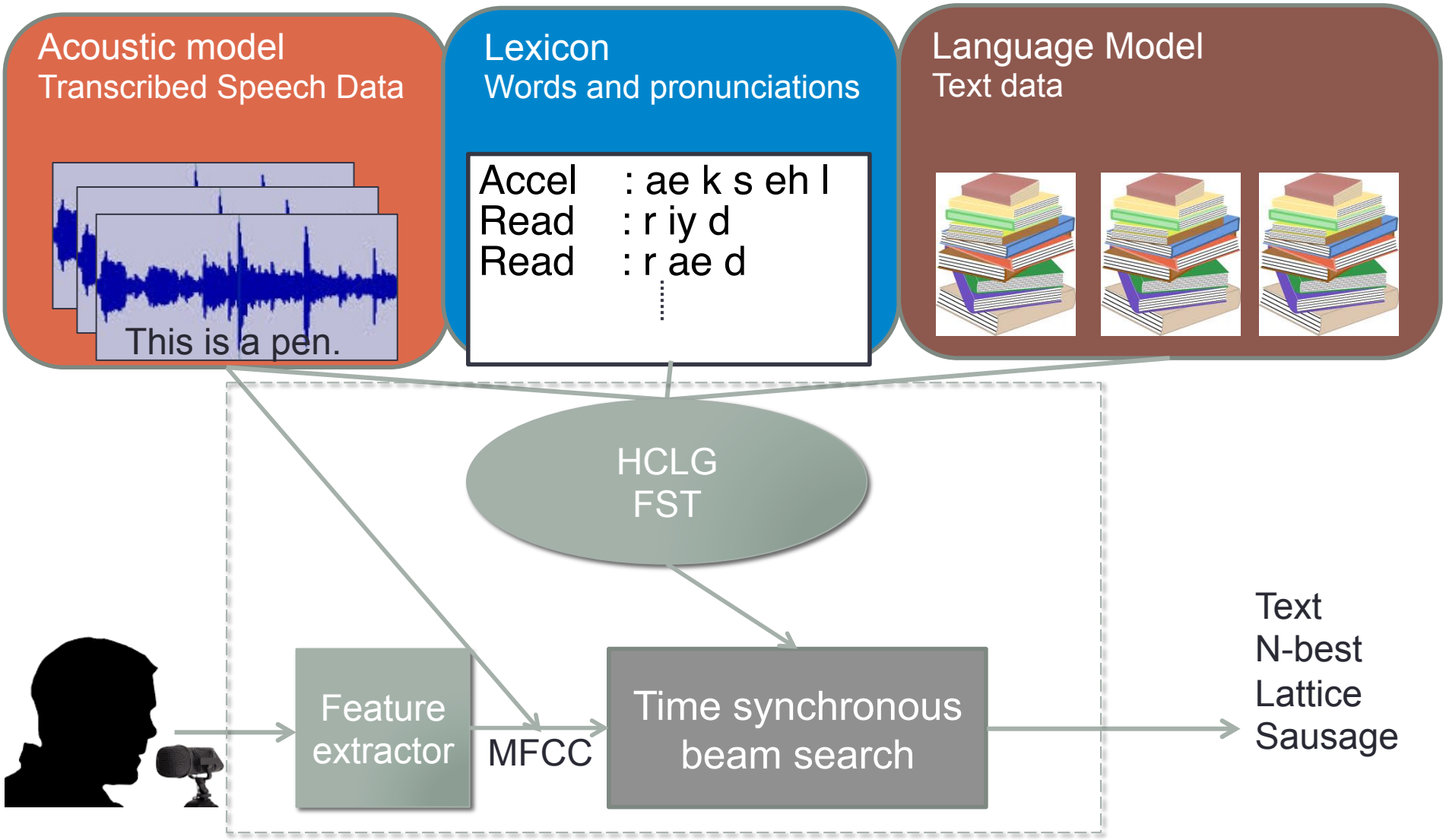
DEEP NEURAL NETWORKS

Quiz

- thank you
- you know about ASR
- you know about gowajee
- thank you gowajee
- what is ASR all about

- Using bigrams find the probability of the following sentences. Don't forget about <start> and <end>
- P(you know about ASR) = ?
- P(thank you know about) = ?

Inside the recognizer



Machine learning basics

- Discriminative vs Generative models
- Regression
  - Cost/Loss function
  - Gradient descent
  - Decision boundary
  - Overfitting vs Underfitting

Generative Models

- HMMs, GMMs, and n-grams are generative models
- Learn the model for each class y given input features x p(x|y). p(x|y) is usually called the likelihood probability.
  - P(x| y = /k/) , P(x| y = /ae/)
- To do classification we want to solve for the best y given input feature x

$$y^* = argmax_y P(y|x)$$

- We can use Bayes' rule

$$y^* = argmax_y \frac{P(x|y)P(y)}{P(x)}$$

Generative Models

$$y^* = argmax_y \frac{P(x|y)P(y)}{P(x)}$$

- P(y) is called the prior probability
- P(x) is ignored since we only care for argmax wrt. Y

- Can we use P(y|x) instead?

$$y^* = argmax_y P(y|x)$$

Discriminative models

- Discriminative models model P(y|x) directly

$$y^* = argmax_y P(y|x)$$

- P(y|x) is called the posterior probability
- Generally, P(y|x) can be any function h(y,x) that gives a score for each class

- Examples of discriminative models support vector machines, logistic regression, neural networks.

Discriminative vs Generative

- Model the posterior P(y|x)
- Care about how to *discriminate* between different classes
- Usually outperforms generative models in classification tasks
- Need to retrain the whole model

- Model the likelihood P(x|y)
- Learns about how x is *generated* from y
- Worse performance but can be used for other tasks (simulate data)
- Easy to add a new class y' – train P(x|y = y')

INTRO TO LINEAR REGRESSION

Predicting amount of rainfall

https://esan108.com/%E0%B8%9E%E0%B8%A3%E0%B8%B0%E0%B9%82%E0%B8%84%E0%B8%81%E0%B8%B4%E0%B8%99%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-%E0%B8%AB%E0%B8%A1%E0%B8%B2%E0%B8%A2%E0%B8%96%E0%B8%B6%E0%B8%87.html

Predicting amount of rainfall

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	**76950**
5	1	0	0	7	**30234**
6	0	3	5	7	**123456**
5	0	3	12	0	**89301**
4	3	0	6	7	**?**

We assume the input features have some correlation with the amount of rainfall.

Can we create a model that predict the amount of rainfall?

What is the output?

What is the input (features)?

Predicting the amount of rainfall

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	**76950**
5	1	0	0	7	**30234**
6	0	3	5	7	**123456**
5	0	3	12	0	**89301**
4	3	0	6	7	**?**

- $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$
- Where θs are the parameter of the model
- Xs are values in the table

(Linear) Regression

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$
- θs are the parameter (or weights)
- We can rewrite

Assume x_0 is always 0

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T \mathbf{x}$$

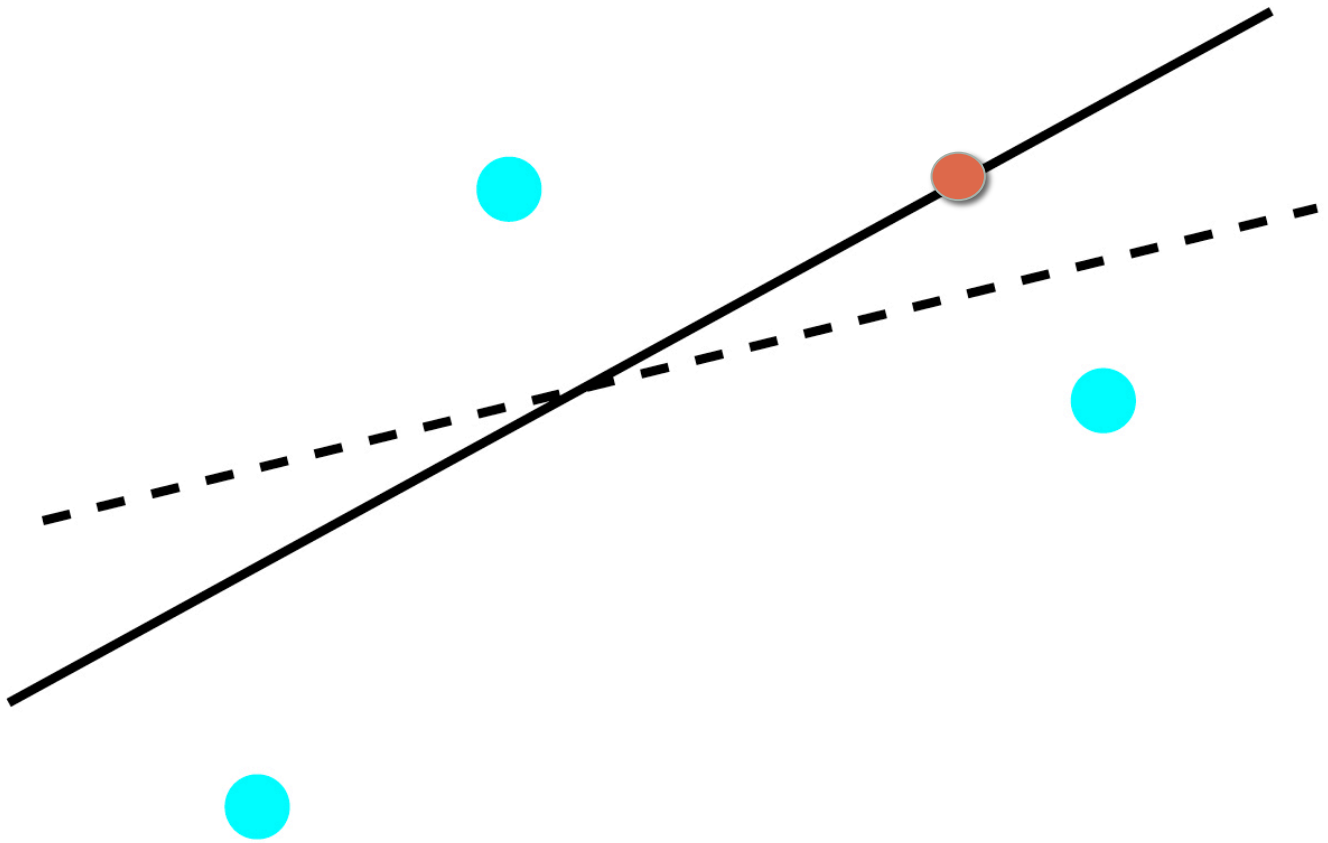
- Notation: vectors are bolded
- Notation: vectors are column vectors

Picking **θ**

- In order to pick **θ** we need to quantify performance.
- How to quantify best performance?

$$h_\theta(x) = \Sigma_{i=0}^{n} \theta_i x_i = \theta^T \mathbf{x}$$

$$h_\theta(x) = \Sigma_{i=0}^{n} \theta_i x_i = \theta^T \mathbf{x}$$



https://chem.libretexts.org/Textbook_Maps/Analytical_Chemistry_Textbook_Maps/Map%3A_Analytical_Chemistry_2.0_(Harvey)/05_Standardizing_Analytical_Methods/5.4%3A_Linear_Regression_and_Calibration_Curves

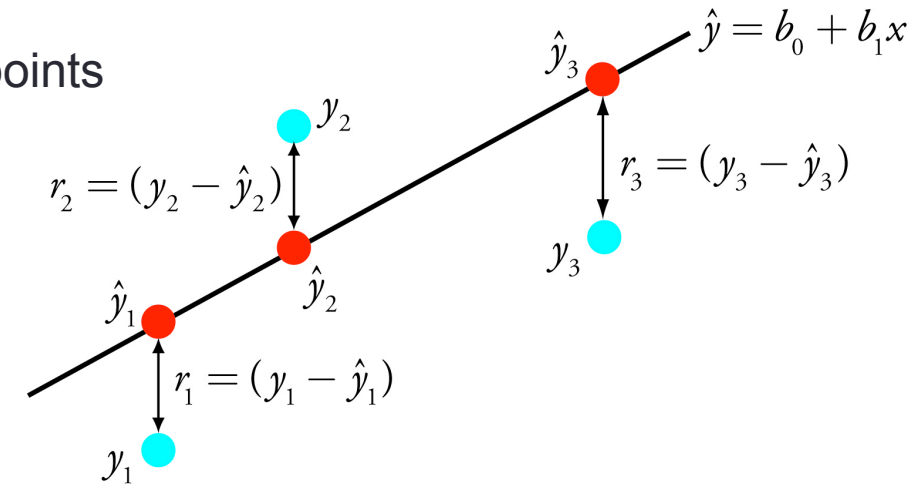
Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

We want to pick **θ** that minimize the loss

m is number of training data points



Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

We want to pick **θ** that minimize the loss

Scaling

$$\frac{m}{2}J(\theta) = \frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

Same **θ** minimizes both function

Picking θ

- How to quantify best performance?
  - Square error loss function

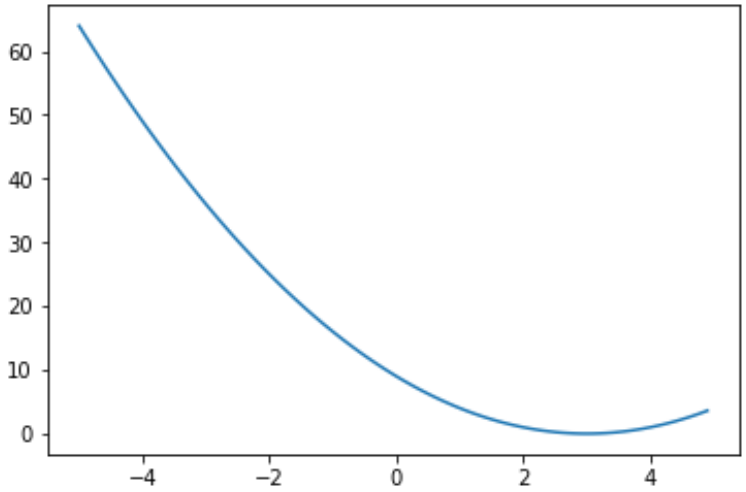
$$J(\theta) = \frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

- How to pick θ to minimize J
  - Random until you get the best performance?
  - Can we do better than random chance?

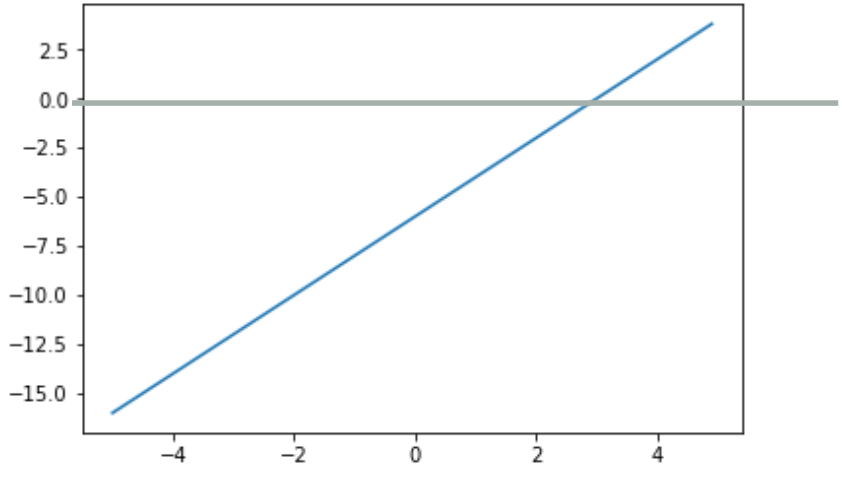
Minimizing a function

- You have a function
  - $y = (x - a)^2$
- You want to minimize Y with respect to x
  - $dy/dx = 2x - 2a$
  - Take the derivative and set the derivative to 0
    - (And maybe check if it's a minima, maxima or saddle point)
- We can also go with an iterative approach

Gradient descent



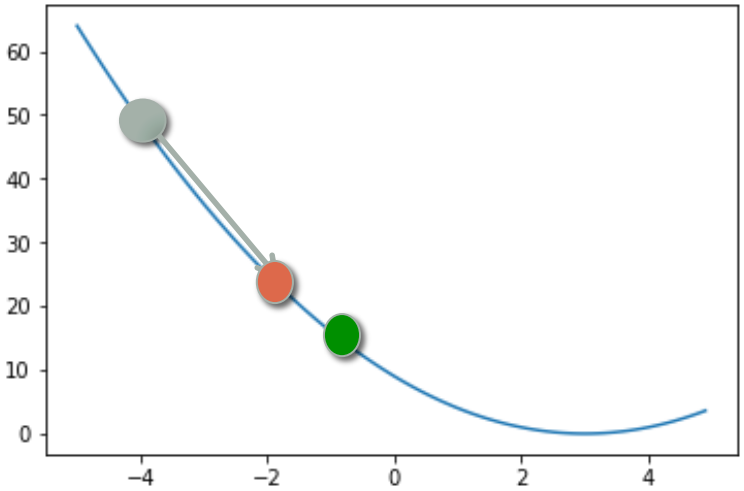
y



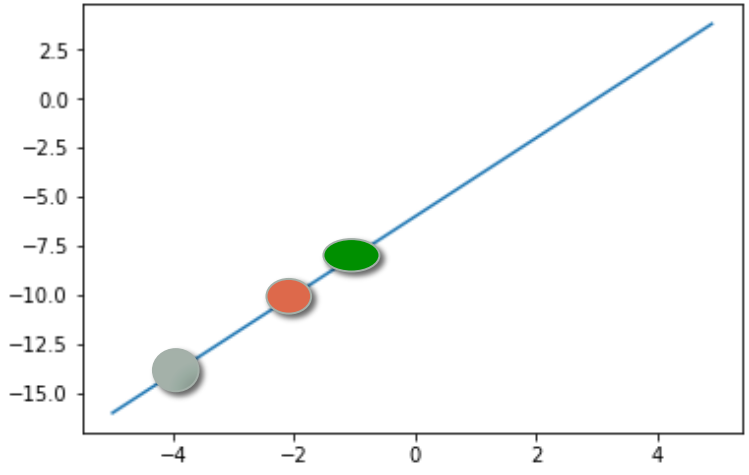
dy/dx

First what does dy/dx means?

Gradient descent



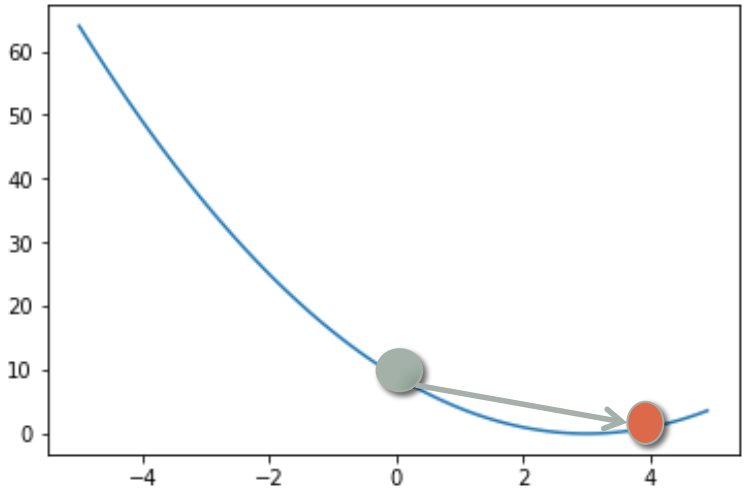
y



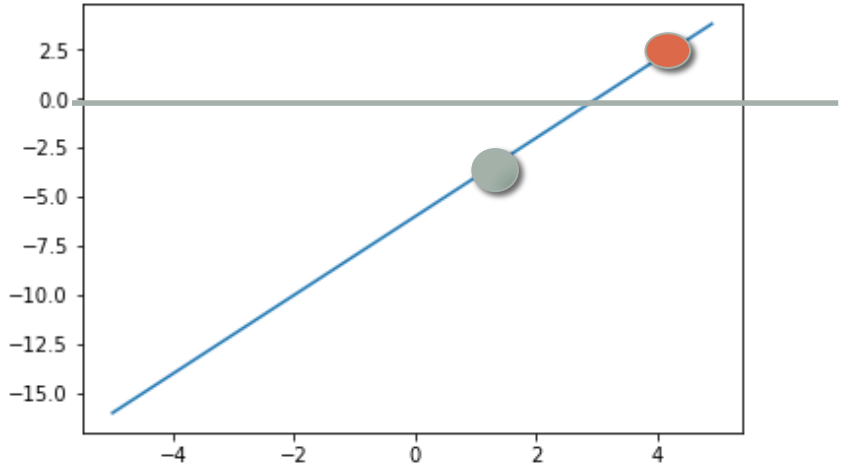
dy/dx

Move along the negative direction of the gradient
The bigger the gradient the bigger step you move

Gradient descent



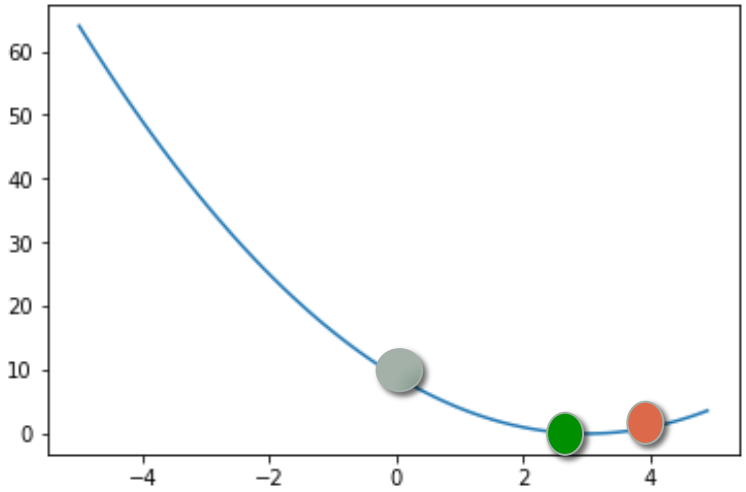
y



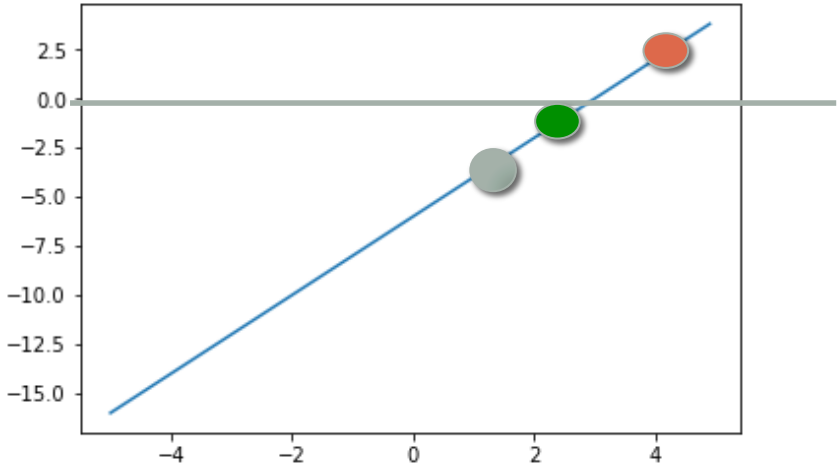
dy/dx

What happens when you overstep?

Gradient descent



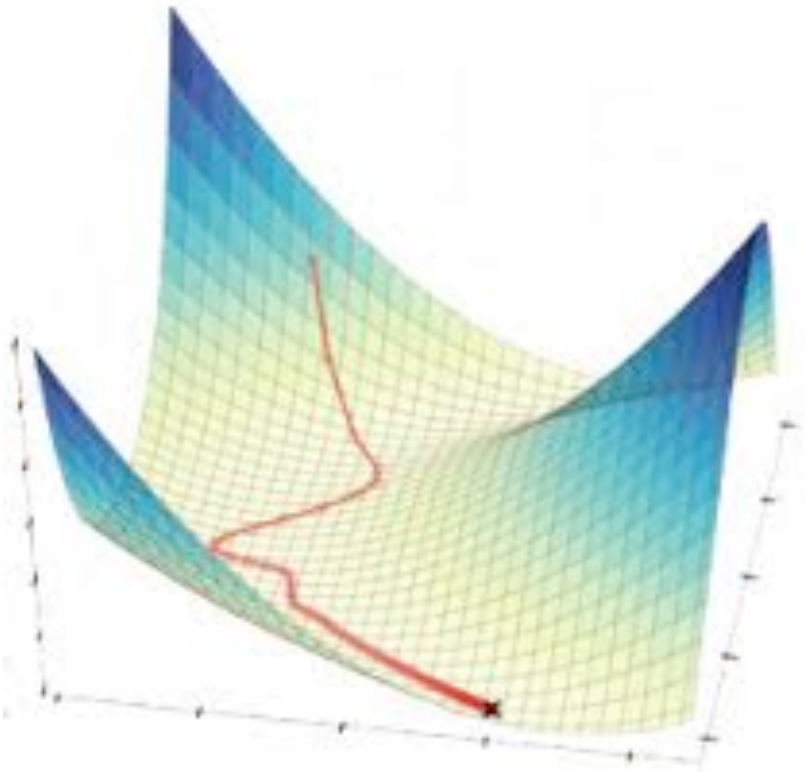
y



dy/dx

If you over step you can move back

Gradient descent in 3d



http://dsdeepdive.blogspot.com/2016/03/optimizations-of-gradient-descent.html

Formal definition

- $y = f(x)$

- Pick a starting point x_0

- Moves along -dy/dx
- $x_{n+1} = x_n - r * dy/dx$
- Repeat till convergence
- r is the learning rate
    - Big r means you might overstep
    - Small r and you need to take more steps

Picking **θ**

- How to quantify best performance?
  - Square error loss function

$$J(\theta) = \frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2$$

- How to pick **θ** to minimize J
  - Gradient descent

LMS regression with gradient descent

$$\frac{\partial J}{\partial \theta_j} = -\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

$$\theta_j \Leftarrow \theta_j + r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

m = number of training example
i = index of training example
j = index of feature dimension

$\mathbf{x}_i =$ $x^{(1)}$ $x^{(2)}$ $x^{(3)}$ … $x^{(j)}$ … $x^{(J)}$

Batch updates vs mini-batch

$$\theta_j \Leftarrow \theta_j - r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

- Batch updates (considering the whole training data) estimate the Loss function precisely
  - Can takes a long time if m is large
- Updates with a subset of m
  - We now have an estimate of the loss function
    - This can lead to a wrong direction, but we get faster updates
  - Called Stochastic Gradient Descent (SGD) or incremental gradient descent

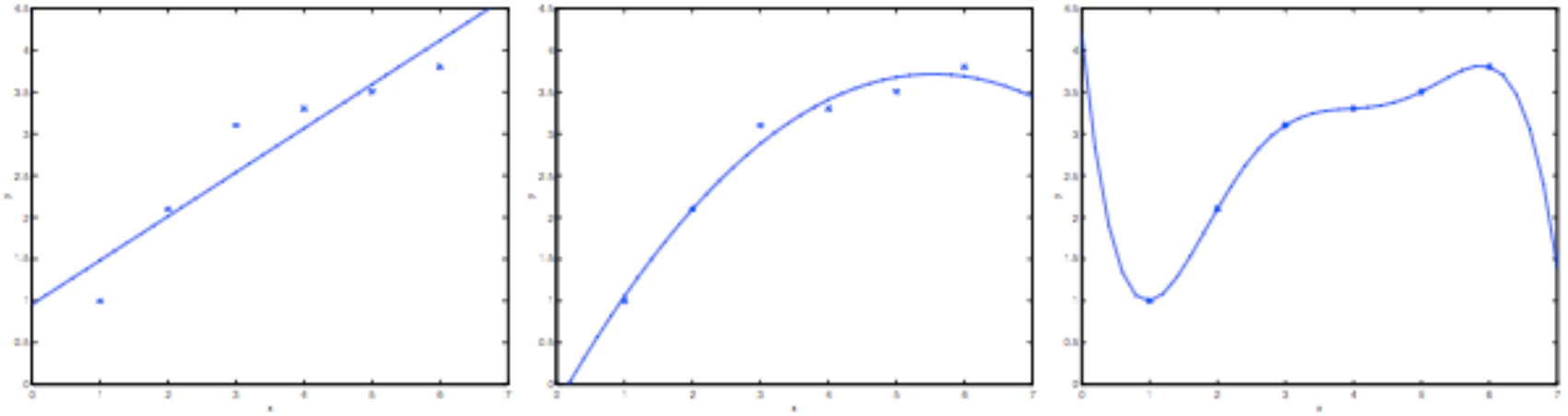
Regression with non-linear features

- If we add extra features that are non-linear
  - For example x^2

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	**76950**
5	1	0	0	7	**30234**
6	0	3	5	7	**123456**
5	0	3	12	0	**89301**
4	3	0	6	7	**?**

- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 + \theta_6 x_1^2 + \ldots$
- These can be considered as additional features
- We can now have a line that is non-linear

Overfitting Underfitting



Adding more non-linear features makes the line more curvy
(Adding more features also means more model parameters)

The curve can go directly to the outliers with enough parameters.
We call this effect overfitting
For the opposite case, having not enough parameters to model the data is called underfitting

Predicting floods

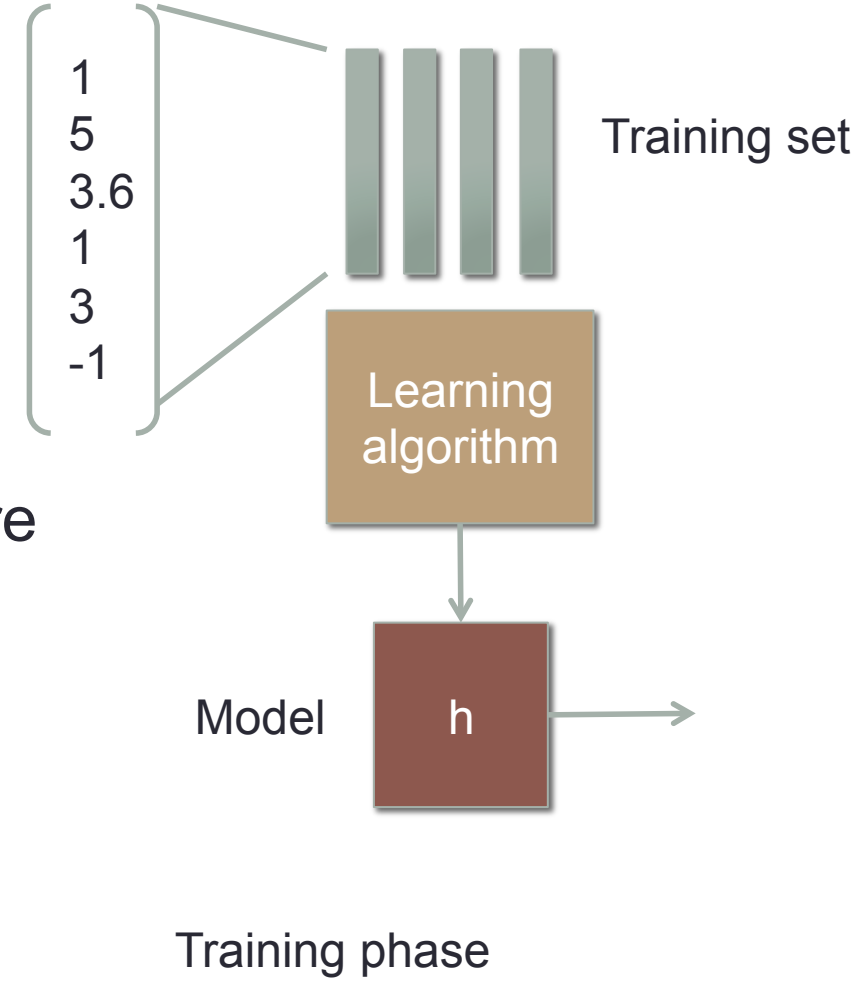
Cloth	Corn	Grass	Water	Beer	Flood?
4	6	3	10	0	**yes**
5	1	0	0	7	**yes**
6	0	3	5	7	**no**
5	0	3	12	0	**yes**
4	3	0	6	7	**?**

So far we talk about predicting an amount what if we want to do classification

Let’s start with a binary choice. Flood or no flood

Flood or no flood

- What would be the output?
- y = 0 if not flooded
- y = 1 if flooded
- Anything in between is a score for how likely it is to flood



Can we use regression?

- Yes
- $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$

- But
- What does it mean when h is higher than 1?
- Can h be negative? What does it mean to have a negative flood value?

- Not that great but we'll use it for now. (Real answer: logistic regression)

Can we use regression?

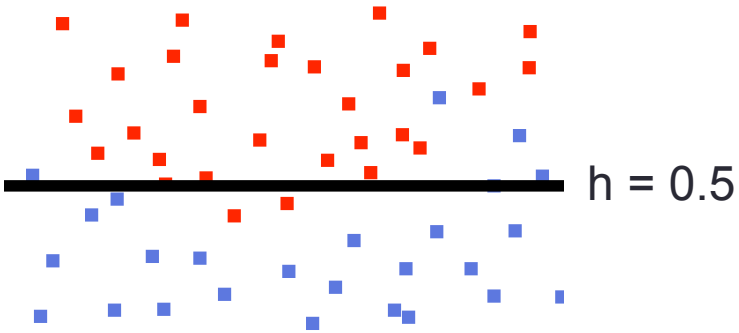
- Yes
- $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$

- Training data
- y = 0 if not flooded
- y = 1 if flooded

- Prediction
- If h > 0.5 flood
- If h < 0.5 no flood

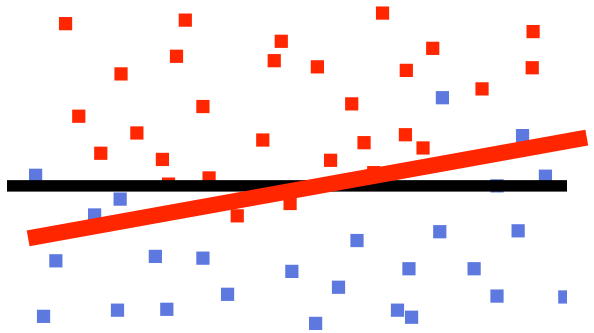
Decision boundary

- The line where our decision flips



Decision boundary

- As we train the model with SGD the decision boundary line will keep shifting

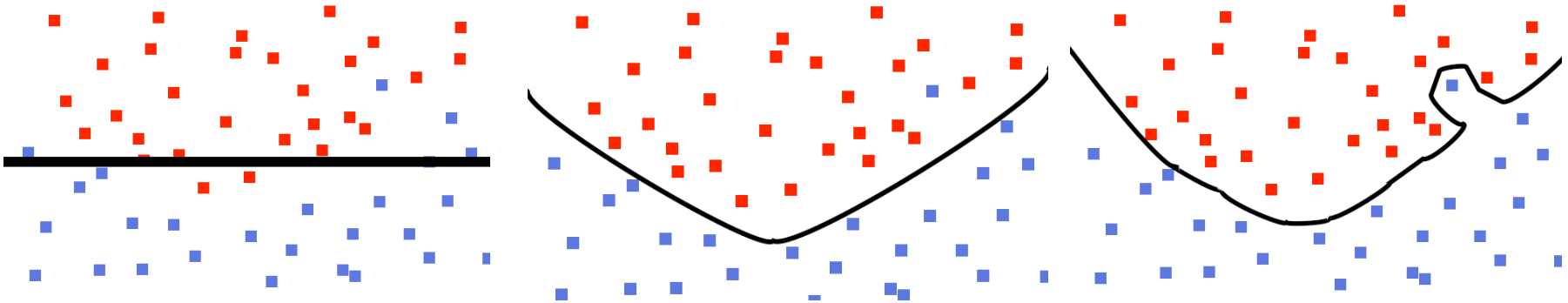


After one update

$$\theta_j \Leftarrow \theta_j - r\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x}_i)x_i^{(j)}$$

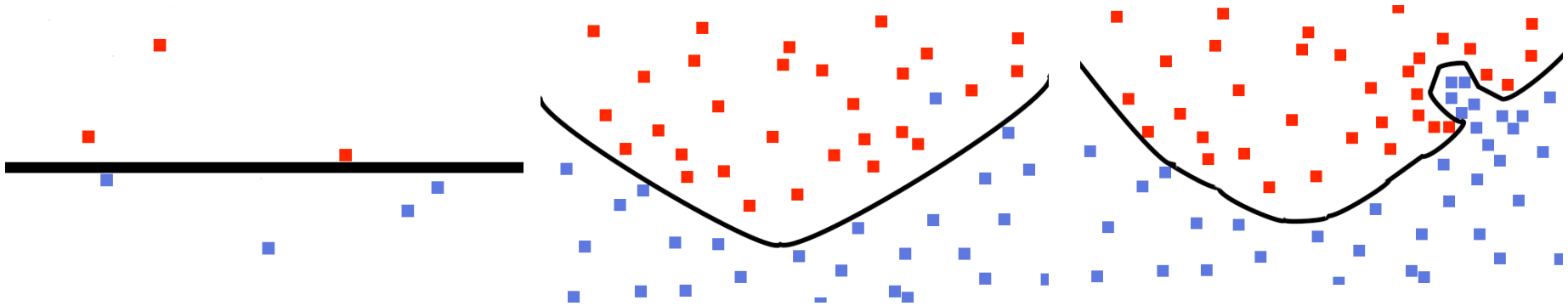
Decision boundary & overfitting

- Usually the more complex the model, the more complex the decision boundary
- One way to gauge the complexity of the model is the number of parameters we need to learn
  - Linear regression with 2 features – 3 features
  - Linear regression with 2 features and x^2 features – 6 features
  - 2 class with Gaussian models – 2x2 parameters
  - Unigram model with V = 10 – 9 parameters



Overfitting vs Underfitting

- It is hard to know beforehand how which model is the one that does not overfit or underfit
- The best model complexity depends on amount of data
- The best way to combat overftting is to evaluate multiple models on the dev set
- Data limitation is also known as the data sparsity problem



Decision boundary

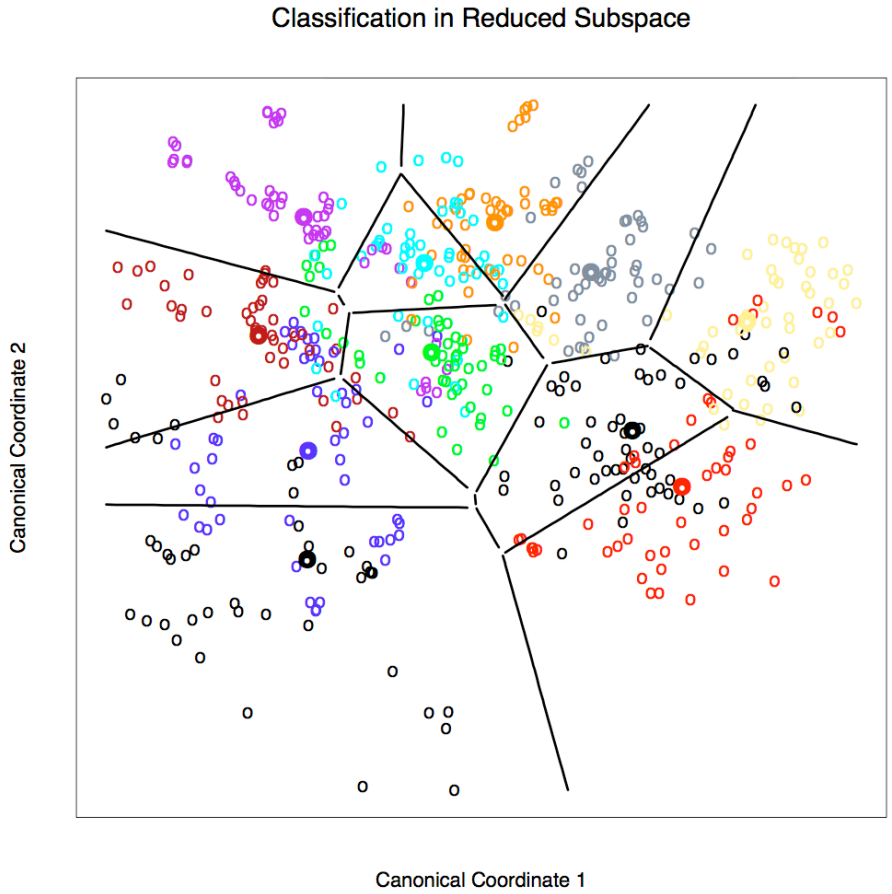


FIGURE 4.11. *Decision boundaries for the vowel training data, in the two-dimensional subspace spanned by the first two canonical variates. Note that in any higher-dimensional subspace, the decision boundaries are higher-dimensional affine planes, and could not be represented as lines.*

For higher dimensional inputs the decision boundary is a hyperplane

Trevor Hastie, The Elements of Statistical Learning, 2009

DEEP NEURAL NETWORKS

DNNs (Deep Neural Networks)

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

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%

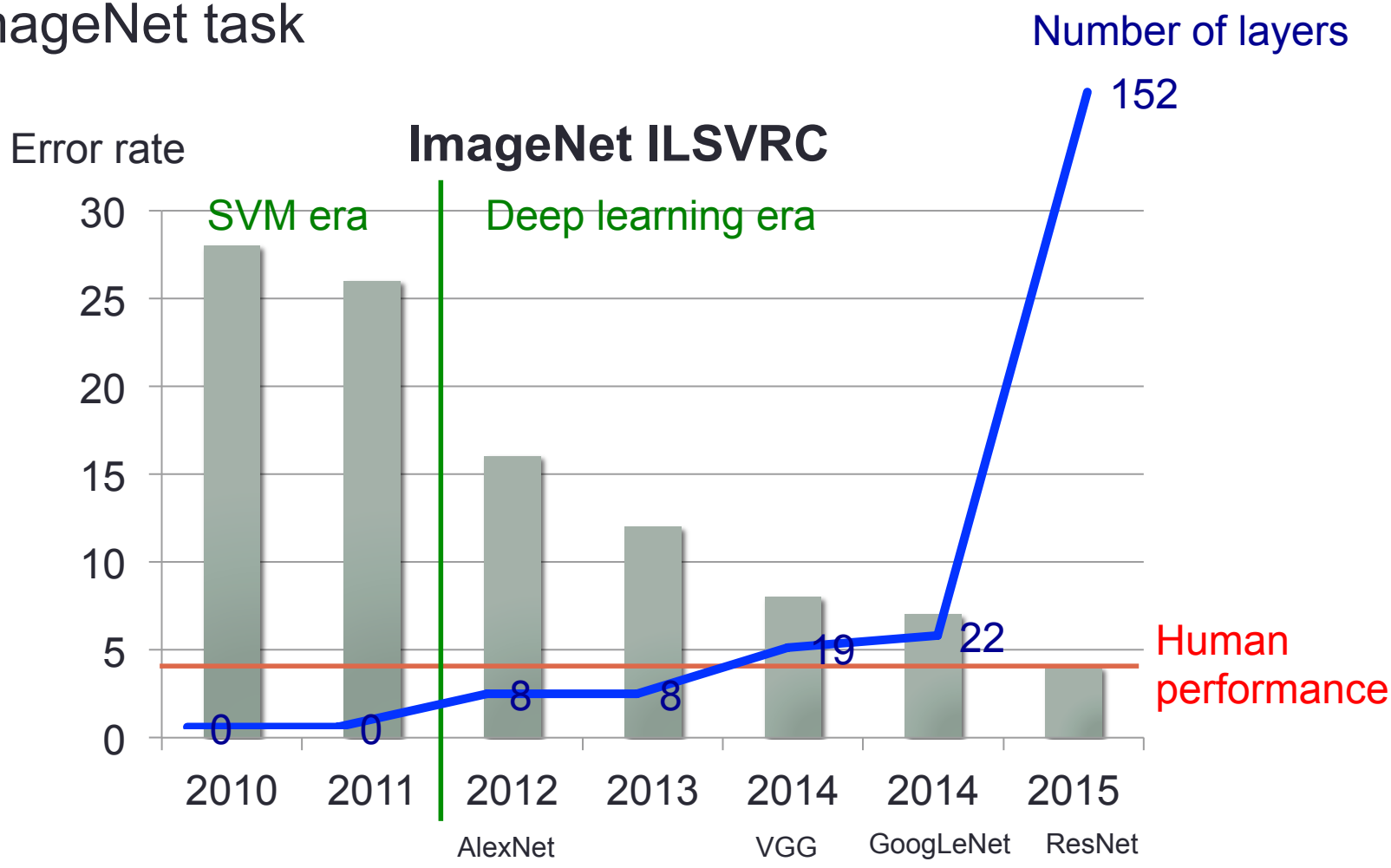
Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition,” 2012.

ImageNet – object classification challenge

Alex, Krizhevsky, Imagenet classification with deep convolutional neural networks, 2012

Wider and deeper networks

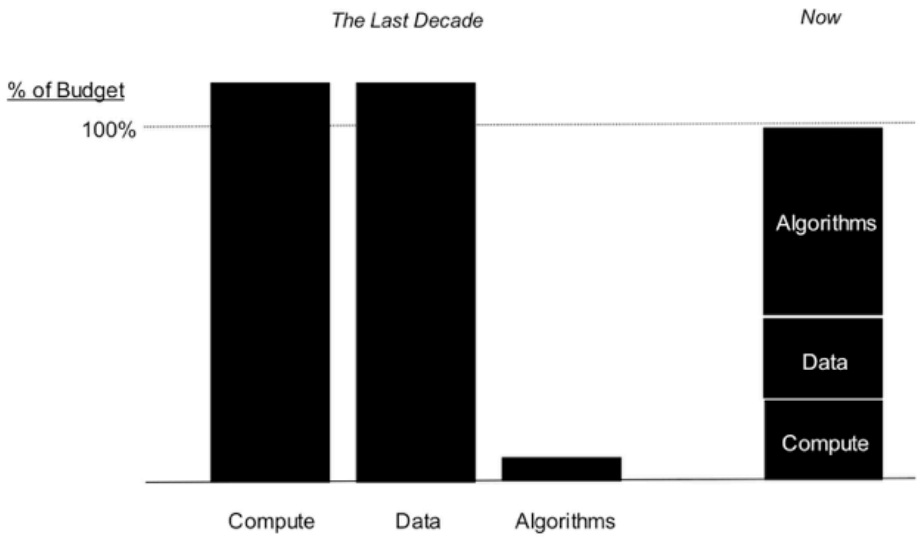
- ImageNet task



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

Why now

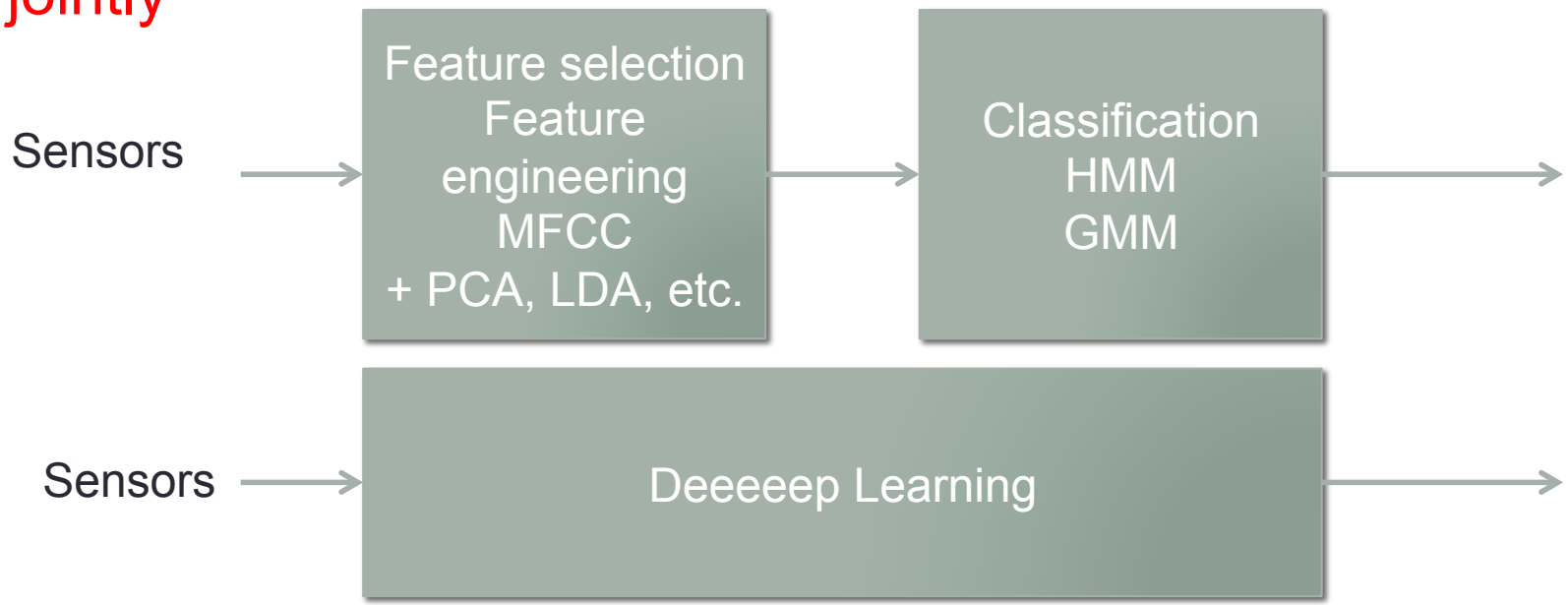
- Neural Networks has been around since 1990s
- Big data – DNN can take advantage of large amounts of data better than other models
- GPU – Enable training bigger models possible
- Deep – Easier to avoid bad local minima when the model is large



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

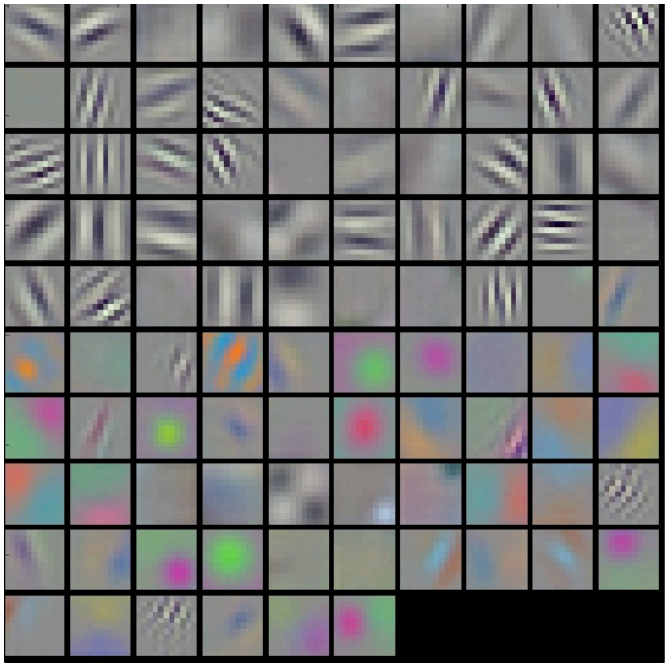
Why is deep learning good

- Traditional machine learning approaches need feature engineering (MFCC, SIFT, HoG, etc.)
  - Features are based on human understanding of the phenomena
  - Have some simplifying assumptions
- Deep learning performs modeling and feature engineering jointly



Why is deep learning good

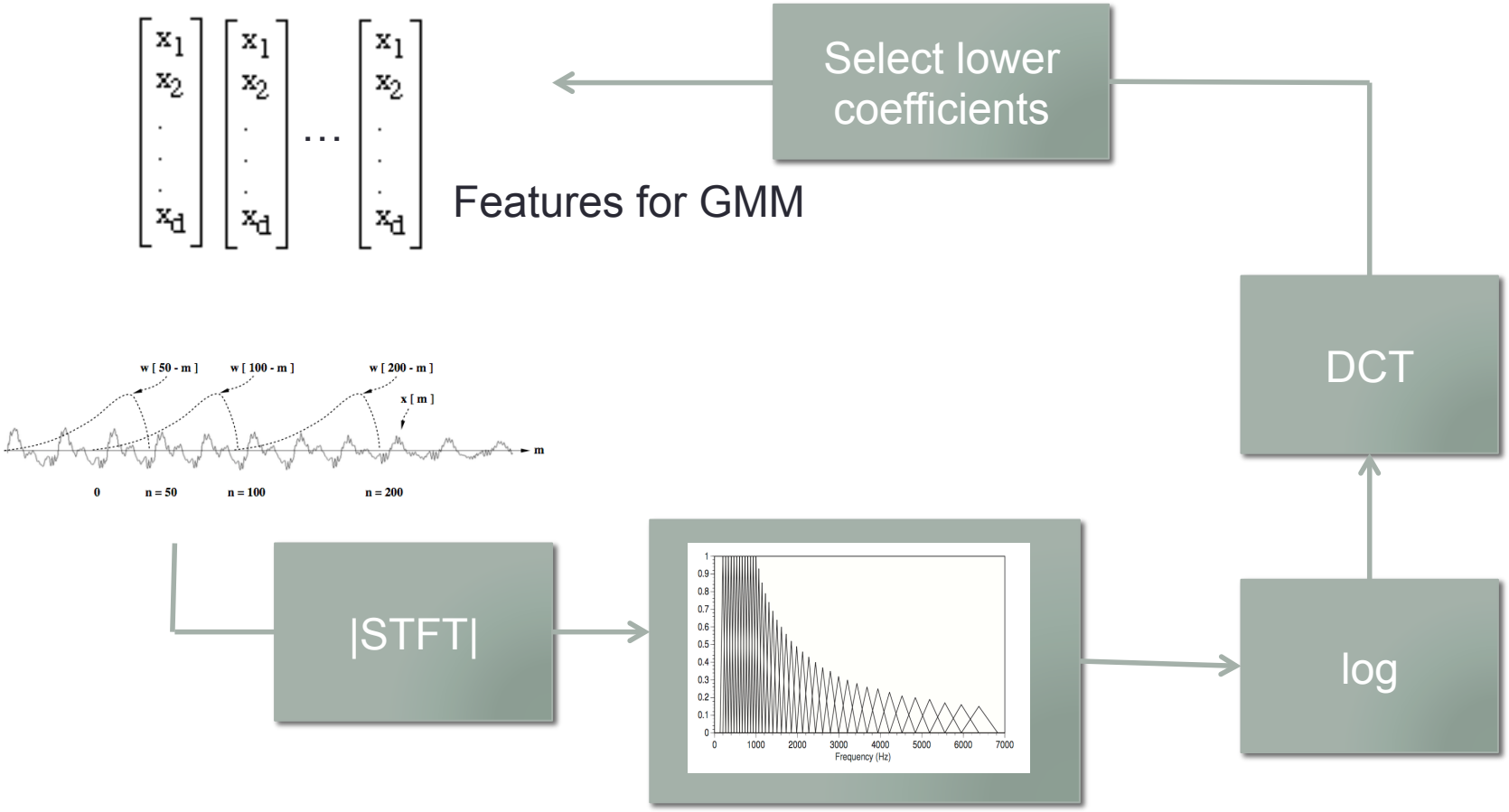
- DNN can handle higher dimensional input feature



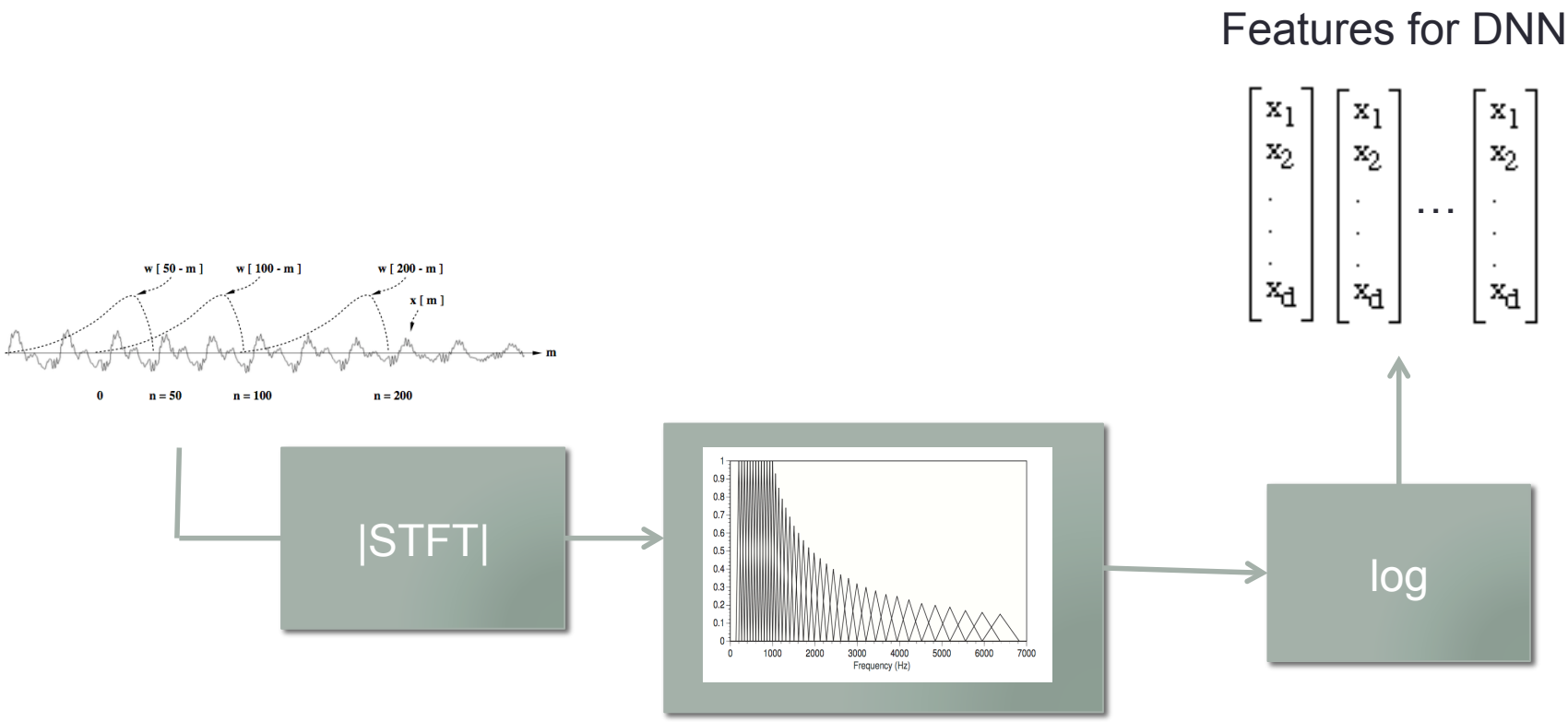
Features learned from a CNN for a vision task.
Inputs are now raw pixels instead of descriptors.

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

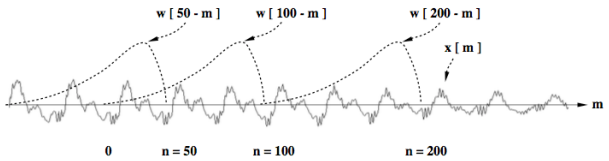
MFCC (Mel Frequency Cepstral Coefficient)



MFCC (Mel Frequency Cepstral Coefficient)



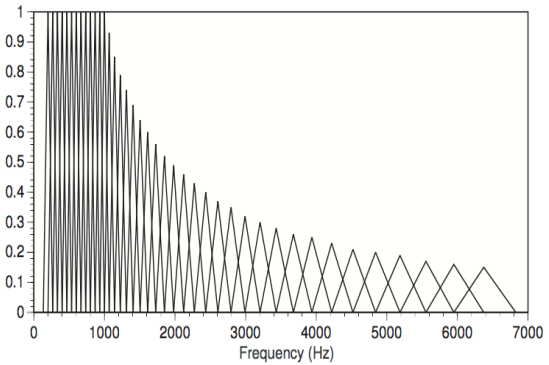
MFCC (Mel Frequency Cepstral Coefficient)



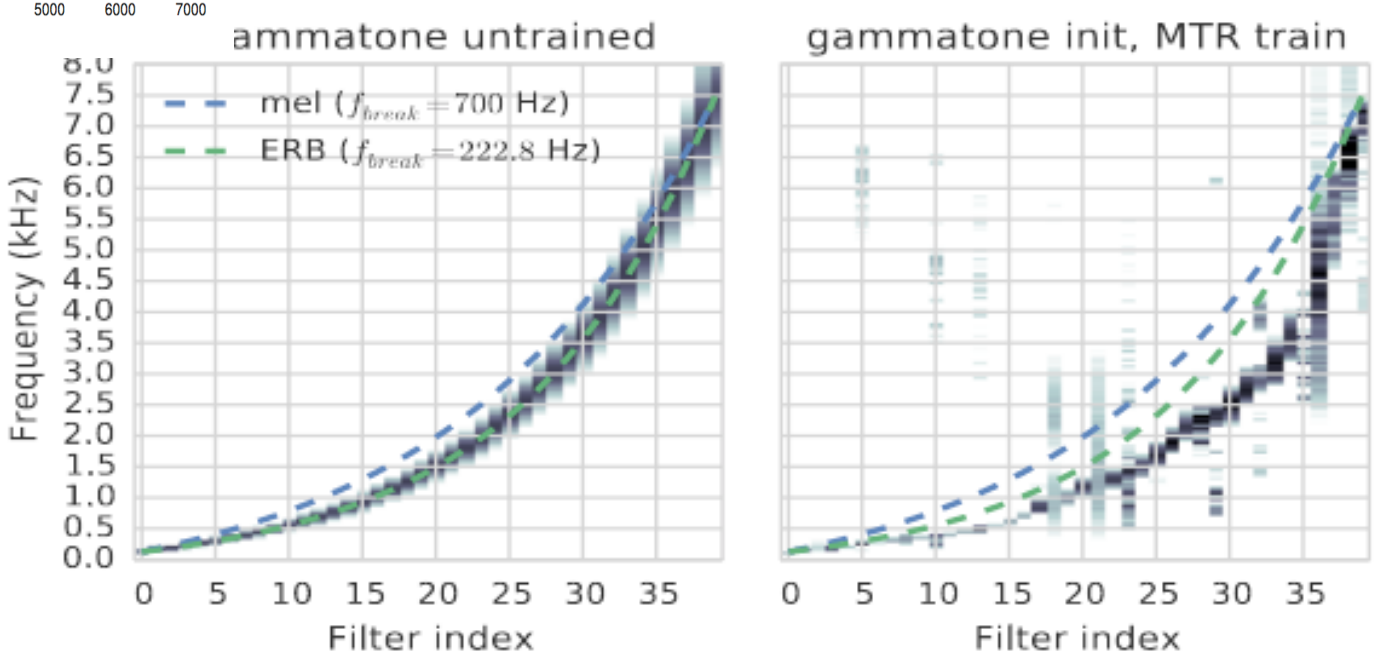
Features for DNN

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix} \cdots \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix}$$

Learning the Mel filters



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



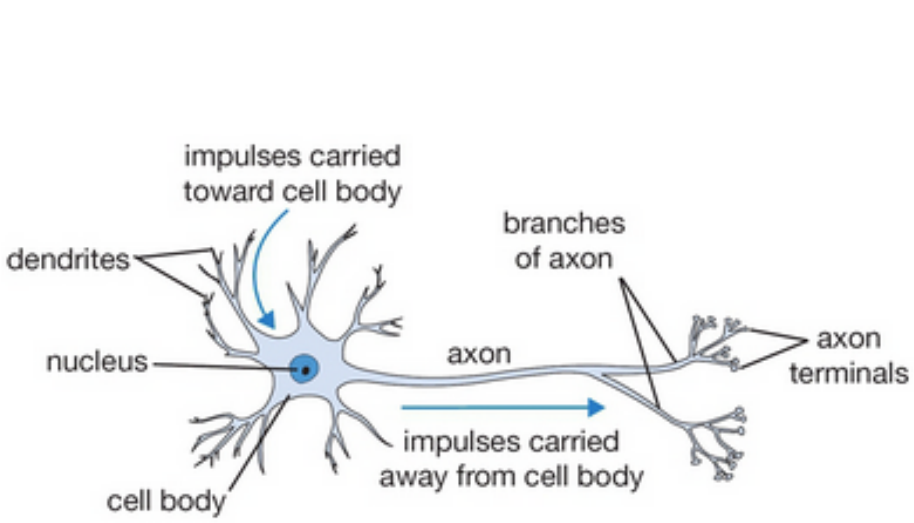
Learning the Speech Front-end with Raw Waveform CLDNNs, 2015

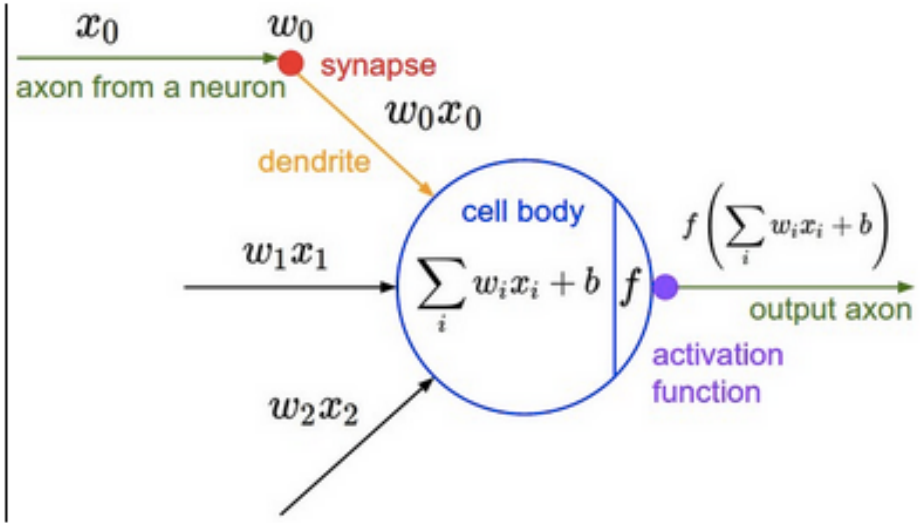
DNNs architectures

- Fully connected networks
  - Neuron
  - Softmax layer
- Convolutional neural networks
- Recurrent neural networks
- Gated recurrent units
- Long short-term memory networks

Fully connected networks

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

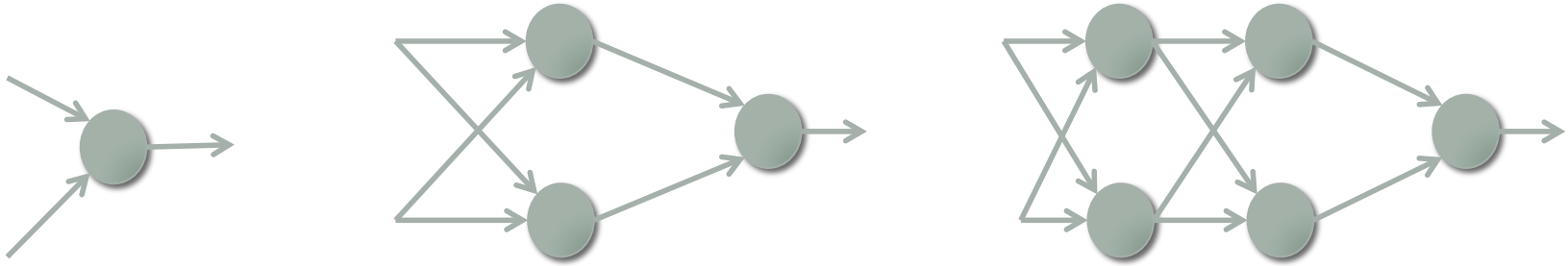


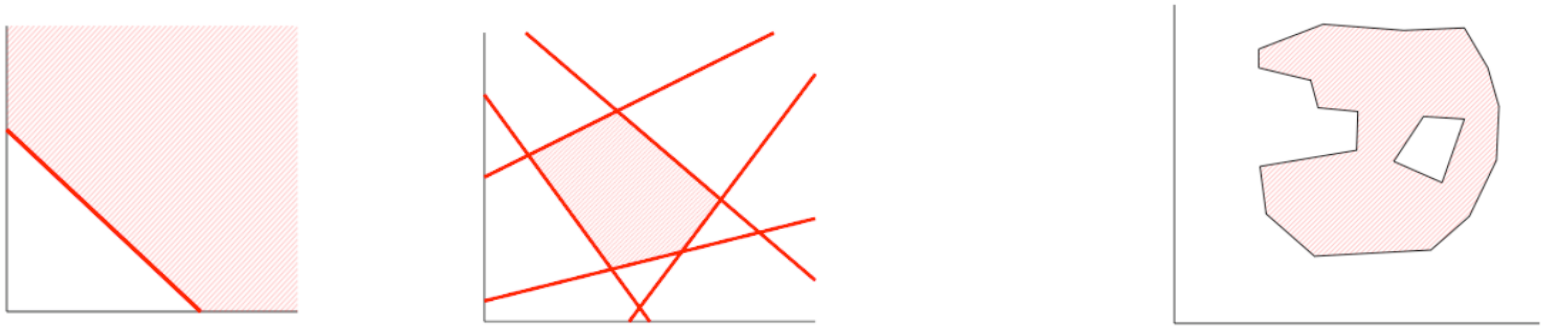


http://cs231n.github.io/neural-networks-1/

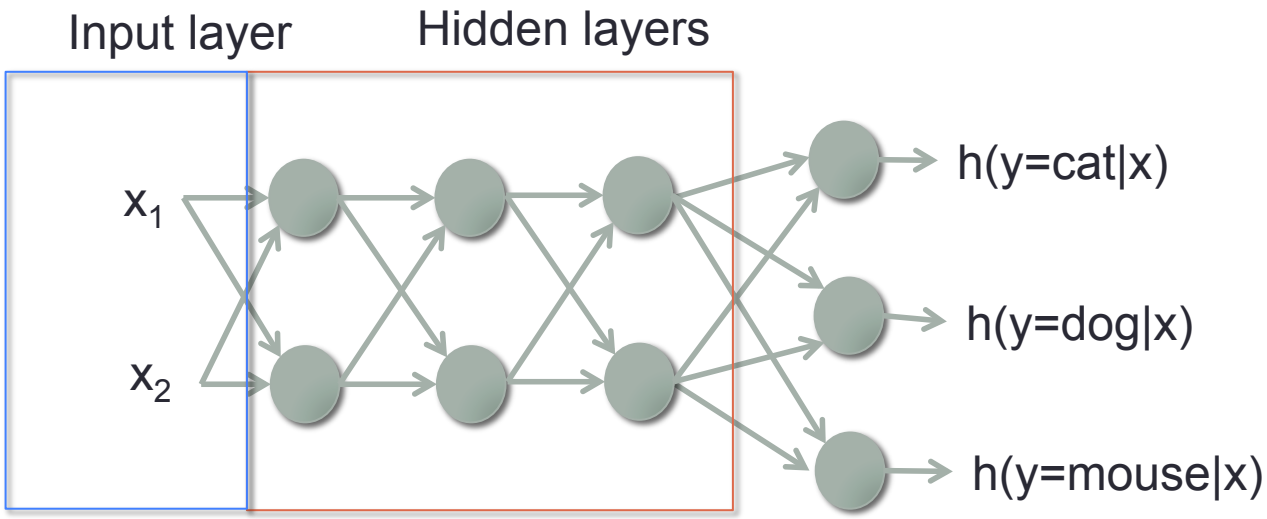
Combining neurons

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





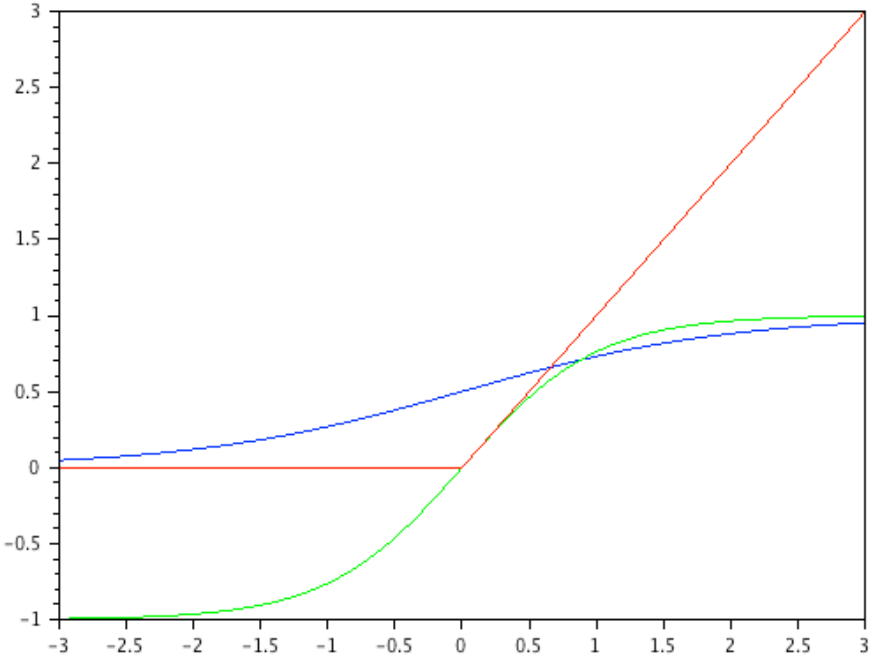
Terminology



Input should be scaled to have zero mean unit variance

Non-linearity

- The F 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
- tanh
- Rectified Linear Unit (ReLU)
- Most popular is ReLU and its variants (Fast to train, and more stable)
  - Personally, I find sigmoid gives slightly better accuracy



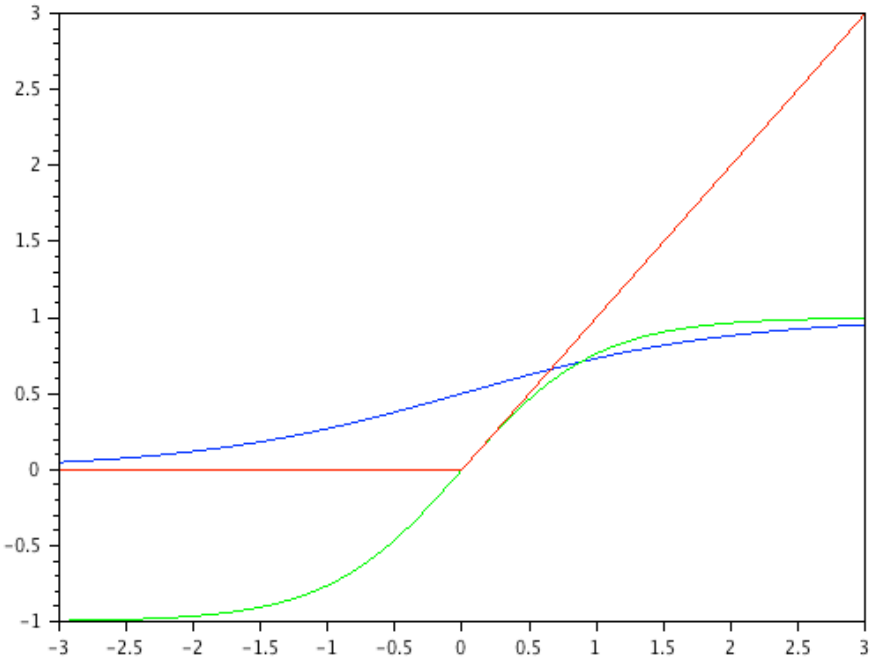
Non-linearity

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

$$tanh(x)$$

- Rectified Linear Unit (ReLU)

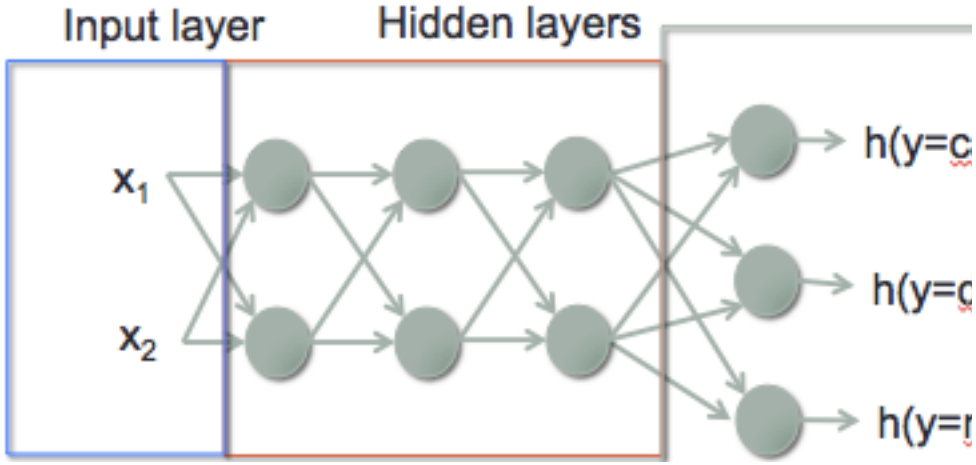
$$max(0, x)$$



Output layer – Softmax layer

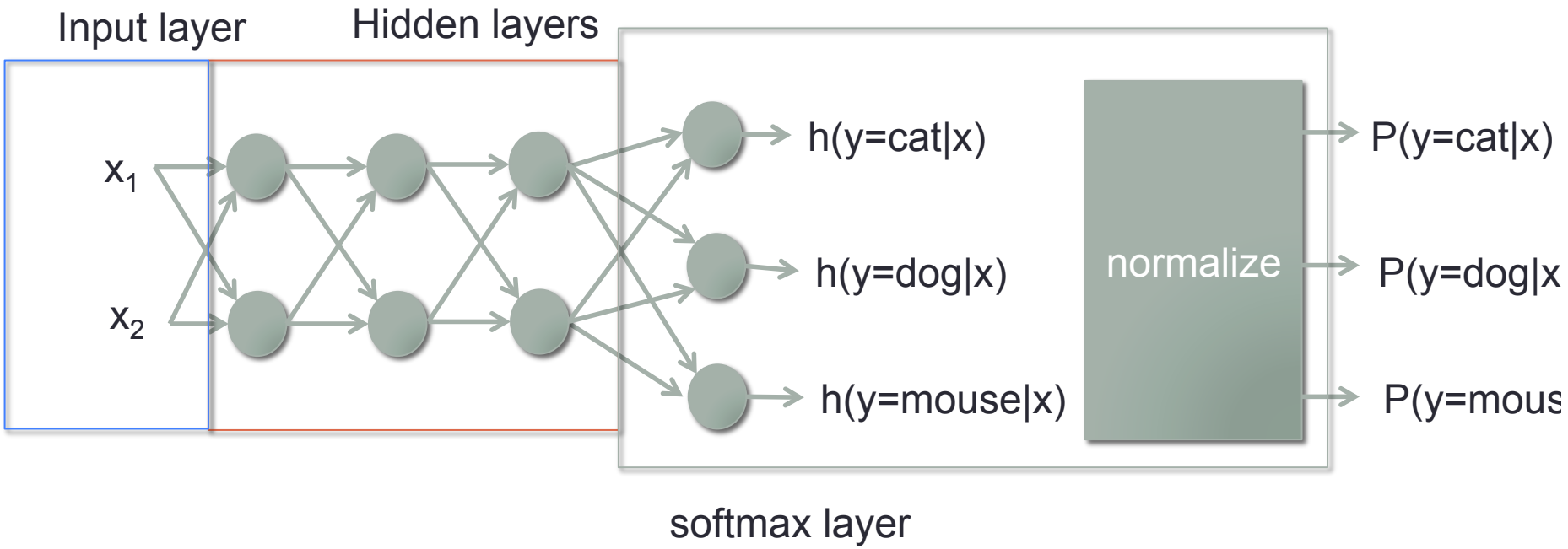
- We usually wants the output to mimic a probability function (0<=P<=1,sums to 1)
- Current setup has no such constraint
- Takes the exponent
- Add a normalization!

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



Softmax layer

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



Neural network training

- Initialize the network (weight and bias)
- Define an objective function
- Minimize it with respect to the network parameters
  - Back propagate using stochastic gradient descent (SGD)

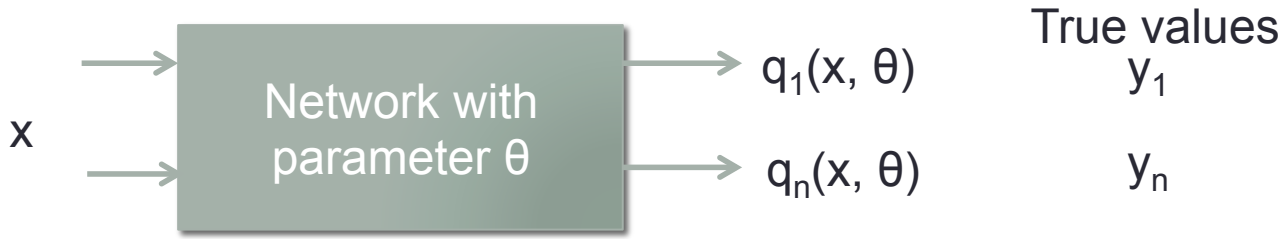
Objective function (Loss function)

- Can be any function that summarizes the performance into a single number
- Cross entropy
  - Used for softmax outputs (probabilities)

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

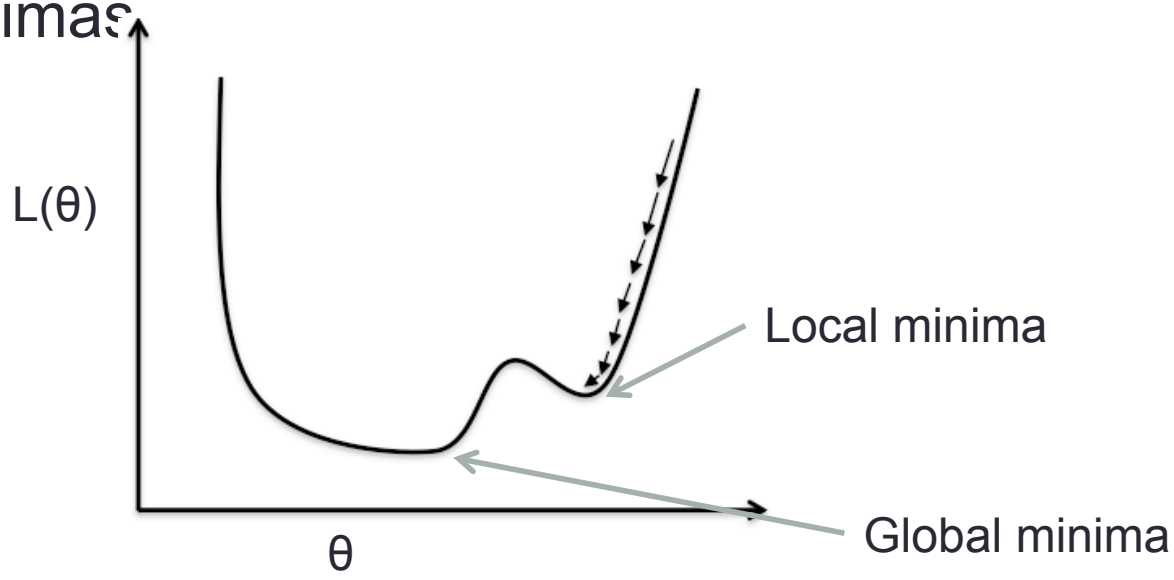
- Sum of square errors
  - Used for any real valued outputs

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



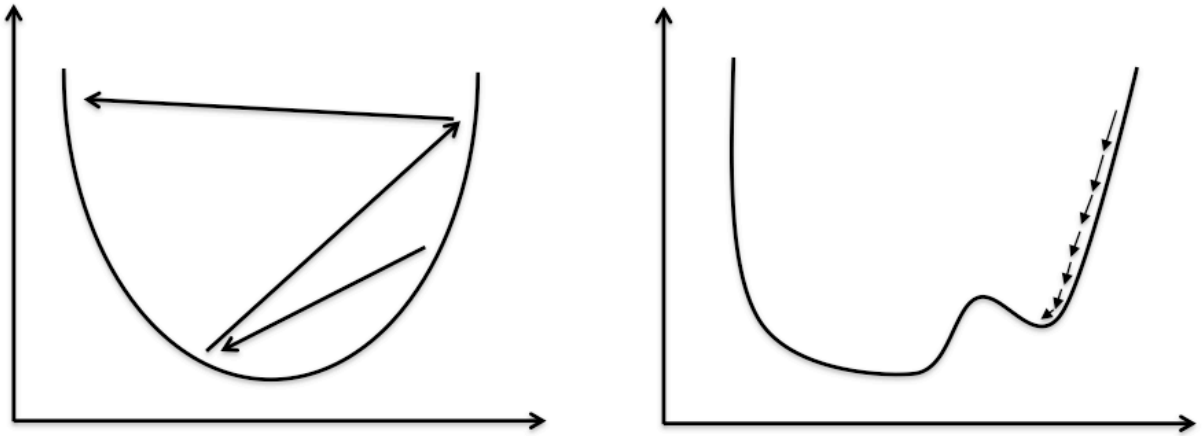
Minimization using gradient descent

- We want to minimize L with respect to θ
  - Differentiate with respect to θ
- L is a very complex function that has no closed-form solution
- Iteratively improve the model by local gradients
- Deep models are less likely to get stuck in bad local minimas



Learning rate

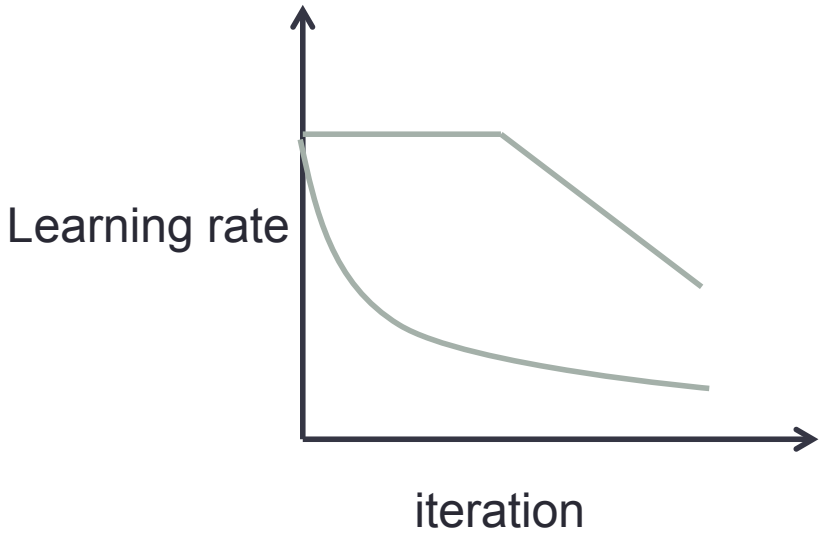
- How fast to go along the gradient direction is controlled by the learning rate
- Too large models diverge
- Too small the model get stuck in local minimas and takes too long to train



http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Learning rate scheduling

- Usually starts with a large learning rate then gets smaller later
- Depends on your task
- Automatic ways to adjust the learning rate : Adagrad, Adam, etc.



Neural network training

- Initialize the network (weight and bias)
- Define an objective function
- Minimize it with respect to the network parameters
  - Back propagate using stochastic gradient descent (SGD)

Back propagation

- Forward pass
  - Pass the value of the input until the end of the network
- Backward pass
  - Compute the gradient starting from the end and passing down gradients using chain rule

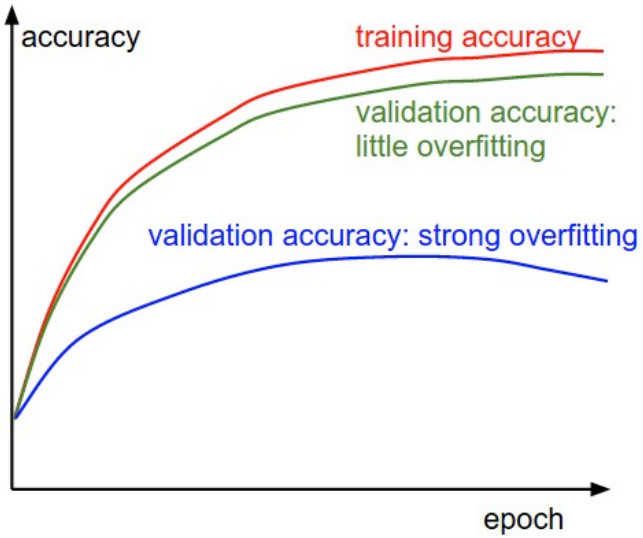
Examples to read

https://alonalj.github.io/2016/12/10/What-is-Backpropagation/

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Overfitting

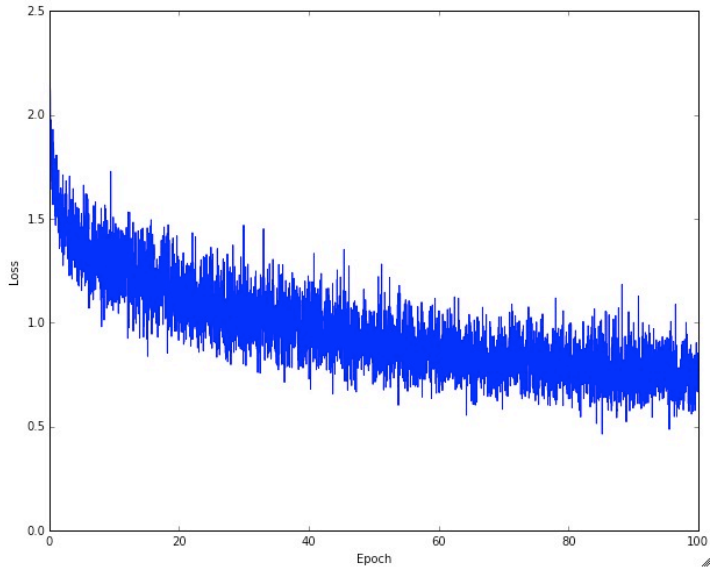
- You can keep doing back propagation forever!
- The training loss will always go down
- But it overfits
- Need to monitor performance on a held out set
- Stop or decrease learning rate when overfit happens



http://cs231n.github.io/neural-networks-3/

Monitoring performance

- Can monitor many criterions
  - Loss function
  - Classification accuracy
  - WER - preferred but expensive to monitor
- Sometimes these disagree
- Actual performance can be noisy, need to see the trend



http://cs231n.github.io/neural-networks-3/

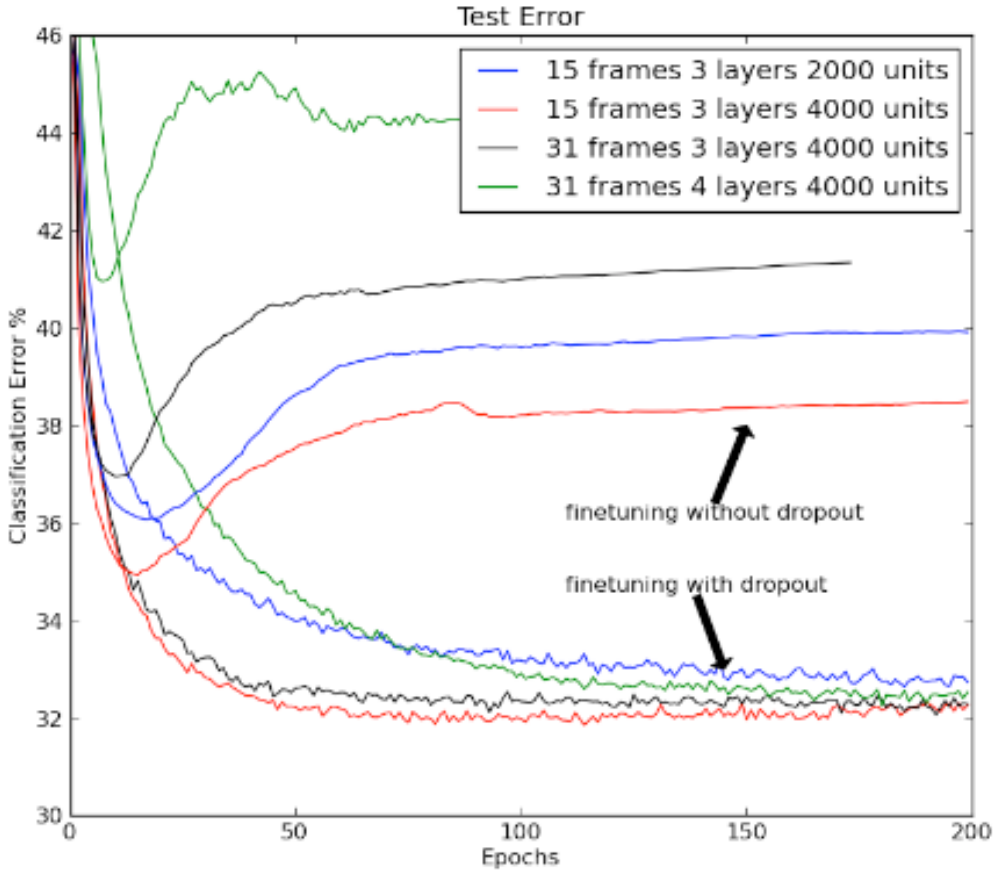
Reducing overfitting - dropout

- A *RECENT* (2012) regularization technique for reducing overfitting
- Randomly turn off different subset of neurons during training
  - Network no longer depend on any particular neuron
  - Force the model to have redundancy – robust to any corruption in input data
  - A form of performing model averaging (assemble of experts)
- Now a standard technique

Hinton, Geoffrey "Improving neural networks by preventing co-adaptation of feature detectors" 2012

Dropout on TIMIT

- A phoneme recognition task



Hinton, Geoffrey "Improving neural networks by preventing co-adaptation of feature detectors" 2012

Vanishing/Exploding gradient

- Backprop introduces many multiplications down chain
- The gradient value gets smaller and smaller
  - The deeper the network the smaller the gradient in the lower layers
  - Lower layers changes too slowly (or not at all)
  - Hard to train very deep networks (>6 layers)
- The opposite can also be true. The gradient explodes from repeated multiplication
  - Put a maximum value for the gradient (Gradient clipping)

Residual networks

- Or Highway networks
- Adds a direct connection between layers (bypass non linearity
- Gradient now has two components (direct connection and traditional direction)

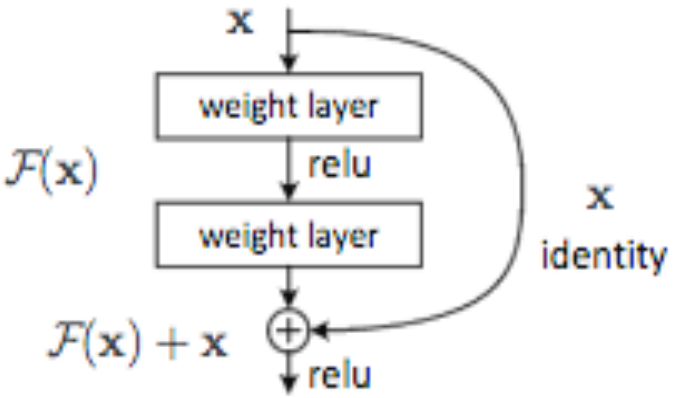


Image recognition task

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Kaiming He, Deep Residual Learning for Image Recognition, 2015

Residual networks

ASR task

System	#layers	with overlap	no overlap
LSTMP	3	50.7	41.7
LSTMP	8	52.6	43.8
HLSTMP	3	50.4	41.2
HLSTMP	8	50.7	41.3

Y. Zhang, Highway Long Short-Term Memory RNNs for Distant Speech Recognition 2016

DNNs architectures

- Fully connected networks
  - Neuron
  - Softmax layer
  - Training – SGD and backprop
- Convolutional neural networks
- Recurrent neural networks
- Gated recurrent units
- Long short-term memory networks

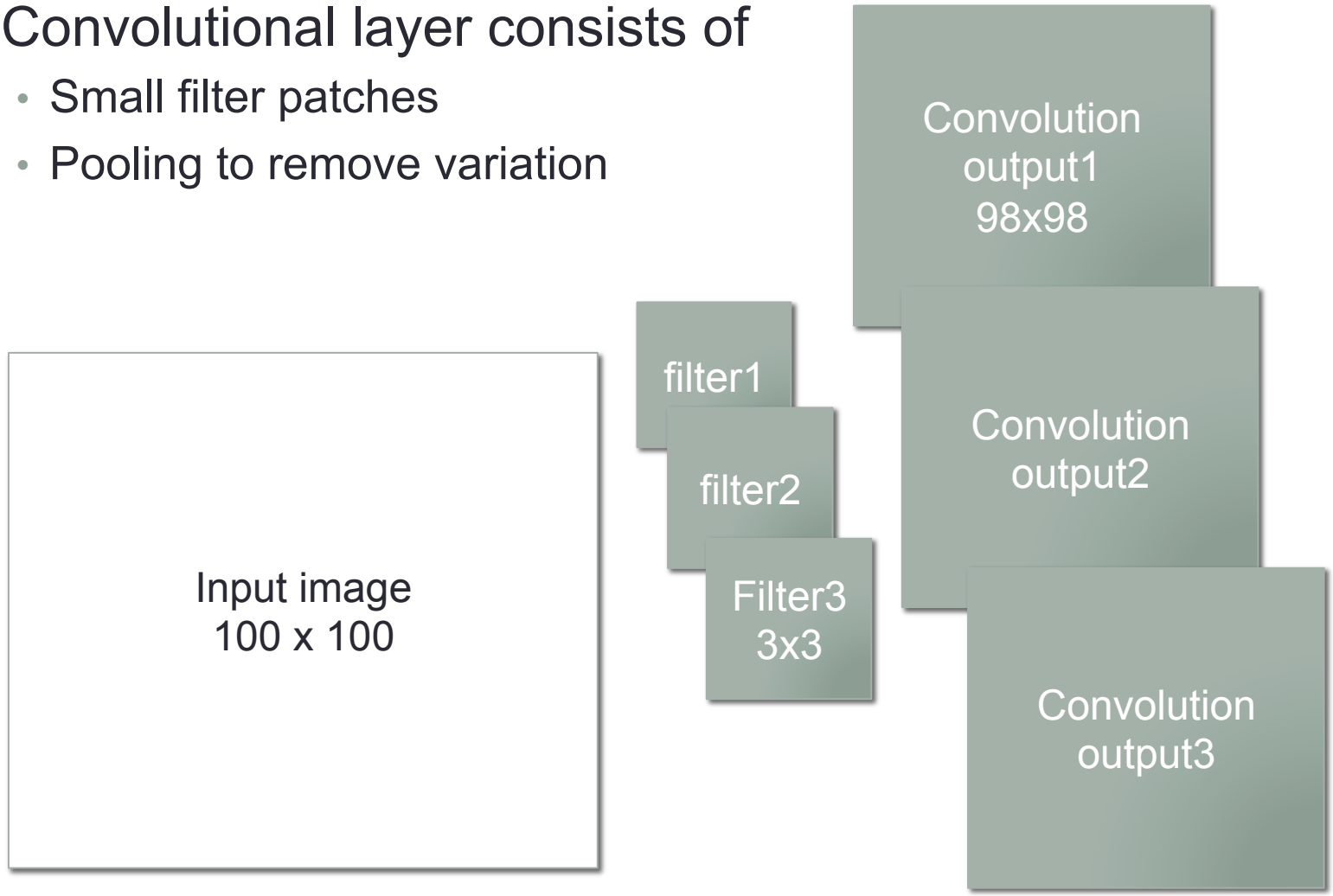
Convolutional Neural Networks (CNNs)

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

- For ASR, the sound made by male and female has different shifts in the format frequencies
- Can we use the same parameters to learn that a cat exists regardless of location
  - Shift-invariance

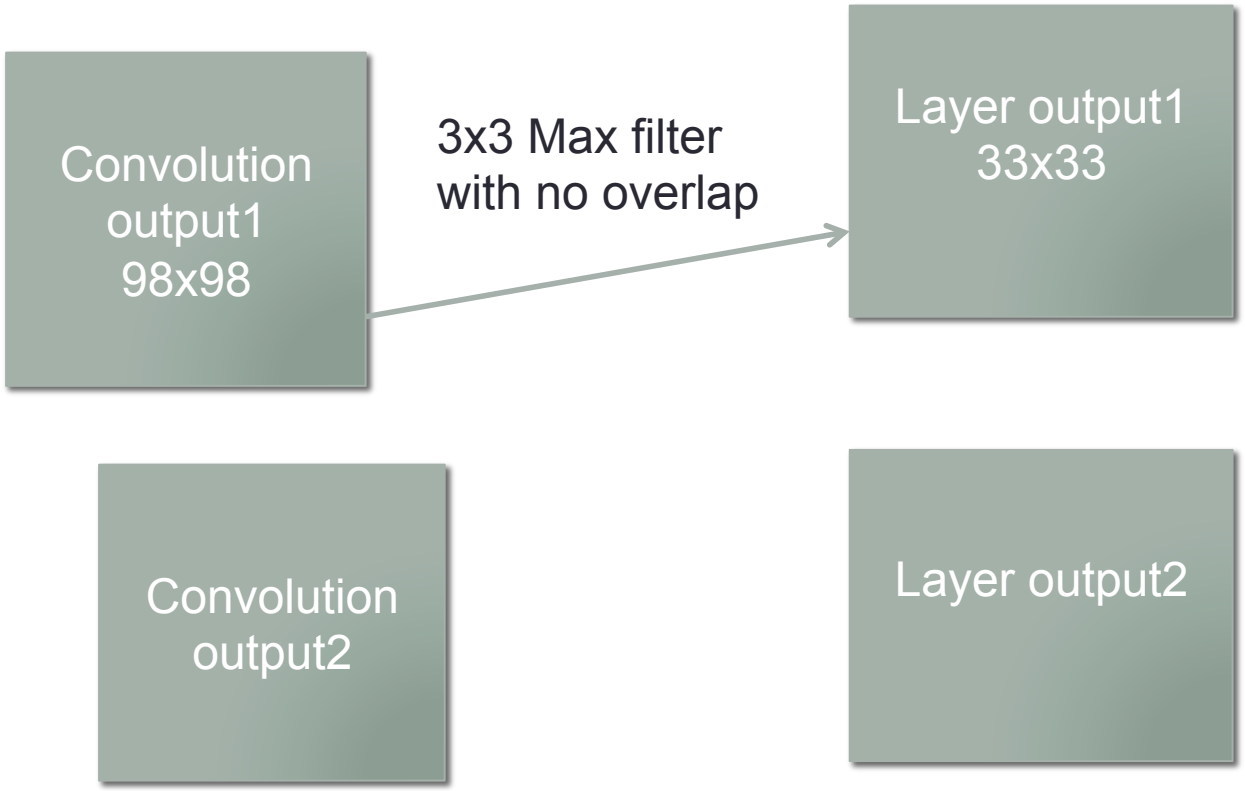
Convolutional filters

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



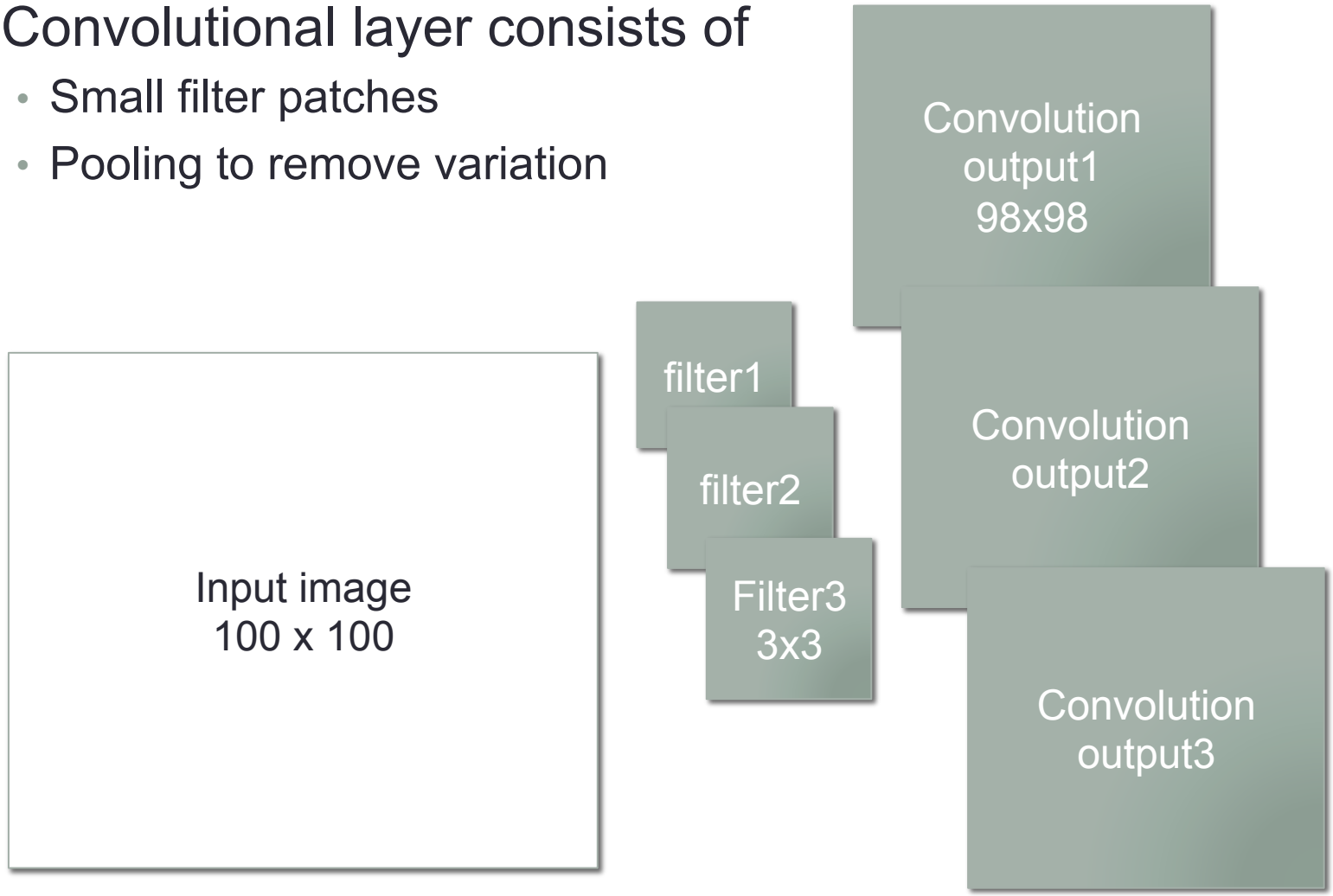
Convolutional filters

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

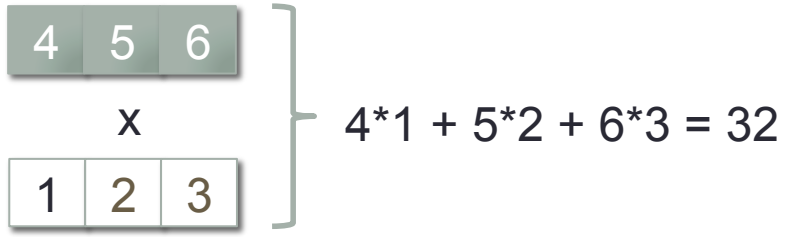


Convolutional filters

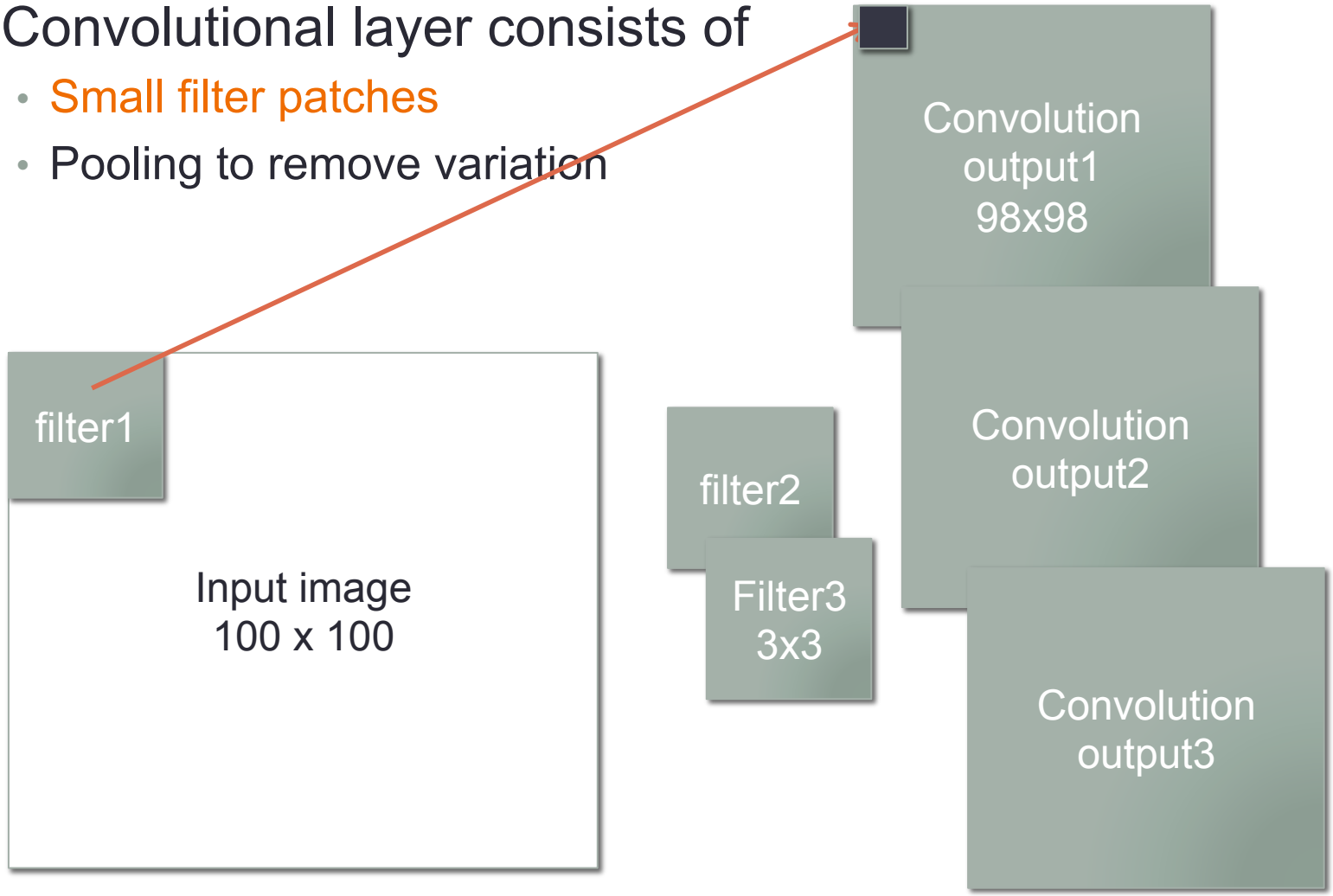
- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



Convolutional filters

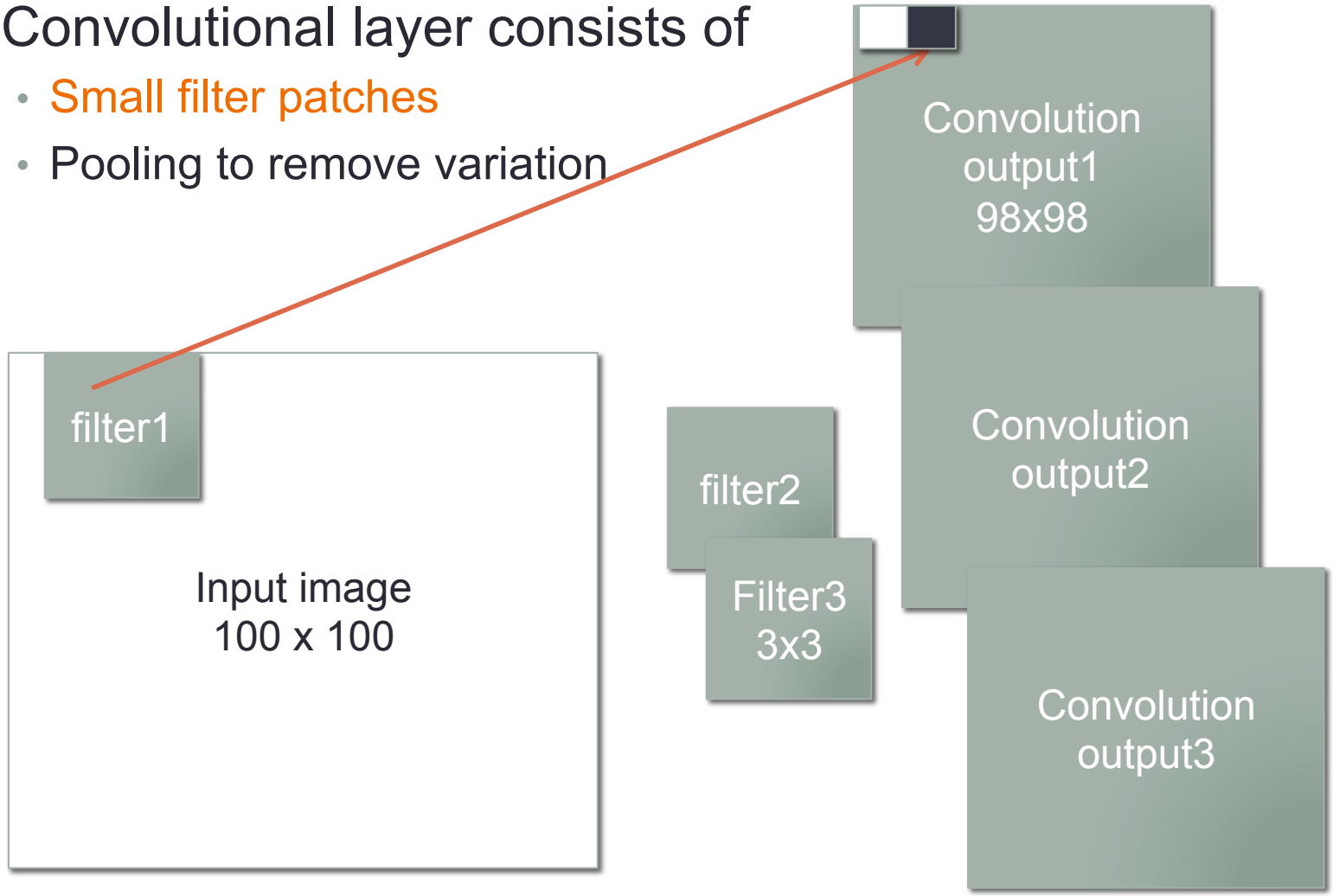


- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



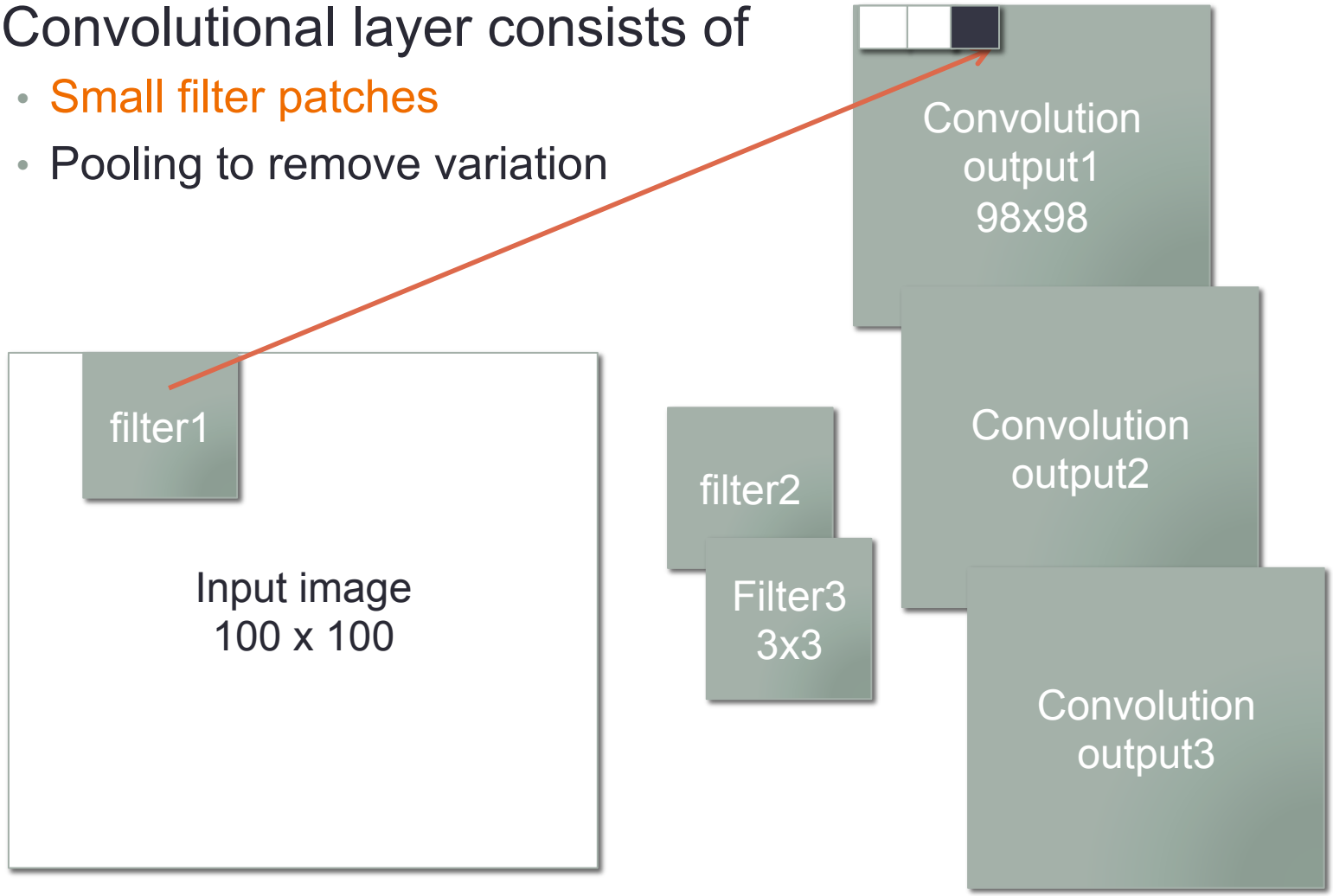
Convolutional filters

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



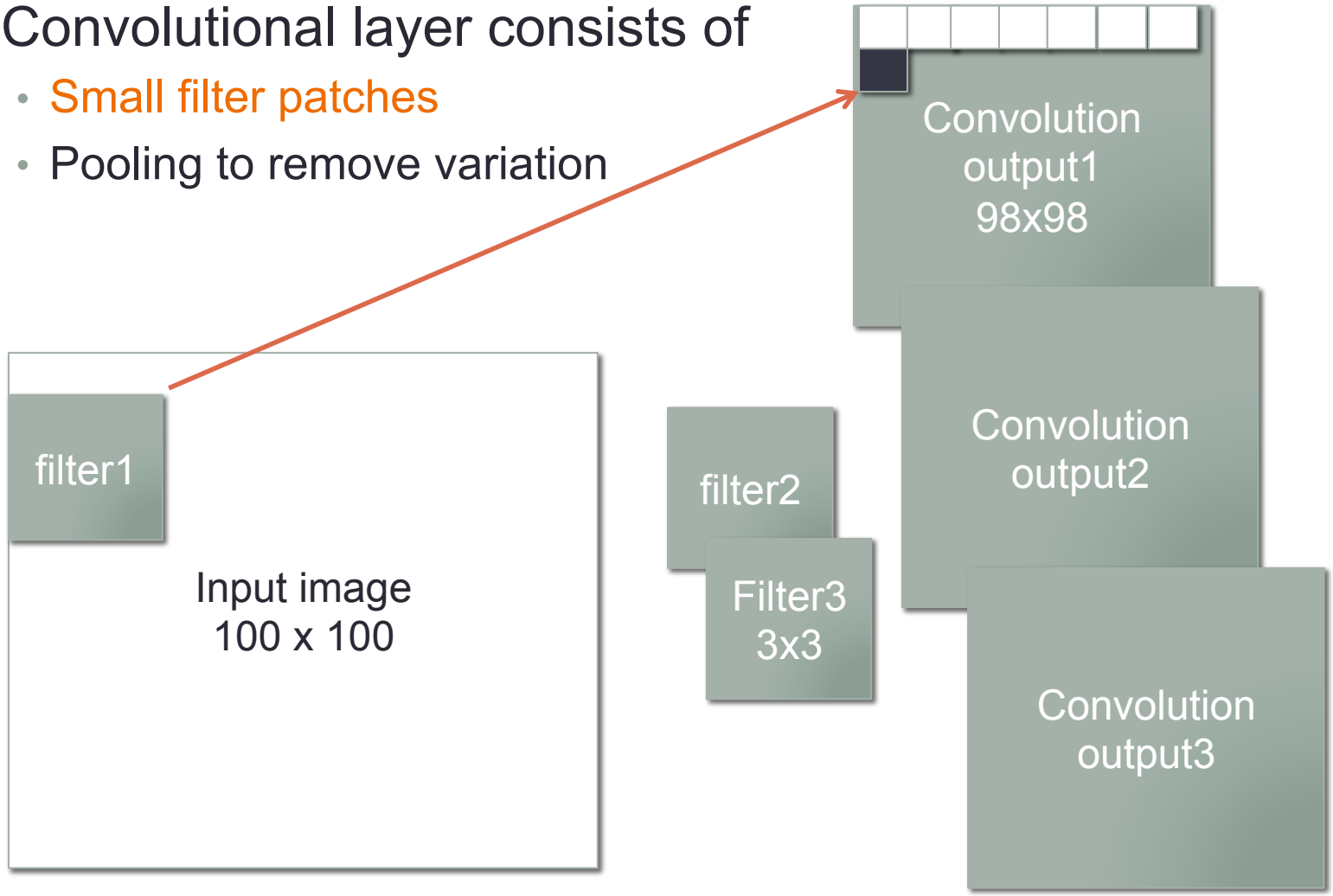
Convolutional filters

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



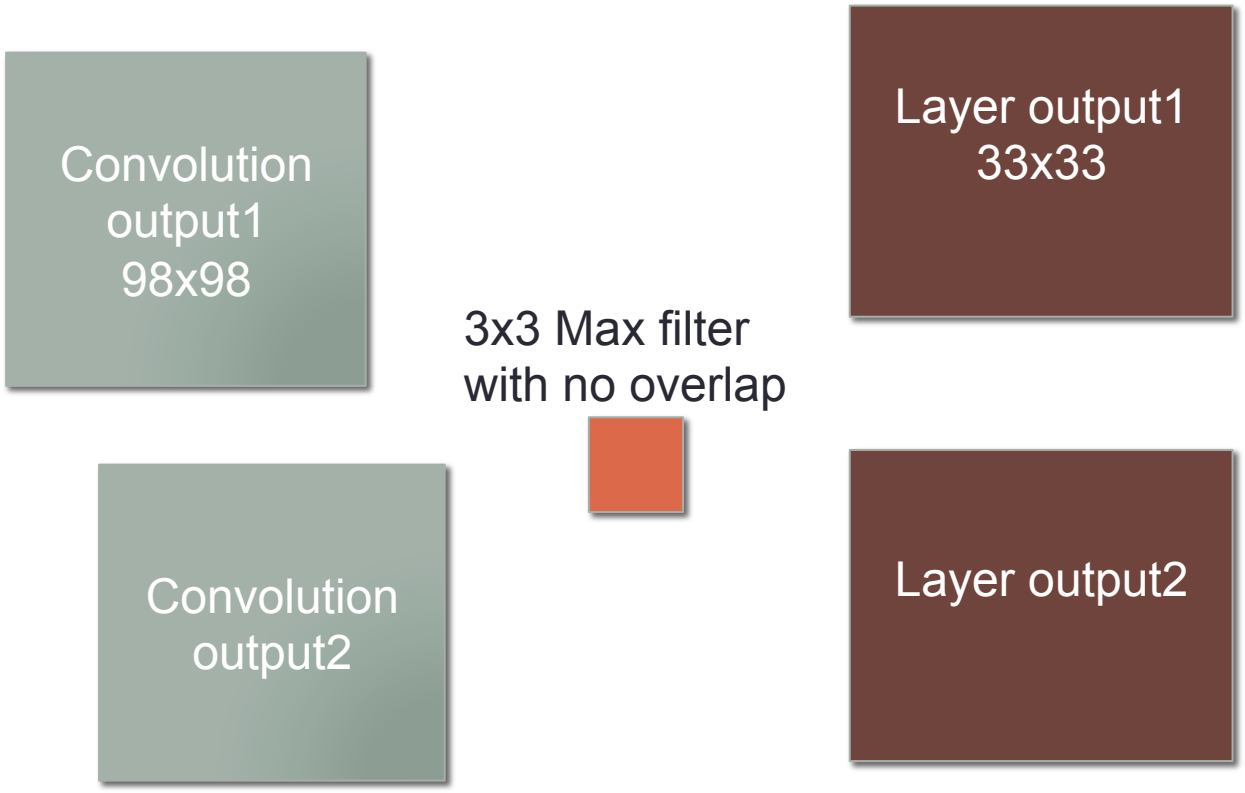
Convolutional filters

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



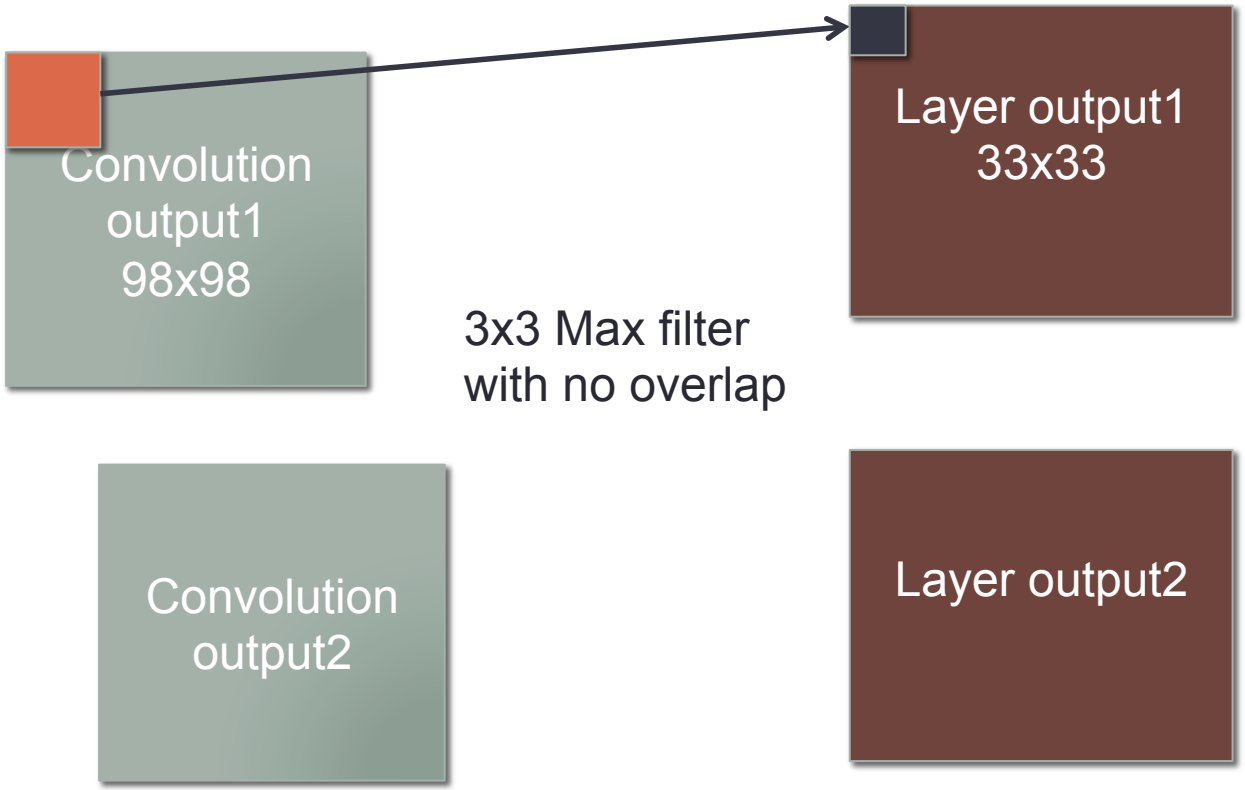
Pooling/subsampling

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



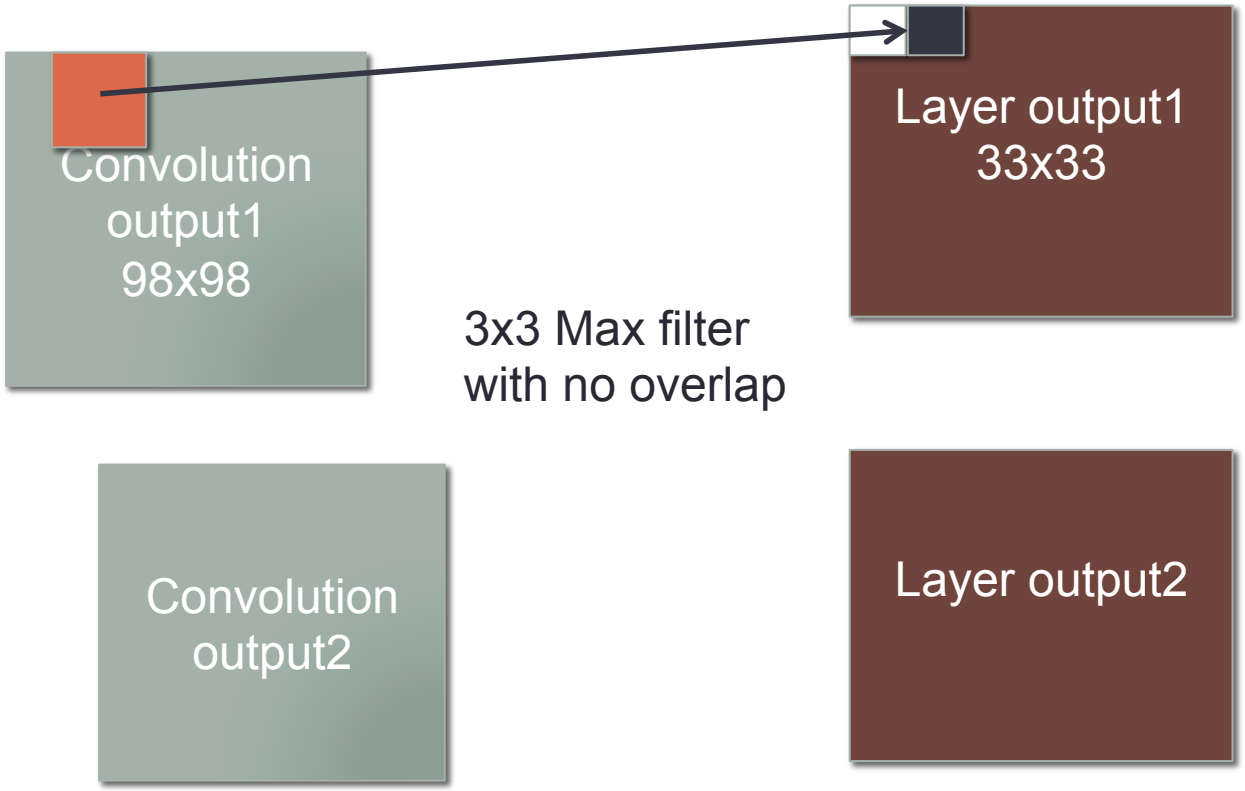
Pooling/subsampling

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



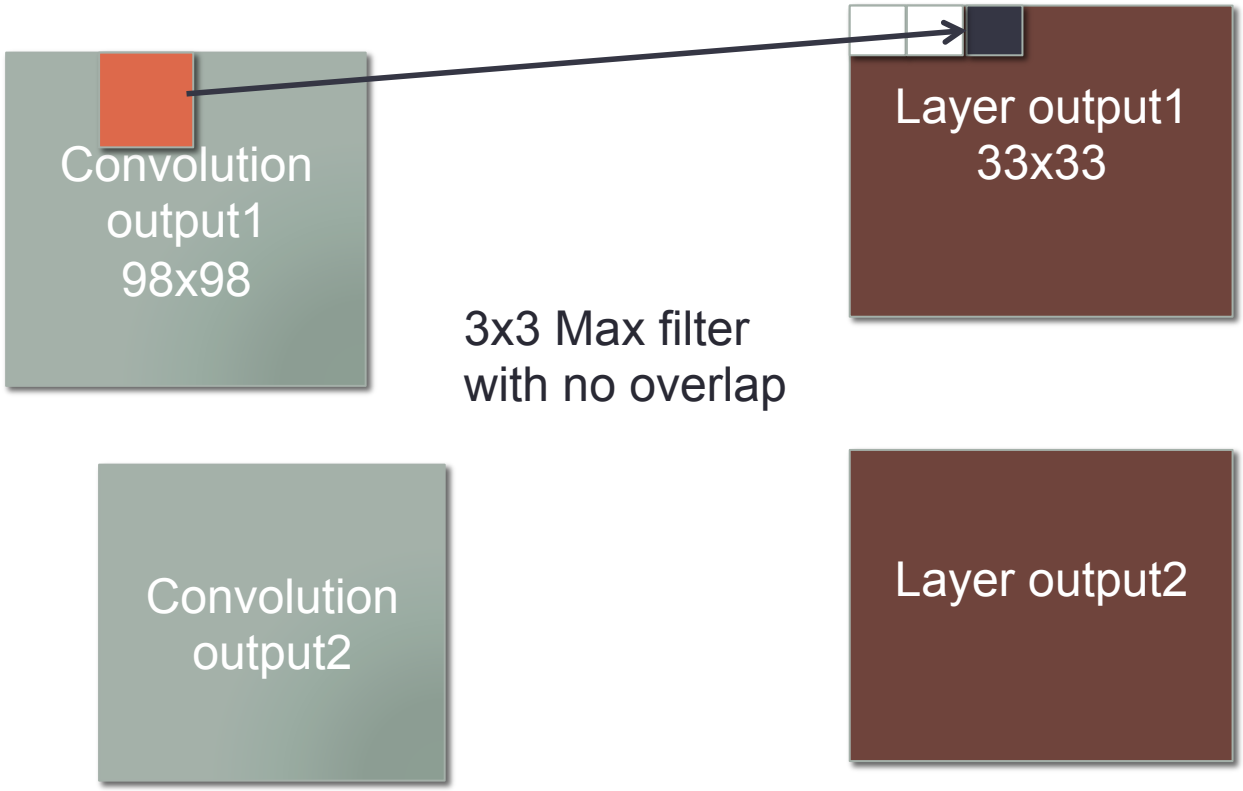
Pooling/subsampling

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



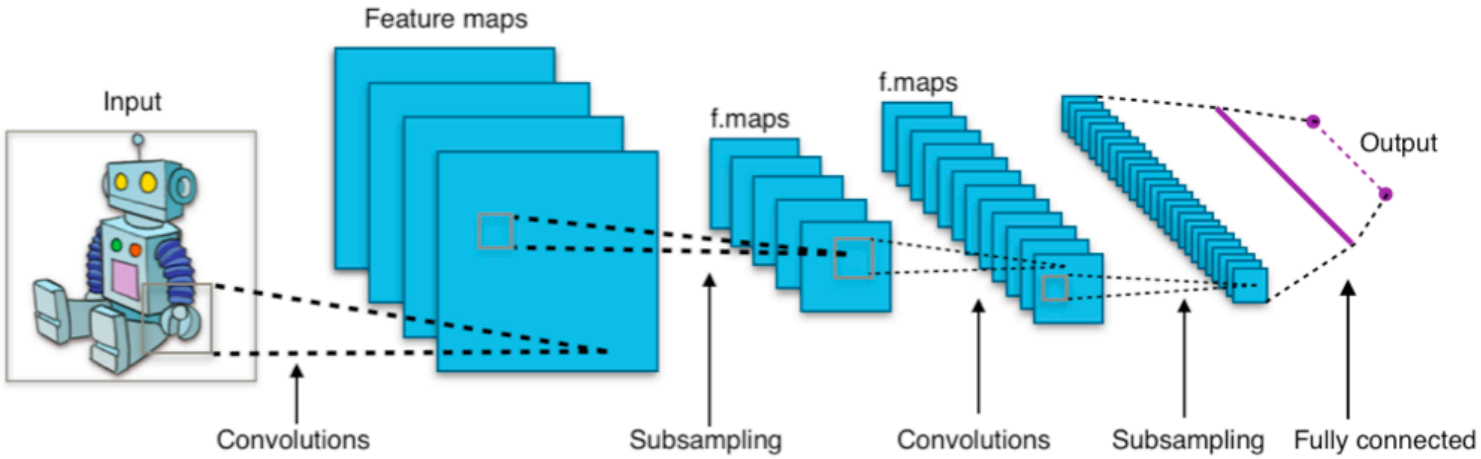
Pooling/subsampling

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



CNN

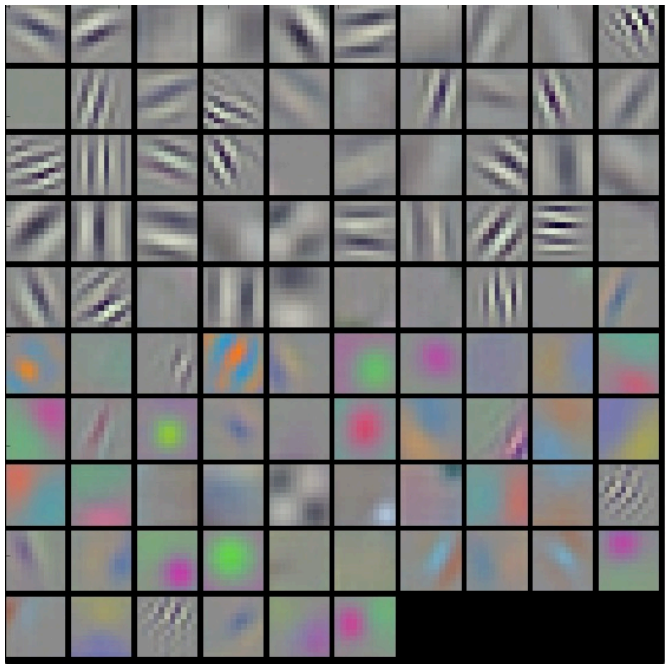
- Filter size, number of filters, and pooling rate are all parameters
- Usually followed by a fully connected network at the end
  - CNN is good at learning low level features
  - DNN combines the features into high level features and classify



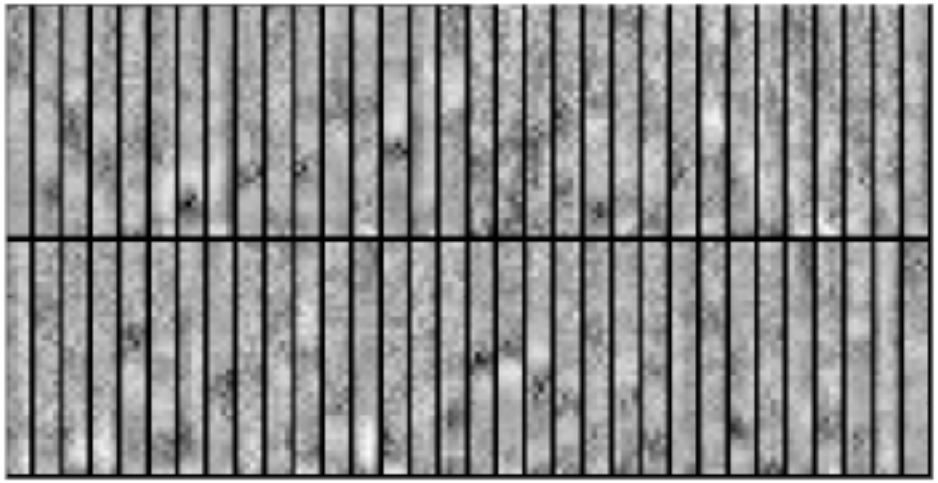
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

CNN

- The filter can learn interesting features
  - Most are quite random



Vision



ASR

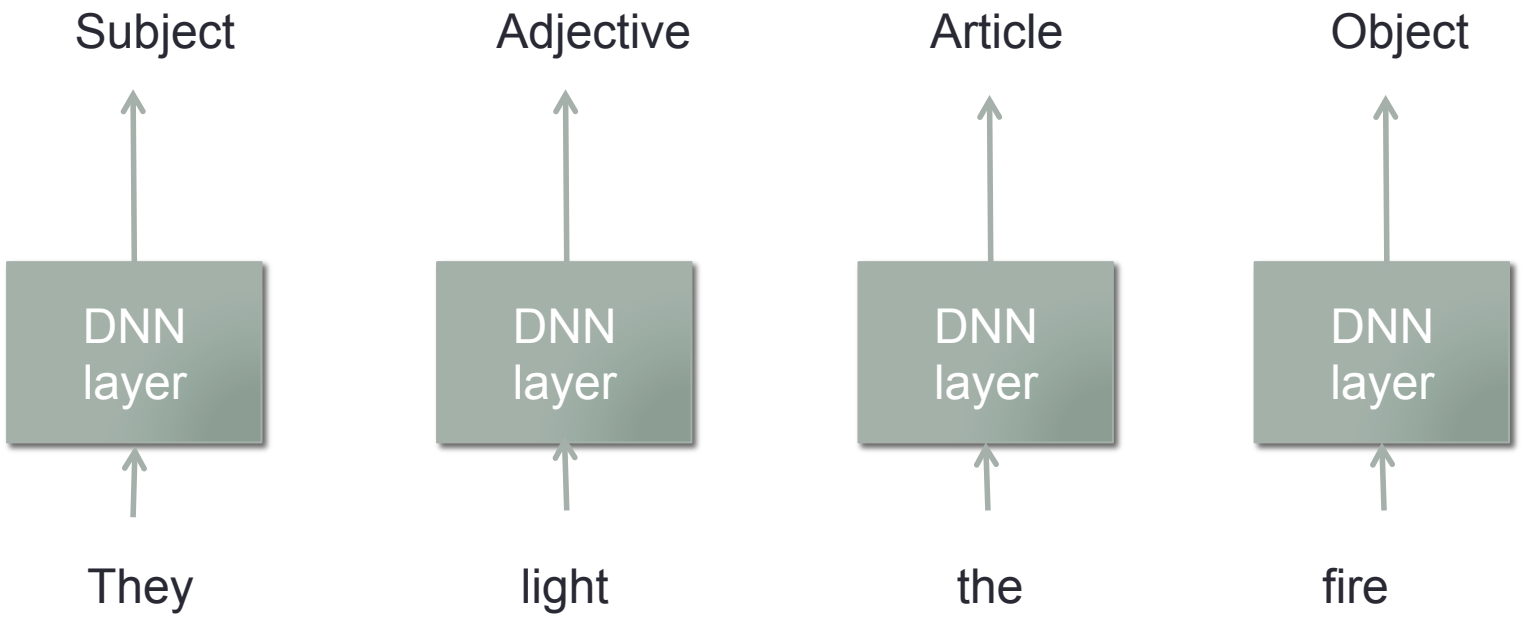
D. Harwath "Deep Multimodal Semantic Embeddings for Speech and Images 2015

Recurrent neural network (RNN)

- Many tasks has temporal relationship
  - Speech – CVC structure
  - NLP – open bracket should follow by an end bracket
- DNN parameters are fixed (for the forward pass). Thus, it does not have the memory to keep track of these relationships
- RNN try to force the model to take into account of previous inputs and outputs

Recurrent neural network (RNN)

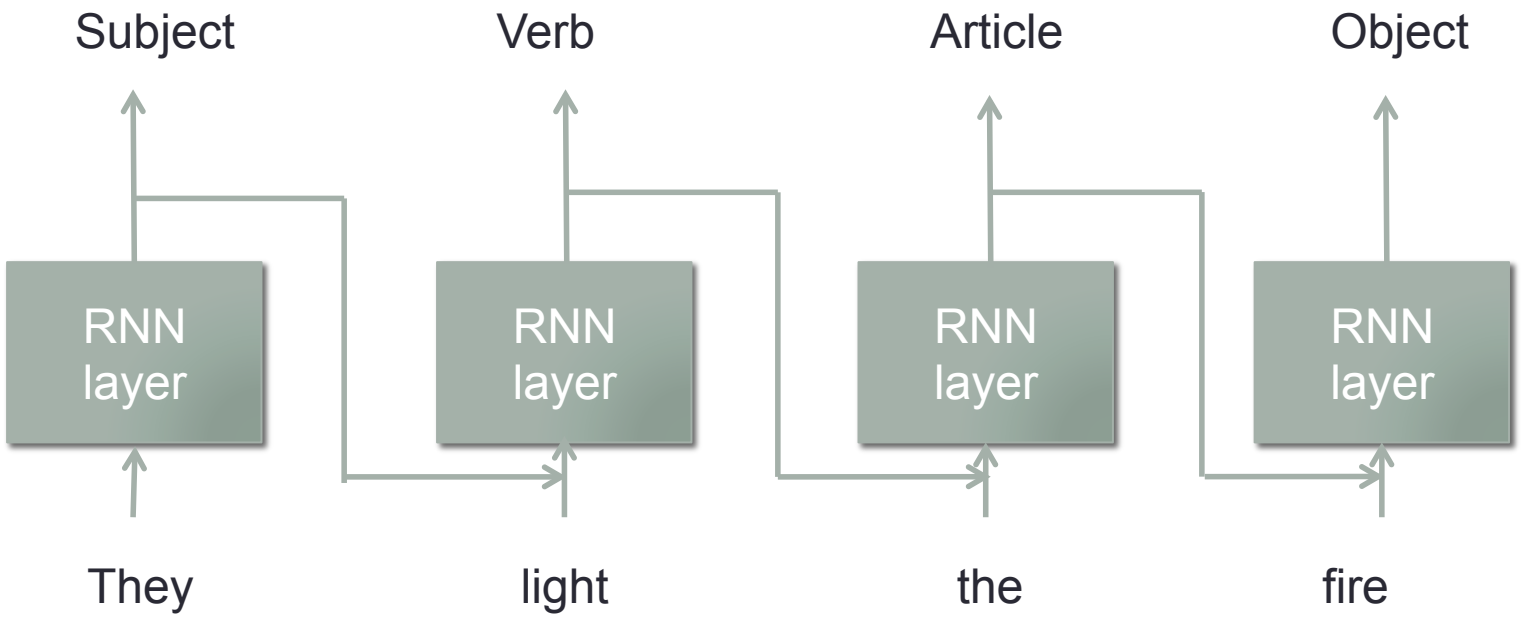
- DNN framework



Problem1: need a way to remember the past

Recurrent neural network (RNN)

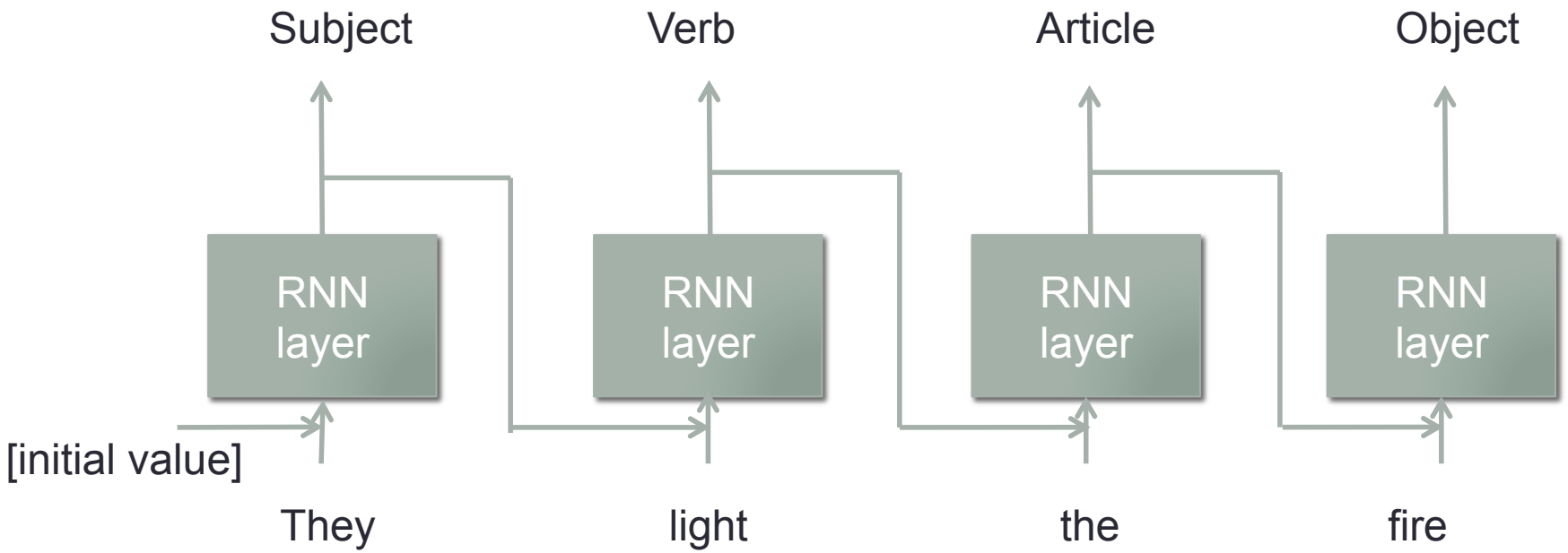
- RNN framework



Output of the layer encodes something meaningful about the past

Recurrent neural network (RNN)

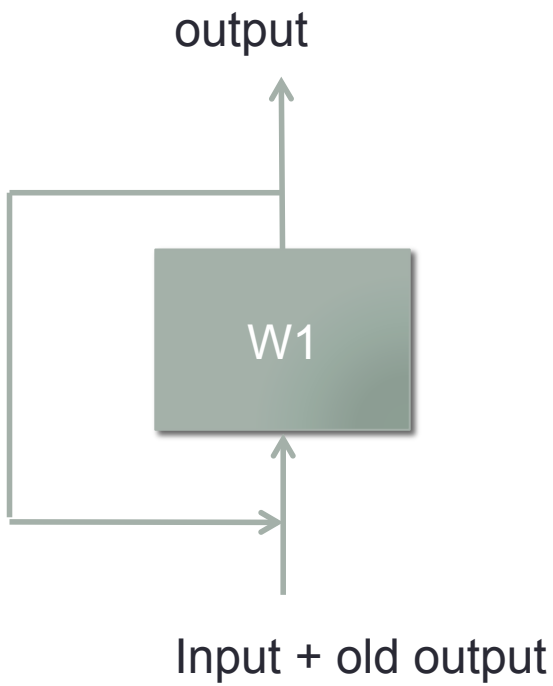
- RNN framework



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

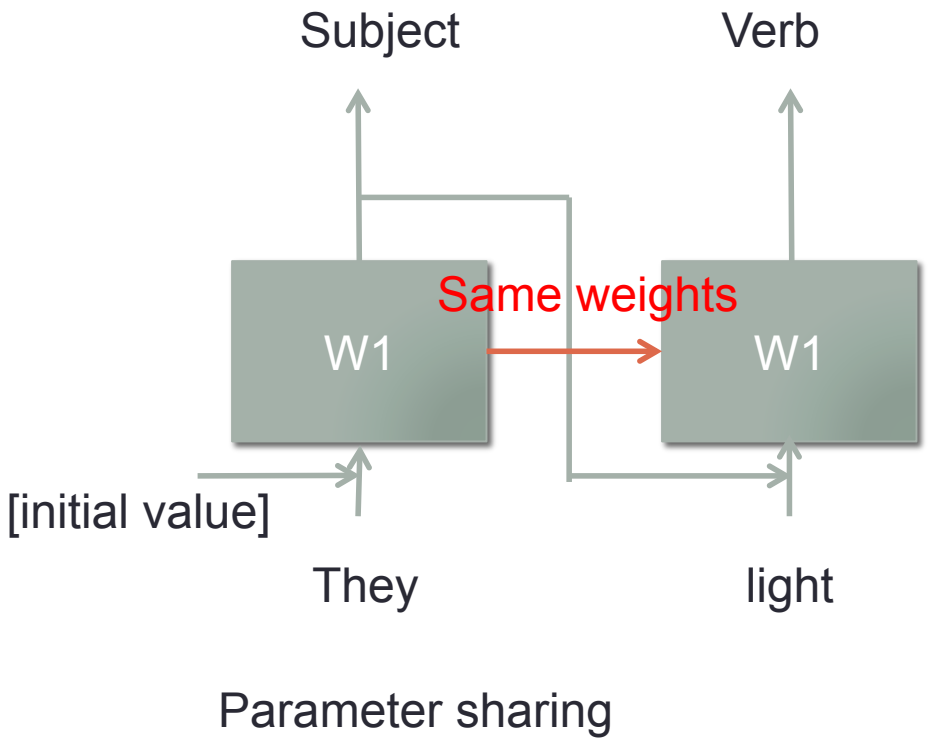
Recurrent neural network (RNN)

- Unrolling of a recurrent layer.



Recurrent neural network (RNN)

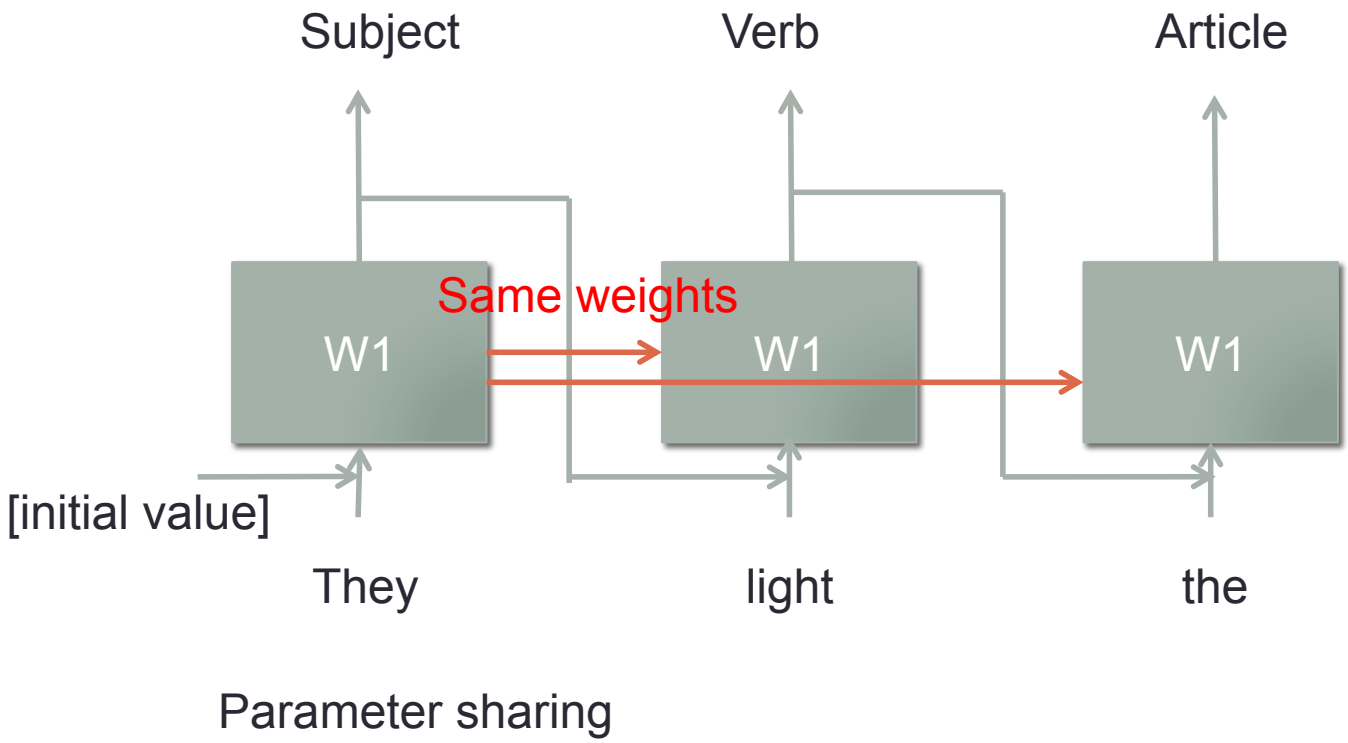
- Unrolling of a recurrent layer.



Parameter sharing

Recurrent neural network (RNN)

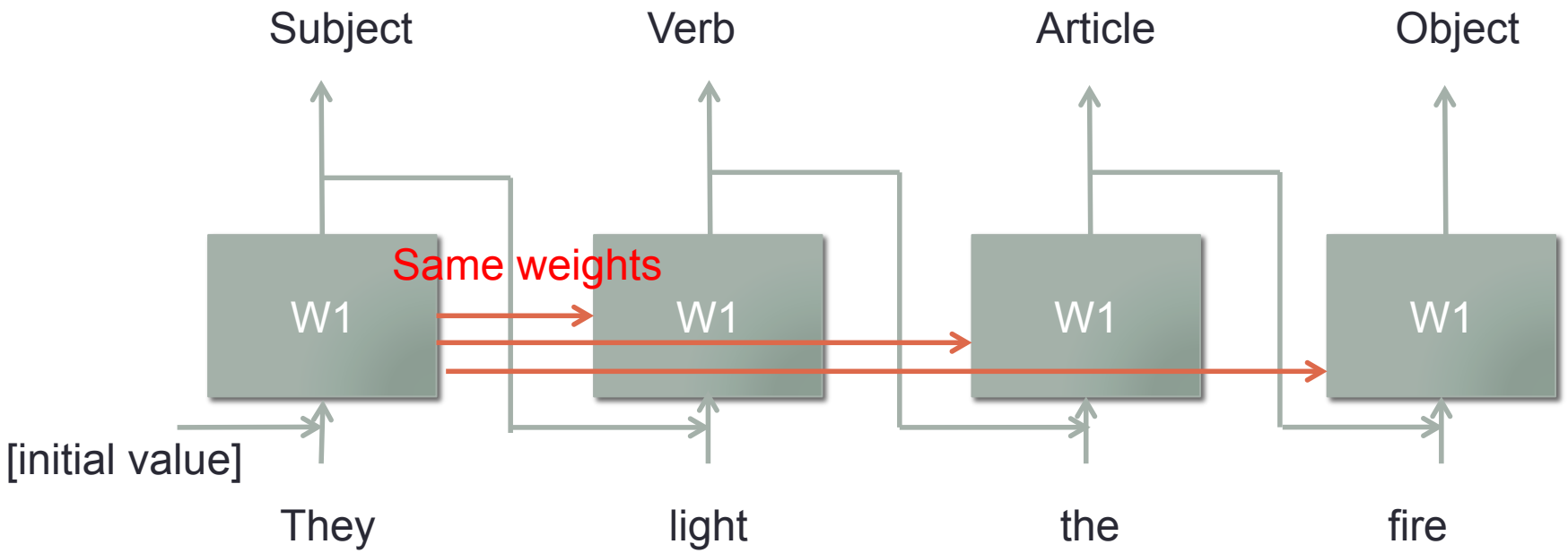
- Unrolling of a recurrent layer.



Parameter sharing

Recurrent neural network (RNN)

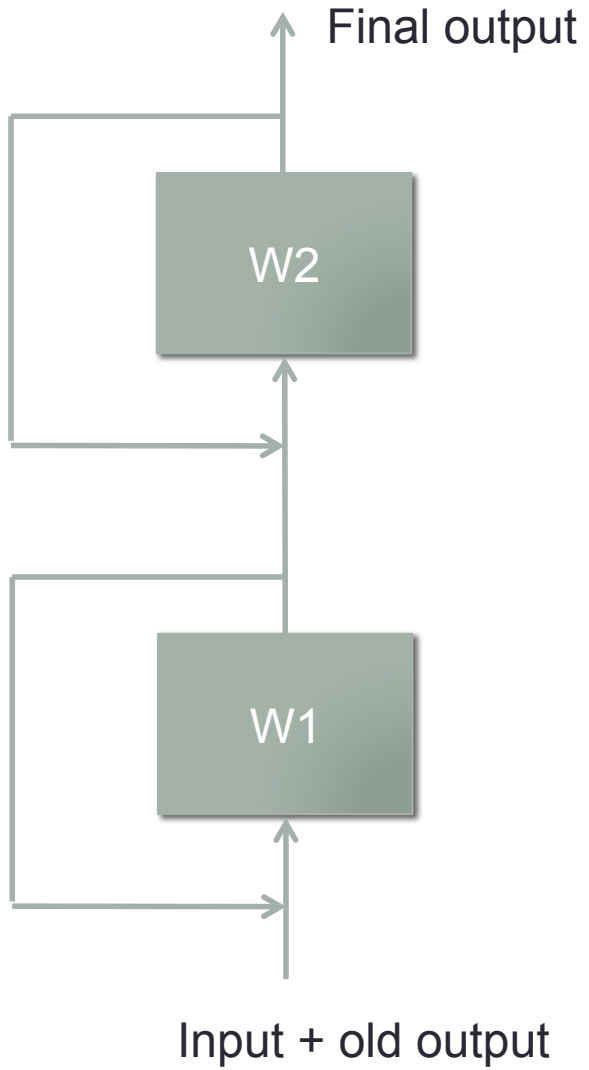
- Unrolling of a recurrent layer.



Parameter sharing

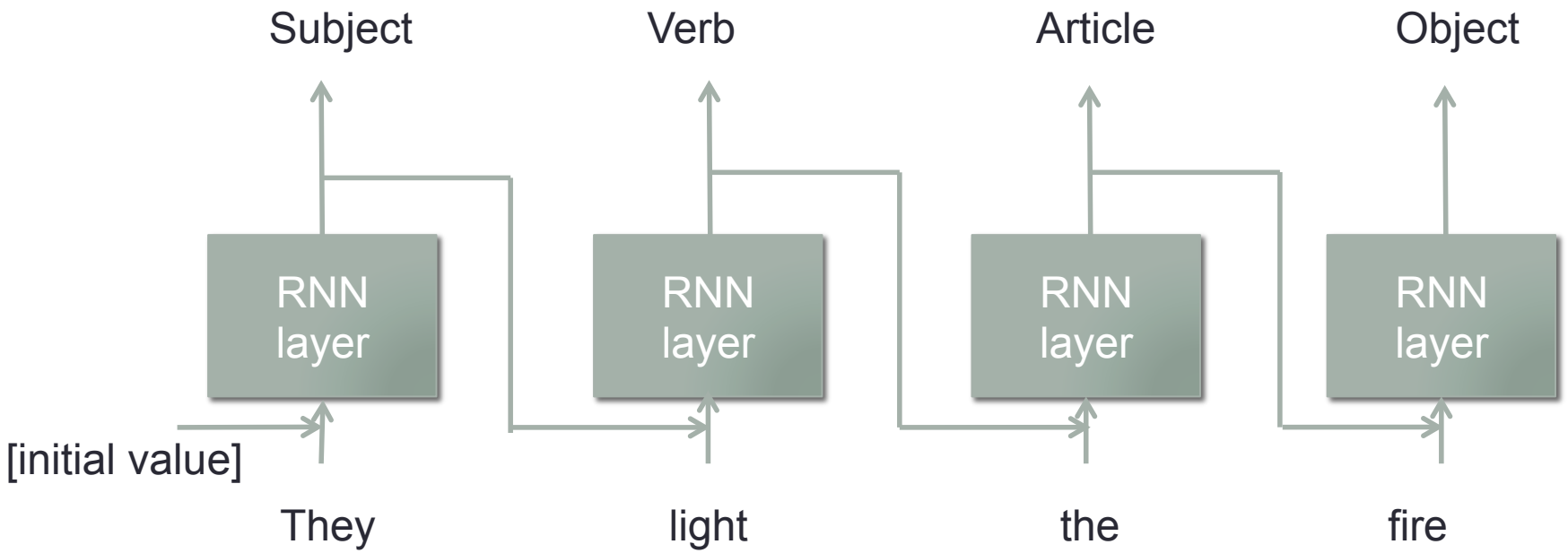
Recurrent neural network (RNN)

- Stacks of recurrent layer



Training a recurrent neural network

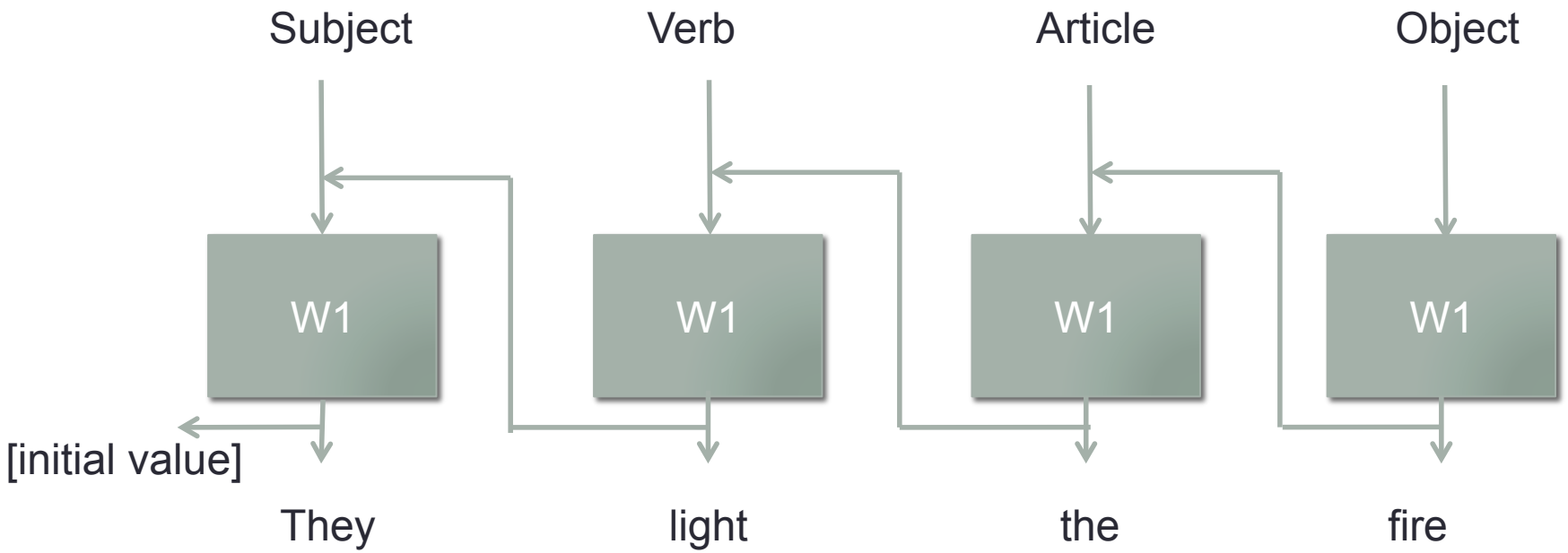
- RNN framework



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

Training a recurrent neural network

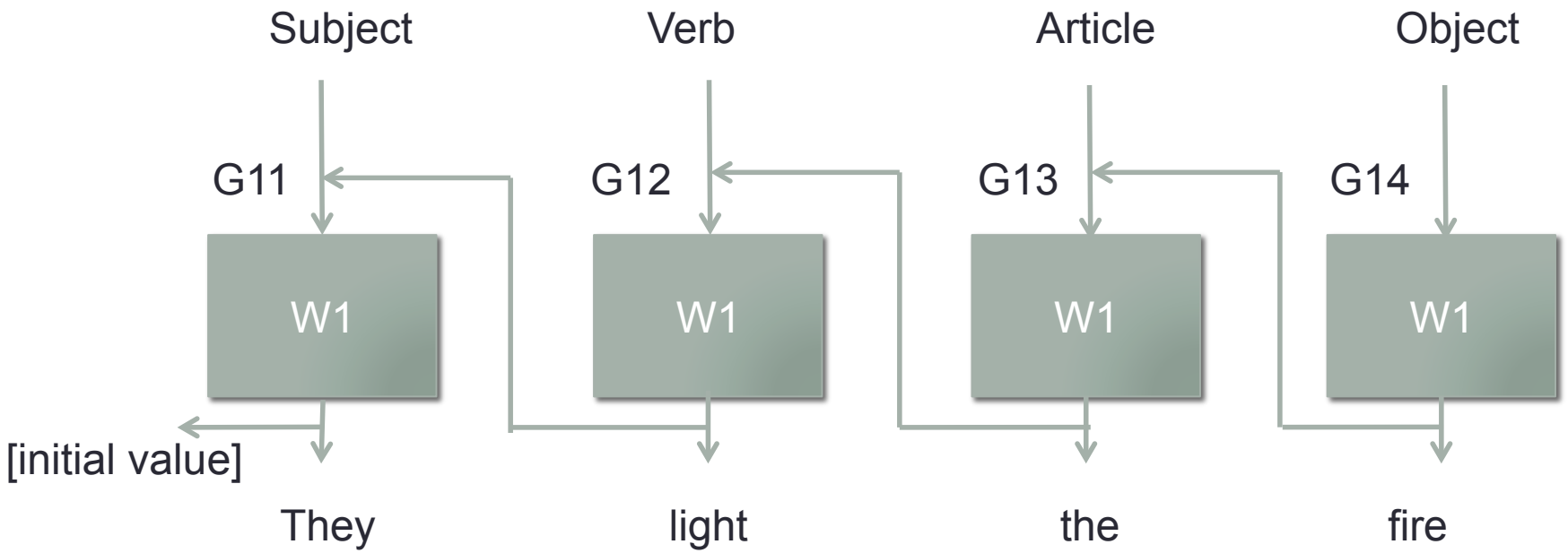
- Backward Computation graph



Backpropagation through time (BPTT)

BPTT

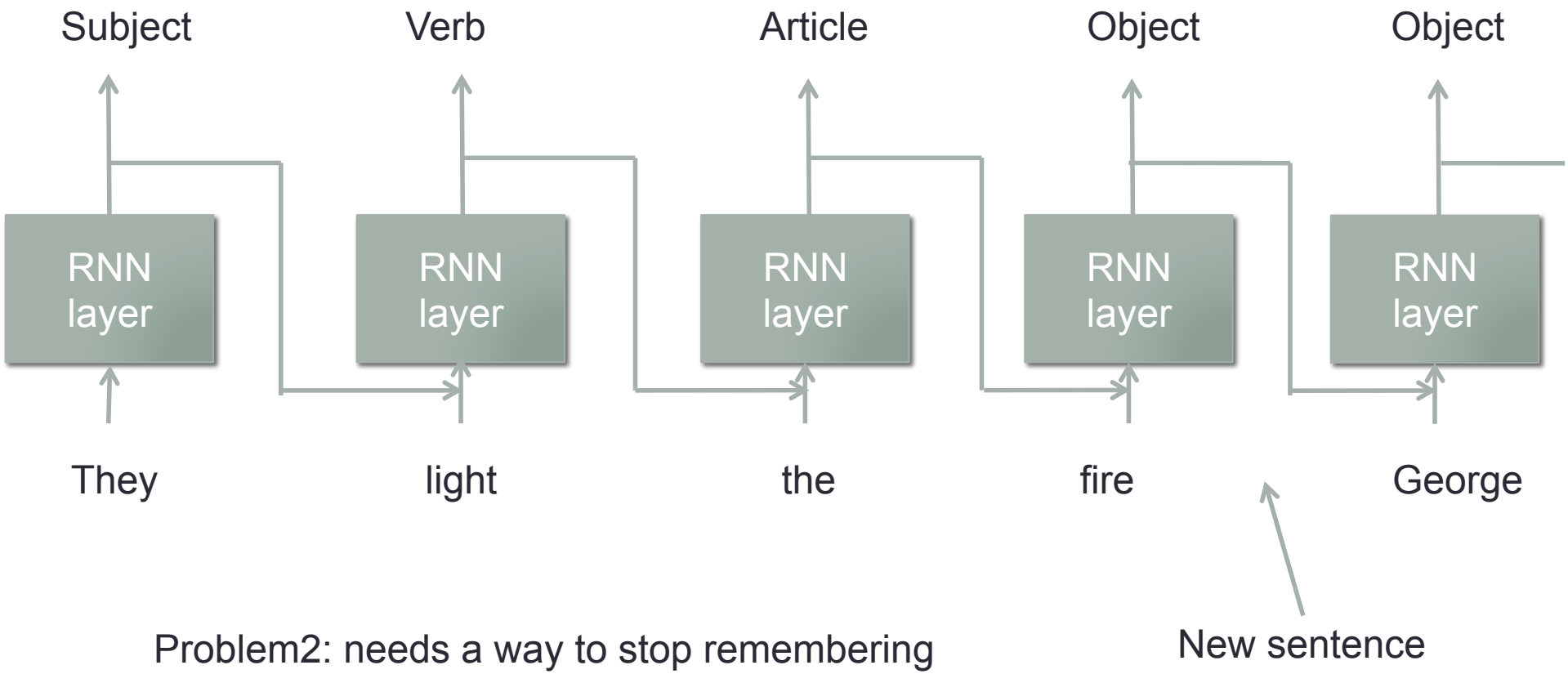
- Backward Computation graph



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

Problem1: cannot deal with infinitely long recurrent
Gradient explosion, vanishing

Recurrent neural network (RNN)

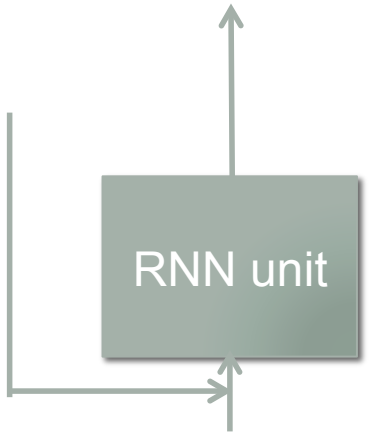


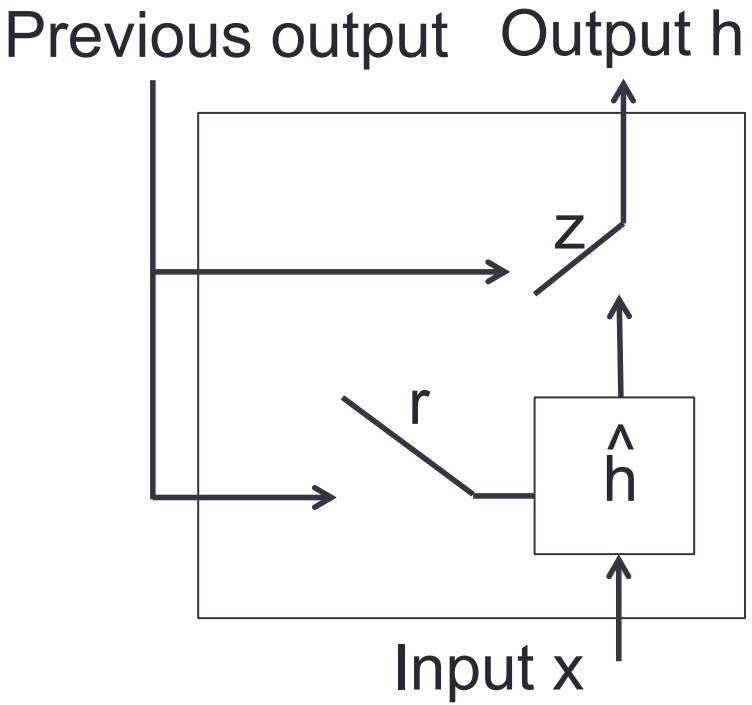
Problem2: needs a way to stop remembering

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

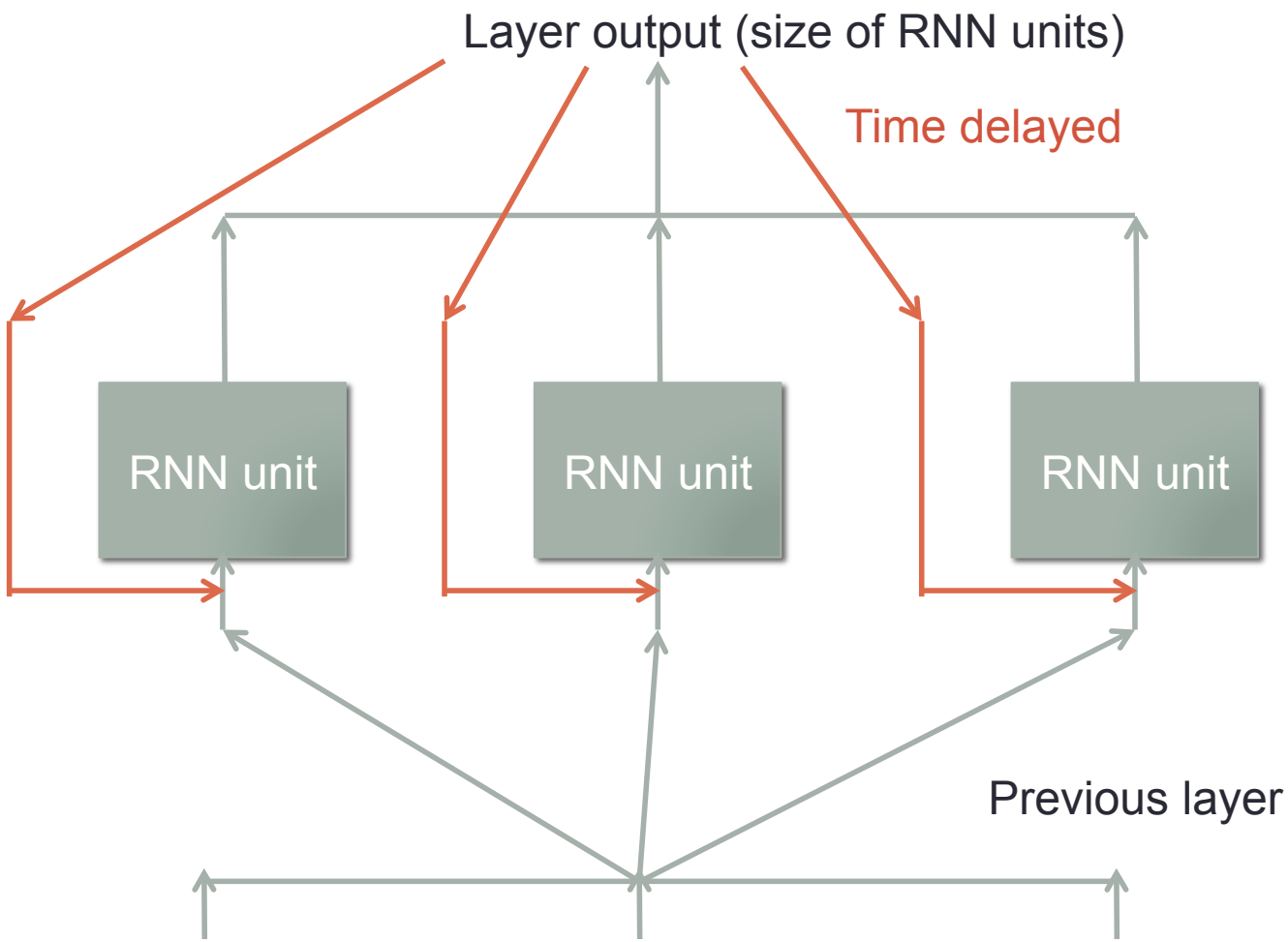
Gated Recurrent Unit (GRU)

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

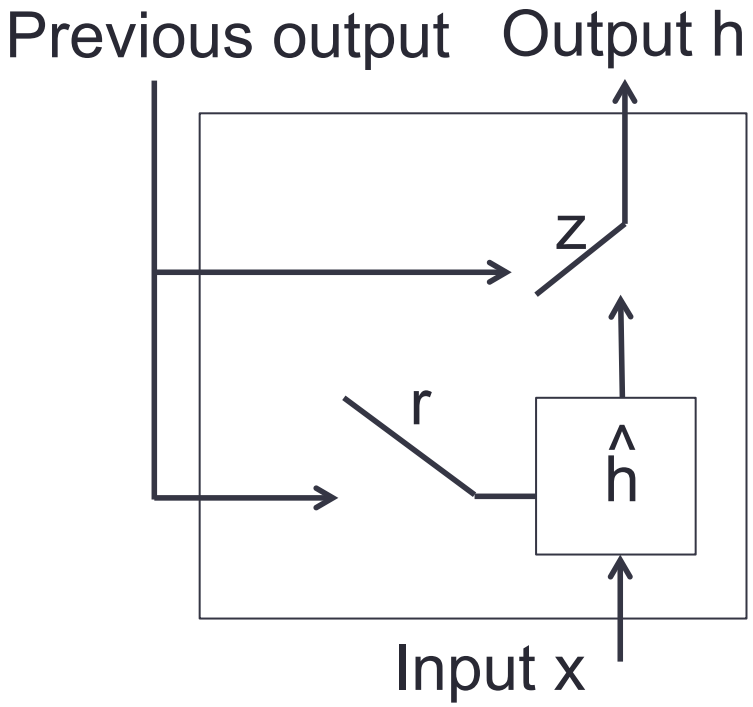




Gated Recurrent Unit (GRU) layer



Gated Recurrent Unit (GRU)

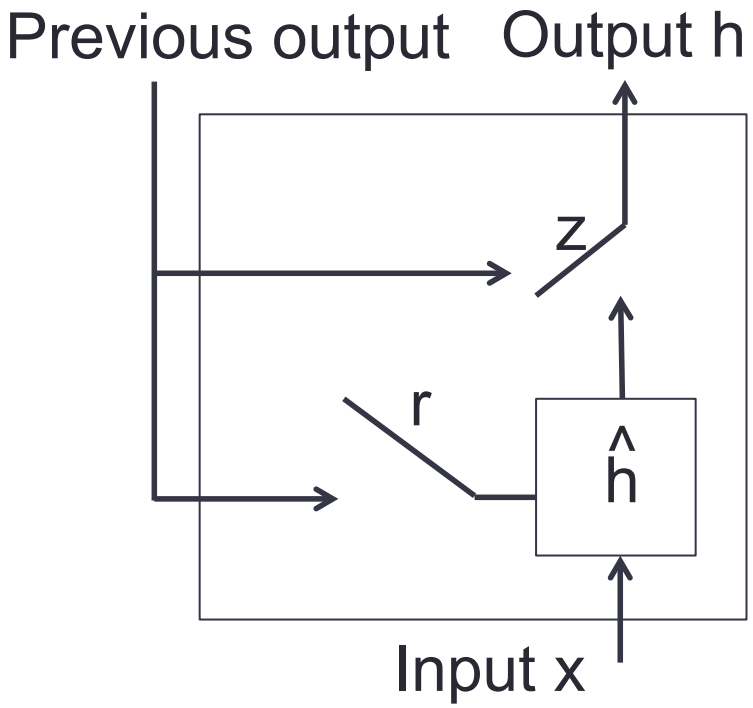


Neuron index

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

time index

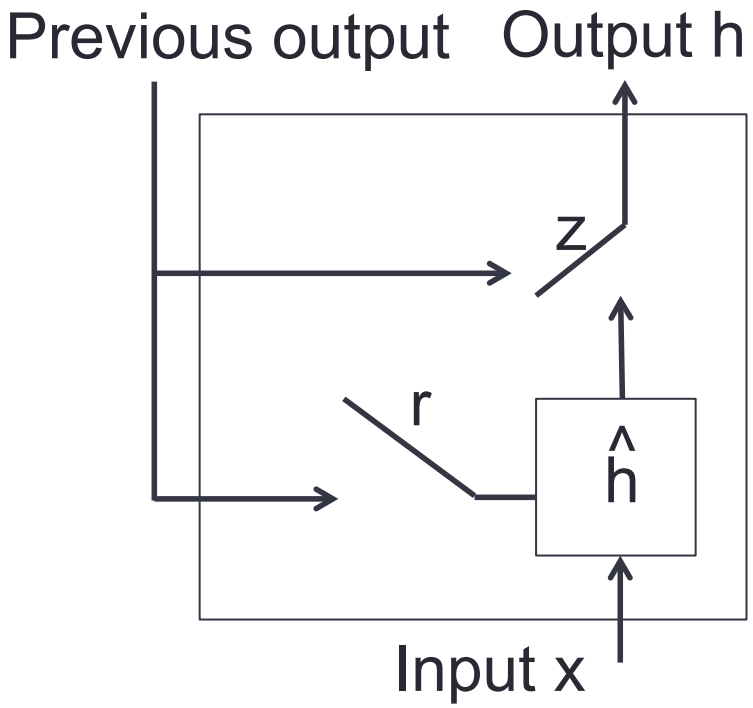
Gated Recurrent Unit (GRU)



$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

One GRU neuron output (scalar)

Gated Recurrent Unit (GRU)



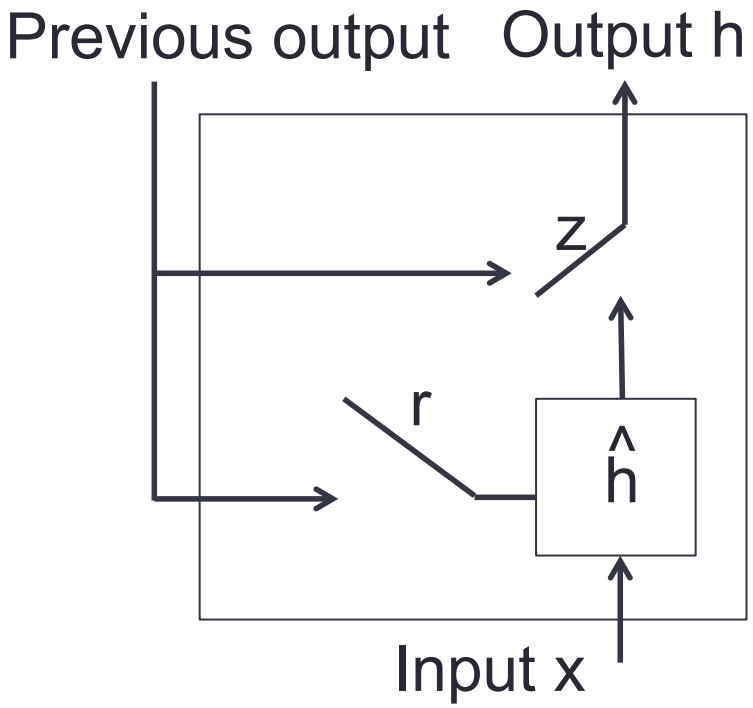
$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

Element-wise product

Linear transform with matrix multiply

Gated Recurrent Unit (GRU)



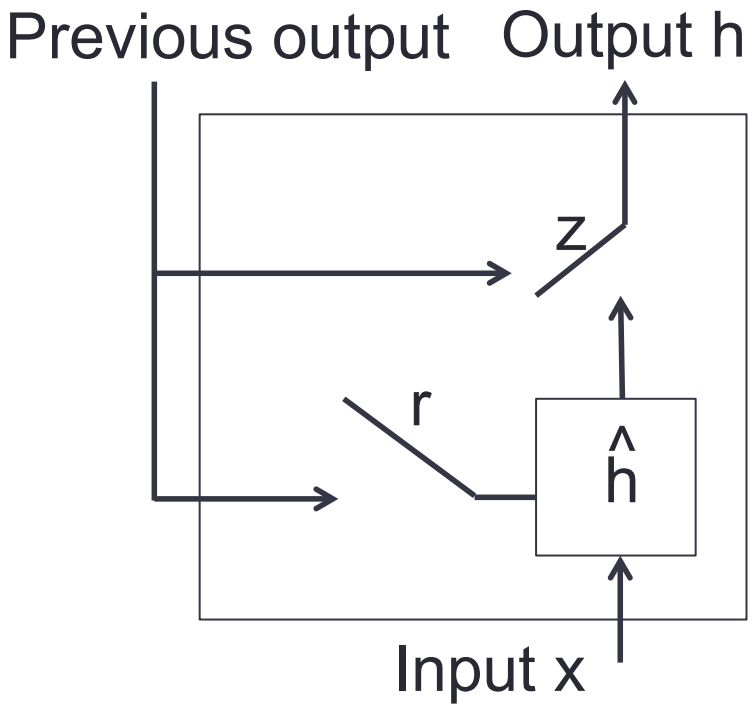
$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

Takes the j-th element

Bound the output

Gated Recurrent Unit (GRU)



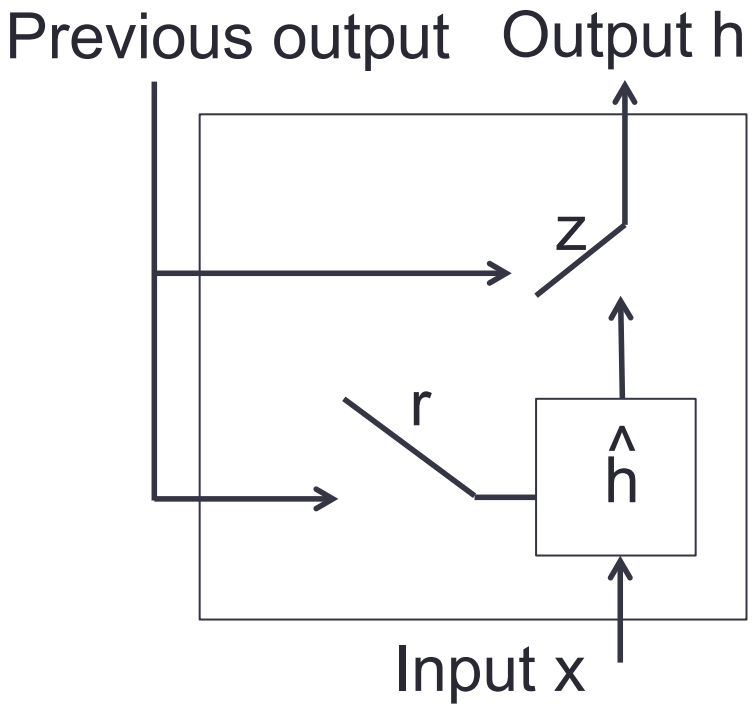
$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$z_t^j = sigmoid^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

Indicates a different set of weights

Gated Recurrent Unit (GRU)



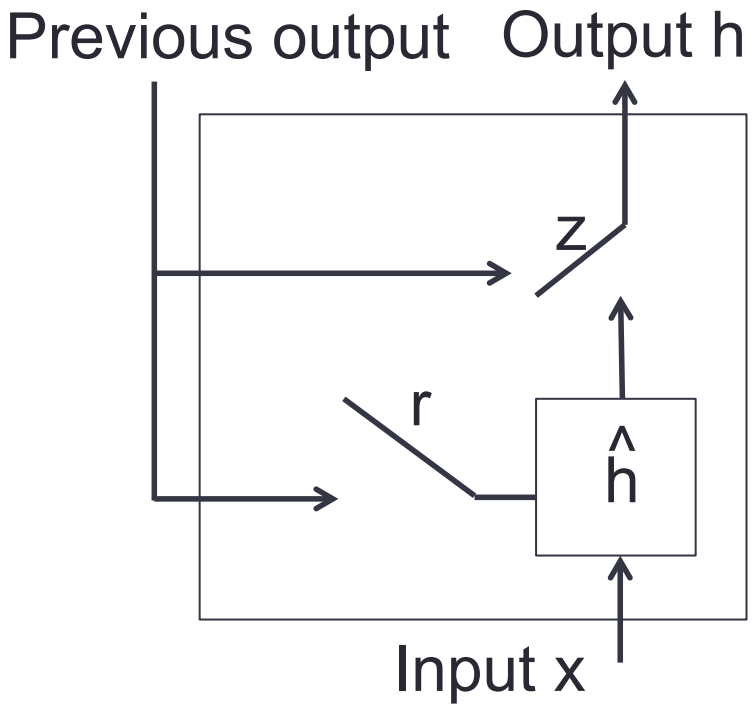
$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$z_t^j = \underline{sigmoid^j}(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

Bounds the output to 0 to 1 for interpolation

Gated Recurrent Unit (GRU)



$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

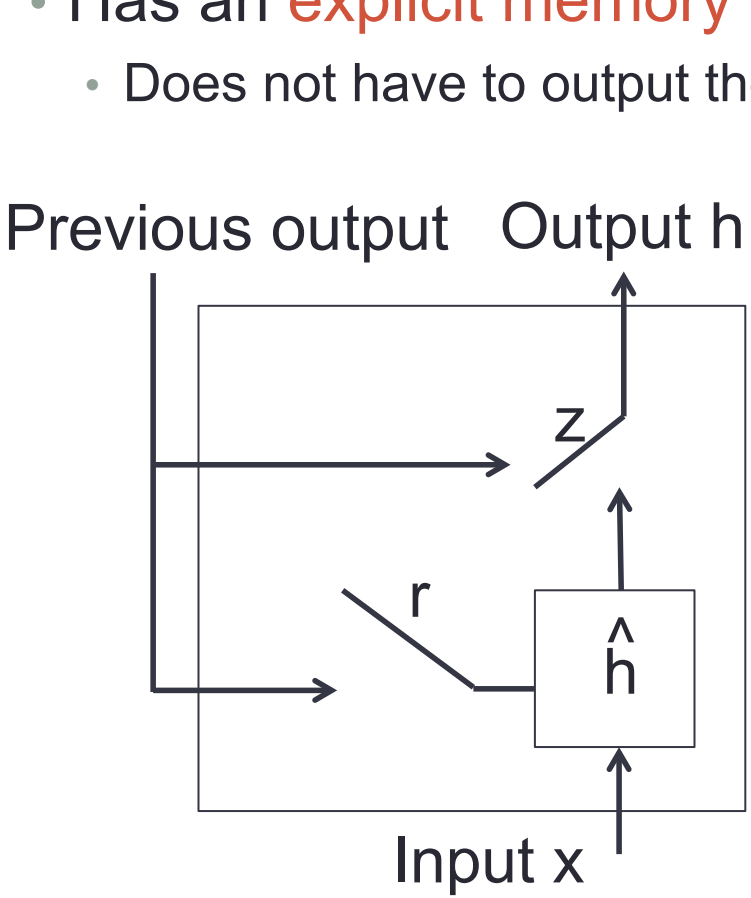
$$\hat{h}_t^j = tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

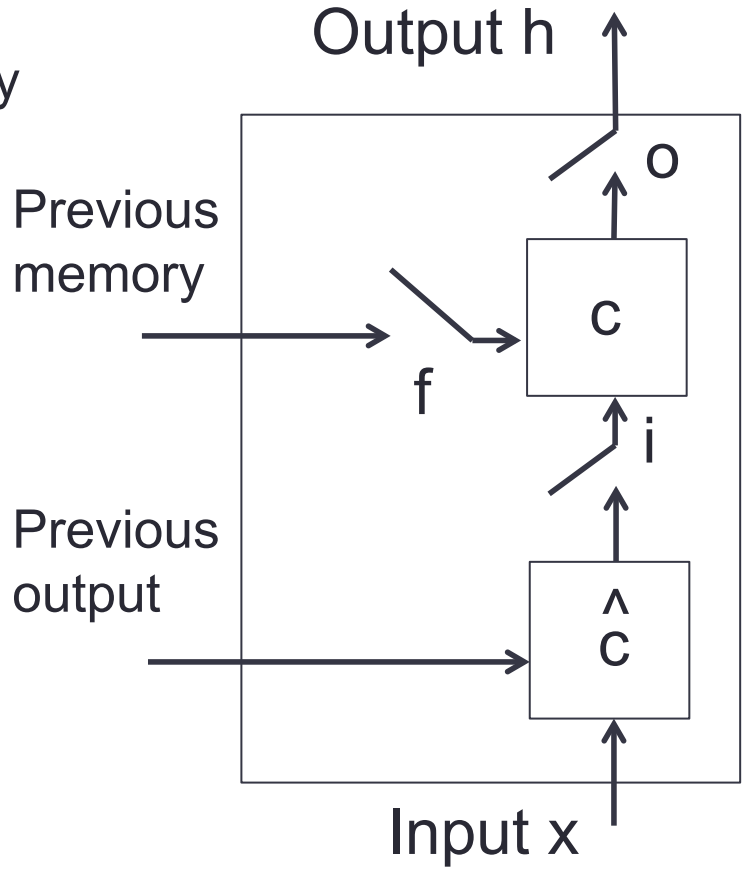
$$z_t^j = sigmoid^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

$$r_t^j = sigmoid^j(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1})$$

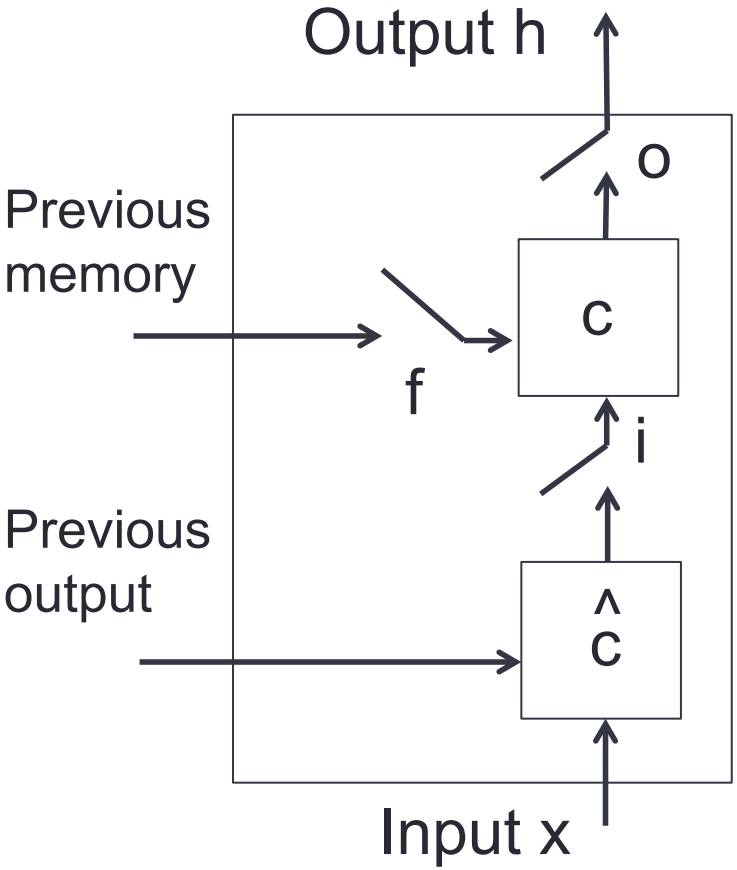
Long Short-Term Memory (LSTM)

- Have 3 gates, forget (f), input (i), output (o)
- Has an explicit memory cell (c)
  - Does not have to output the memory





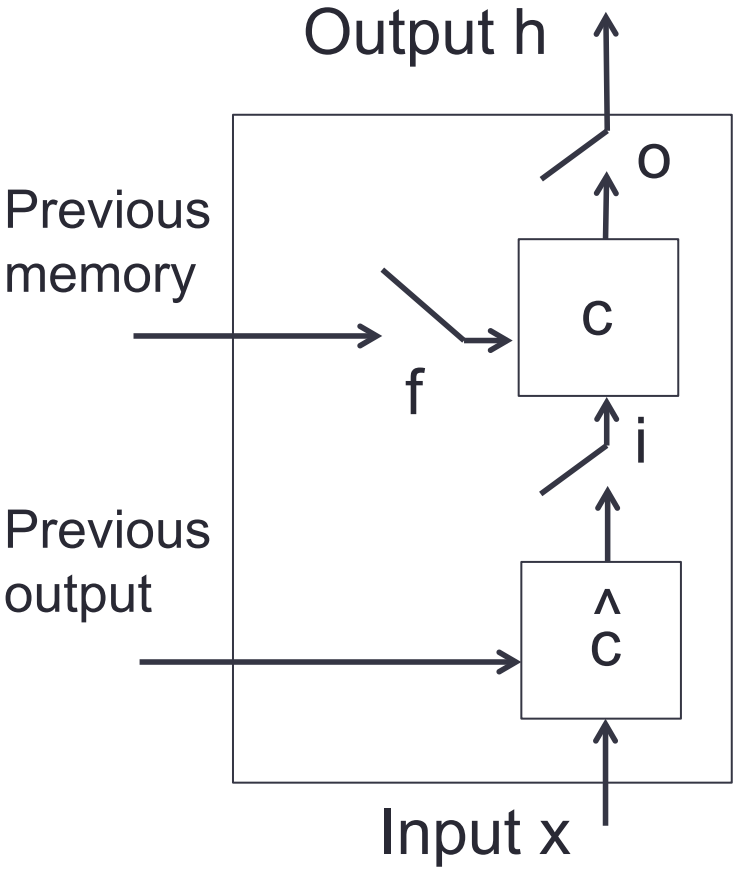
Long Short-Term Memory (LSTM)



$$i_t^j = F^j(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c_{t-1}})$$
$$o_t^j = F^j(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c_t})$$
$$f_t^j = F^j(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_j\mathbf{c_{t-1}})$$

Contribution from memory
"Peephole connection"

Long Short-Term Memory (LSTM)



$$i_t^j = F^j(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c_{t-1}})$$

$$o_t^j = F^j(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c_t})$$

$$f_t^j = F^j(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_j \mathbf{c_{t-1}})$$

$$h_t^j = o_t^j tanh(c_t^j)$$

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \hat{c}_t^j$$

$$\hat{c}_t^j = tanh^j(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})$$

LSTMs

- Now the predominant model in speech recognition
- DNN – 11.3% LSTM - 10.5% on Google voice search
- Usually combined with CNN and DNN for real application

Hasim Sak, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling, 2014

LSTM memories remember meaningful things

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                       struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
            df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

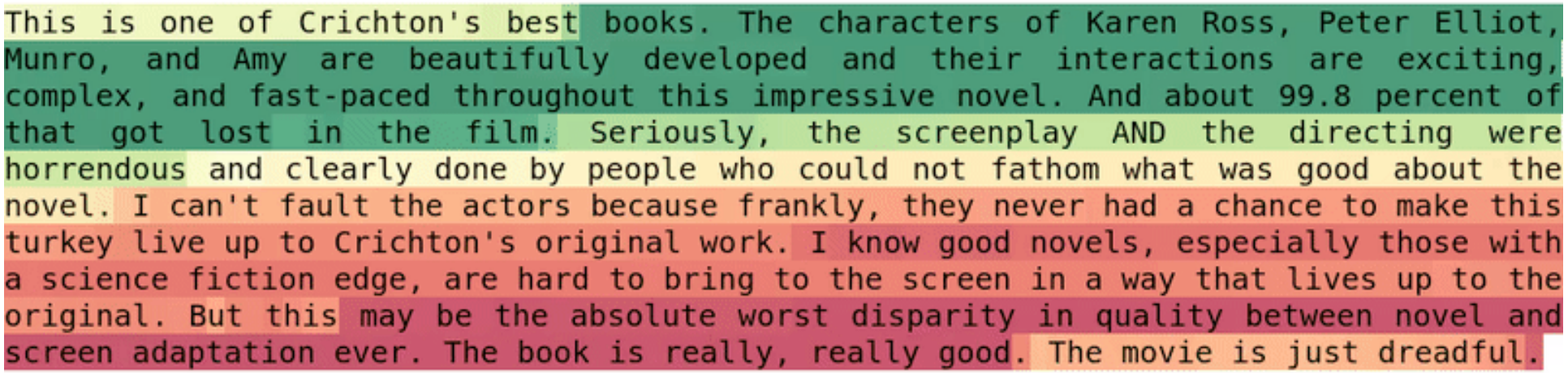
Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Andrej Karpathy, Visualizing and Understanding Recurrent Networks, 2015

The sentiment neuron

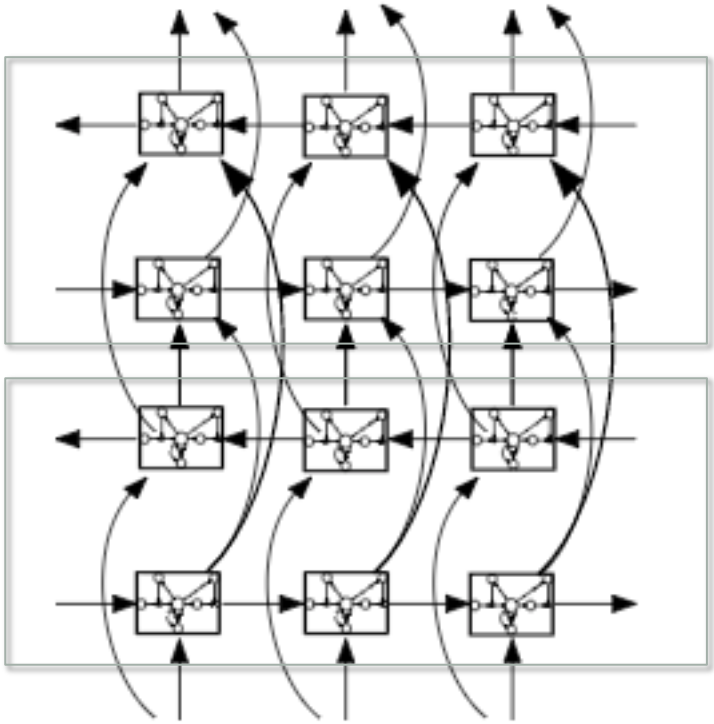
- An LSTM cell that automatically encodes sentiment from Amazon's review training data.



https://blog.openai.com/unsupervised-sentiment-neuron/

Bi-directional LSTMs

- Uni-directional LSTMs only consider information from the past
- Bi-directional LSTMs consider information from the future and the past
- For each bi-directional layer, there are two LSTMs, one from the past, one from the future.



Hasim Sak, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling, 2014

Bi-directional LSTMs

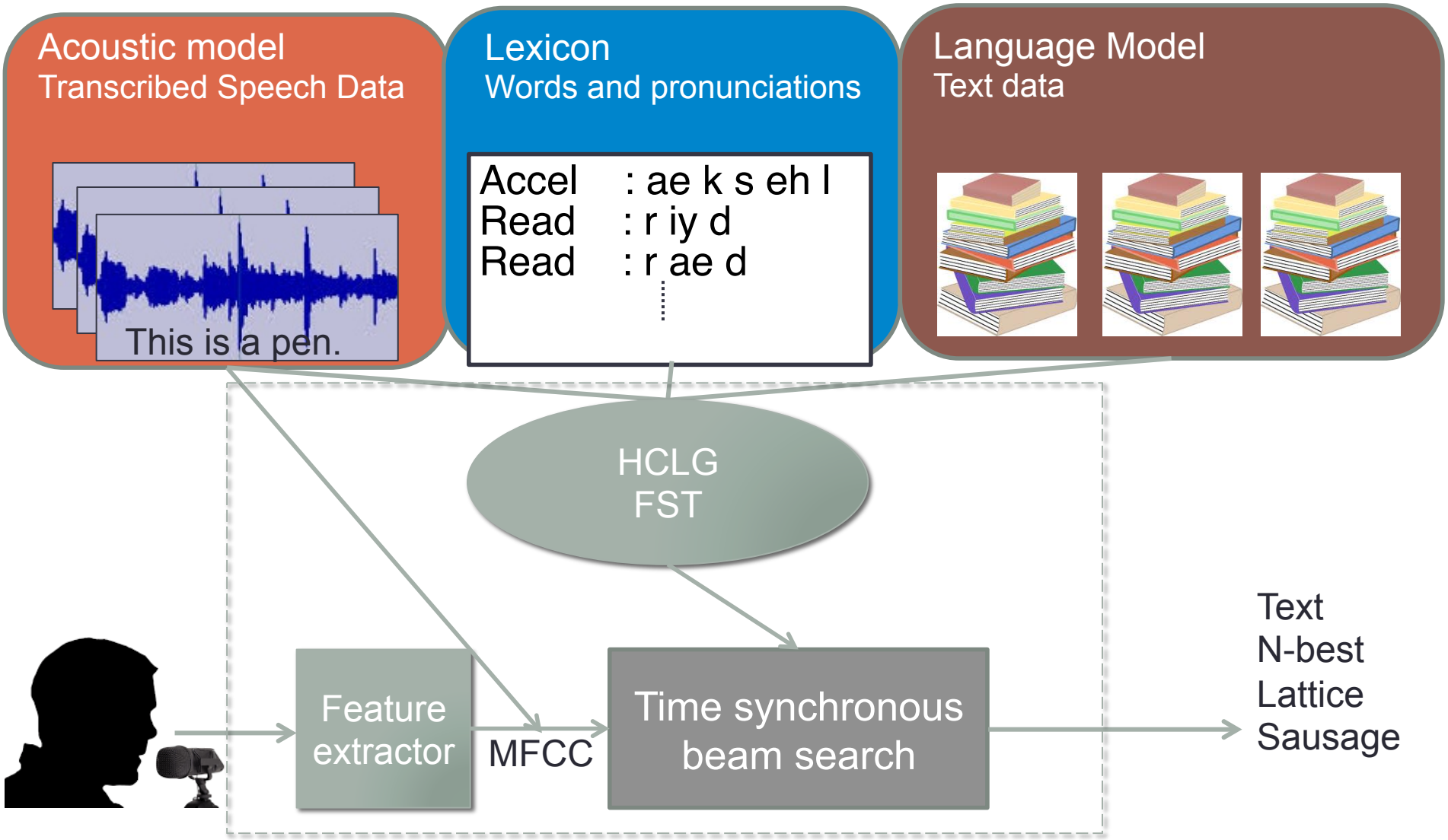
- Some improvement over uni-directional
- For real-time application, only consider a limited amount of frames in the future

Uni-directional vs Bi-directional LSTM on meeting transcription

LSTMP	3	50.7
HLSTMP	3	50.4
BLSTMP	3	48.5
BHLSTMP	3	48.3

Y. Zhang, Highway Long Short-Term Memory RNNs for Distant Speech Recognition 2016

Deep learning in ASR?



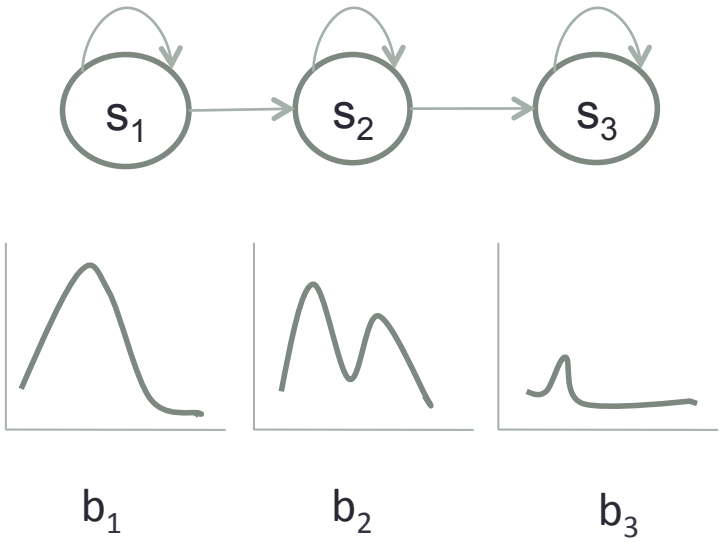
Deep learning in AM

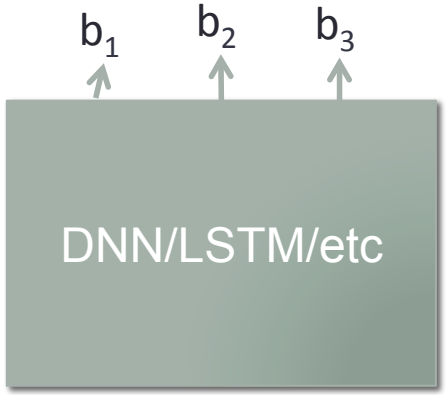
Two main approaches

- Hybrid DNN-HMM
- Tandem

Hybrid DNN-HMM approach

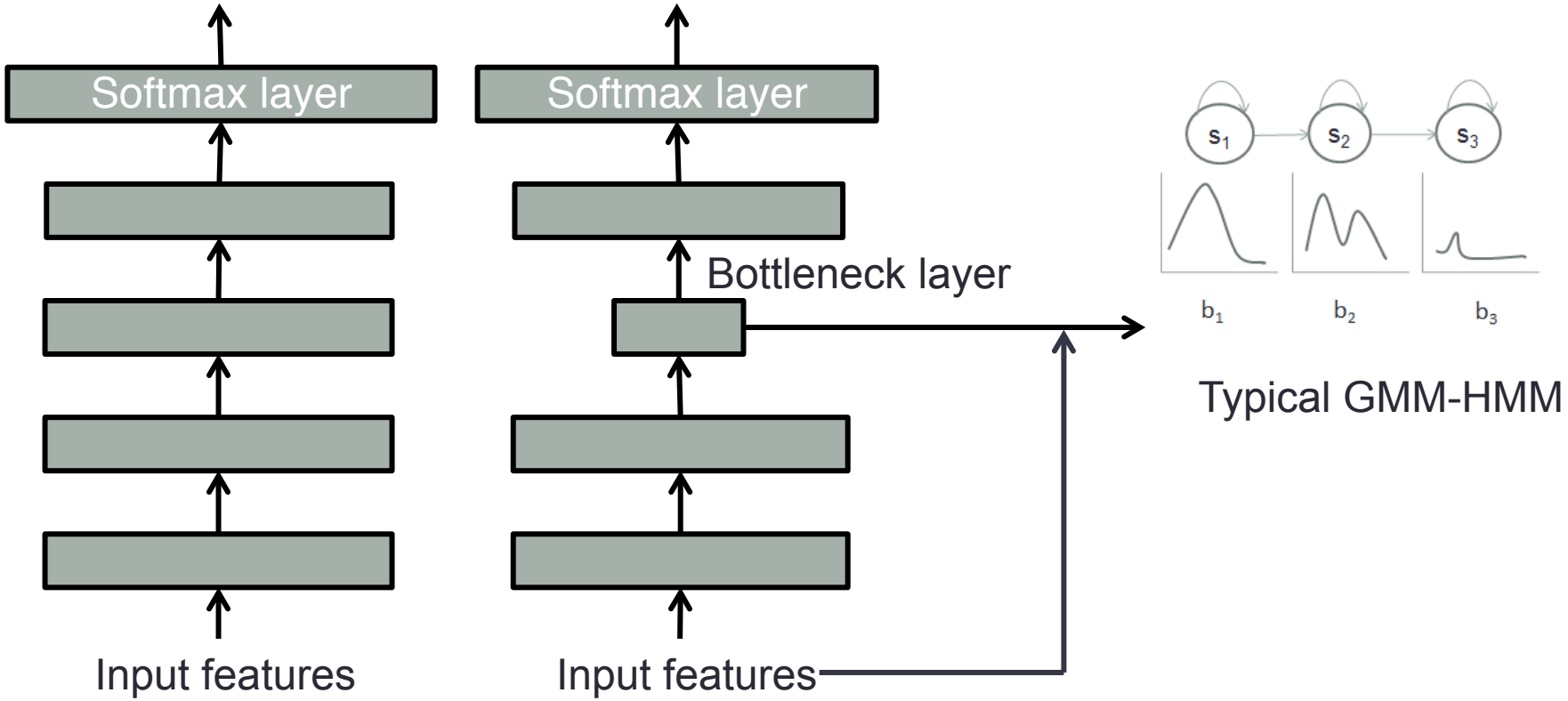
- A typical speech recognizer uses the GMM-HMM framework
  - Emission probabilities are modeled by a GMM
- Instead, model emission probabilities with a DNN
  - DNN gives posteriors, while GMM gives likelihoods. Convert DNN outputs to likelihoods by removing the priors.





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.



Hybrid vs. Tandem

	Hybrid	Tandem
Training		

Both methods still need a starting HMM-GMM model to get alignments and class labels!

Deep learning in LM

- Use recurrent networks (RNN/LSTM/GRU) to predict the next word given current word (history is in the memory)
- Input and output has dimension V
  - Need modifications for large vocab (1m words+)
- However, we cannot model RNNs as FST (infinite memory in time – means infinite number of FST states)
- Can only use RNN-LM in rescoring
  - Generate n-best with regular n-grams
  - Remove LM n-gram scores and add RNN-LM scores
  - Re-rank the n-best

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

Deep learning in LM

- 1B word google benchmark. Perplexity results rather than WER.

Model	Test Perplexity
Large Ensemble (Chelba et al., 2013)	43.8
RNN+KN-5 (Williams et al., 2015)	42.4
RNN+KN-5 (Ji et al., 2015a)	42.0
RNN+SNM10-skip (Shazeer et al., 2015)	41.3
Large Ensemble (Shazeer et al., 2015)	41.0
Our 10 best LSTM models (equal weights)	26.3
Our 10 best LSTM models (optimal weights)	26.1
10 LSTMs + KN-5 (equal weights)	25.3
10 LSTMs + KN-5 (optimal weights)	25.1
10 LSTMs + SNM10-skip (Shazeer et al., 2015)	**23.7**

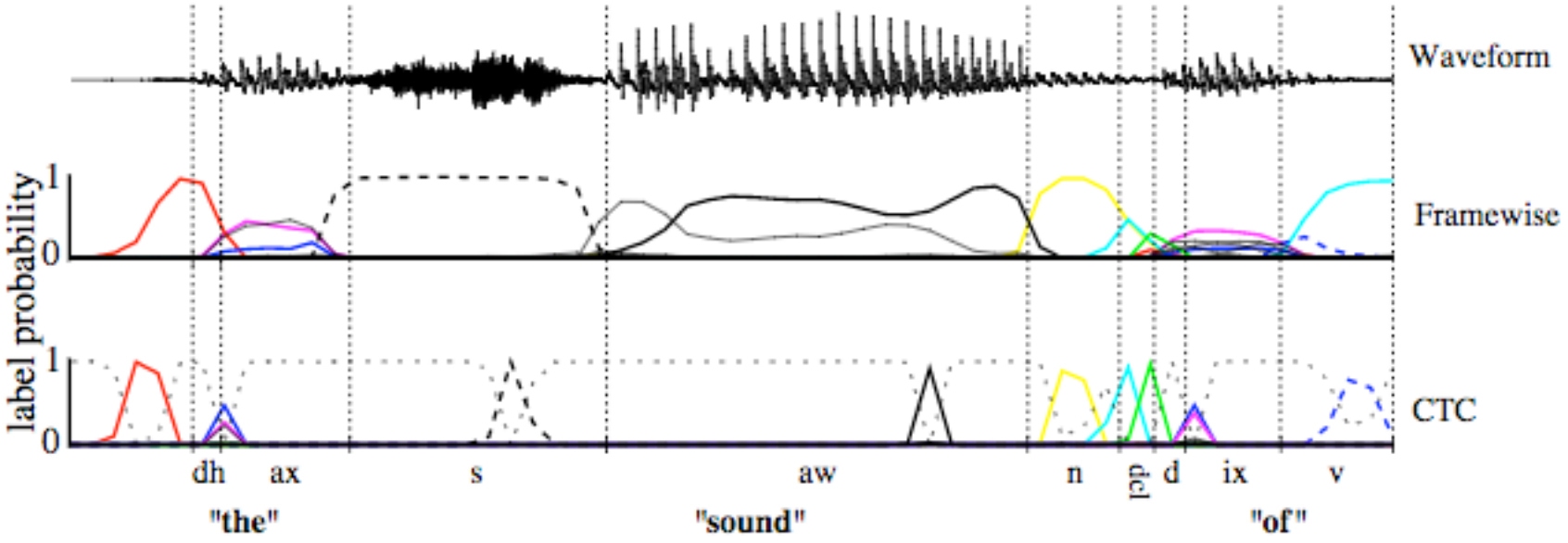
Jozefowiz, R., Exploring the Limits of Language Modeling, 2016

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
- Needs large amounts of data (1000+ hours) – people also tried on smaller data, but need typical models as a starting point (for alignments).
- Next class!

Connectionist Temporal Classification (CTC)

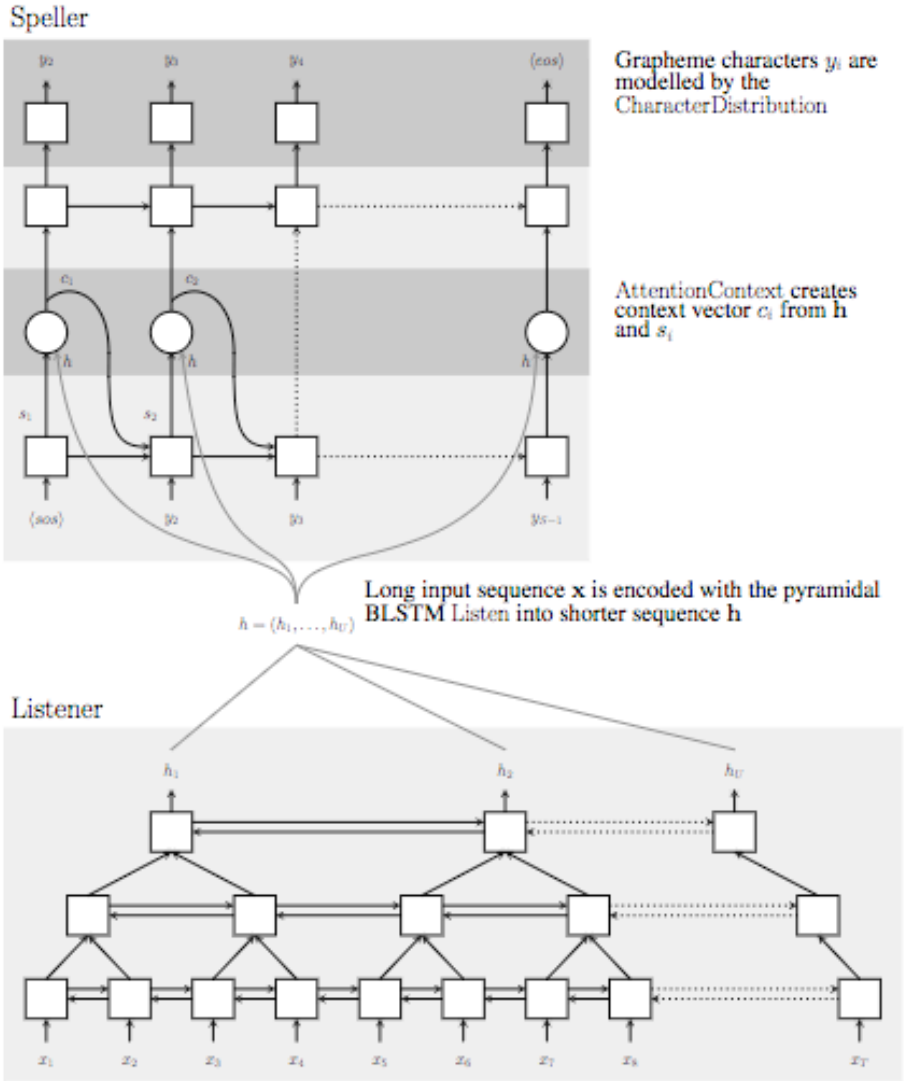
- A kind of end-to-end modeling
- Only output words/characters at certain spikes, while DNN-HMM emits every frame
- Has a background class for no output

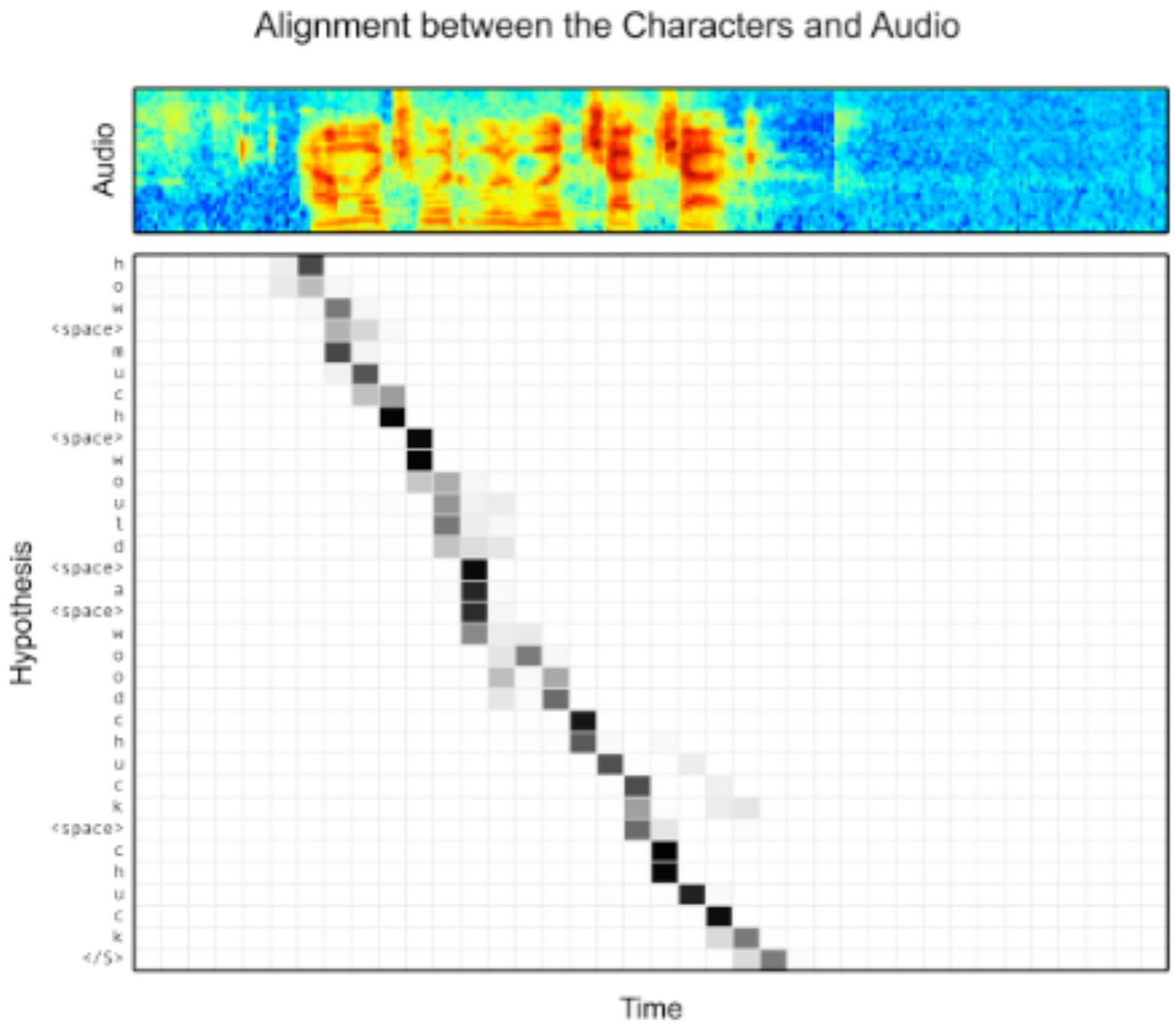


Graves, A. Connectionist Temporal Classification, 2006

Attention mechanism

- Automatically select which frame/input is of current interest





Chan, W. Listen, Attend and Spell, 2015

Attention model in NLP

- Attention model can help with long context sentences for NLP

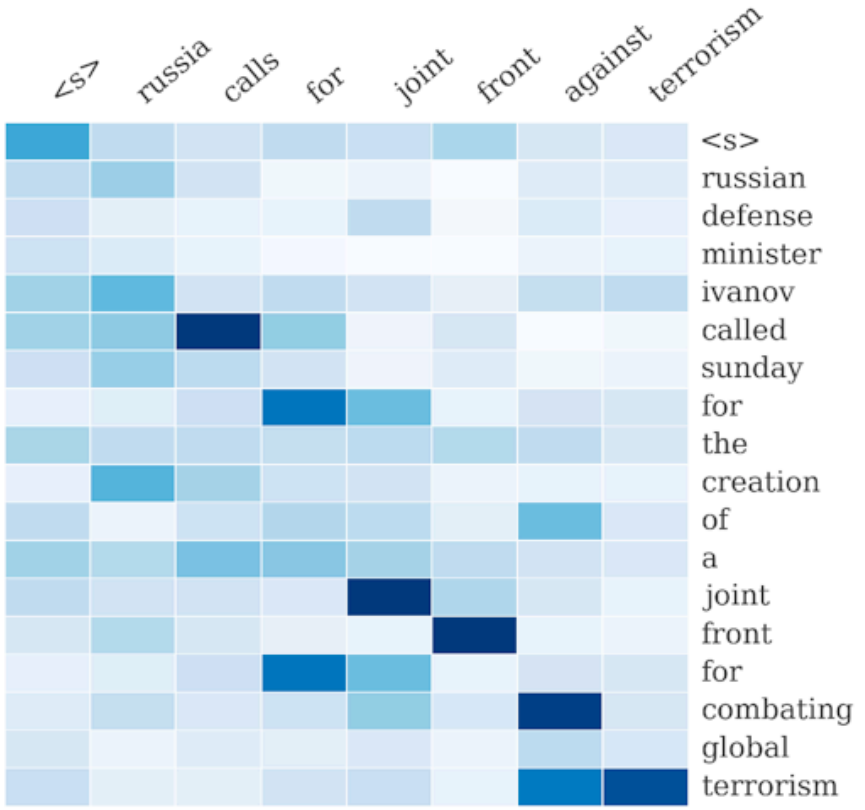


Figure 1: Example output of the attention-based summarization (ABS) system. The heatmap represents a soft alignment between the input (right) and the generated summary (top). The columns represent the distribution over the input after generating each word.

Rush A., A Neural Attention Model for Sentence Summarization, 2015

Other uses of Neural network models

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

Word2Vec

- Recall, we can understand the meaning of the word from its context
- Train a neural network that predicts what are likely words around an input word
- Use the hidden activations as features (just like bottleneck features in ASR)

Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Word2vec

- The vector representations (embeddings) has interesting properties
- Similar meaning words are near each other (euclidean distance).
- You can also add and subtract meanings.
- Word2vec is the most used NLP feature currently (for pretty much all NLP tasks)

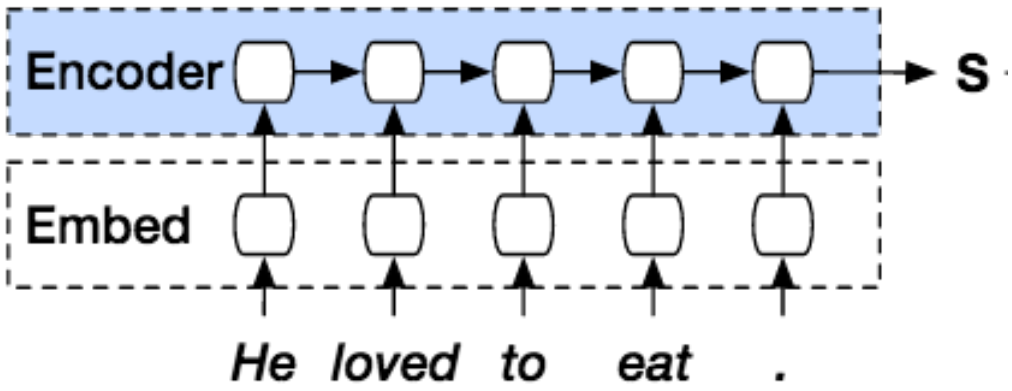
Thai word2vec demo: http://bit.ly/2s0SNHl

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

Sentence encoder

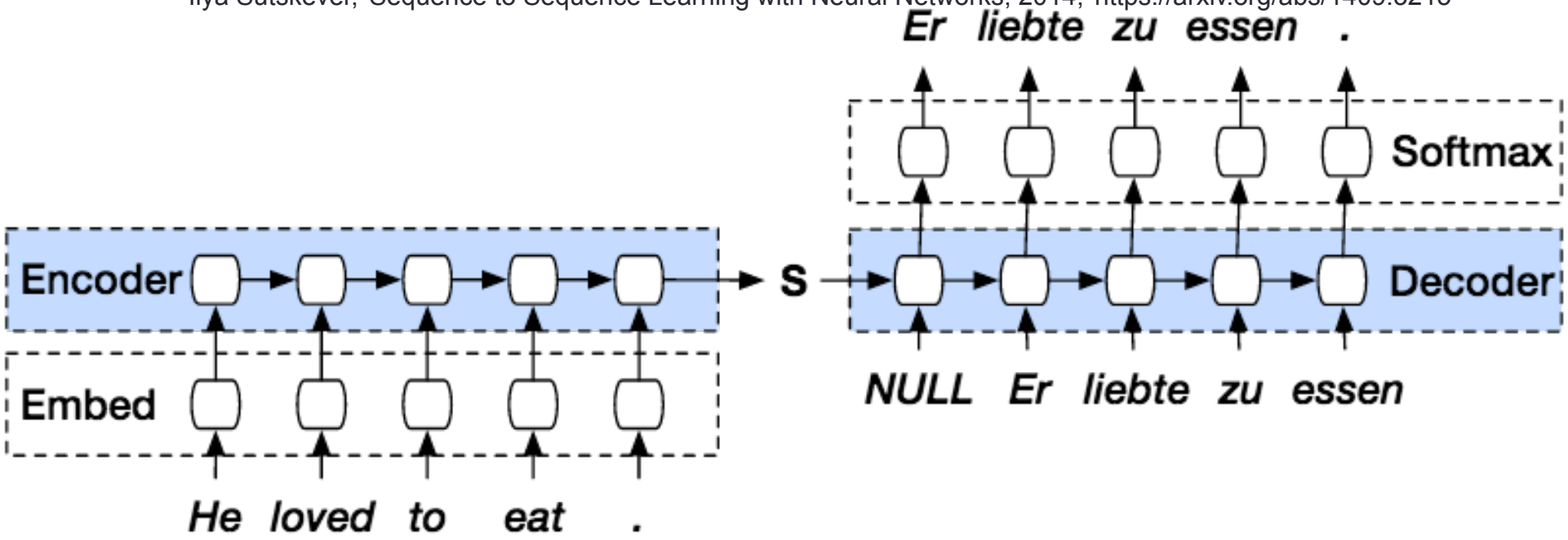
- How can we represent sentences?
- Similarly, use the internal states as the representation
- LSTMs/GRUs are good for sequence task



Machine translation using encoder-decoder

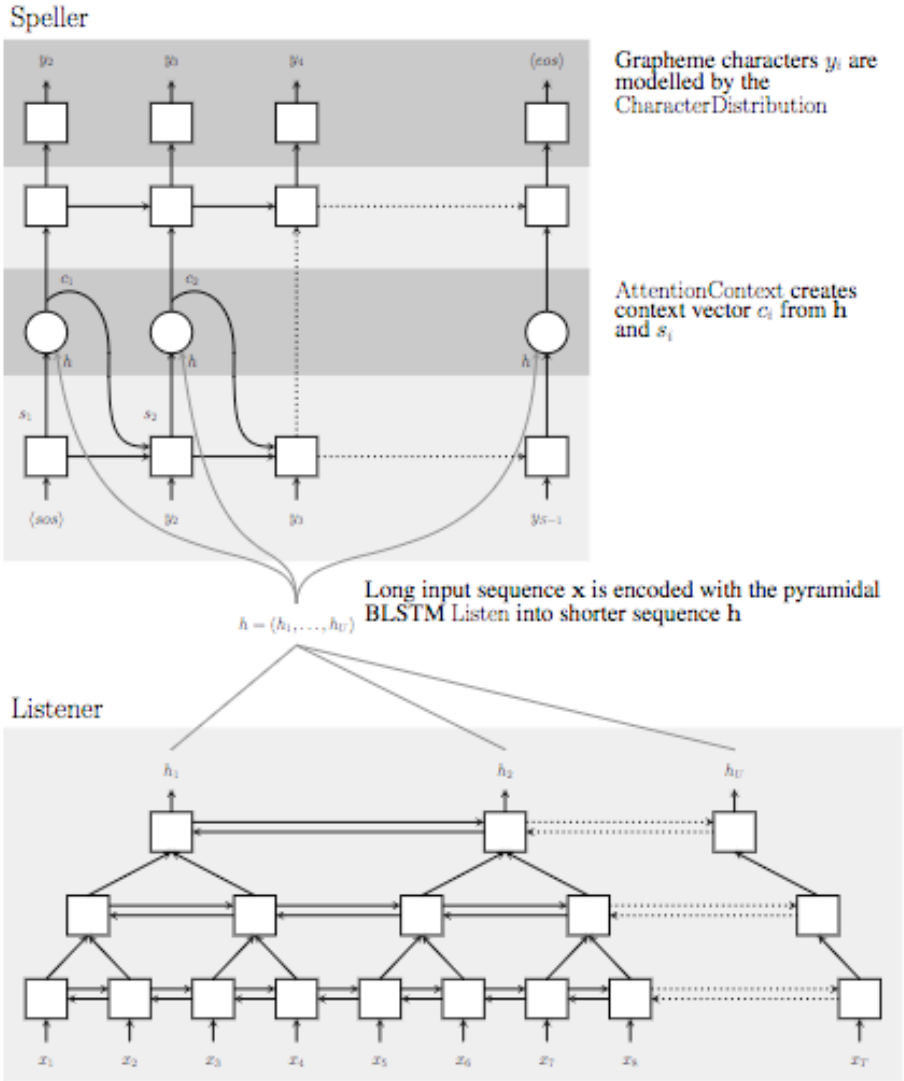
- Use another LSTM as the decoder
- Input to the LSTM is the encoded sentence and the previously output word

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

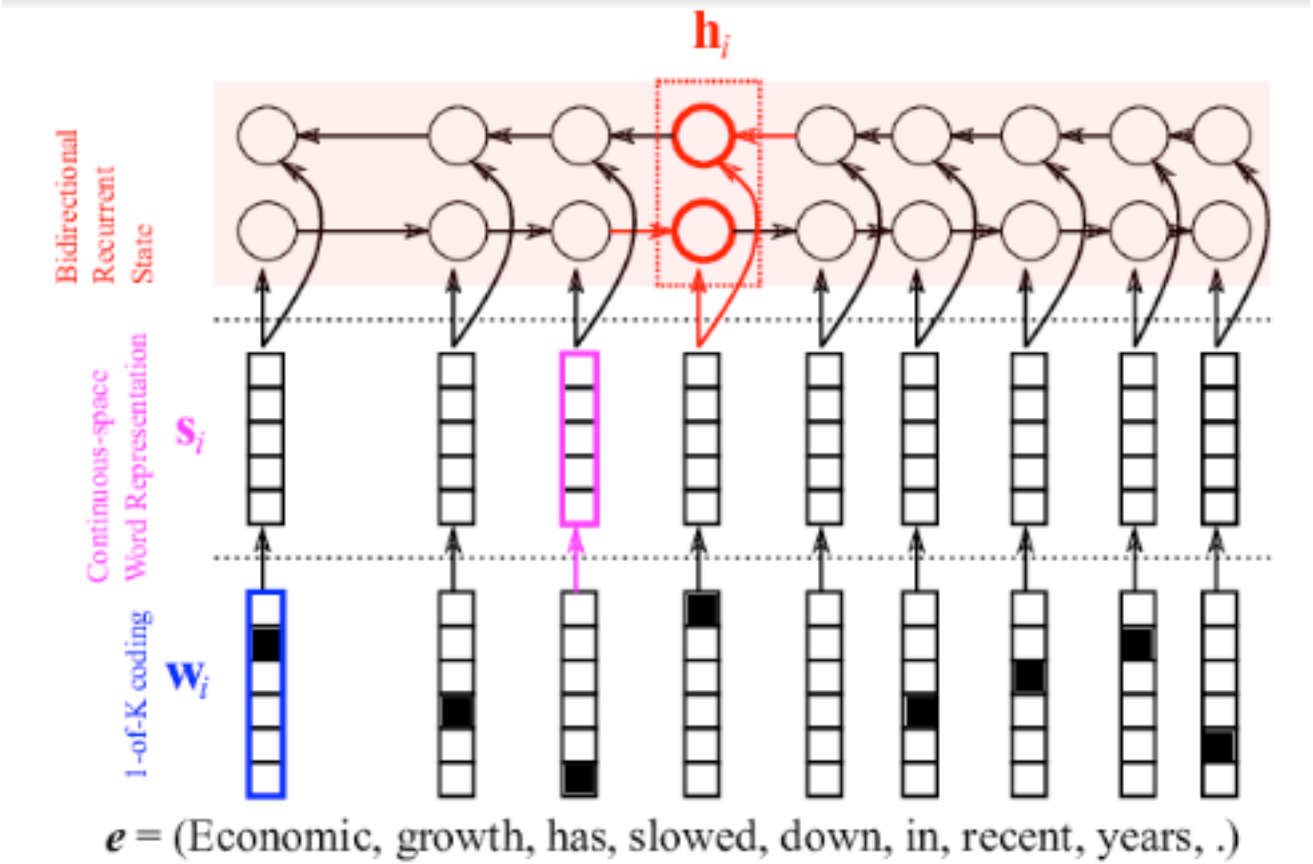


Attention mechanism

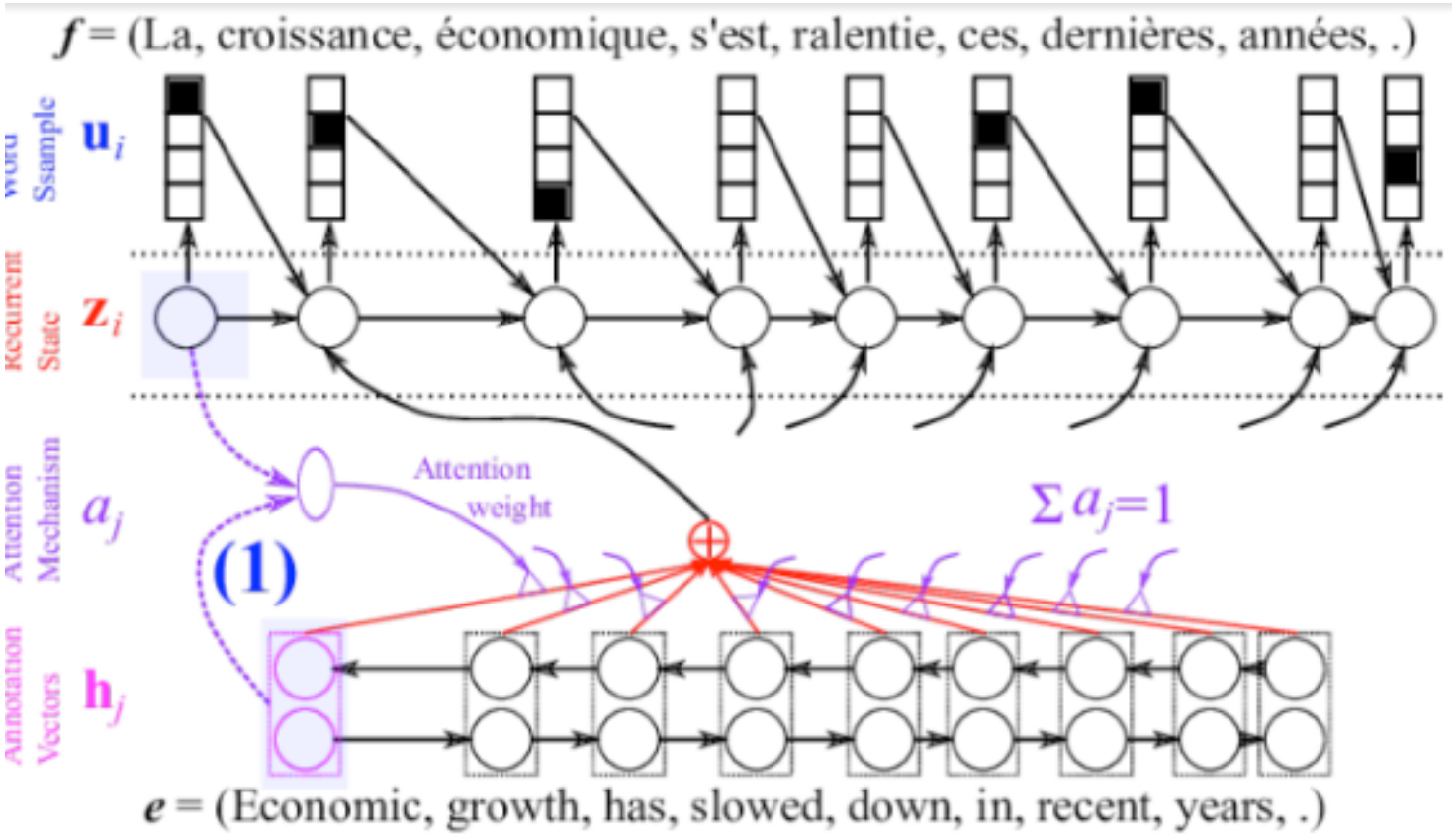
- Automatically select which frame/input is of current interest



Encoder

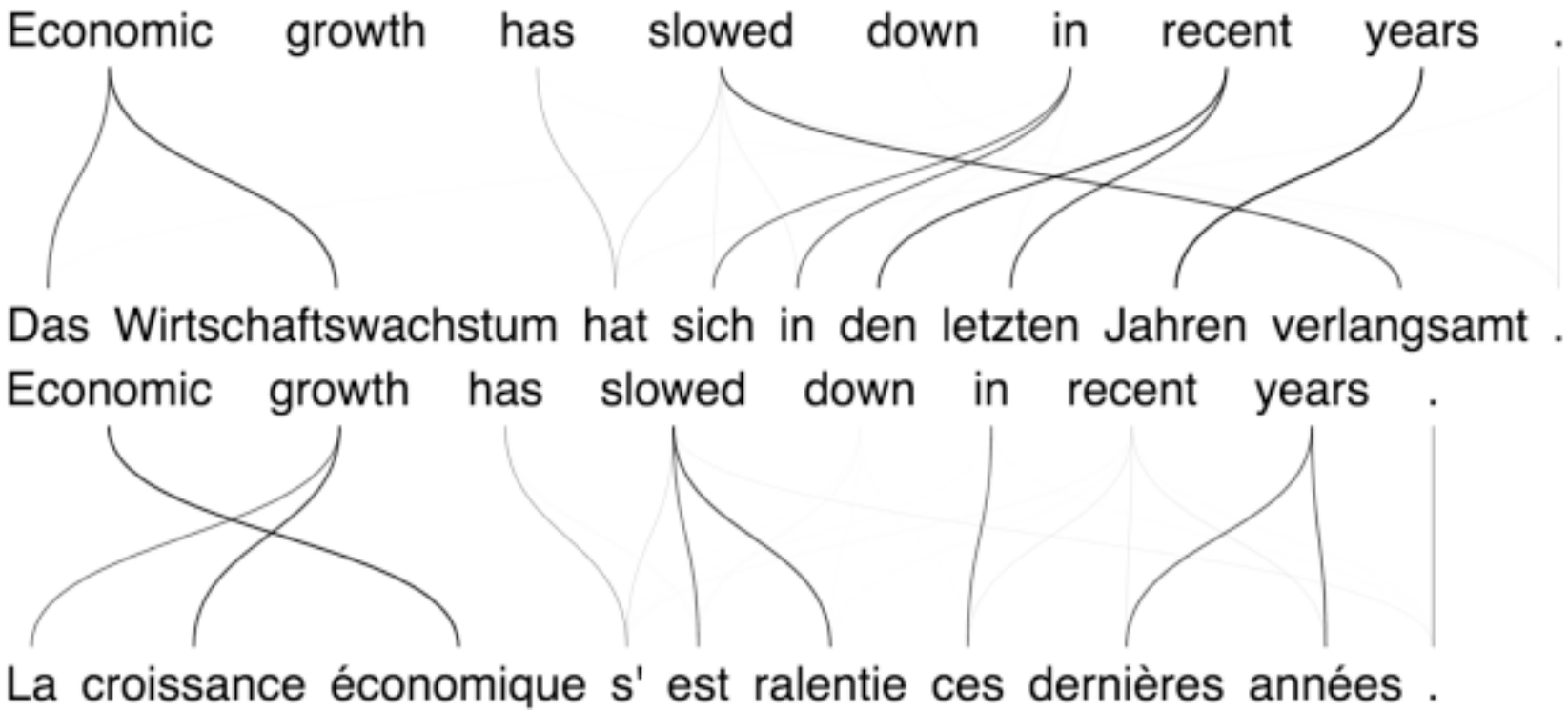

$\mathbf{h}_i$
Bidirectional Recurrent State
Continuous-space Word Representation
$\mathbf{s}_i$
1-of-K coding
$\mathbf{w}_i$
$\boldsymbol{e}$ = (Economic, growth, has, slowed, down, in, recent, years, .)

Decoder



http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attention visualization



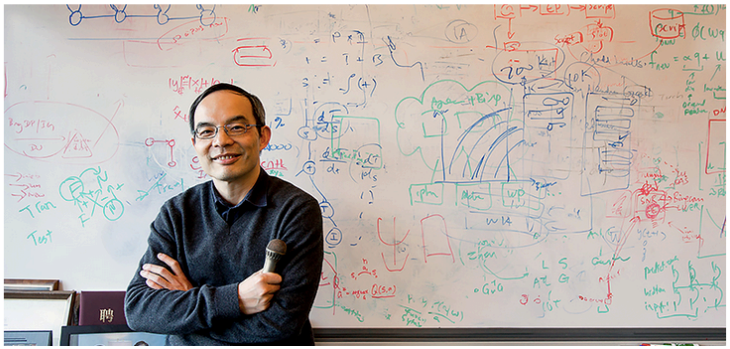
What's a good candidate for DNN applications

- Anything human can complete in 1 second
- More sophisticated tasks still in progress. Many already surpass human capabilities
  - Machine Translation
  - AutoML
  - Reinforcement learning : AlphaGo Zero
  - Chemical synthesis
- Biggest bottleneck is still labeled data

https://blogs.microsoft.com/ai/machine-translation-news-test-set-human-parity/

Microsoft reaches a historic milestone, using AI to match human performance in translating news from Chinese to English

Mar 14, 2018 | Allison Linn

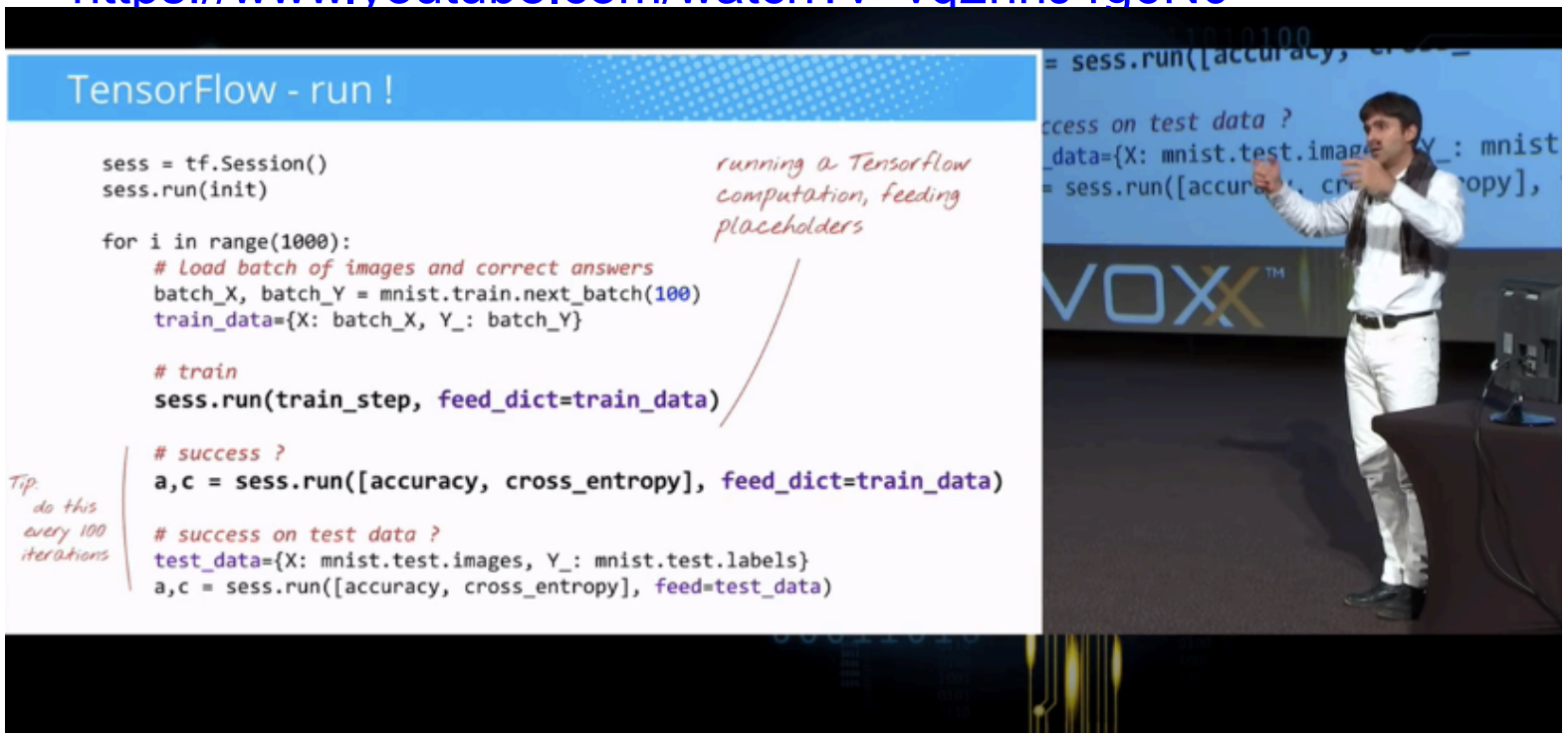


Tricks to reduce training data

- Transfer learning
  - Learning feature in one task and use it in another
- Multi-task training
  - Train the network to learn extra things
- Unsupervised training – helps find better initialization
  - Auto-encoders
- Generative Adversarial Network (GAN)
  - Discriminative meets Generative models!
  - Very popular area

Where to learn more

- Time commitment 3 hours:
  - Tensorflow and deep learning - without a PhD by Martin Görner (other talks also available on newer topics)
  - https://www.youtube.com/watch?v=vq2nnJ4g6N0

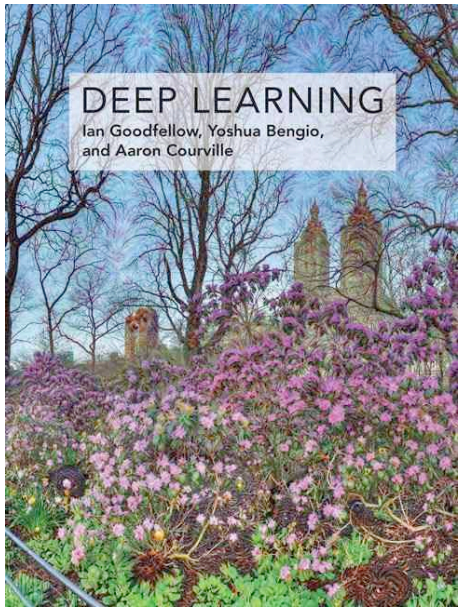


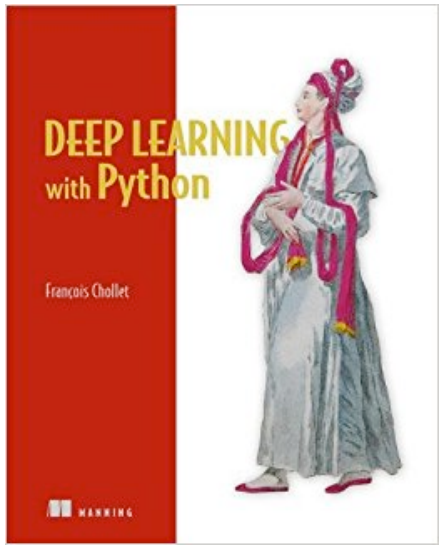
Where to learn more

- Fei-Fei Li and Andrew Karpathy's Computer Vision class (Stanford cs231n 2015/2016)
  - Full course about deep learning (vision oriented)
  - http://cs231n.stanford.edu/
- Pattern Recognition & NLP courses
- Google ML course
- https://developers.google.com/machine-learning/crash-course/ml-intro

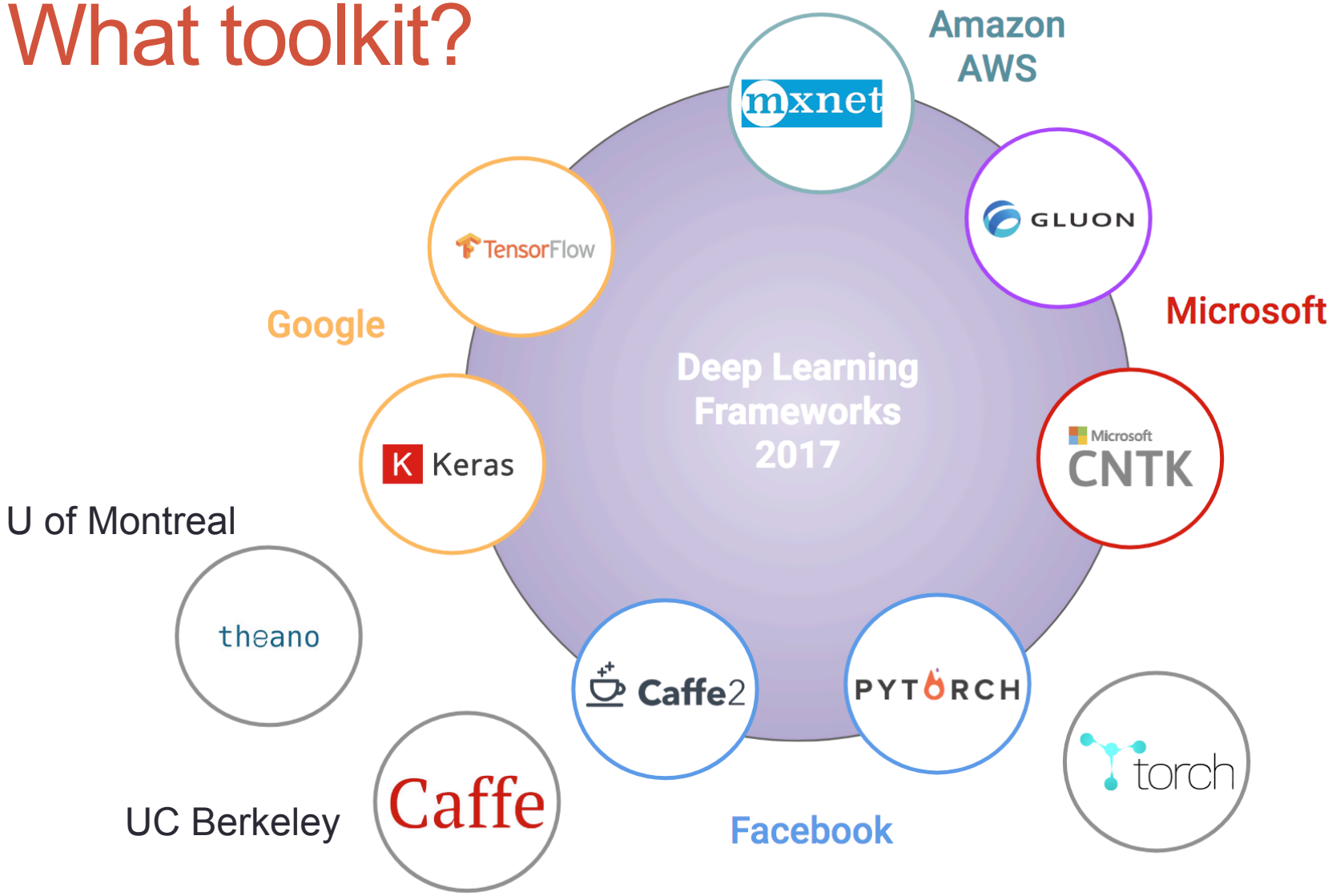
Where to learn more

- http://www.deeplearningbook.org/
  - Good coverage
  - more like a big review article
- Deep learning with python
  - Lots of Keras code examples





What toolkit?



https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750

Tools

- Tensorflow + Keras frontend (python)
  - Probably the best place to start if you have no particular task in mind
- PyTorch
  - Good for creating new networks
- Caffe
  - Geared towards vision tasks
- Kaldi
  - Only for ASR, small community
  - nnet and nnet2 for fully connected
  - nnet3 for more sophisticated things
  - cntk : microsoft generic toolkit, easily integrated with Kaldi