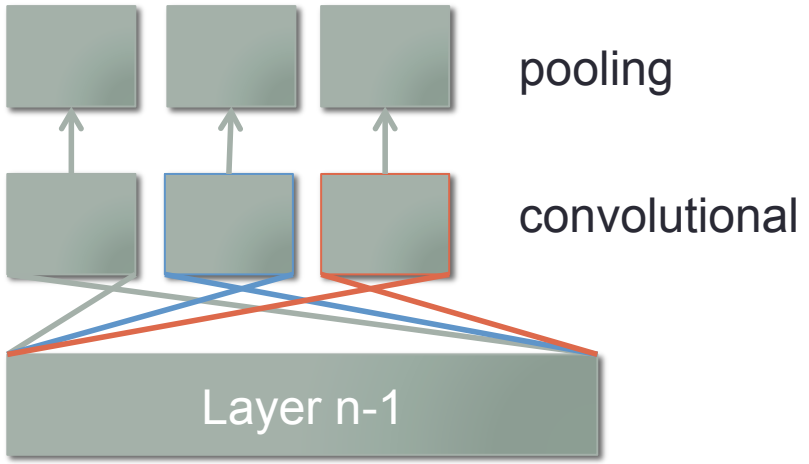
Parameter sharing in convolution neural networks

- $W^T x$



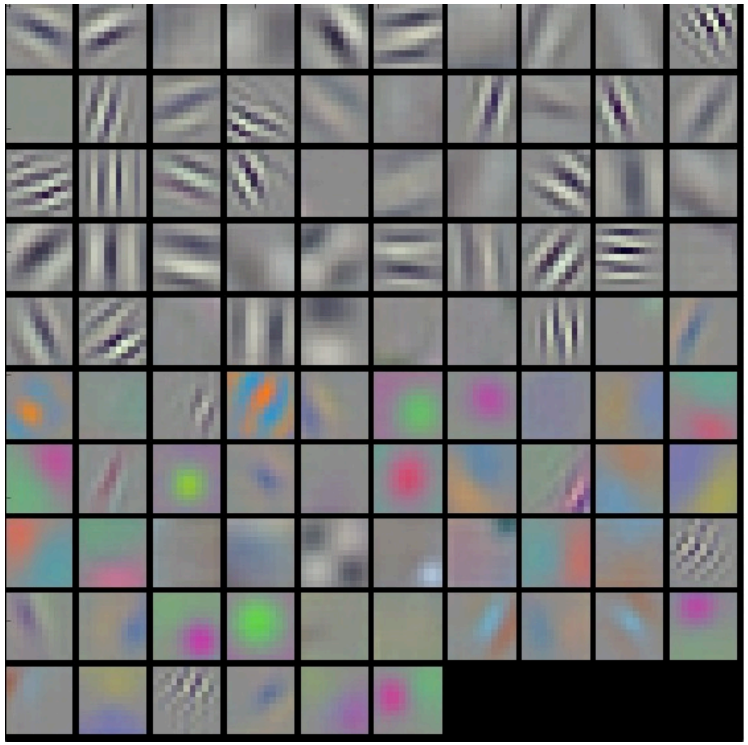
- Cats at different location might need two neurons for different locations in fully connect NNs.
- CNN shares the parameters in 1 filter
- The network is no longer fully connected





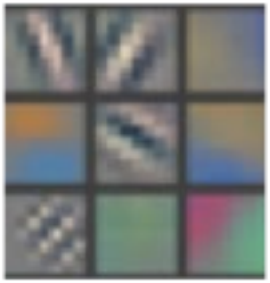
Visualizing convolutional layers

- We can visualize the weights of the filter
- “Matched filters”

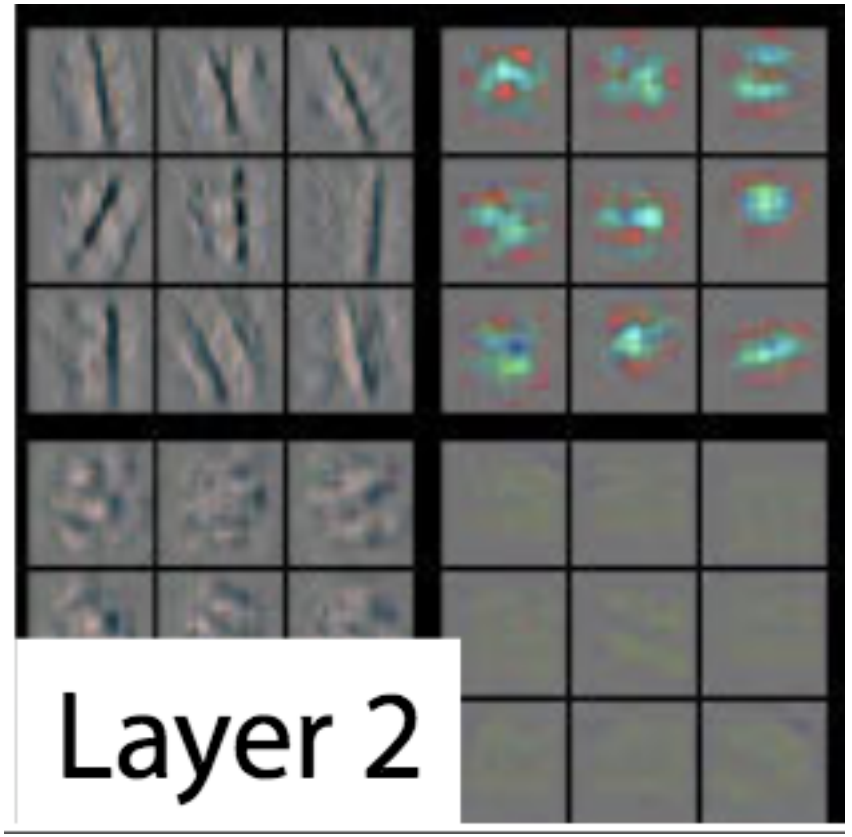


http://cs231n.github.io/understanding-cnn/

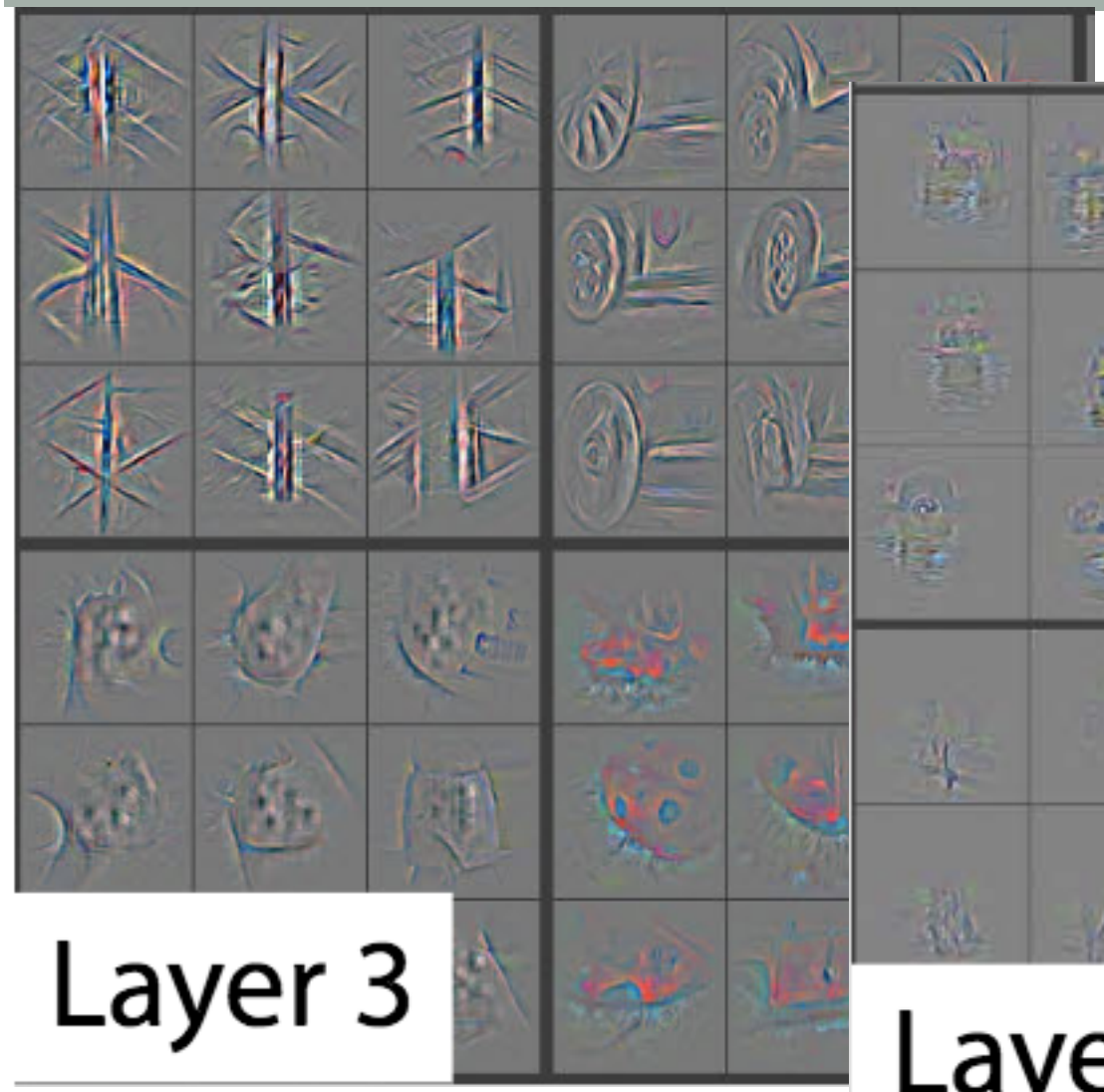
Higher layer captures higher-level concepts

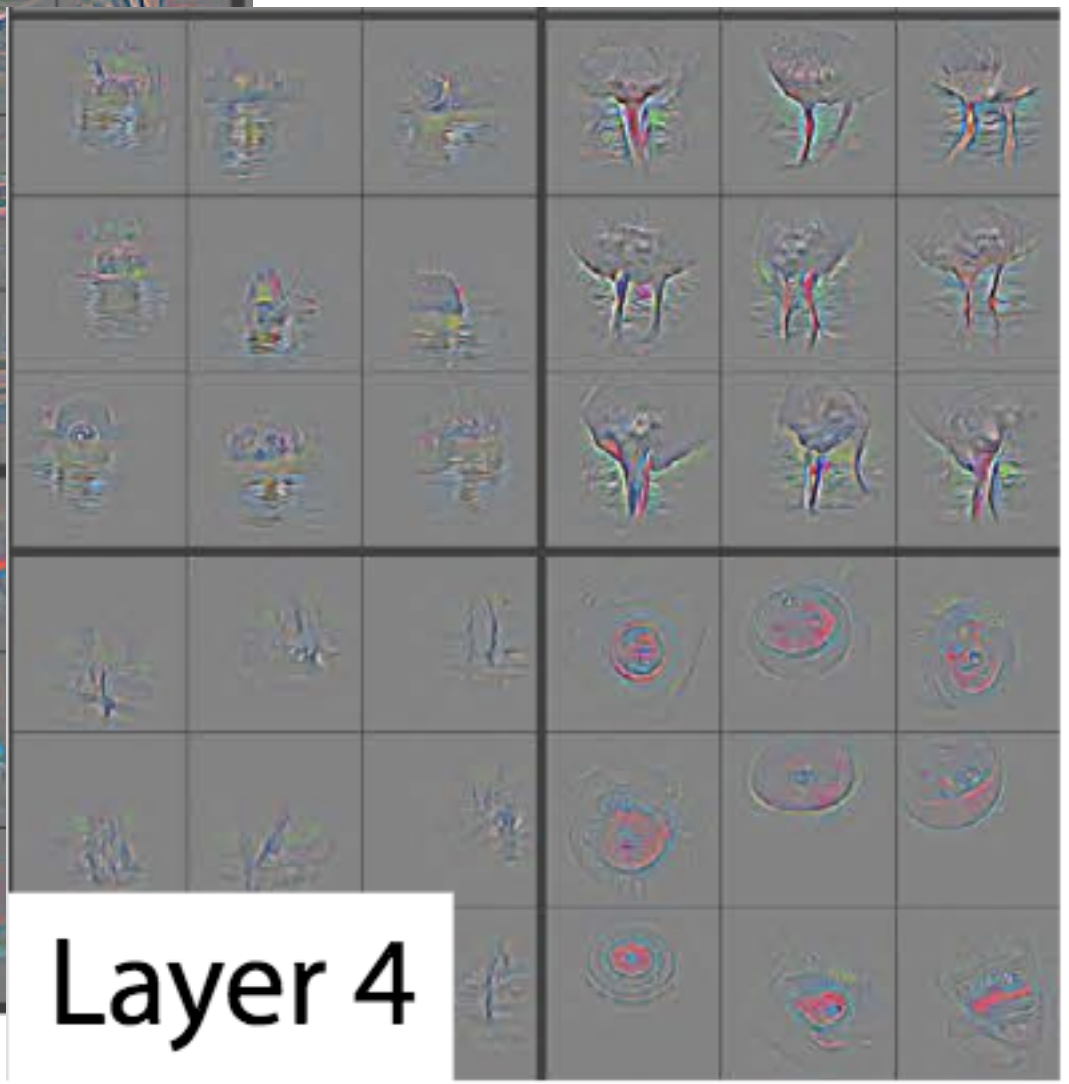


Layer 1



https://arxiv.org/pdf/1311.2901.pdf


Layer 3


Layer 4

CNN

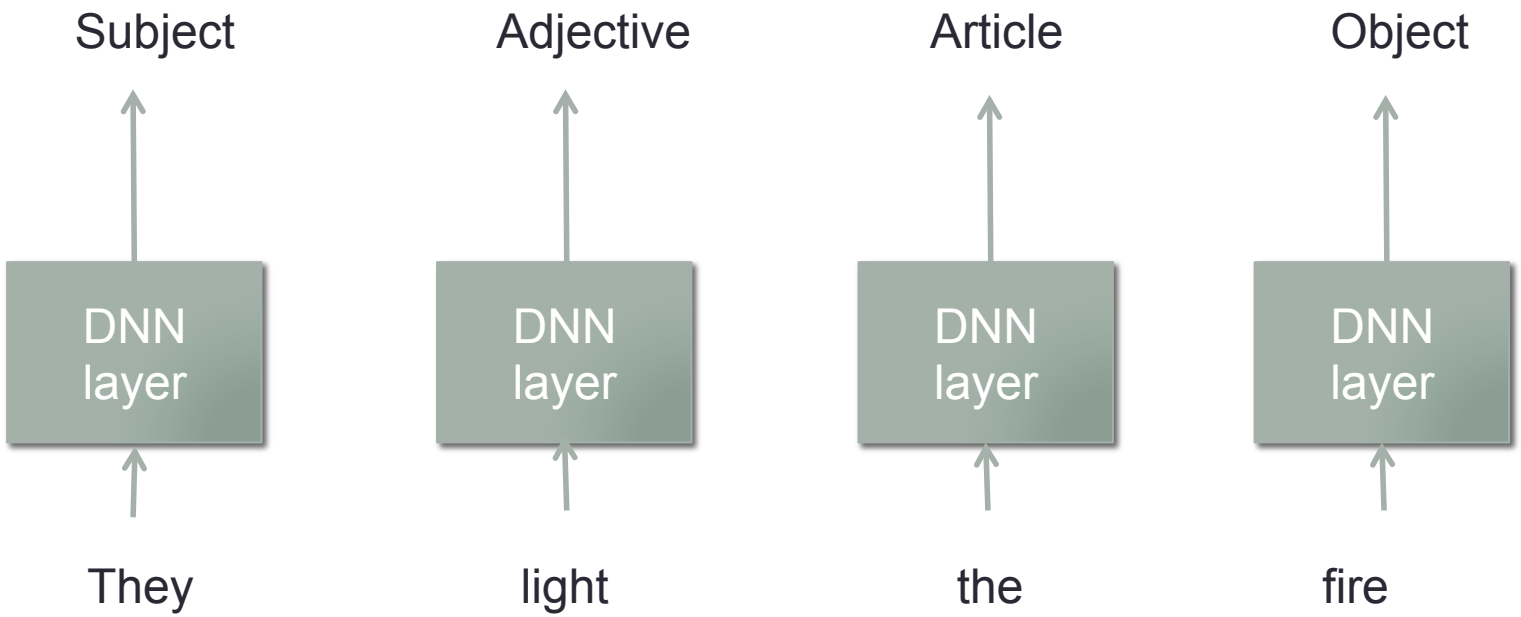
- Convolutional layer
- Subsampling

- Sharing of parameters in space

- Sharing of parameters in time?

Recurrent neural network (RNN)

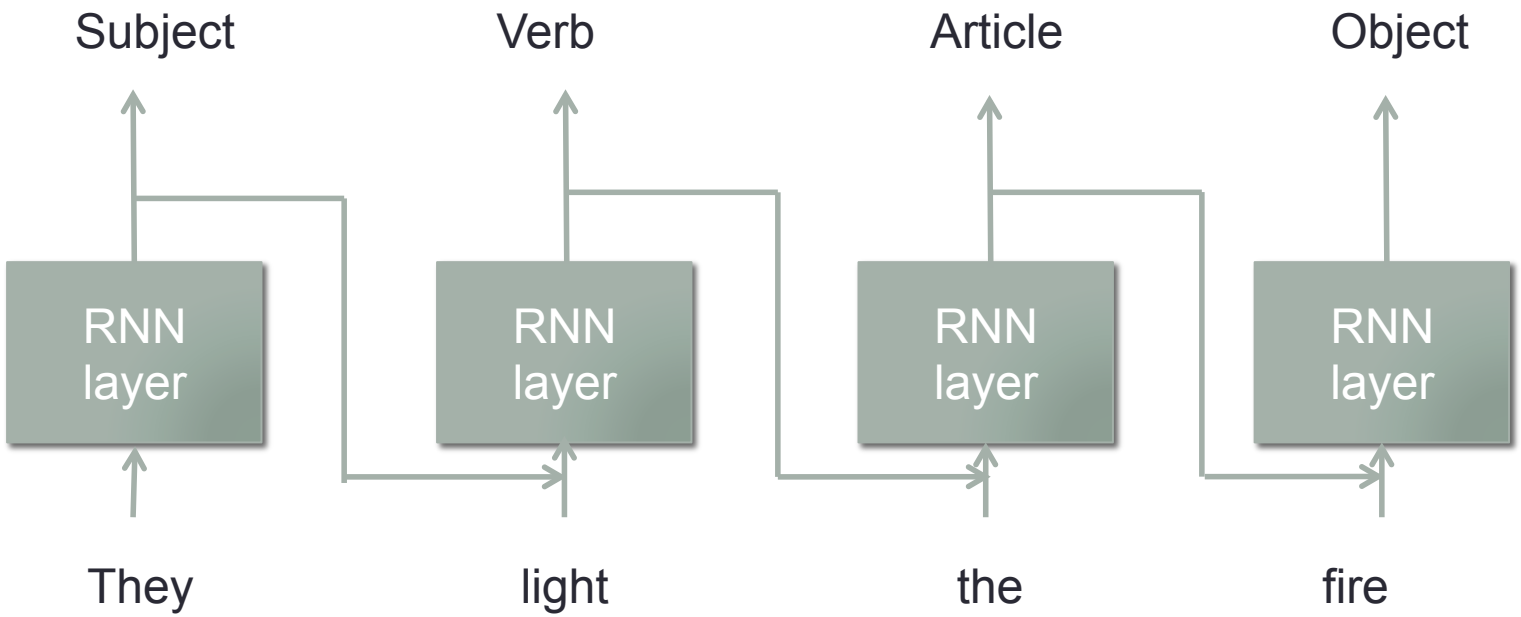
- DNN framework



Problem1: need a way to remember the past

Recurrent neural network (RNN)

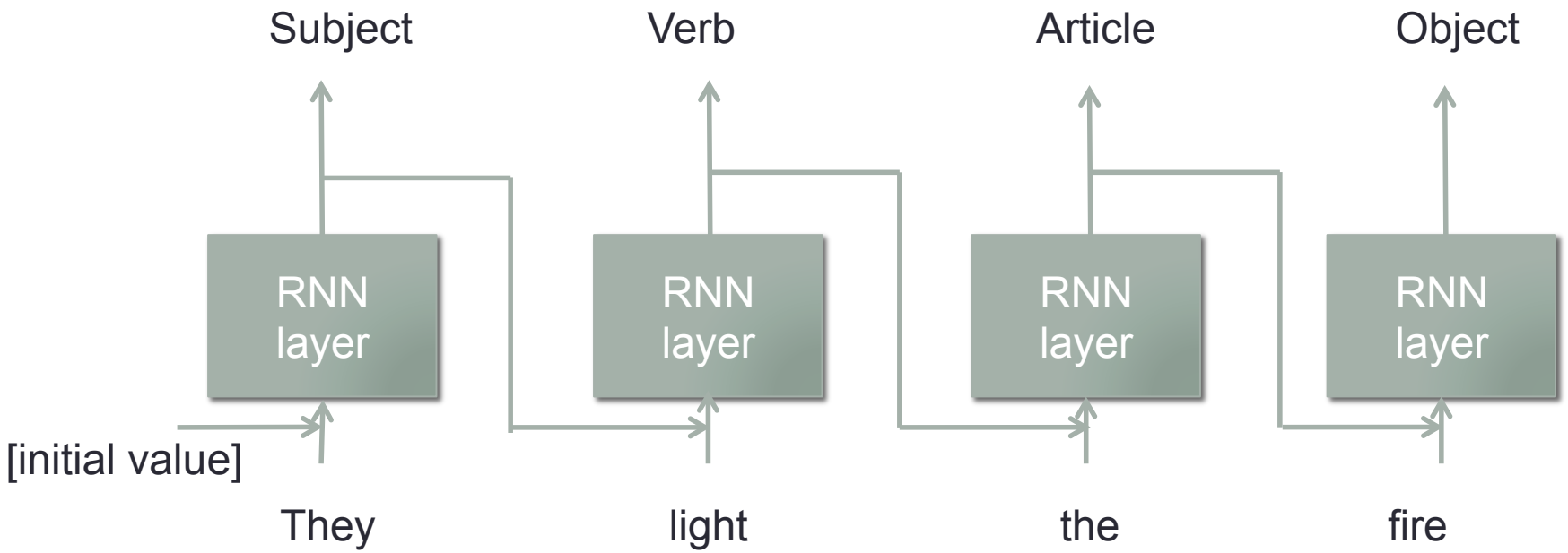
- RNN framework



Output of the layer encodes something meaningful about the past

Recurrent neural network (RNN)

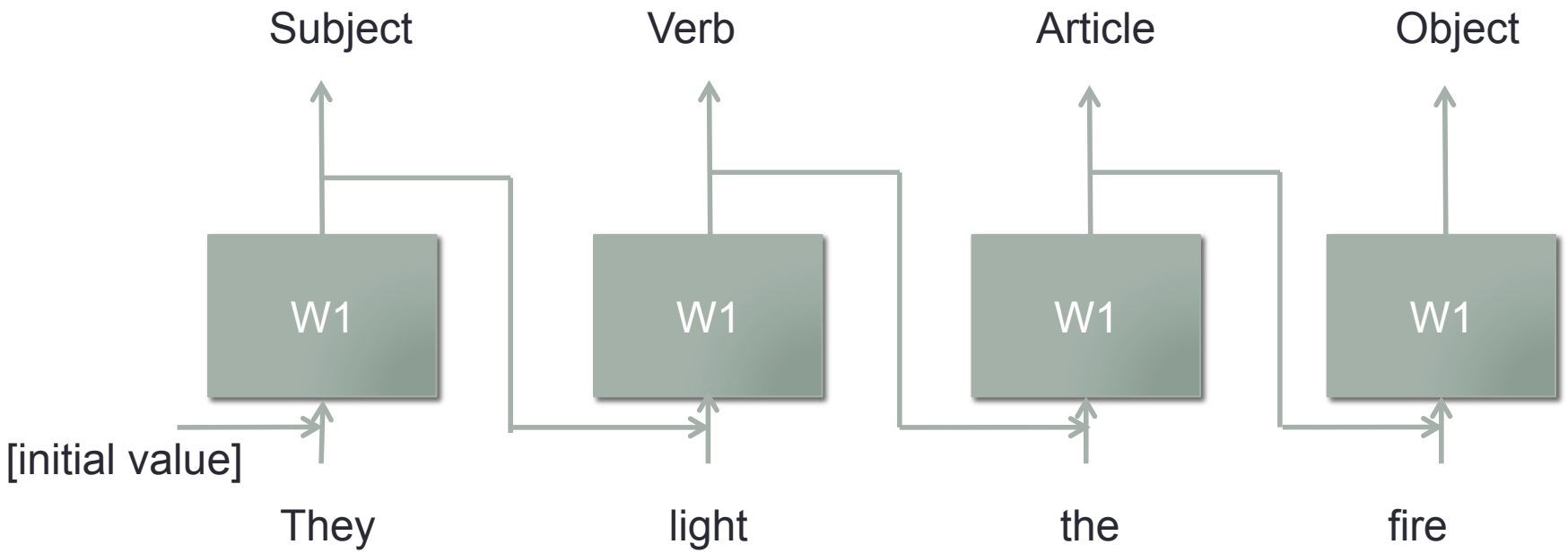
- RNN framework



New input feature = [original input feature, output of the layer at previous time step]

Training a recurrent neural network

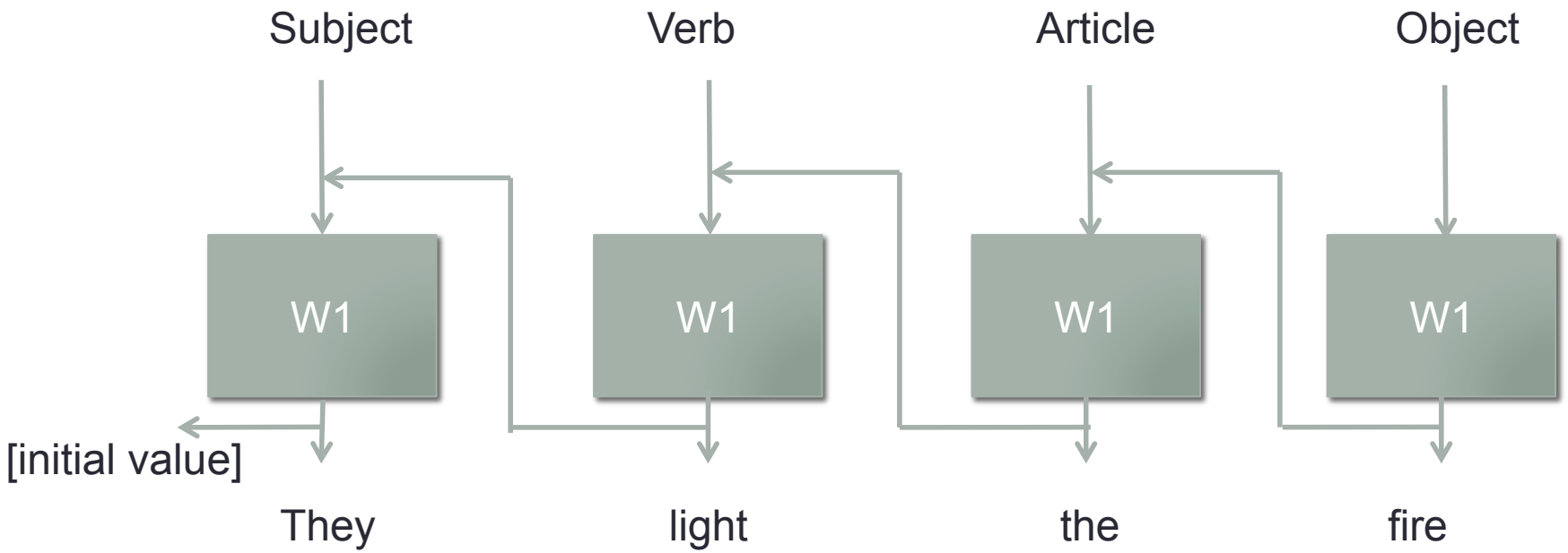
- Computation graph



Parameter sharing

Training a recurrent neural network

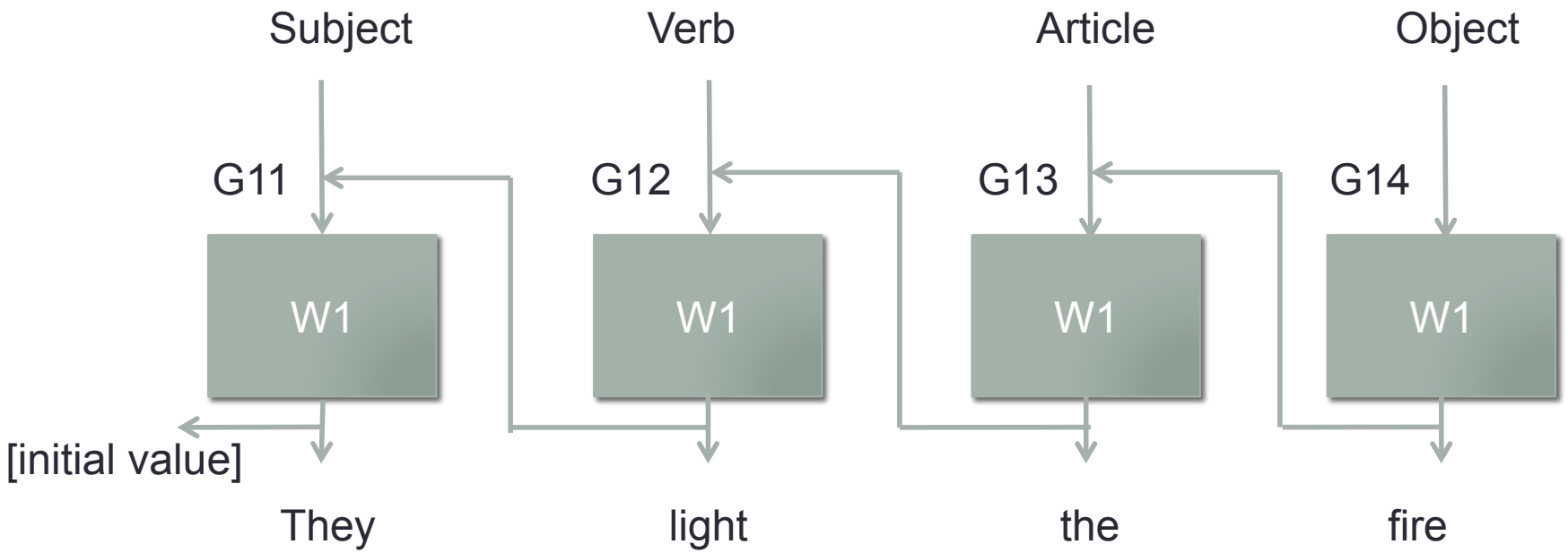
- Backward Computation graph



Backpropagation through time (BPTT)

BPTT

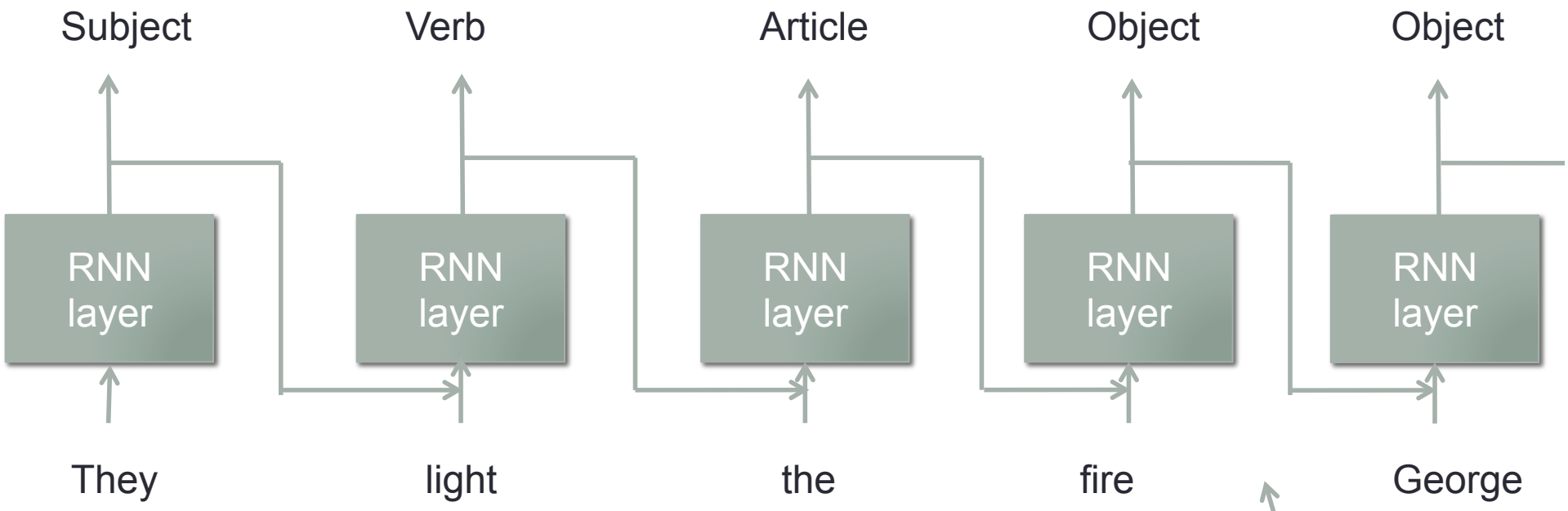
- Backward Computation graph



W1 <- W1 + G11 + G12 + G13 + G14

Cannot deal with infinitely long recurrent
Gradient explosion, vanishing

Recurrent neural network (RNN)

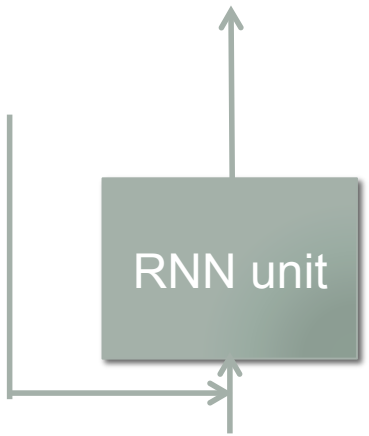


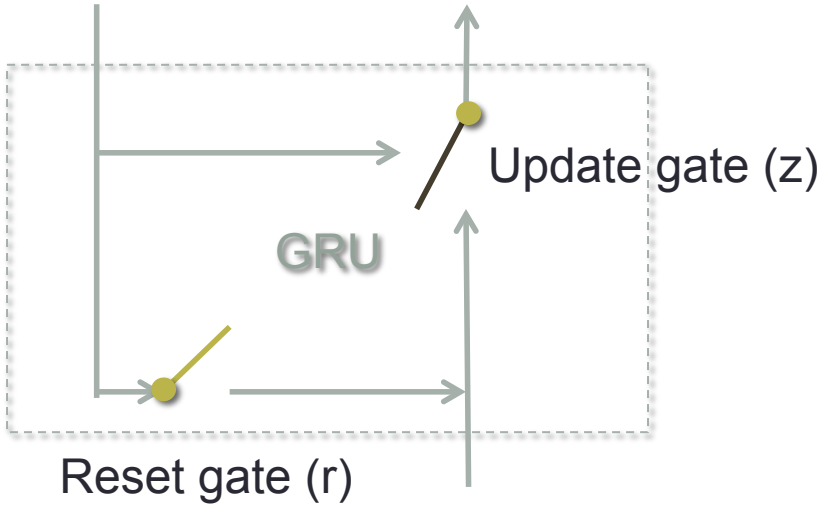
Problem2: needs a way to stop remembering

Can the network learn when to start and stop remembering things?

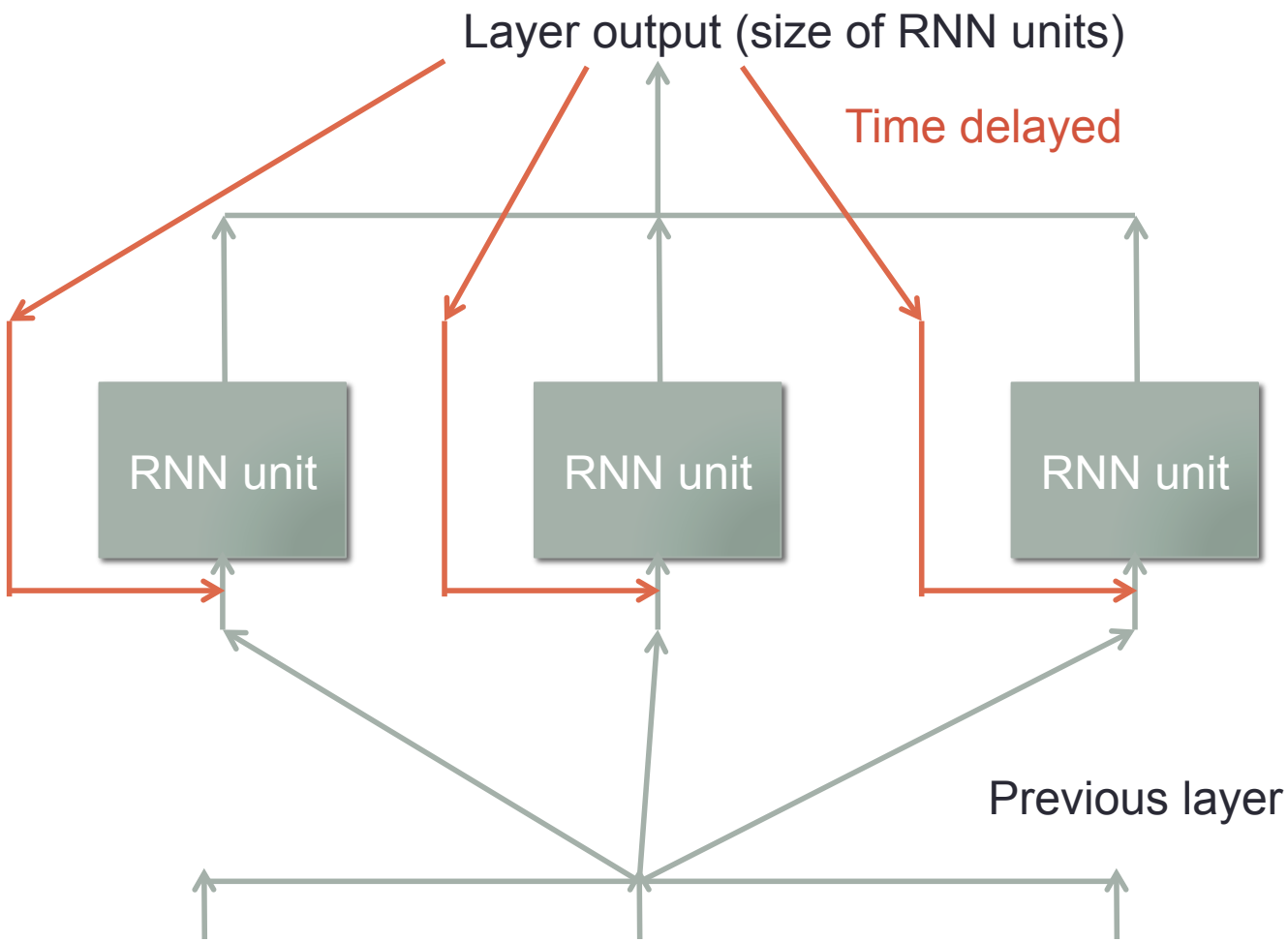
Gated Recurrent Unit (GRU)

- Forms a Gated Recurrent Neural Networks (GRNN)
- Add gates that can choose to reset (r) or update (z)



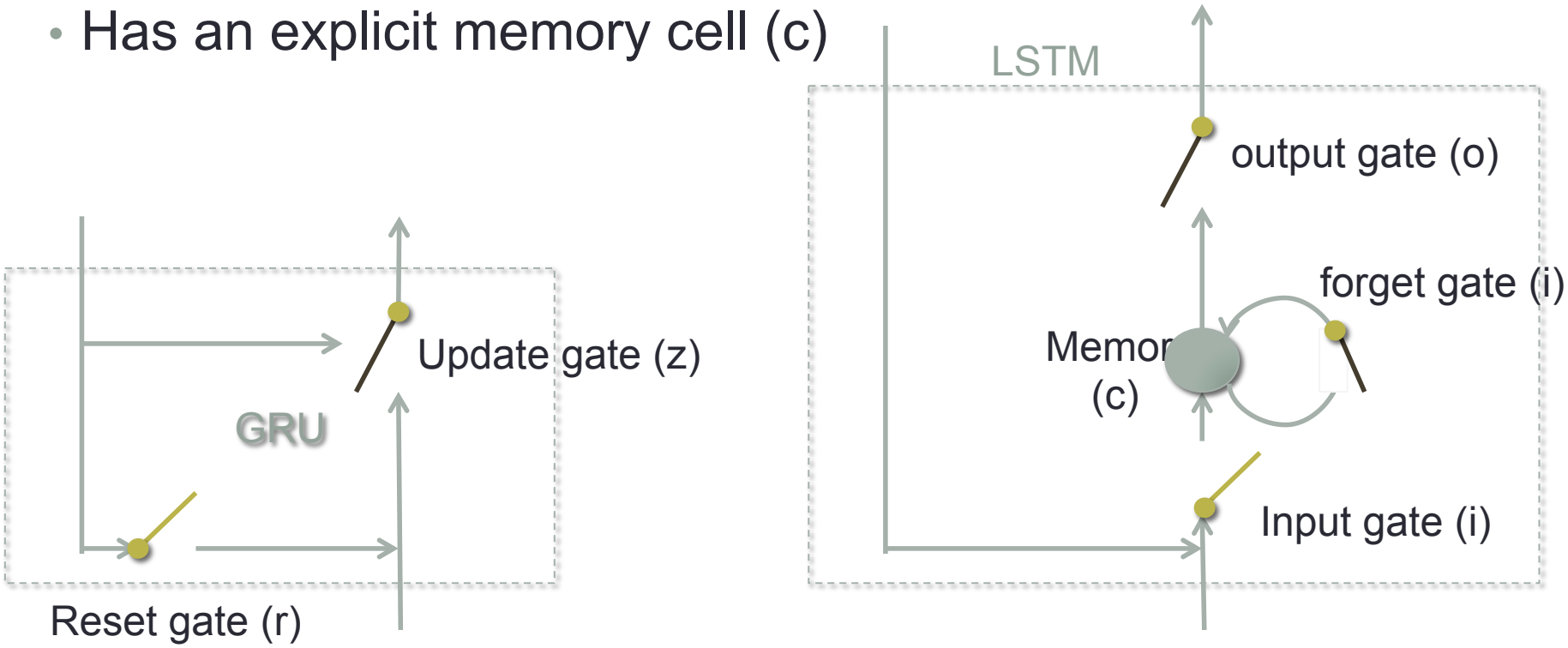


Gated Recurrent Unit (GRU) layer



Long Short-Term Memory (LSTM)

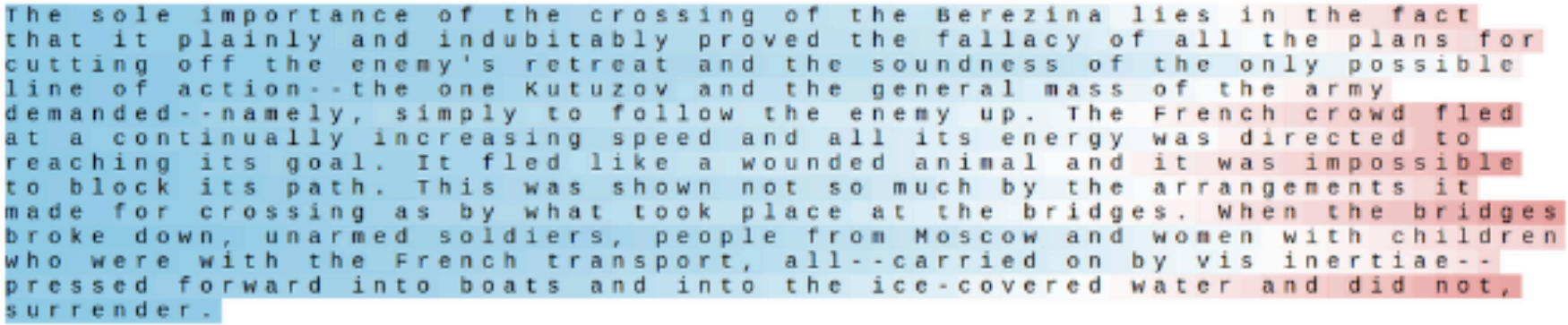
- Have 3 gates, forget (f), input (i), output (o)
- Has an explicit memory cell (c)



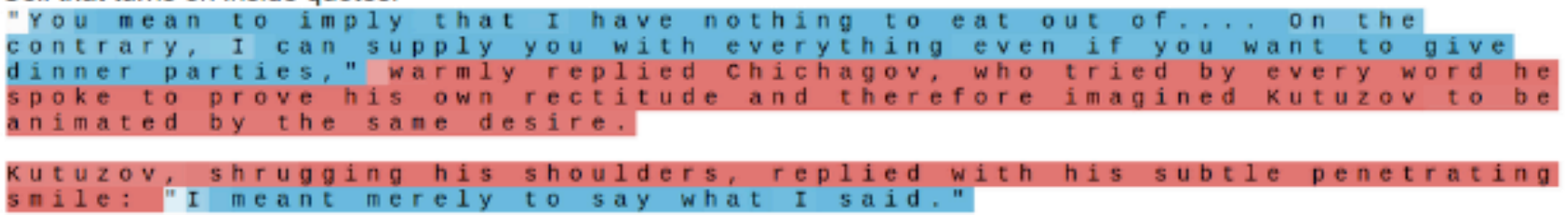
Both works for data with time dependency. Try GRU first

LSTM remembers meaningful things

Cell sensitive to position in line:



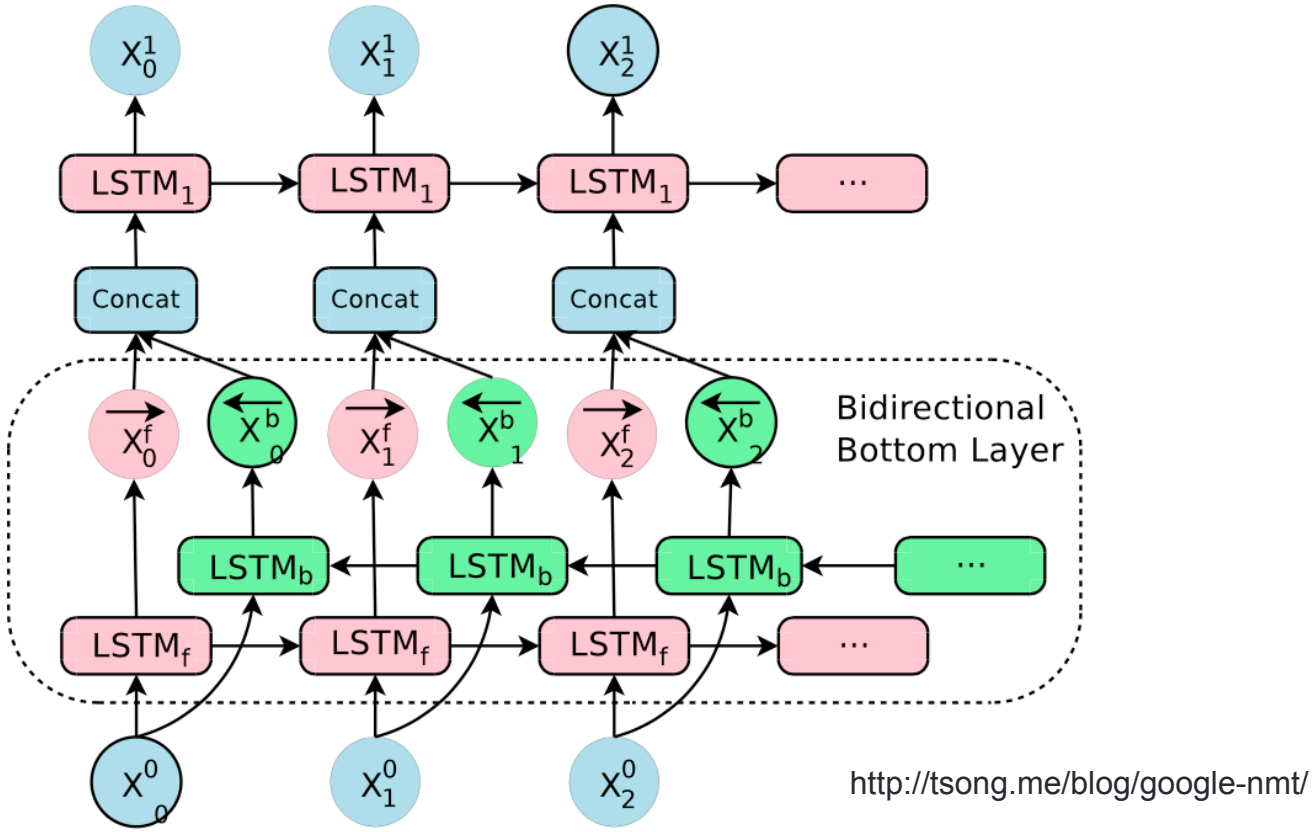
Cell that turns on inside quotes:



Andrej Karpathy, Visualizing and Understanding Recurrent Networks, 2015, https://arxiv.org/abs/1506.02078

Bi-directional LSTM

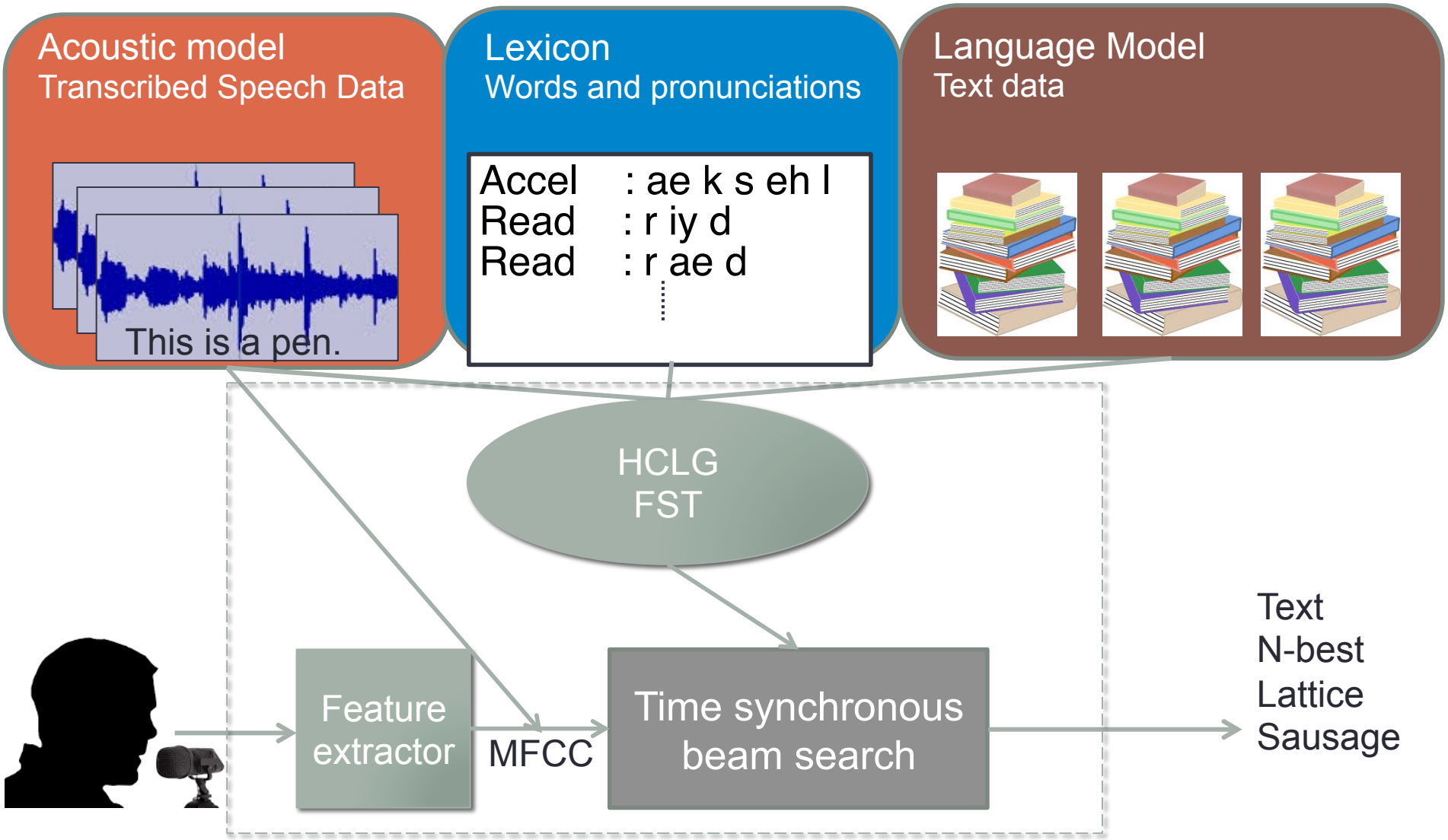
- The previous GRU/LSTM only goes backward in time (uni-directional)
- Most of the time information from the future is useful for predicting the current output



http://tsong.me/blog/google-nmt/

DEEP LEARNING IN AUTOMATIC SPEECH RECOGNITION

Deep learning in ASR?



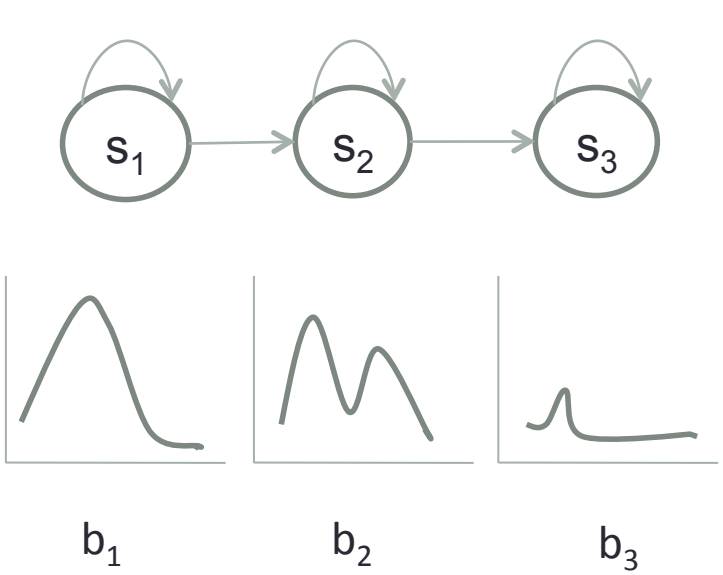
Deep learning in AM

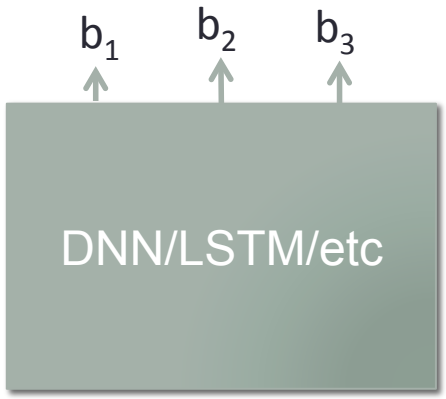
Two main approaches

- Hybrid DNN-HMM
- Tandem

Hybrid DNN-HMM approach

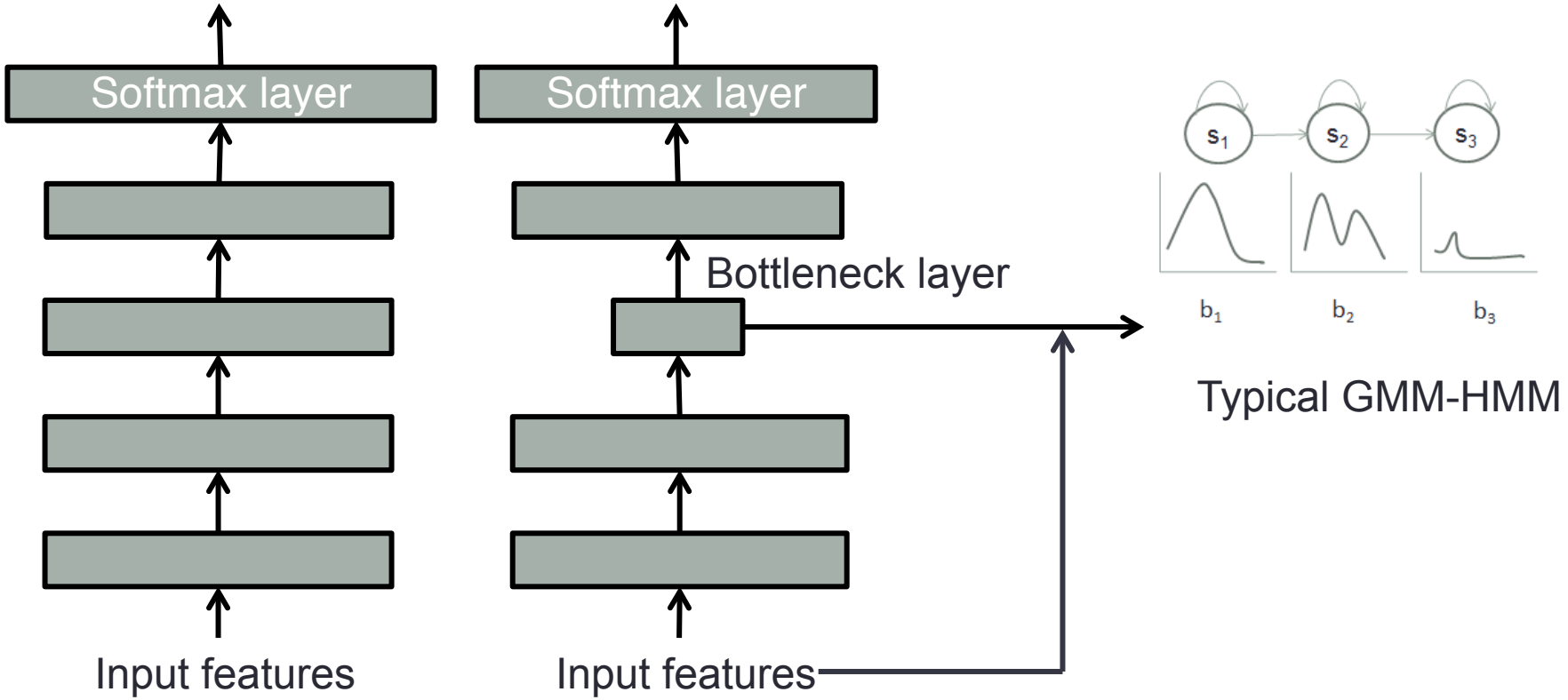
- A typical speech recognizer uses the GMM-HMM framework
  - Emission probabilities are modeled by a GMM
- Instead, model emission probabilities with a DNN





Tandem approach

- Use the DNN to generate good features to feed into the general GMM-HMM framework.
- Typically done by placing a narrow hidden layer in the network.



DNN in ASR results

	WER
Traditional	64.5
Tandem	55.4
Hybrid	55.6

Similar performance. Using tandem or hybrid depends on application.

Method	WER
LSTM	18.0
CNN+LSTM	17.6
LSTM+DNN	17.6
CLDNN	**17.3**

Table 5. WER, CLDNN

Using a combination of CNN+LSTM+fully connected is best

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43455.pdf

Deep learning in LM

- WSJ task with RNN

Model	DEV WER	EVAL WER
Lattice 1 best	12.9	18.4
Baseline - KN5 (37M)	12.2	17.2
Discriminative LM [8] (37M)	11.5	16.9
Joint LM [9] (70M)	-	16.7
Static 3xRNN + KN5 (37M)	11.0	15.5

Around 10% relative gain over n-gram

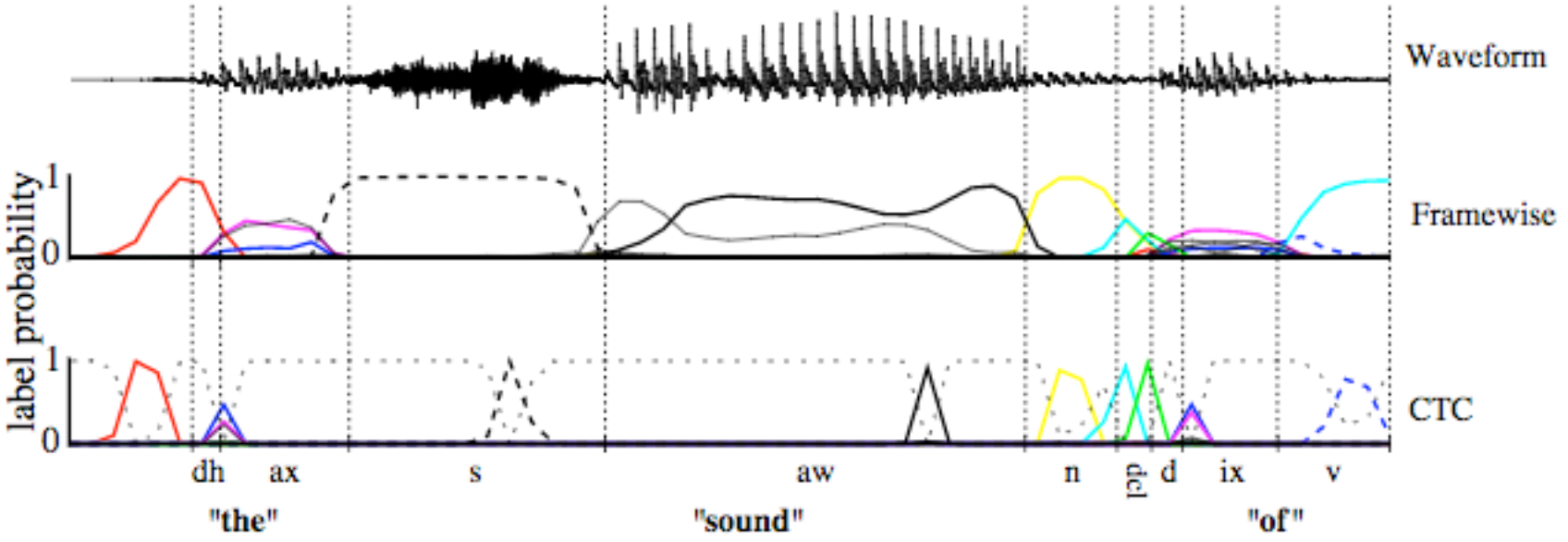
Mikolov T, Recurrent Neural Network Based Language Model, 2010

End-to-End models

- Instead of the typical feature extraction followed by AM +Lexicon+LM, train a BIG network that goes from waveform to characters/words

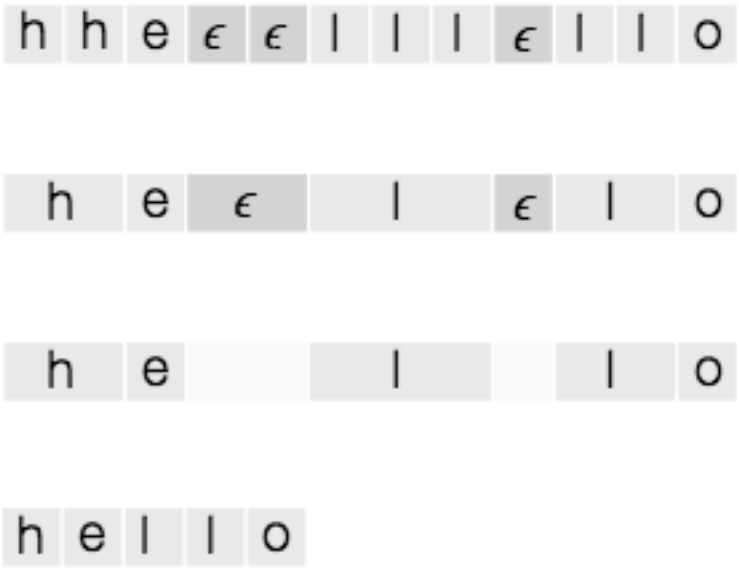
Connectionist Temporal Classification (CTC)

- A kind of end-to-end modeling
- Only output words/characters at certain spikes, while DNN-HMM emits every frame
- Has a background class for no output



Graves, A. Connectionist Temporal Classification, 2006

CTC output

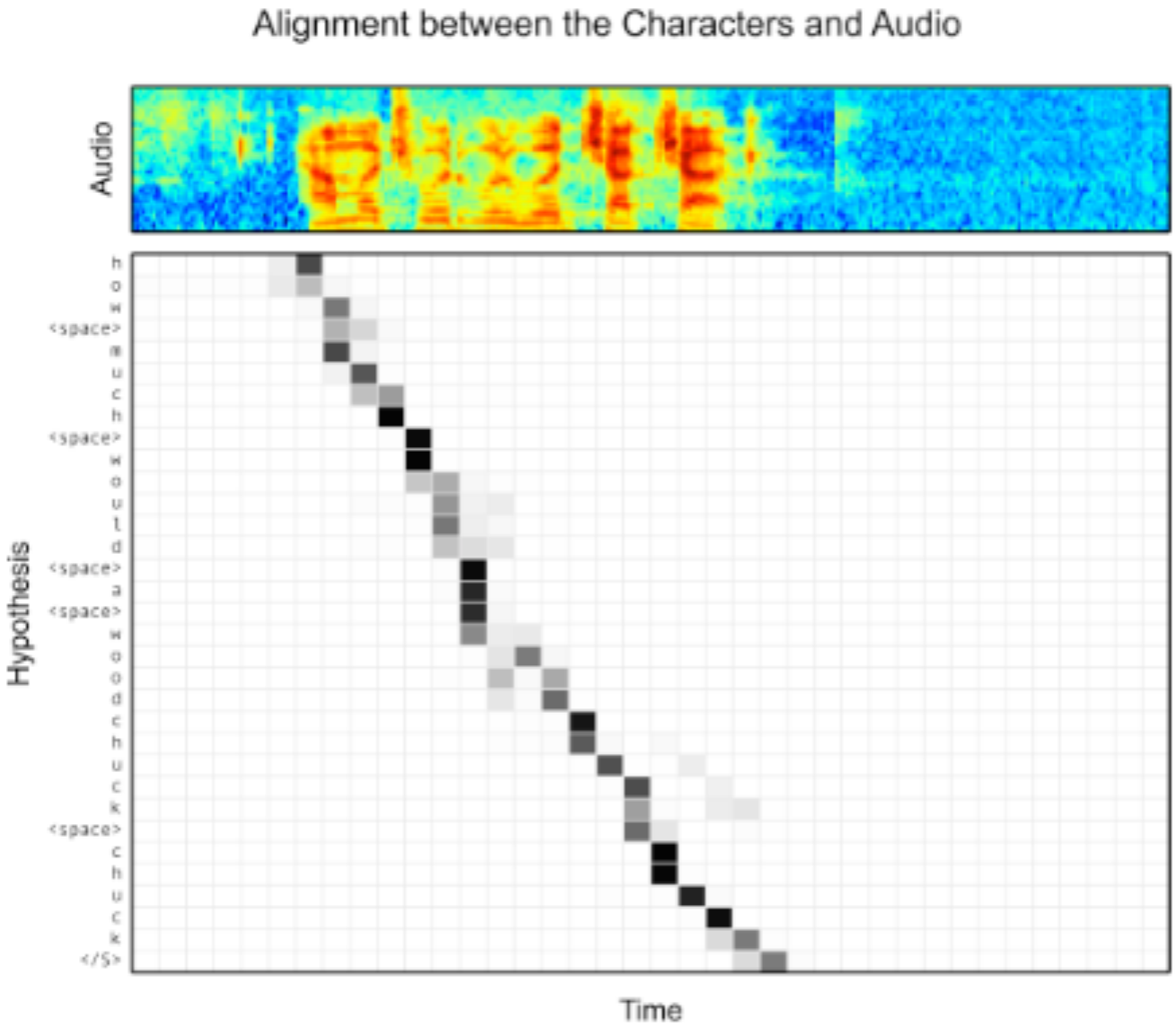


First, merge repeat characters.

Then, remove any ϵ tokens.

The remaining characters are the output.

https://distill.pub/2017/ctc/



Chan, W. Listen, Attend and Spell, 2015

Other uses of Neural network models

- Word meaning representation (Word2Vec)
- Machine translation (Encoder-Decoder framework)

Encoder-decoder

- We know neural networks can learn representations internally
- Can we use those internal representations?

One hot encoding

- Categorical representation is usually represented by one hot encoding

- Categorical representations examples:
  - Words in a vocabulary, characters in Thai language

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

- Sparse representation
  - Spare means most dimension are zero

One hot encoding

- Sparse – but lots of dimension
  - Curse of dimensionality
- Does not represent meaning.

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

|Apple – Bird| = |Bird – Cat|

Getting meaning into the feature vectors

- You can add back meanings by hand-crafted rules
- Old-school NLP is all about feature engineering
- Word segmentation example:
  - Cluster numbers
  - Cluster letters
- Concatenate them
- 1 = [0 0 0 0 1 0 0 0, 1, 0]
- ก = [0 0 0 1 0 0 0 0, 0, 1]
- า = [1 0 0 0 0 0 0 0, 0, 2]
- Which rules to use?
  - Try as many as you can think of, and do feature selection or use models that can do feature selection

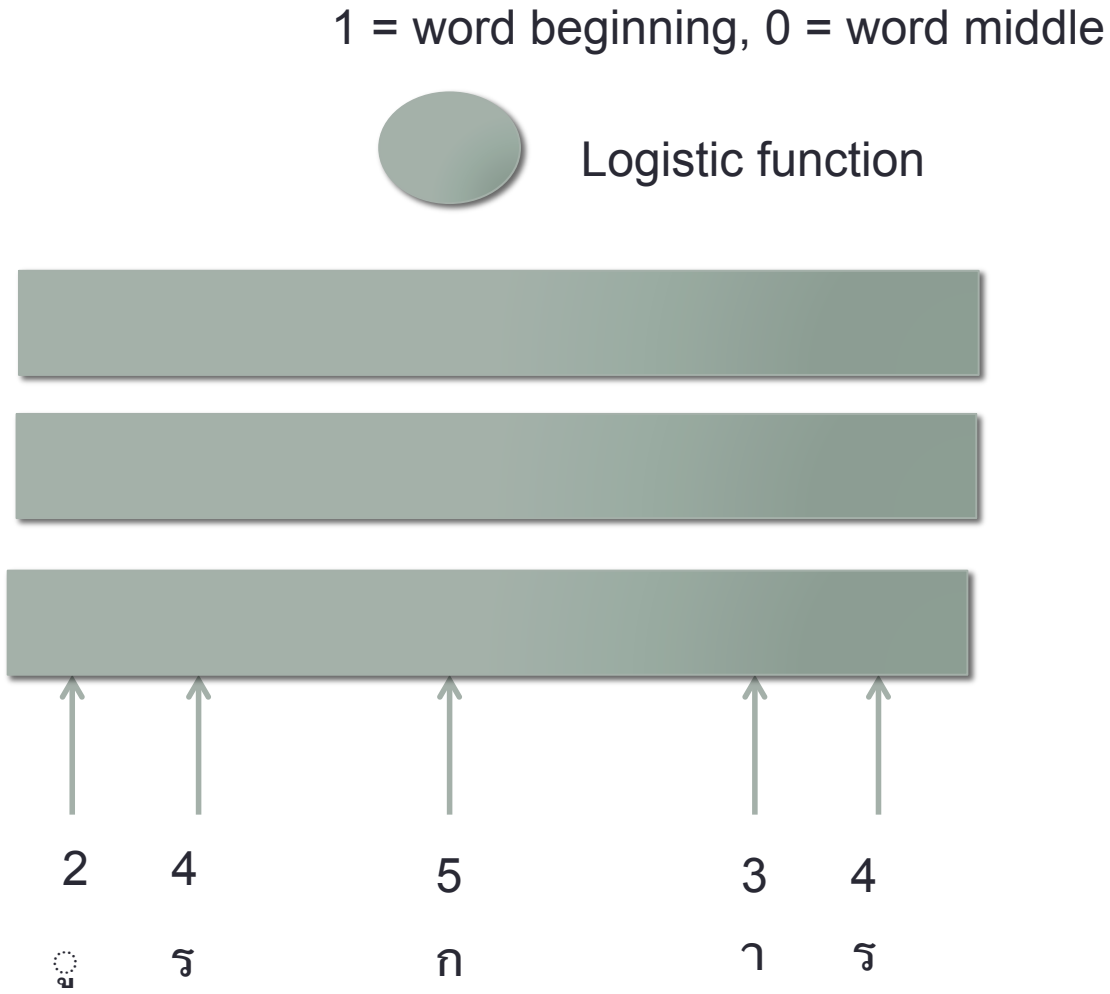
Dense representation

- We can encode sparse representation into a lower dimensional space
  - F: R^N -> R^M, where N>M

Apple -> 1 -> [1, 0, 0, 0, …] -> [2.3, 1.2]
Bird -> 2 -> [0, 1, 0, 0, …] -> [-1.0, 2.4]
Cat -> 3 -> [0, 0, 1, 0, …] -> [-3.0, 4.0]

- We can do this by using an embedding layer

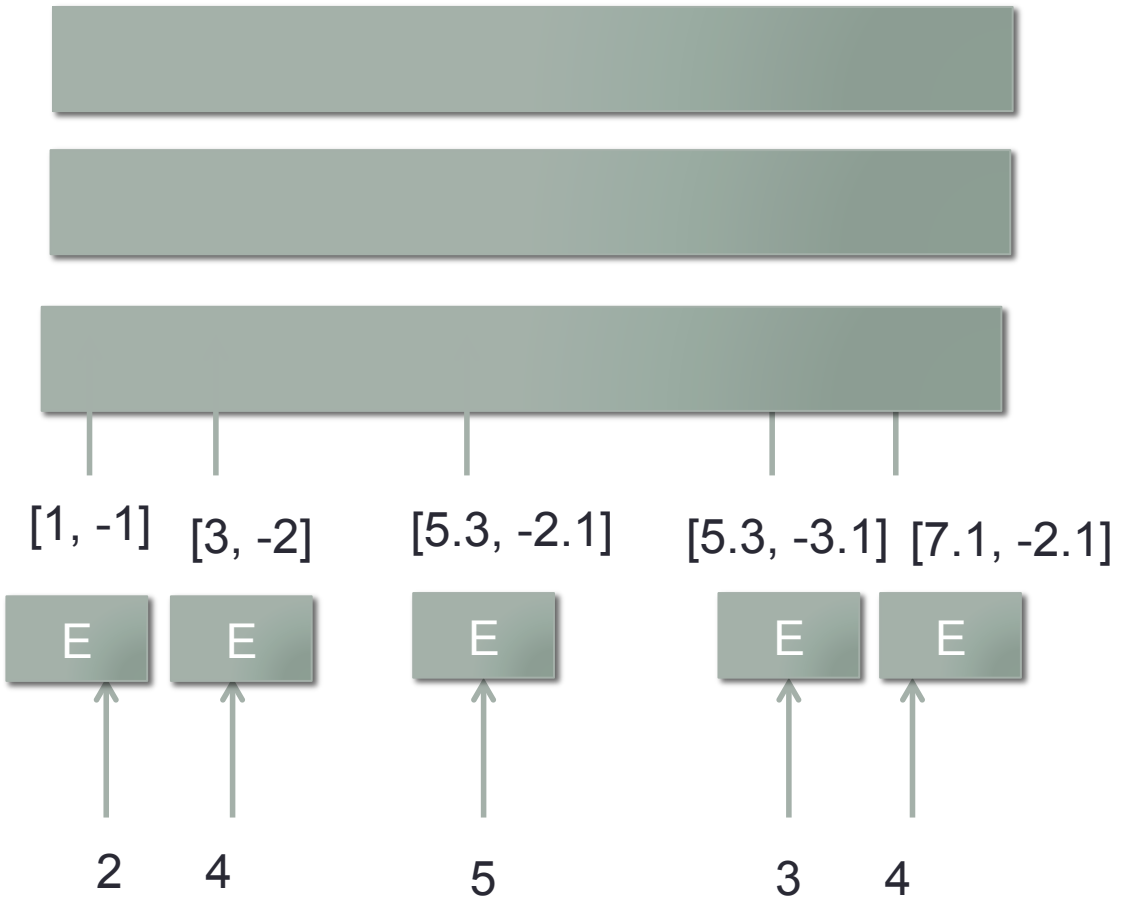
Word segmentation with fully connected networks



Adding embedding layer

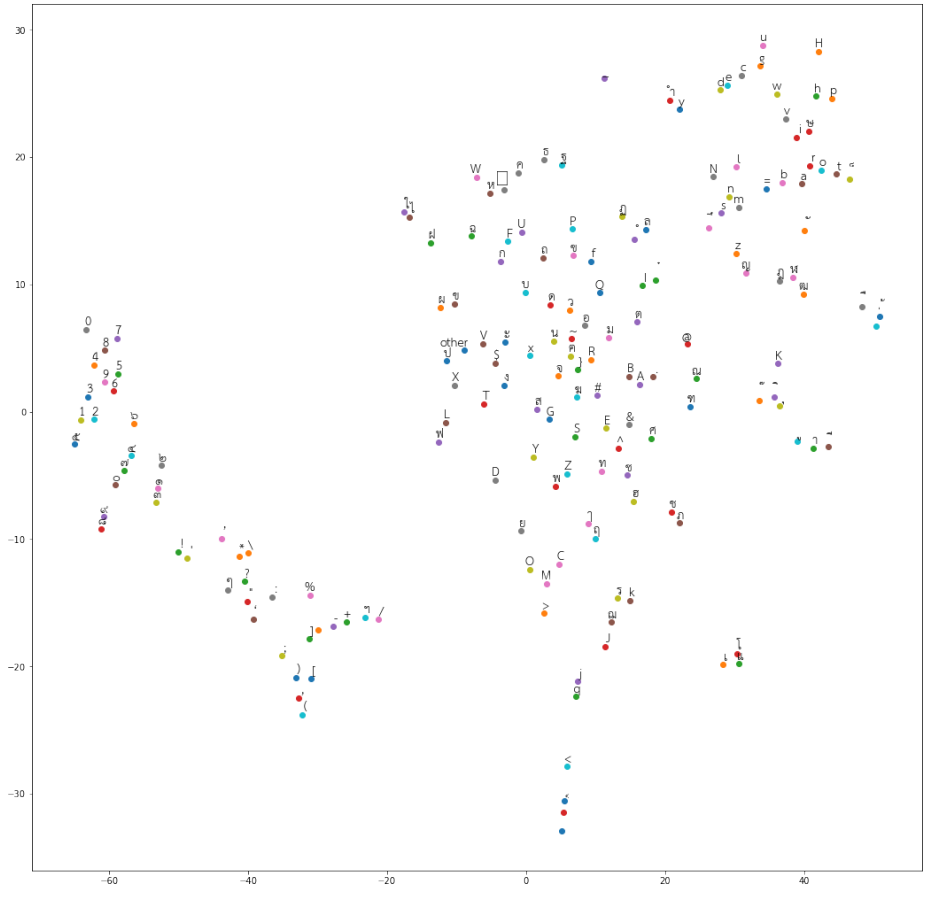
Embedding layer shares the same weights

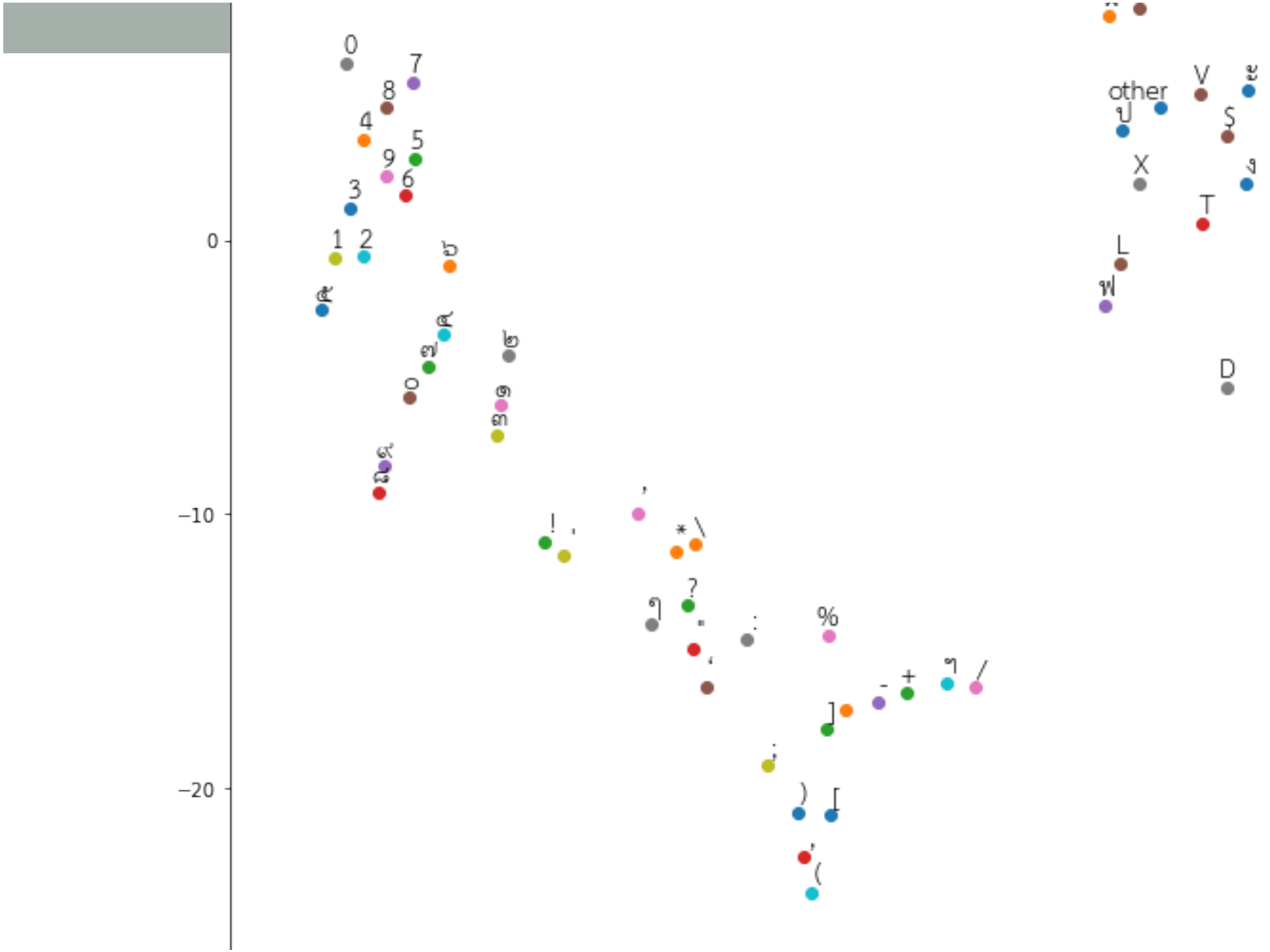
Parameter sharing!

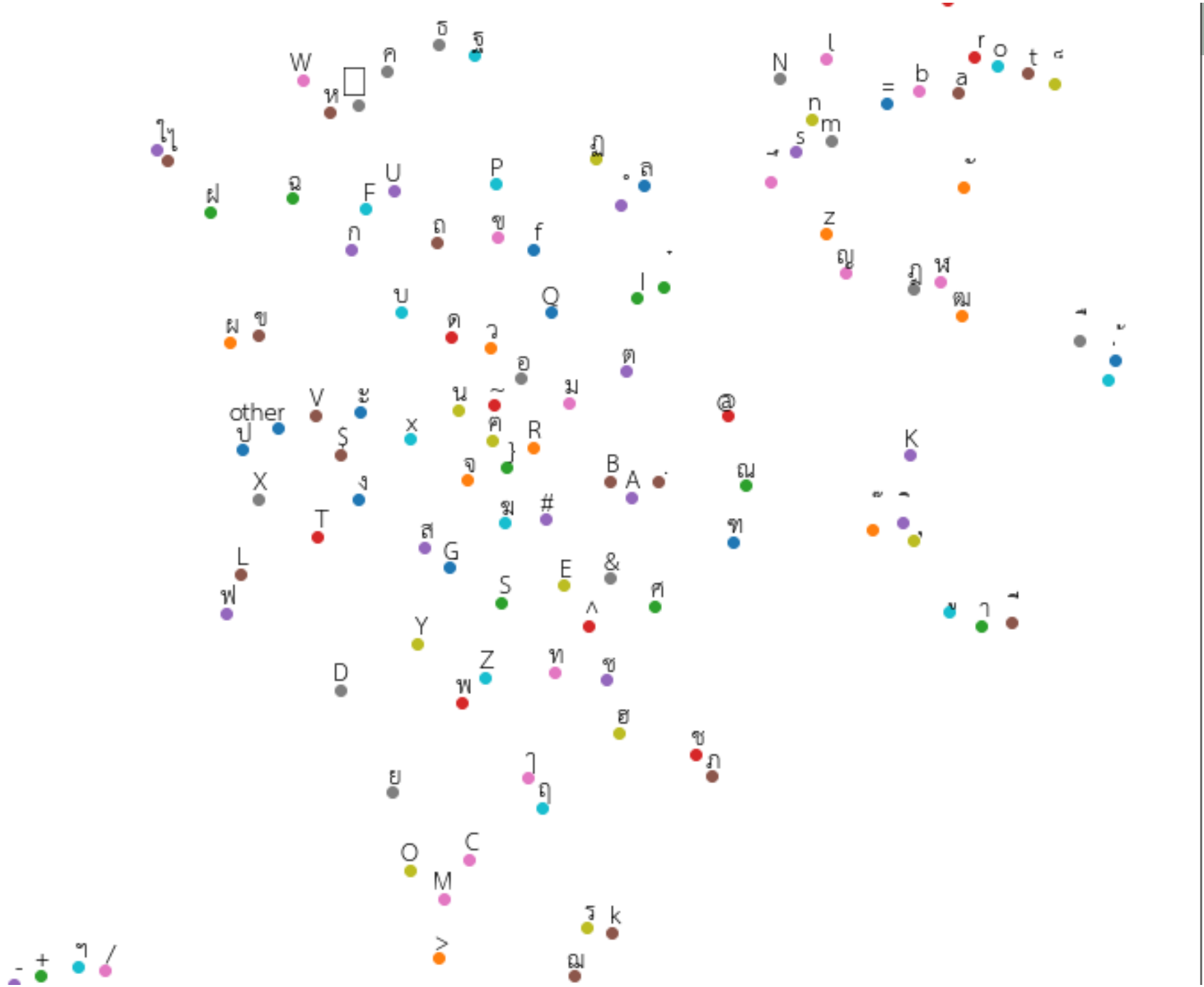


Embedding and meaning (semantics)

- Meaning is inferred from the task
- Embedding of 32 dimensions -> t-SNE into 2 dimension for visualization
- Automatically!

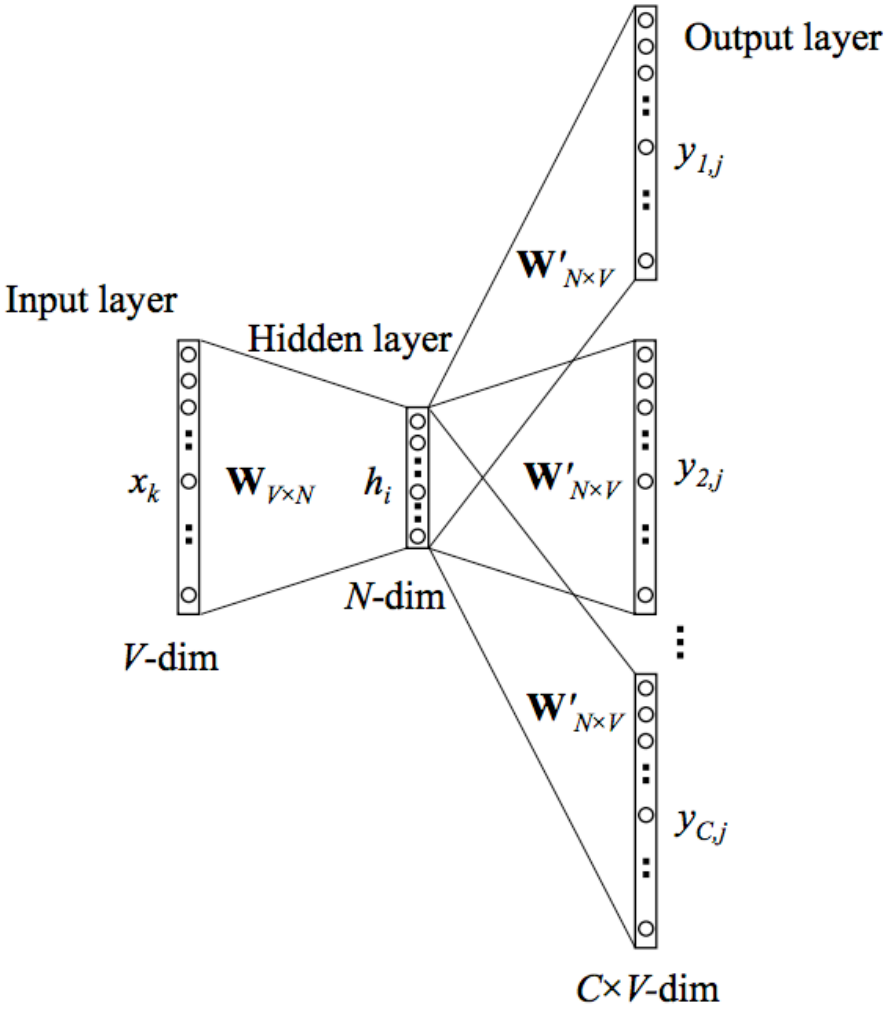



0
−10
−20
other


other

Word2Vec

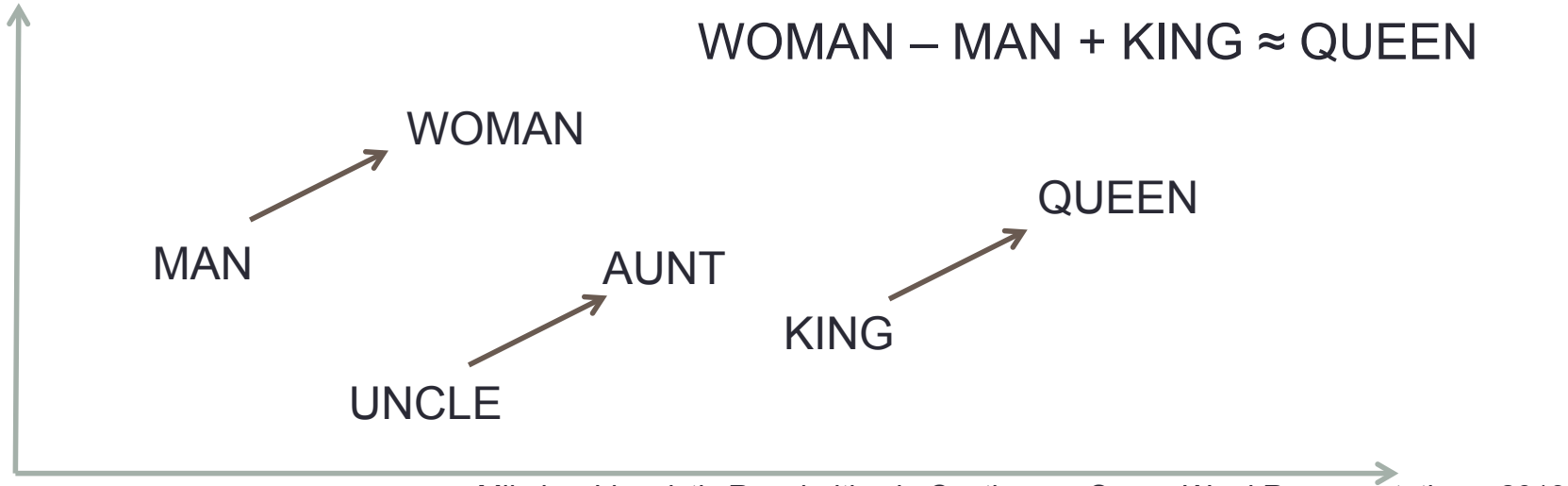
- Maps a word to a vector representation using neural networks
- Input word, predicts word around it
  - Words meaning can be captured by the context words
- Mary ___ an apple.



Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Word2Vec

- Vectors from similar words are near each other
- You can also add and subtract meanings just like numbers!

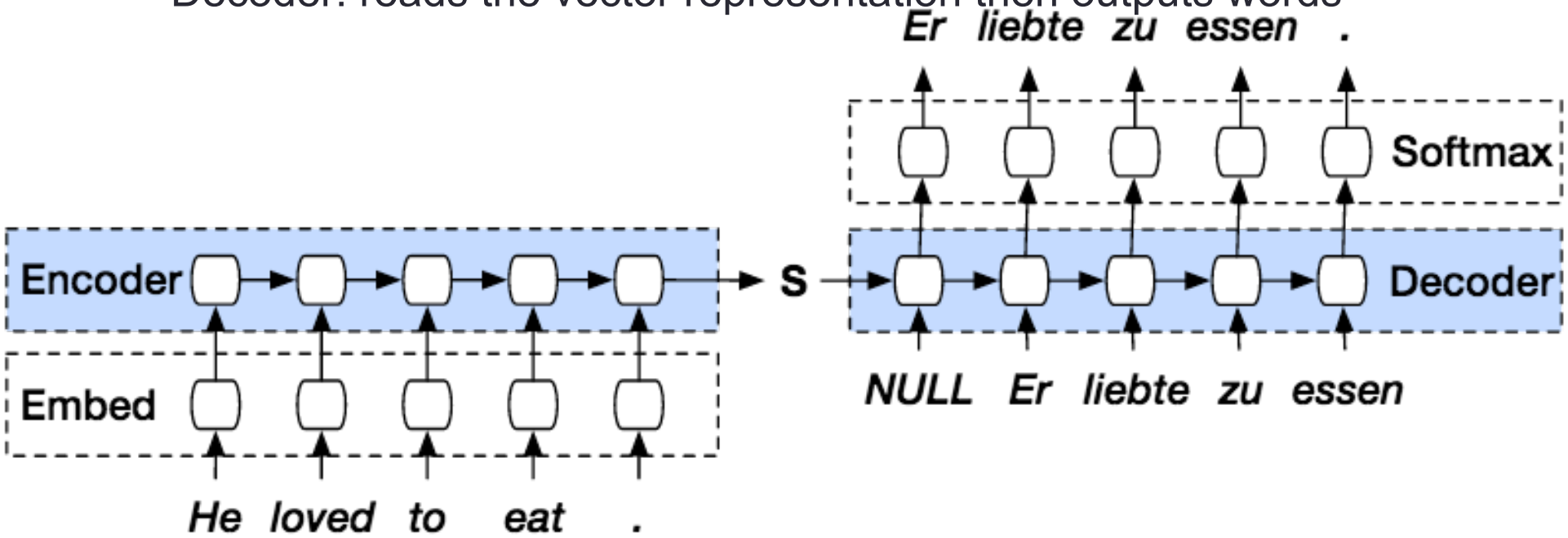


Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Example code for Thai: https://github.com/fooljames/tf-text-workshop

Machine Translation

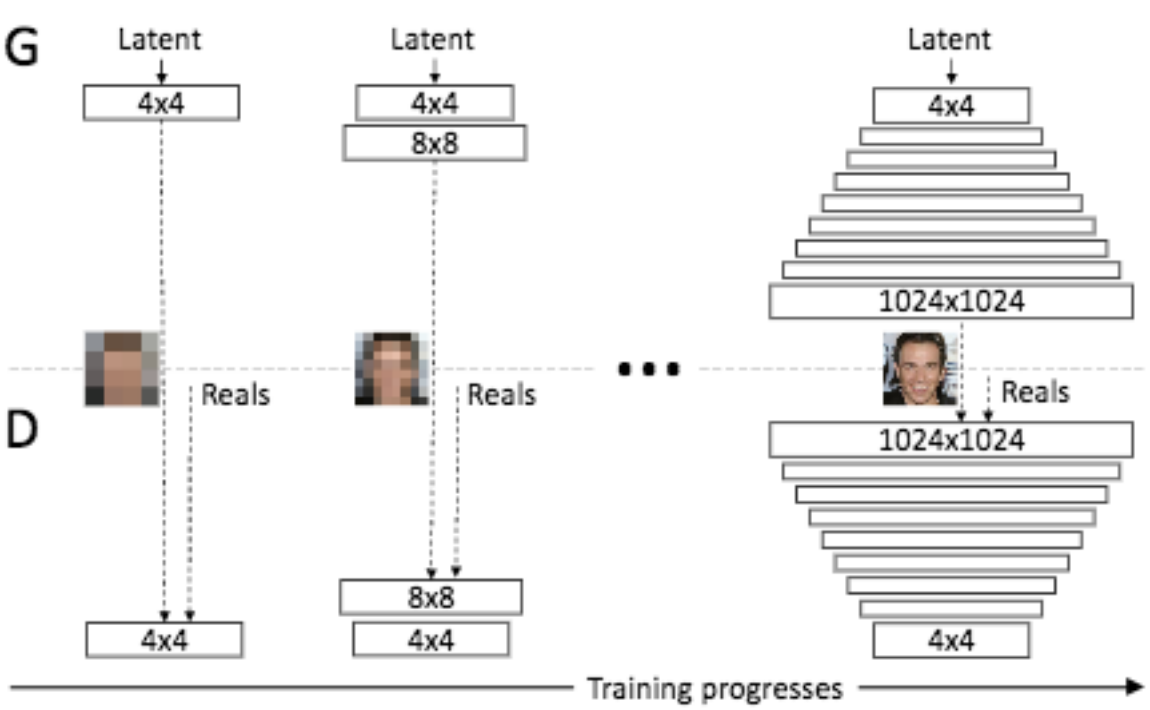
- Given an input sentence. Output another sentence.
- Two RNNs/LSTMs: encoder and decoder
  - Encoder: reads input, generate a vector representation (memory cell of LSTM for example)
  - Decoder: reads the vector representation then outputs words



Ilya Sutskever, Sequence to Sequence Learning with Neural Networks, 2014, https://arxiv.org/abs/1409.3215

Other applications

- Generating fake pictures



http://research.nvidia.com/publication/2017-10_Progressive-Growing-of

Other applications

- Image segmentation

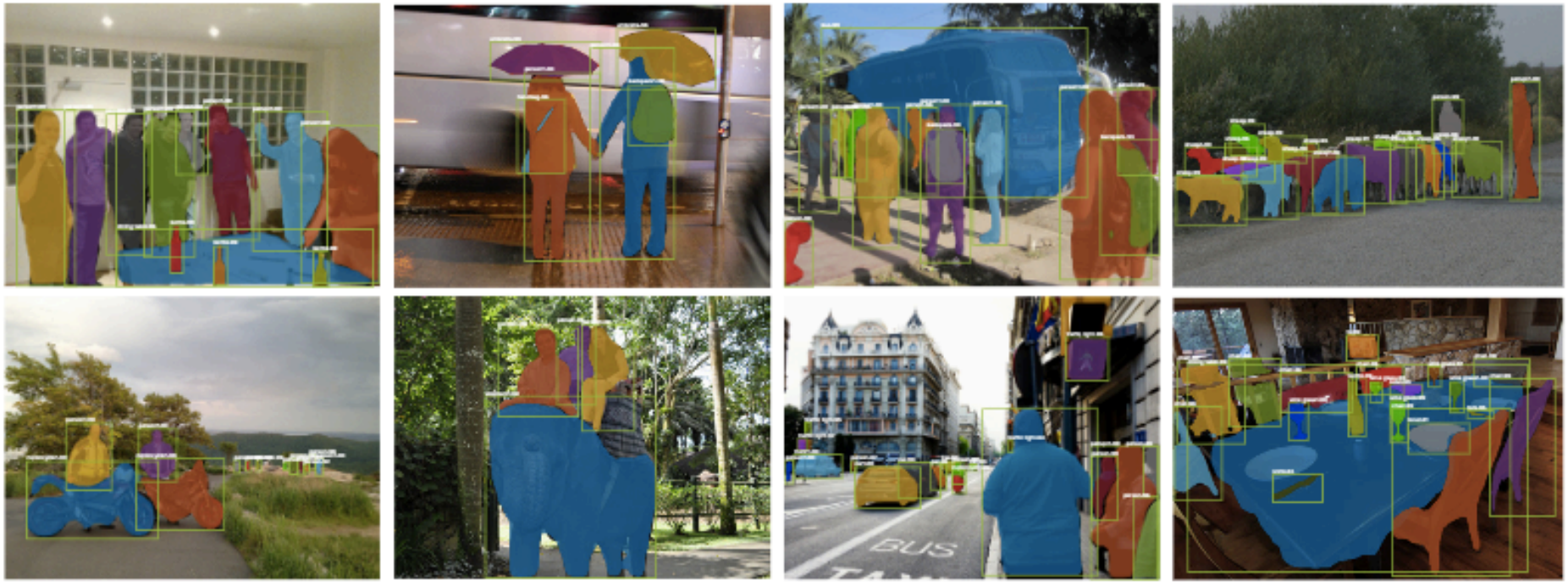
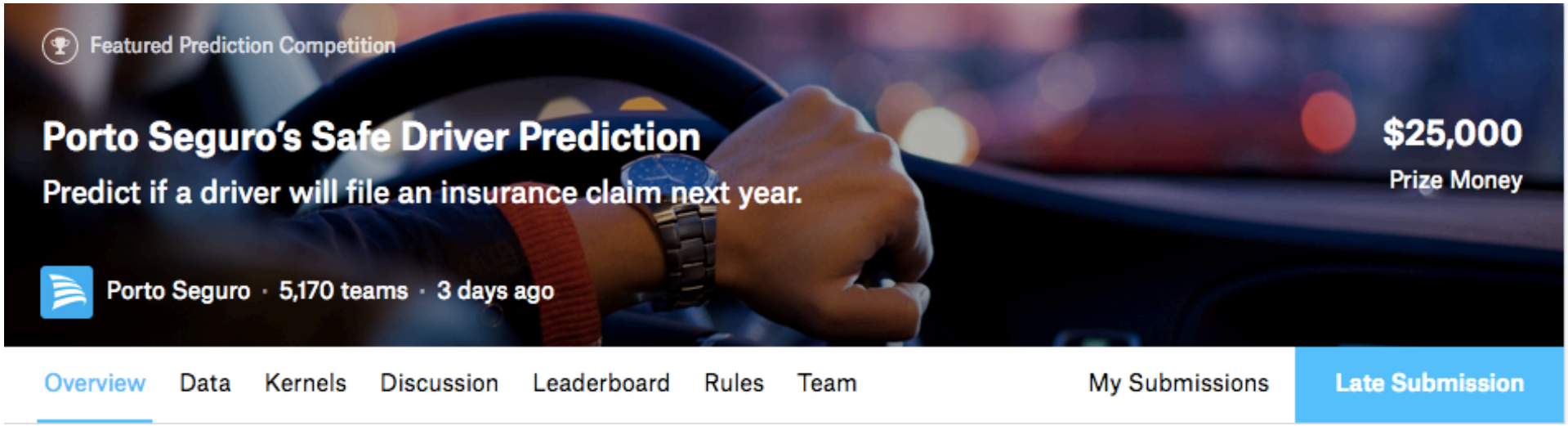


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

https://arxiv.org/pdf/1703.06870.pdf

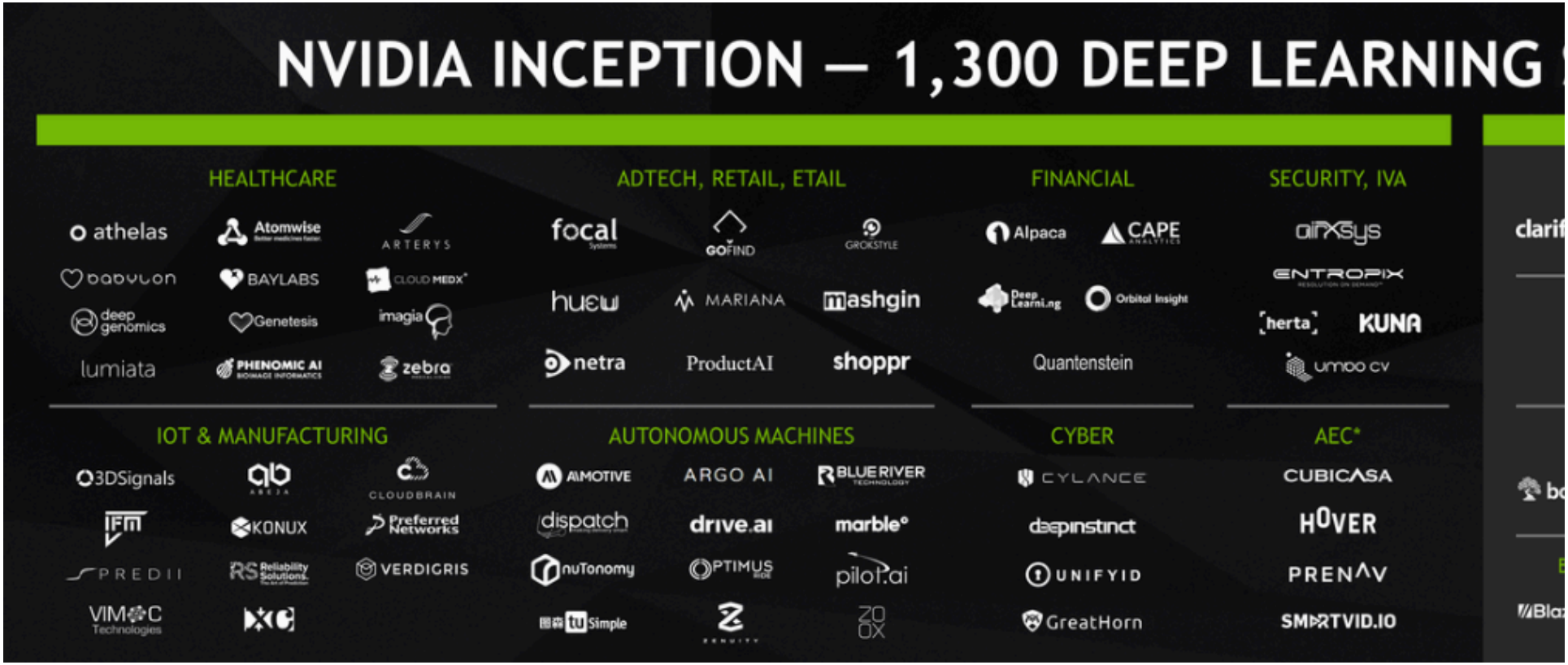
Other applications

- Safe driver prediction



https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/44629

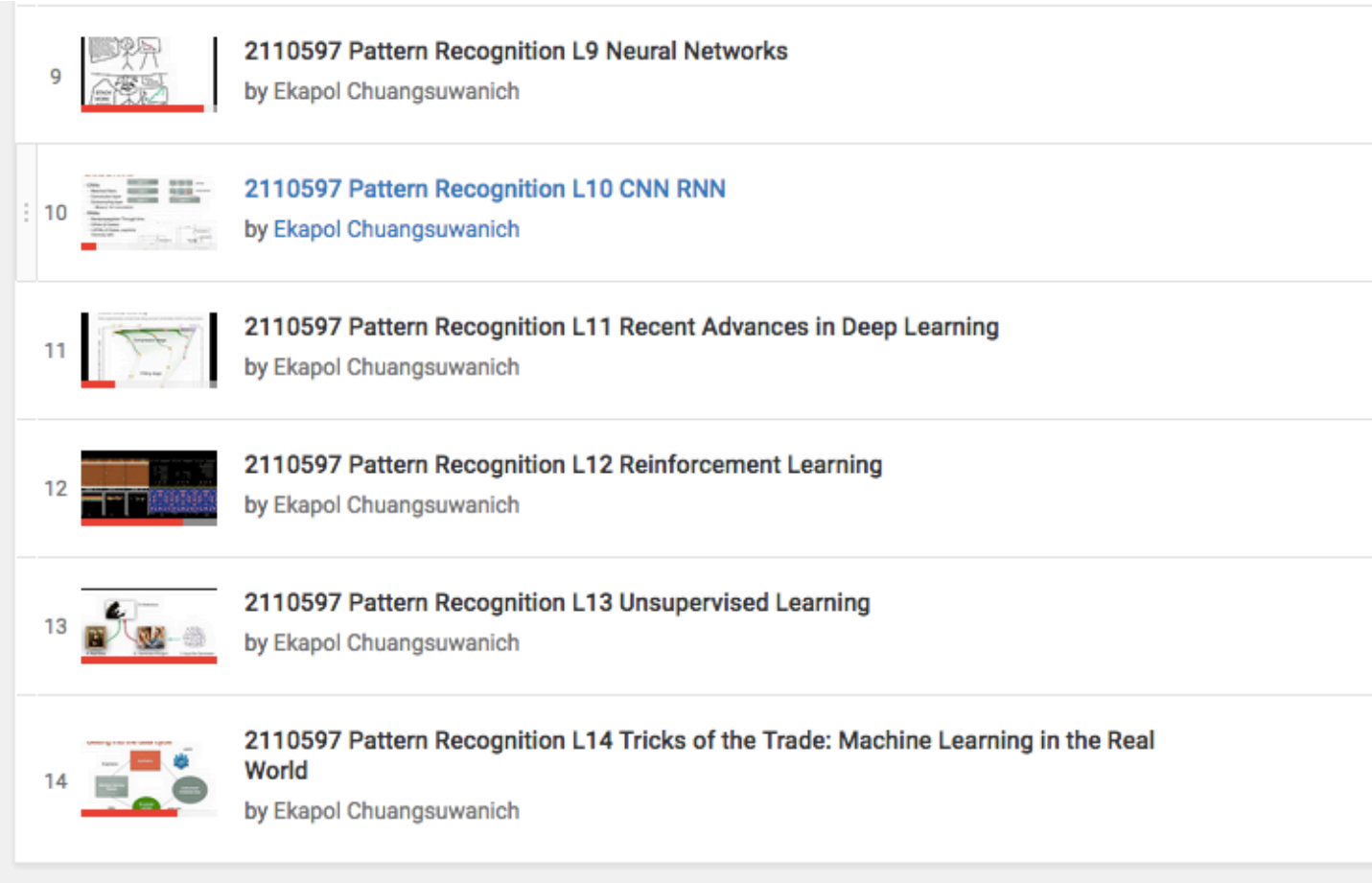
Other applications



https://blogs.nvidia.com/blog/2017/05/09/start-me-up-80-ai-startups-at-gtc-show-how-theyre-changing-industries/

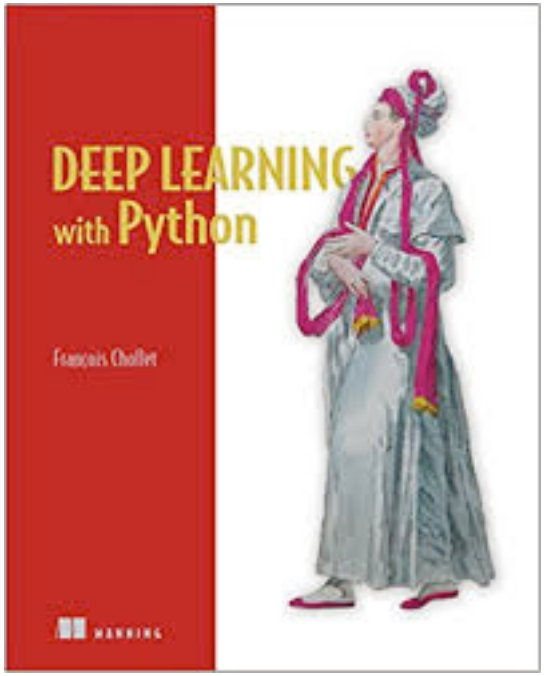
Where to learn more

- https://www.youtube.com/playlist?list=PLcBOyD1N1T-OQd0a6mqjY6gWOuII_stuv&disable_polymer=true



Where to learn more

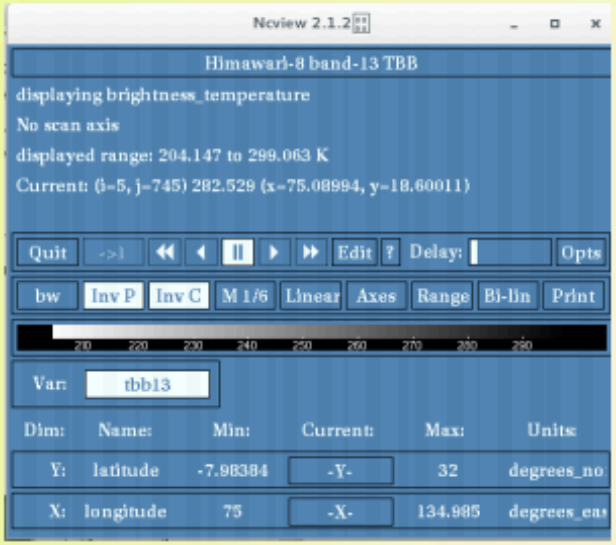
- cs231n
- http://cs231n.stanford.edu/

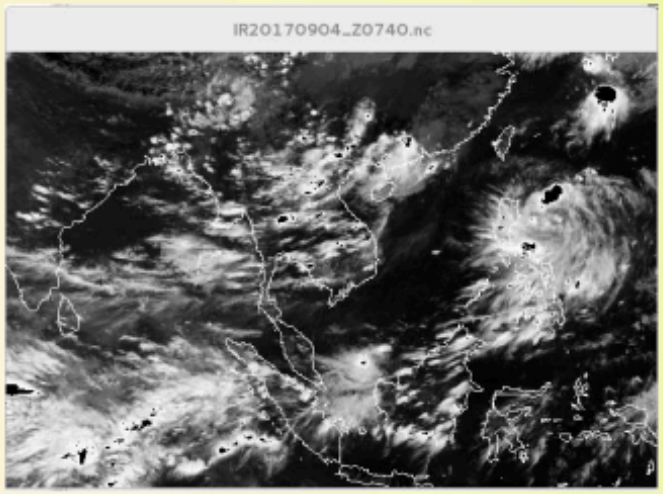


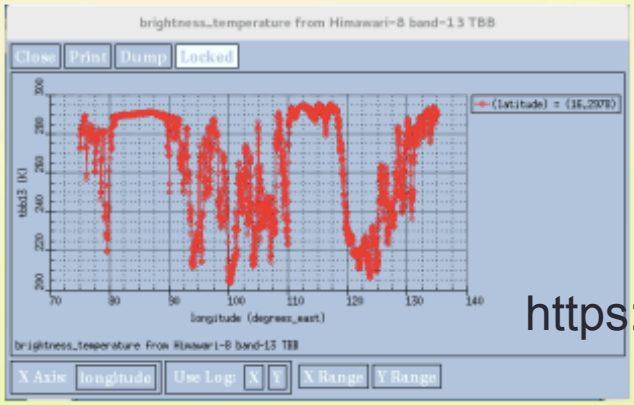
- Deep learning in python
- https://www.manning.com/books/deep-learning-with-python

Possible research topics

- Rain prediction from satellite image







https://arxiv.org/pdf/1506.04214.pdf

Possible research topics

- Thai NLP
  - Internet text normalization
  - PoS tagging
- Call center monitoring