26 - Types of neural networks used in supervised learning

Supervised Learning

Input(x)	Output (y)	Application	
Home features	Price	Real Estate	Std NN
Ad, user info	Click on ad? (0/1)	Online Advertising	
Image	Object (1,...,1000)	Photo tagging	CNN
Audio	Text transcript	Speech recognition	RNN
English	Chinese	Machine translation	
Image, Radar info	Position of other cars	Autonomous driving	Custom/Hybrid

31 - Backpropagation working

How Backpropagation Algorithms Work

Note the output of the model when the "W" value is 3:

Input	Desired Output	Model Output (W=3)
0	0	0
1	2	3
2	4	6

Notice the difference between the actual output and the desired output:

Input	Desired Output	Model Output (W=3)	Absolute Error	Square Error
0	0	0	0	0
1	2	3	1	1
2	4	6	2	4

31 - Backpropagation working 2

If the "W" value is changed to 4, notice the errors:

Input	Desired Output	Model Output (W=3)	Absolute Error	Square Error	Model Output (W=4)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	4	4
2	4	6	2	4	8	16

Backpropagation trains a neural network by assigning random weights to the algorithms and analyzing where the error in the system increases. When errors occur, the difference between the model output and the actual output is calculated. Once calculated, a different weight is assigned and the system is run again, to see if the error is minimized. If the error is not minimized, then an update of parameters is required

To update parameters, weights and biases are adjusted. Biases are located after weights and are in a different layer of a network, always being assigned the value of 1. After the parameters are updated, the process is run again. Once the error is at a minimum, the model is ready to start predicting.

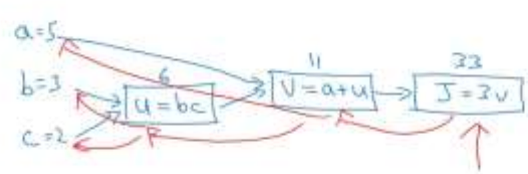
In looking at the diagram below, if "W" also known as weight is changed, then the error of the system goes up, if "W" is changed into a smaller number the error goes down. Once the error is as close to zero as possible that weight is set as the parameter and the model can start predicting.

31 - computational graph

Computation Graph

$J(a,b,c) = 3(a+\underbrace{bc}_{u}) = 3(5+3*2) = 33$

$u = bc$
$v = a+u$
$J = 3v$



34 - vectorization

$z = \underbrace{w^T x}_{} + b$

$w = \begin{bmatrix} \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \end{bmatrix}$ $w \in \mathbb{R}^{n_x}$ $x \in \mathbb{R}^{n_x}$

Non-vectorized:

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

Vectorized

$z = \underbrace{np.dot(w, x)}_{w^T x} + b$

→ GPU } SIMD - single instruction multiple data
→ CPU

34 - vectorization 2

```
a = np.random.rand(1000000)
b = np.random.rand(1000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()

print(c)
print("Vectorized version:" + str(1000*(toc-tic)) +"ms")

c = 0
tic = time.time()
for i in range(1000000):
    c += a[i]*b[i]
toc = time.time()

print(c)
print("For loop:" + str(1000*(toc-tic)) + "ms")
```

```
250286.989866
Vectorized version:1.5027523040771484ms
250286.989866
For loop:474.29513931274414ms
```

34 - vectorization 3

Vectorizing Logistic Regression

$z^{(1)} = w^T x^{(1)} + b$ $\quad z^{(2)} = w^T x^{(2)} + b \quad z^{(3)} = w^T x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)}) \quad a^{(2)} = \sigma(z^{(2)}) \quad a^{(3)} = \sigma(z^{(3)})$

$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$ (n_x, m) $\mathbb{R}^{n_x \times m}$

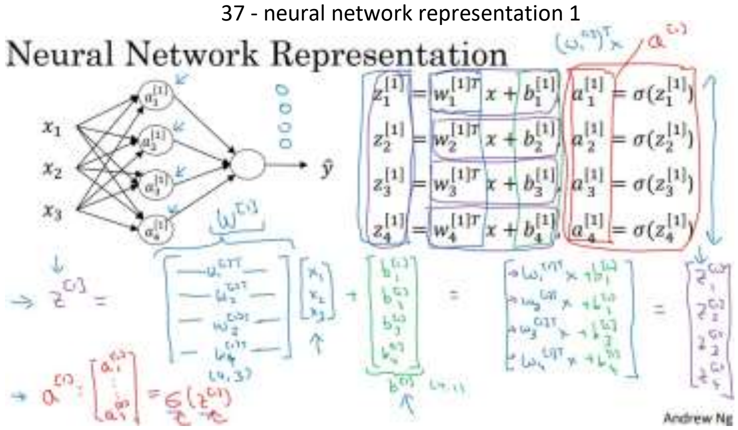
$Z = [z^{(1)}\ z^{(2)} \cdots z^{(m)}] = w^T X + \underbrace{[b\ b \cdots b]}_{1\times m} = [w^T x^{(1)} + b \quad w^T x^{(2)} + b \cdots w^T x^{(m)} + b]$

$Z = np.dot(w.T, X) + b$ $\quad (1,1)\ \mathbb{R}$ "Broadcasting"

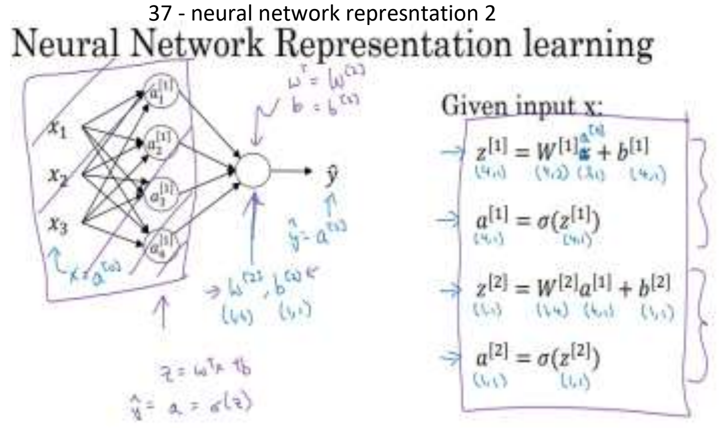
$A = [a^{(1)}\ a^{(2)} \cdots a^{(m)}] = \sigma(Z)$

Andrew Ng

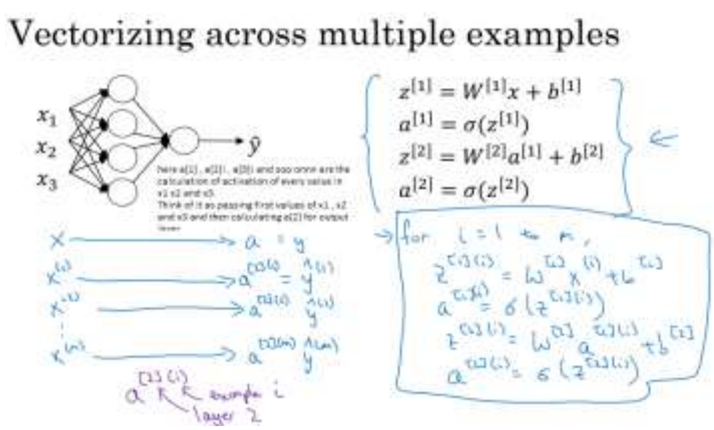
37 - neural network representation 1



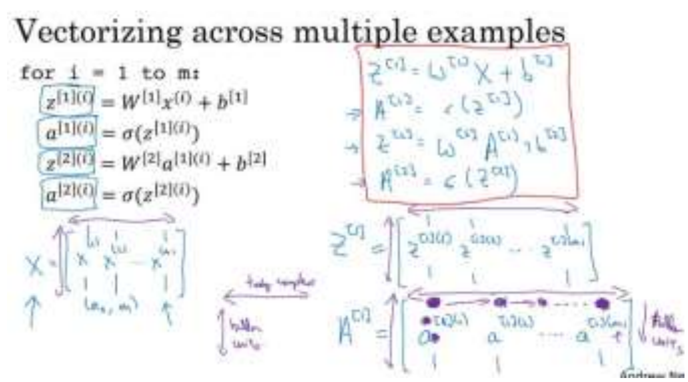
37 - neural network represntation 2



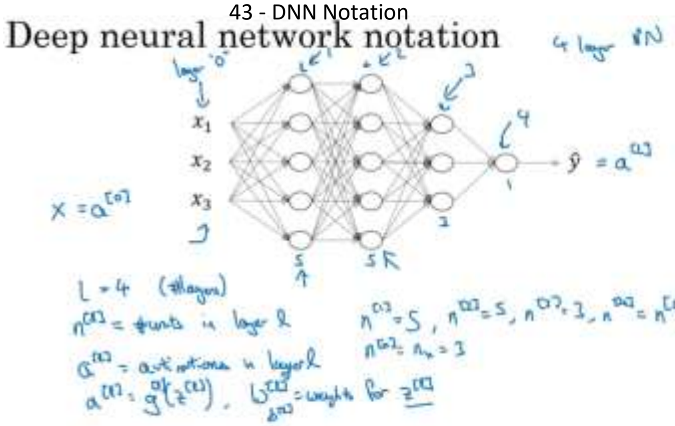
37 - vectorization of across multiple layers



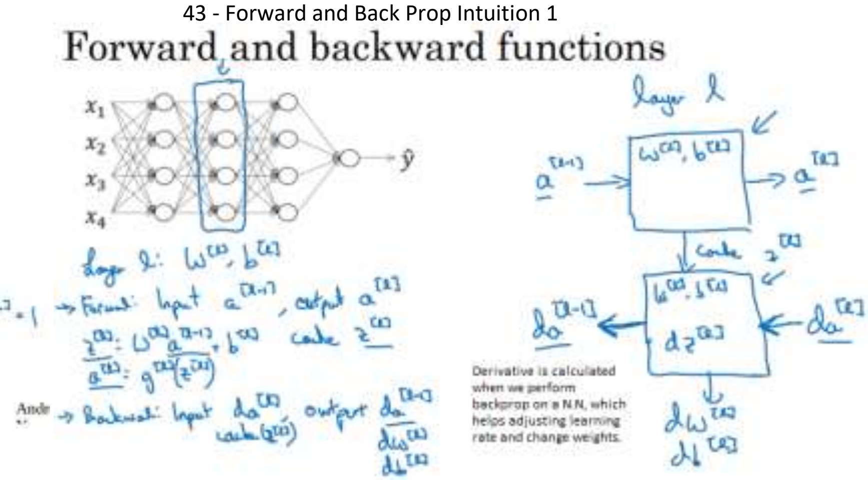
37 - vectorization of multiple training ex 2



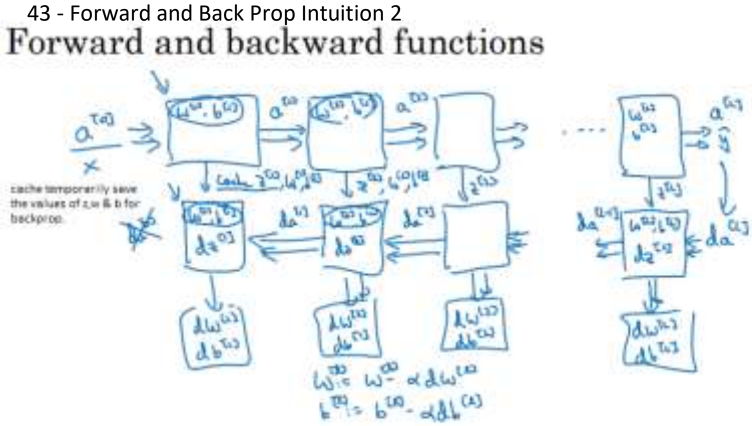
43 - DNN Notation



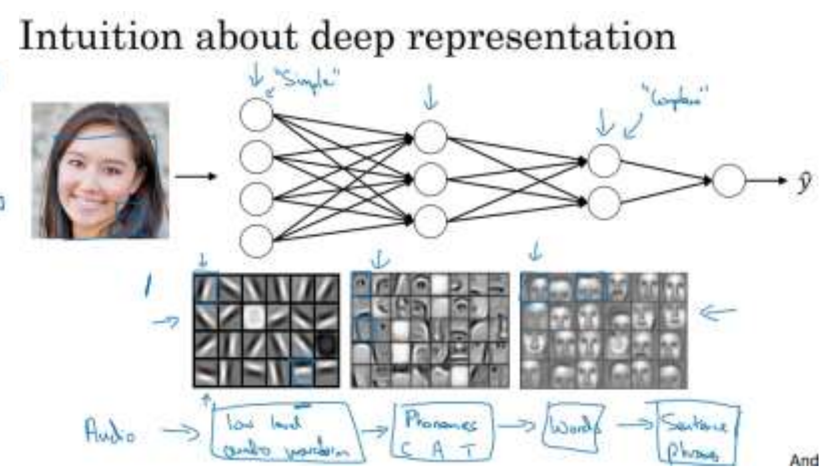
43 - Forward and Back Prop Intuition 1



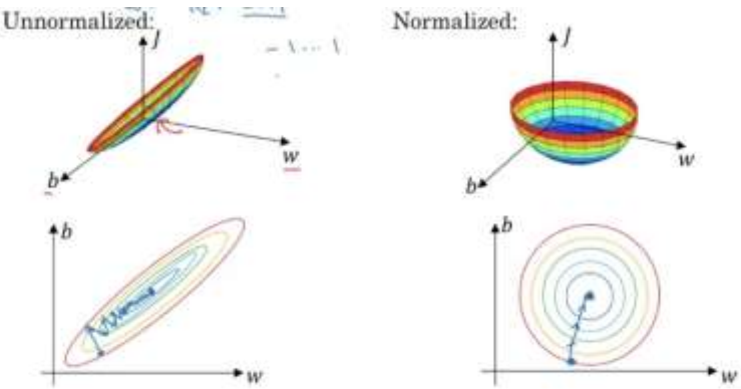
43 - Forward and Back Prop Intuition 2



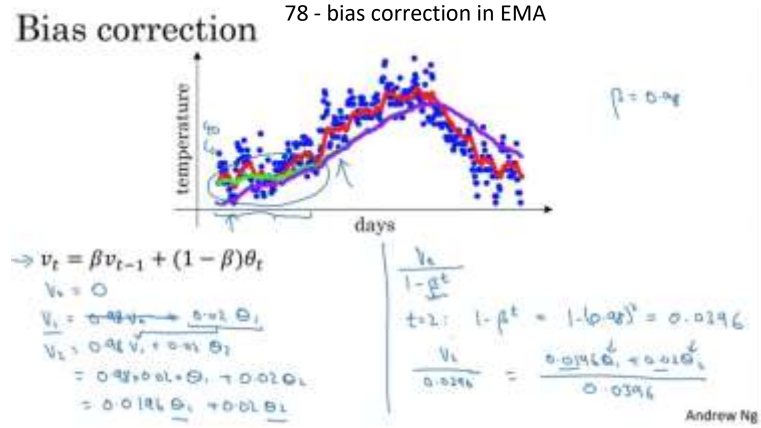
43 - Intuition Deep learning



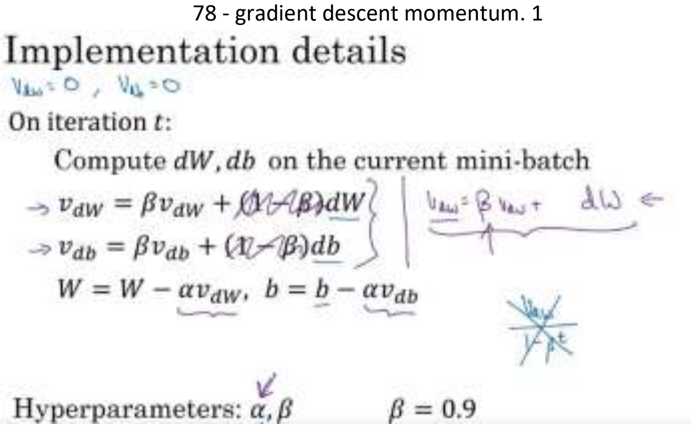
71 normalized inputs



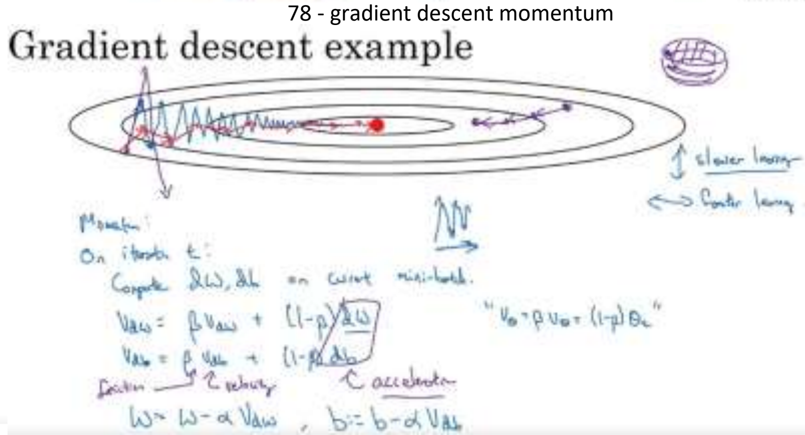
78 - bias correction in EMA



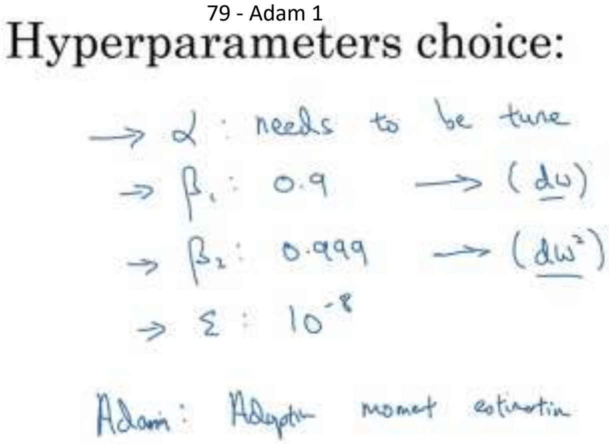
78 - gradient descent momentum. 1



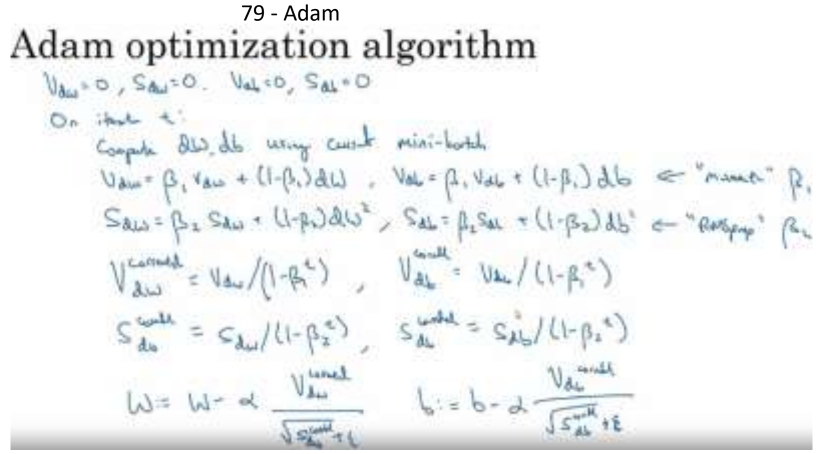
78 - gradient descent momentum



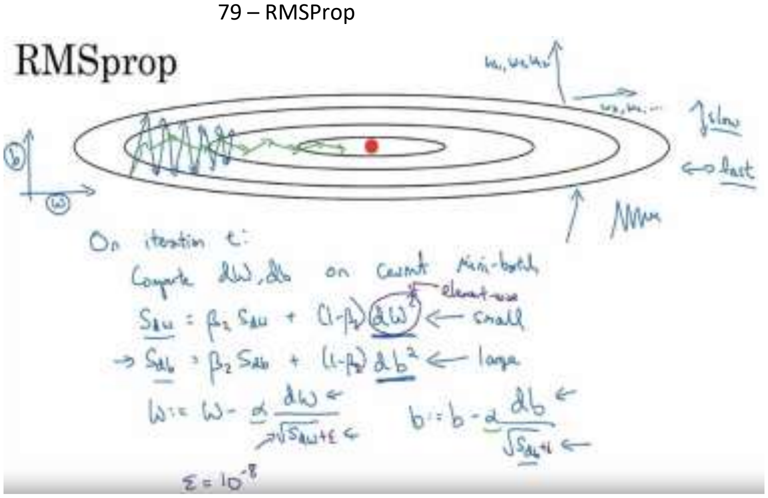
79 - Adam 1



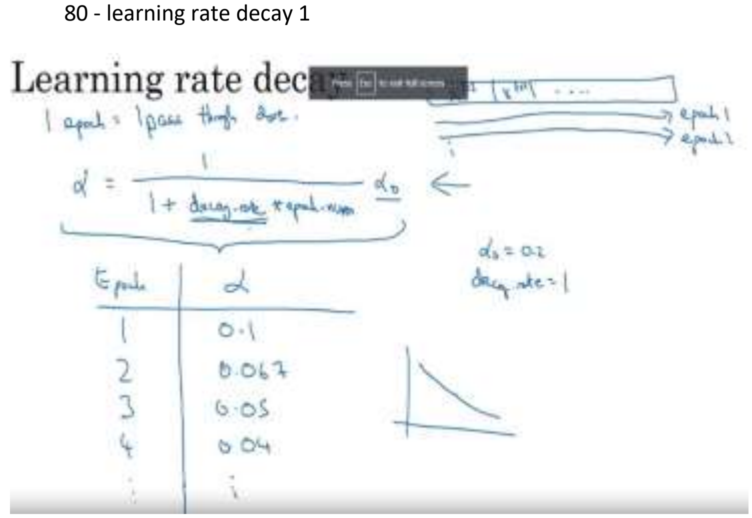
79 - Adam



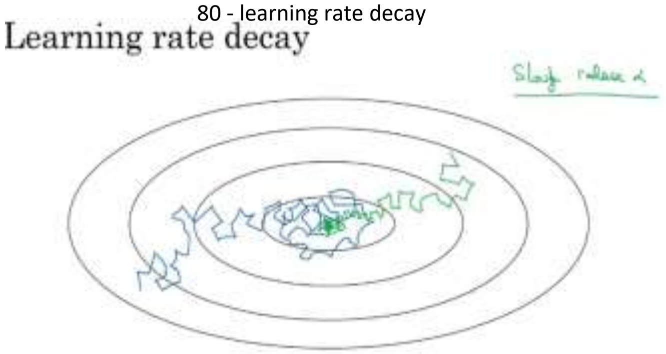
79 – RMSProp



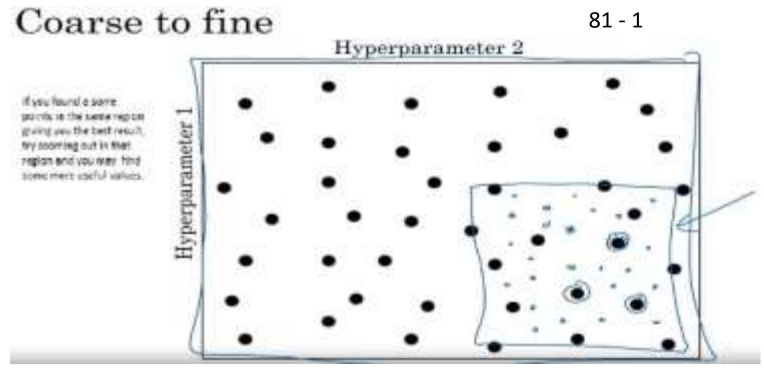
80 - learning rate decay 1



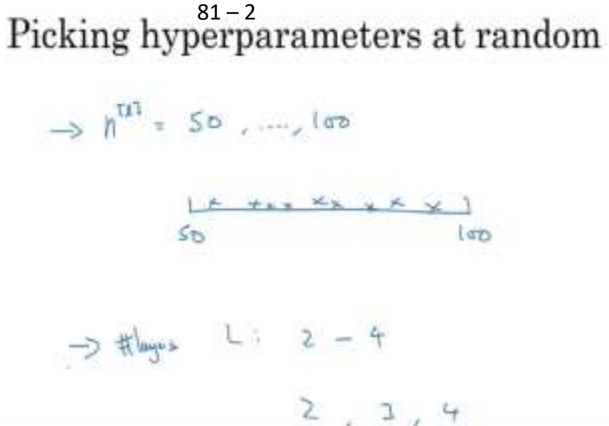
80 - learning rate decay



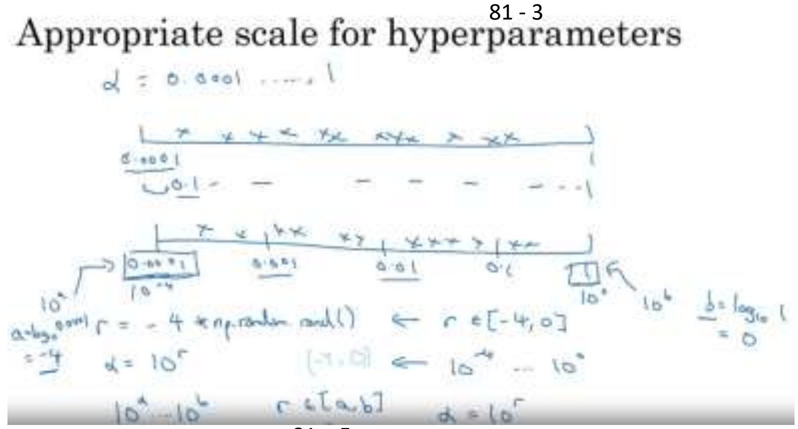
81 - 1



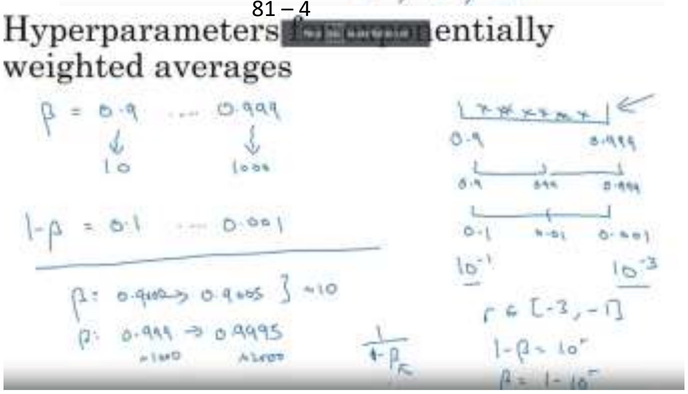
81 – 2



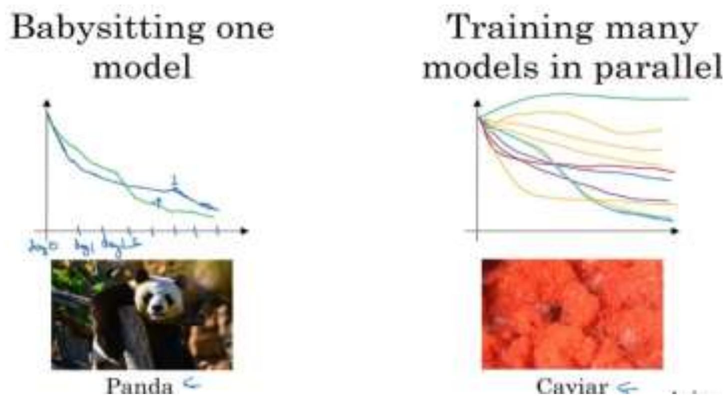
81 - 3



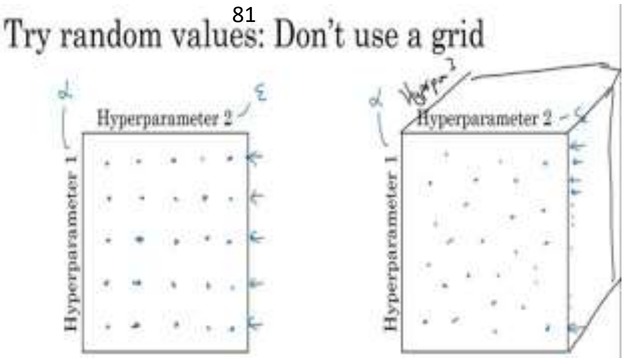
81 – 4



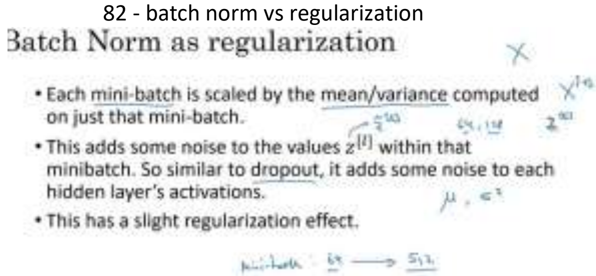
81 – 5



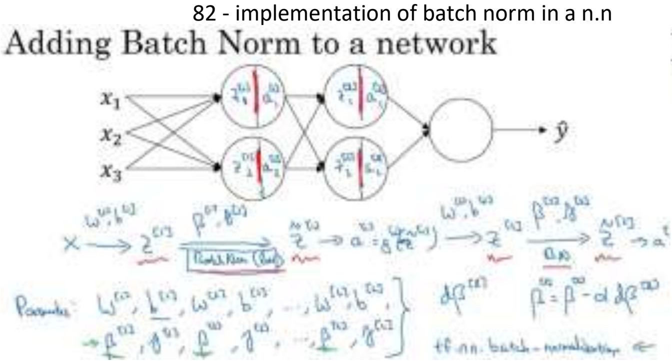
81



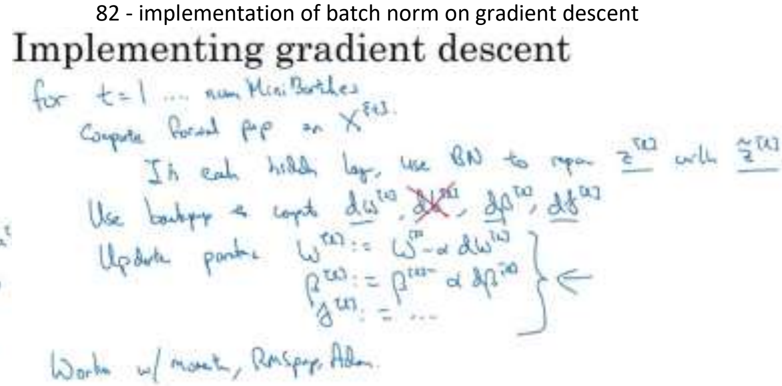
82 - batch norm vs regularization



82 - implementation of batch norm in a n.n



82 - implementation of batch norm on gradient descent



82 - implementation of batch norm

Implementing Batch Norm

Given some intermediate values in NN $z^{(1)}, \dots, z^{(m)}$ ($z^{[l](i)}$)

$\mu = \frac{1}{m}\sum_i z^{(i)}$

$\sigma^2 = \frac{1}{m}\sum_i (z_i-\mu)^2$

$z^{(i)}_{norm} = \frac{z^{(i)}-\mu}{\sqrt{\sigma^2+\epsilon}}$

$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$ learnable parameters of model.

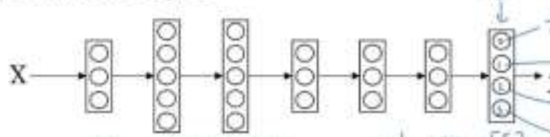
If $\gamma = \sqrt{\sigma^2+\epsilon}$, $\beta = \mu$, then $\tilde{z}^{(i)} = z^{(i)}$

x, $z^{(i)}$

Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$

83 - Softmax

Softmax layer



$\frac{e^5}{176.3} = 0.842$, $\frac{e^2}{176.3} = 0.042$, $\frac{e^{-1}}{176.3} = 0.002$, $\frac{e^3}{176.3} = 0.114$

$z^{[L]} = W^{[L]}a^{[L-1]} + b^{[L]}$ (4,1)

Activation function:

$t = e^{(z^{[L]})}$ (4,1)

$a^{[L]} = \frac{e^{z^{[L]}}}{\sum_{j=1}^{4} t_i}$, $a^{[L]}_i = \frac{t_i}{\sum_{j=1}^{4} t_i}$ (4,1)

$a^{[L]} = g^{[L]}(z^{[L]})$ (4,1)

$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}$

$t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix}$, $\sum_{j=1}^{4} t_j = 176.3$

$a^{[L]} = \frac{t}{176.3}$

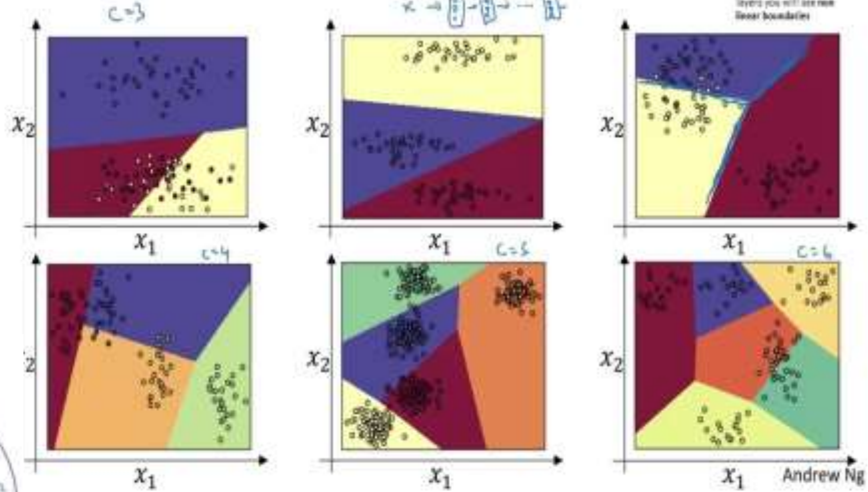
83 - Softmax 1

Softmax examples

$z^{[L]} = W^{[L]}x + b^{[L]}$

$a^{[L]} = \hat{y} = g(z^{[L]})$

this is the result of softmax activations NN with no hidden layers, if you will use hidden layers you will see non linear boundaries



86 - cat classification exmaple

Another cat classification example

optimizing — satisficing

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

cost = accuracy − 0.5 × running time

maximize accuracy

subject to running time ≤ 100 ms

N metrics: 1 optimizing, N−1 satisficing

Wakewords / Trigger words

Alexa, OK Google, Hey Siri, nihaobaidu 你好百度

accuracy. #false positive

maximize accuracy

s.t. ≤ 1 false positive every 24 hours

88 - dev and test for inappropriate images

Another example

Algorithm A: 3% error

Algorithm B: 5% error

→ Dev/test → User images

If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

88 - human level performance

Why compare to human-level performance

Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:

- Get labeled data from humans. (x, y)
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

89 - assumptions related to Supervised Learning

The two fundamental assumptions of supervised learning

1. You can fit the training set pretty well. ≈ Avoidable bias
2. The training set performance generalizes pretty well to the dev/test set. ≈ Variance

89 - bias-variance w human level error 1

Summary of bias/variance with human-level performance

0%

Human-level error (proxy for Bayes error)

"Avoidable bias"

Training error

"Variance"

Dev error

Variance here defines how much of information is model giving off.

89 - bias-variance w human level error 2

Error analysis example

Human (proxy for Bayes error)

Avoidable bias

Training error

Variance

Dev error

Human (proxy for Bayes error)	1%, 0.7%, 0.5%	1%, 0.7, 0.5%	0.4%, 0.5%, 1%
Avoidable bias	4.0%, 4.5%	0.3, 0.5%	0.1%, 0.0%, 0.7%
Training error	5%	1%	0.7%
Variance	1%	4%	0.1%
Dev error	6%	5%	0.8%
	Bias	Variance	

we must set human error in order to get better performance of a model. if we set the human error here to 0.7 then avoidable bias would be 0% means human can't make error so can't your model. Than it would create a problem.

89 - reducing avoidable bias and variance

Reducing (avoidable) bias and variance

Human-level

Avoidable bias →
- Train bigger model
- Train longer/better optimization algorithms
  - momentum, RMSprop, Adam
- NN architecture/hyperparameters search (RNN, CNN)

Training error

Variance →
- More data
- Regularization
  - L2, dropout, data augmentation
- NN architecture/hyperparameters search

Dev error

89 - surpassing human level performance

Surpassing human-level performance

Team of humans	0.5%	0.5%
One human	~~1%~~	1%
Training error	0.6%	0.3%
Dev error	0.8%	0.4%

here we will consider human error as 0.5% , so the avoidable bias is 0.1%(0.6% - 0.5%)

0.1

0.2

here the train error is 0.3% means we have overfitted the model. Here the model has surpassed the human level performance of 0.5% , so your ways of your model making progress are less clear.

0.1%, 0.2%, 0.3%

What is avoidable bias?

90 - error analysis

Look at dev examples to evaluate ideas

90% accuracy
→ 10% error

Should you try to make your cat classifier do better on dogs?

Error analysis: → 5-10 min
- Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

"ceiling"

→ 5%, 5/100, 10% → 9.5%

→ 50%, 50/100, 10% → 5%

90 - error analysis 1

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
⋮	⋮	⋮	⋮		
% of total	8%	43%	61%	12%	

Key to is make a table and put all the pic name in the column you and on the left put all the number of images you are counting manually.
We can see that we have to majorly work on **blurry & Great Cats**

91 - incorrectly labeled data

Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

We can see that we have total error of **10%** means model is **90%** accurate,we have calculated manually that incorrect labeled data is **6%**(hence **0.6** from **10%**).

Even if we fix the incorrectly labeled data we would not be doing much to increase the model's performance , as we have **Blurry & Great Cat** images resisting and causing heavy error in the predictions.

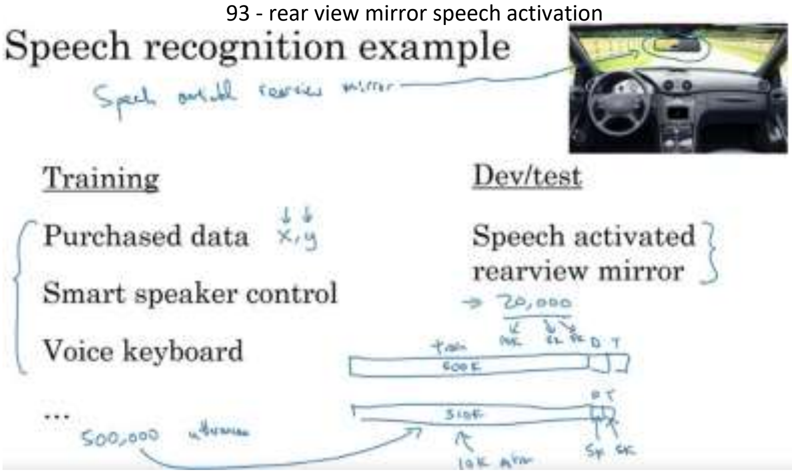
Overall dev set error	10%	2%
Errors due incorrect labels	0.6%	0.6%
Errors due to other causes	9.4%	1.4%

In this case the error is **2%** so solving incorrect labeling will definitly help improving model's performance

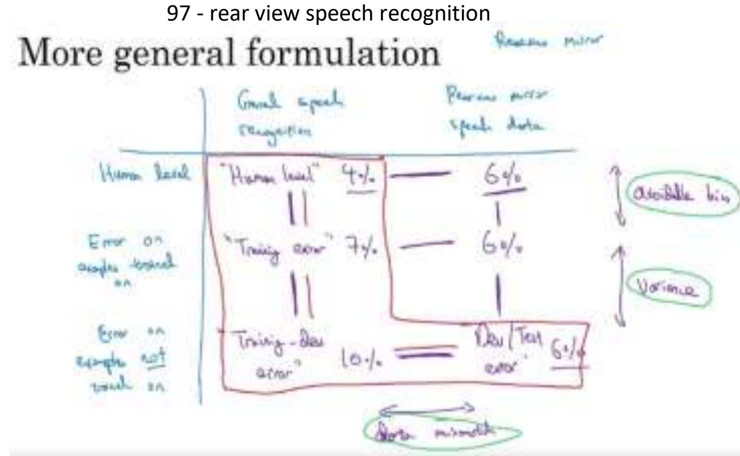
2.1% 1.9%

Goal of dev set is to help you select between two classifiers A & B.

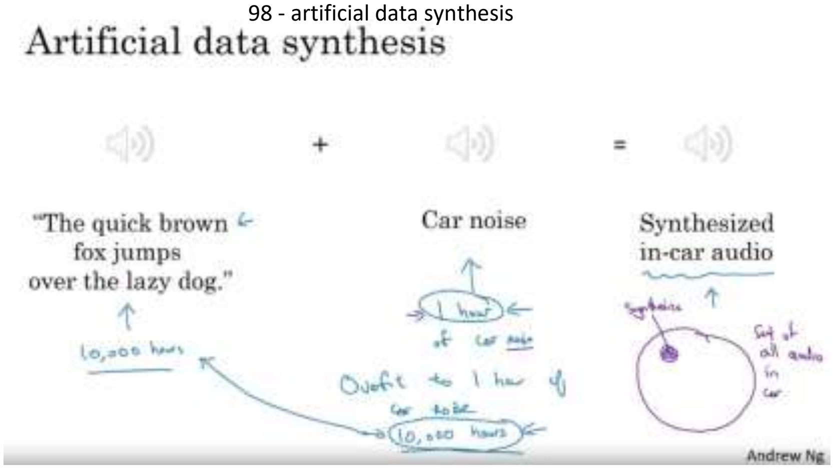
93 - rear view mirror speech activation



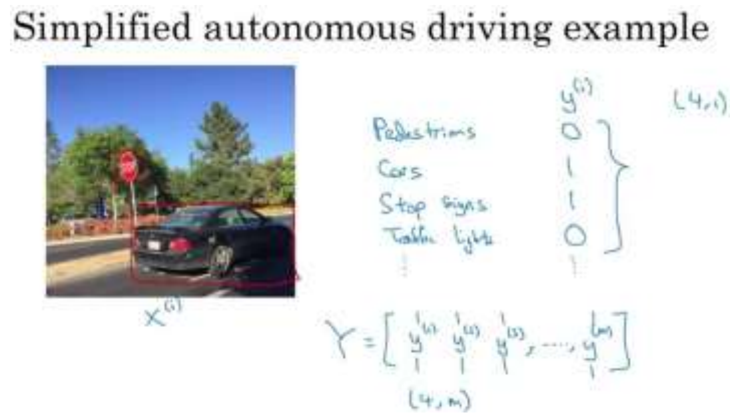
97 - rear view speech recognition



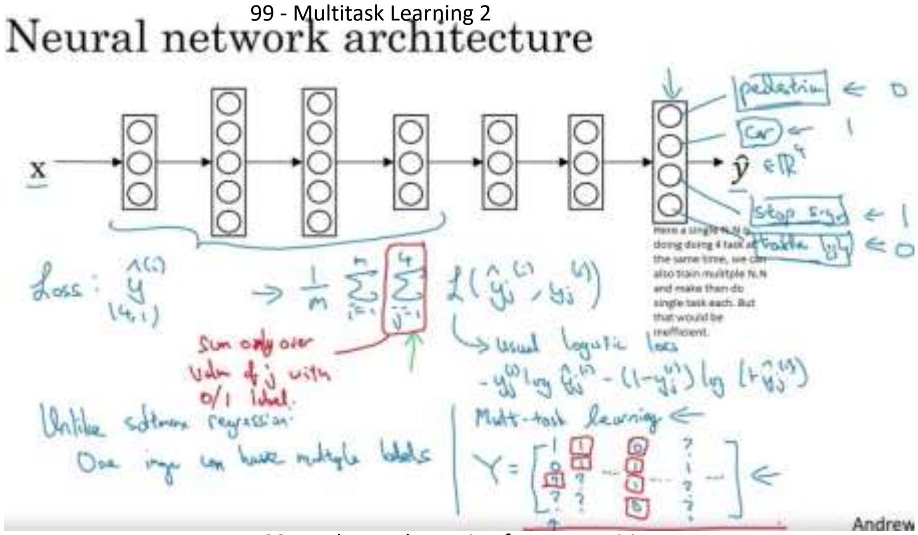
98 - artificial data synthesis



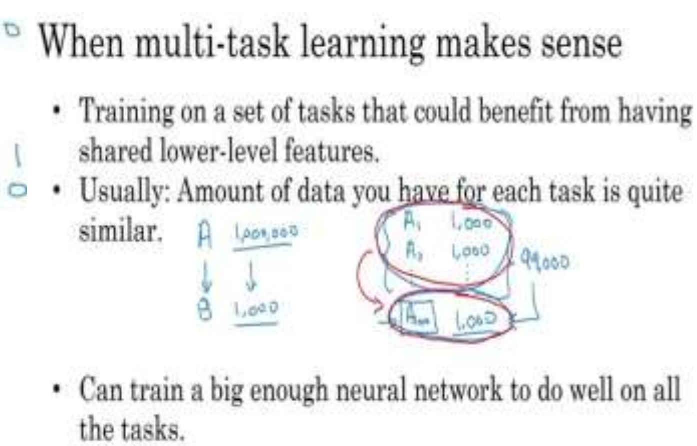
99 - Multitask Learning 1



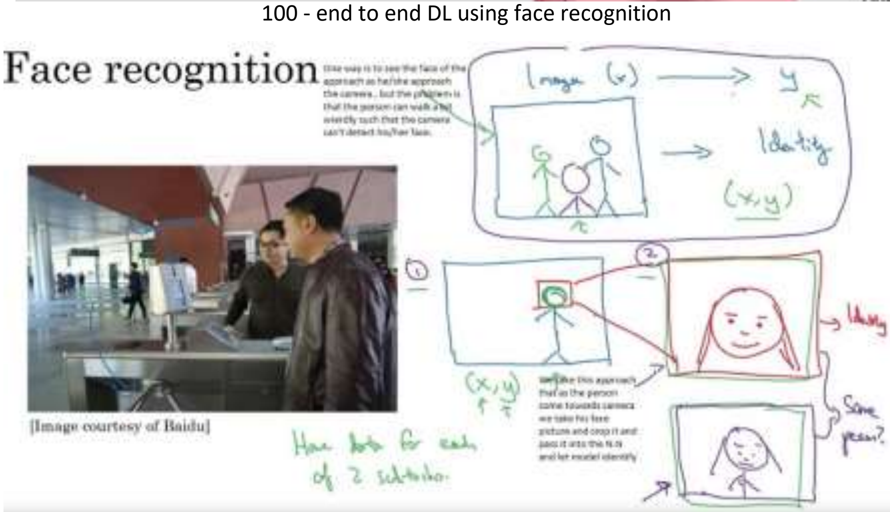
99 - Multitask Learning 2



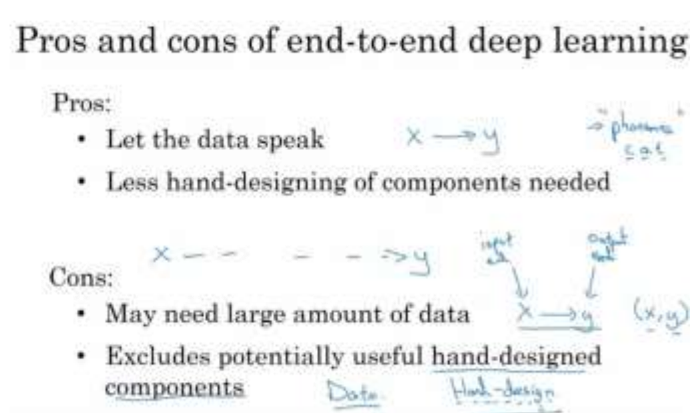
99 - Multitask Learning 3



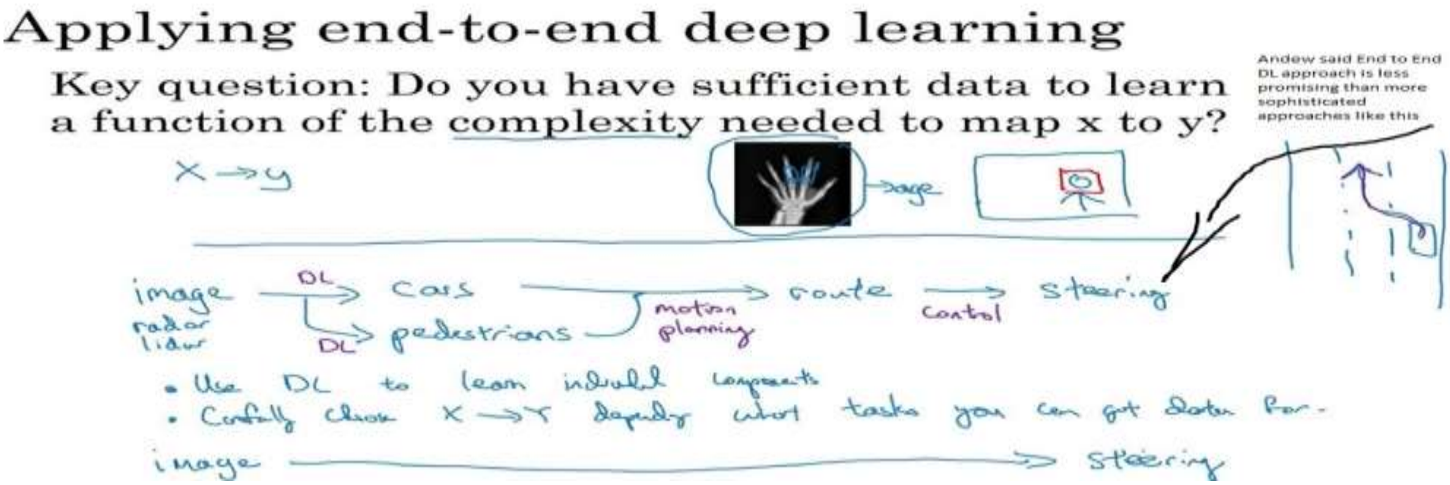
100 - end to end DL using face recognition



100.1 - end to end DL Pros n Cons



100.2 - end to end DL using hand recognition 2



103 - Edge detection In C.N.N

Vertical edge detection

$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

* "convolution"

this is matrix determined by the method you are using.

1	0	-1
1	0	-1
1	0	-1

3×3 filter

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

python: conv-forward
tensorflow: tf.nn.conv2d
keras: Conv2D

If we have no padding then stride of 1 is applied(means we are not skipping any pixel).
The Shape of the resultant image will be **(6-3+1) X (6-3+1)**

103 - Edge detection In C.N.N 1

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6×6

*

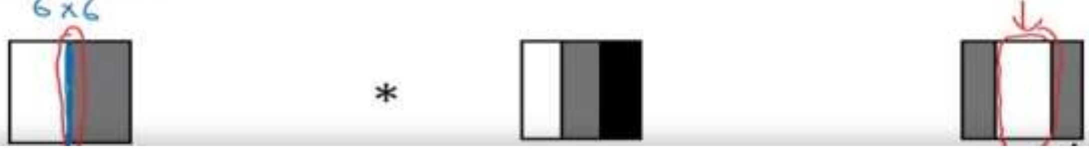
1	0	-1
1	0	-1
1	0	-1

3×3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4×4



103 - Edge detection In C.N.N 3

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

convolution

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$\underbrace{\quad}_{3\times 3}$

we can treat this **3x3 kernel as 9 parameter** and let N.N learn these using **backprop**, so that we can even better result than **Sobel** and **Scharr**.

$=$

45°
70°
73°

Andrew Ng

104 - padding

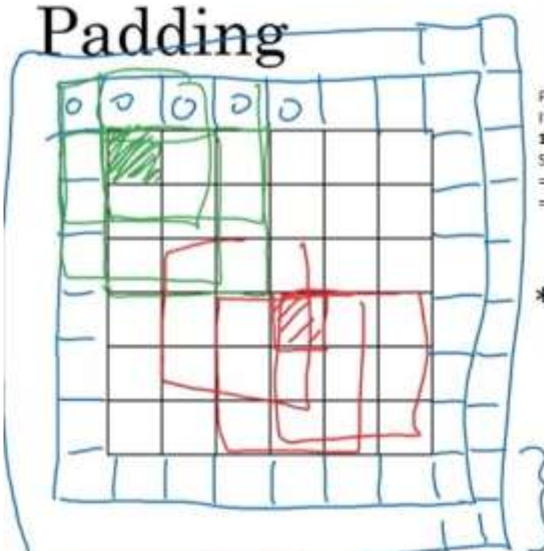
Padding

- shrinky output
- throw away info from edge

If we have a very large N.N the image size will reduce in every step.

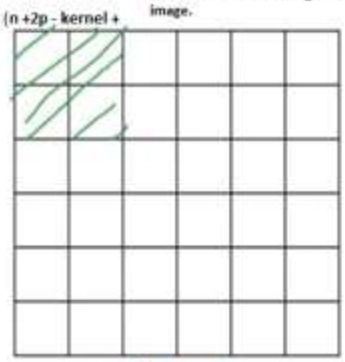
Padding is used bcz if we look at 4x4 box, there are alot of regions that are overlapping, meens pixels on the corners are use much less in the output, so we are throwing away alot of information near the edge of the image.

Padding is the process of adding extra pixel in the image.
If we have 6,6 image we can pad with 1 bit using, **(n +2p - kernel + 1)** x **(n +2p - kernel + 1)**.
So the resulting size will be ,
= (6+2(1) - 3 +1) x (6+2(1) - 3 +1)
= 6 x 6



p=

$*$ 3×3 $f\times f$ $=$



6×6

$\cancel{4\times 4}$

$\underset{n\times n}{6\times 6} \rightarrow 8\times 8$

$n-f+1 \times n-f+1$

$6-3+1=4$

$n+2p-f+1 \times n+2p-f+1$

$6+2-3+1 \times \_\_ = 6\times 6$

$p = \text{padding} = 1$

Andrew Ng

104 - padding 1

Valid and Same convolutions

→ no padding

"Valid": $n\times n \ * \ f\times f \longrightarrow n-f+1 \times n-f+1$

$6\times 6 \ * \ 3\times 3 \longrightarrow 4\times 4$

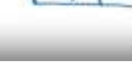
"Same": Pad so that output size is the same as the input size.

$n+2p-f+1 \times n+2p-f+1$

$\cancel{n}+2p-f+1 = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$

$3\times 3 \quad p = \frac{3-1}{2} = 1 \quad | \quad 5\times 5 \quad f=5 \quad p=2$

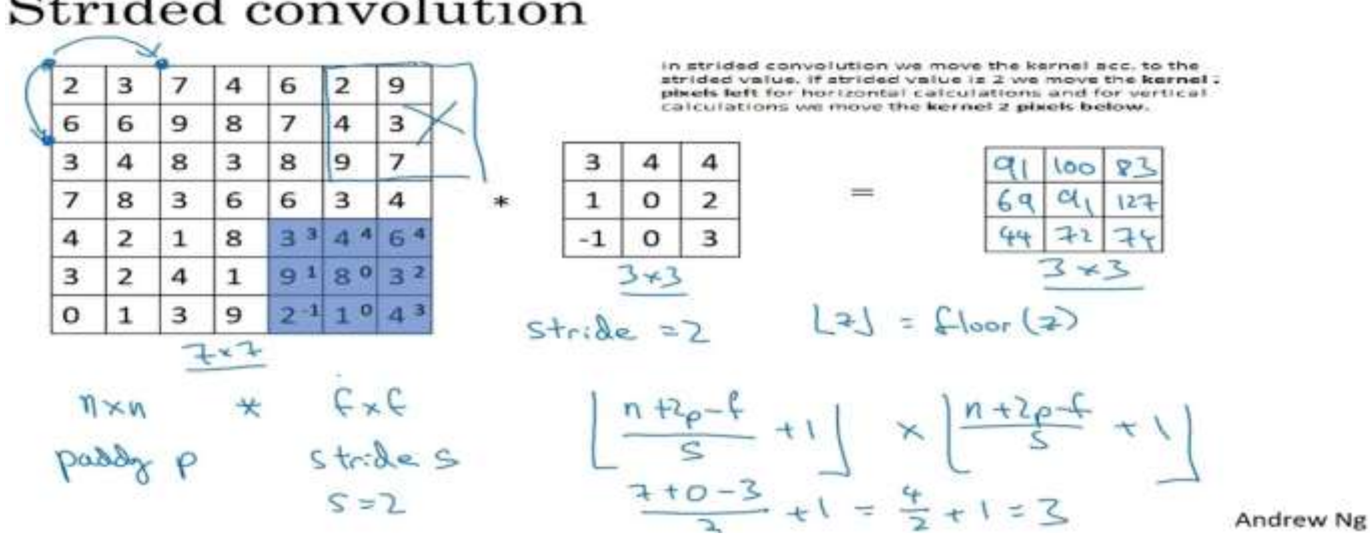
f is usually odd



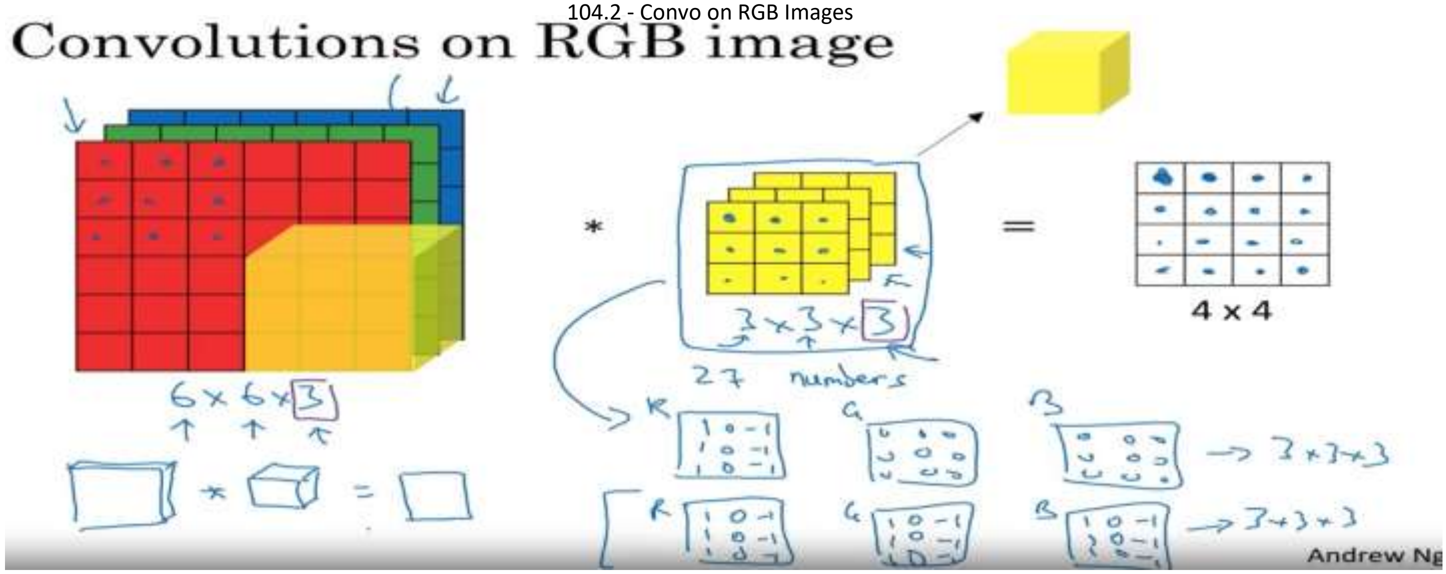
1x1
3x3
5x5
7x7

Andrew Ng

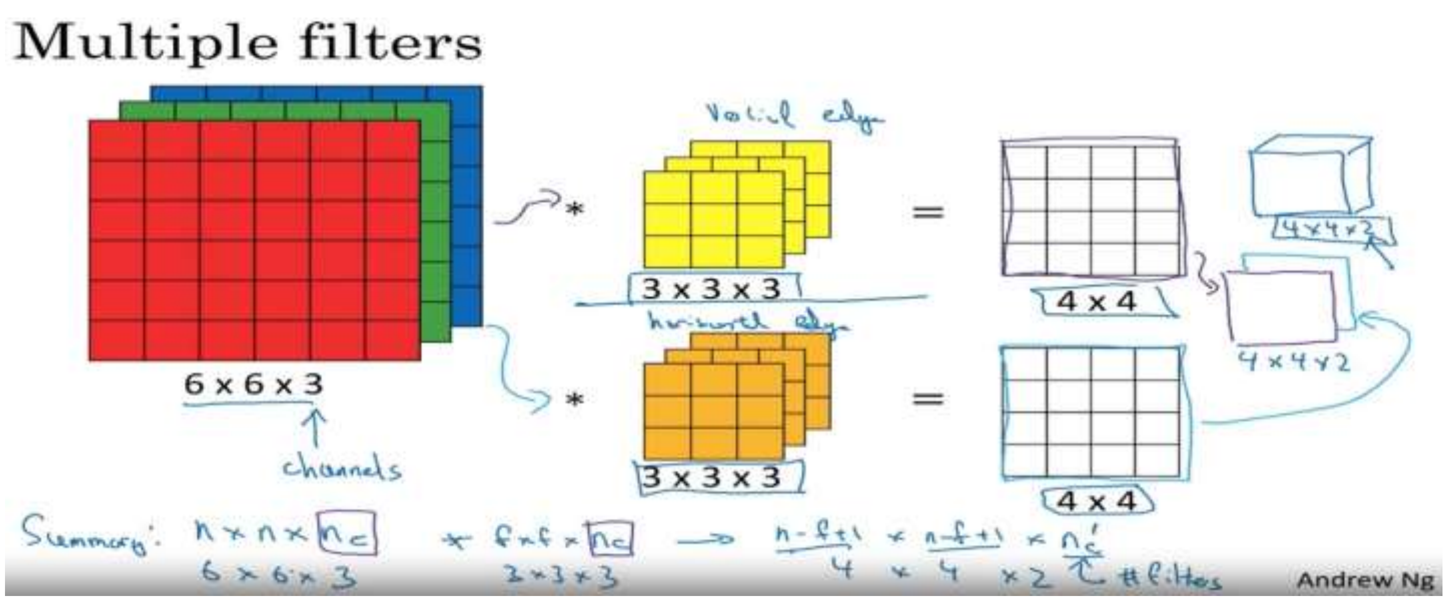
104 - strided convolutions



104.2 - Convo on RGB Images

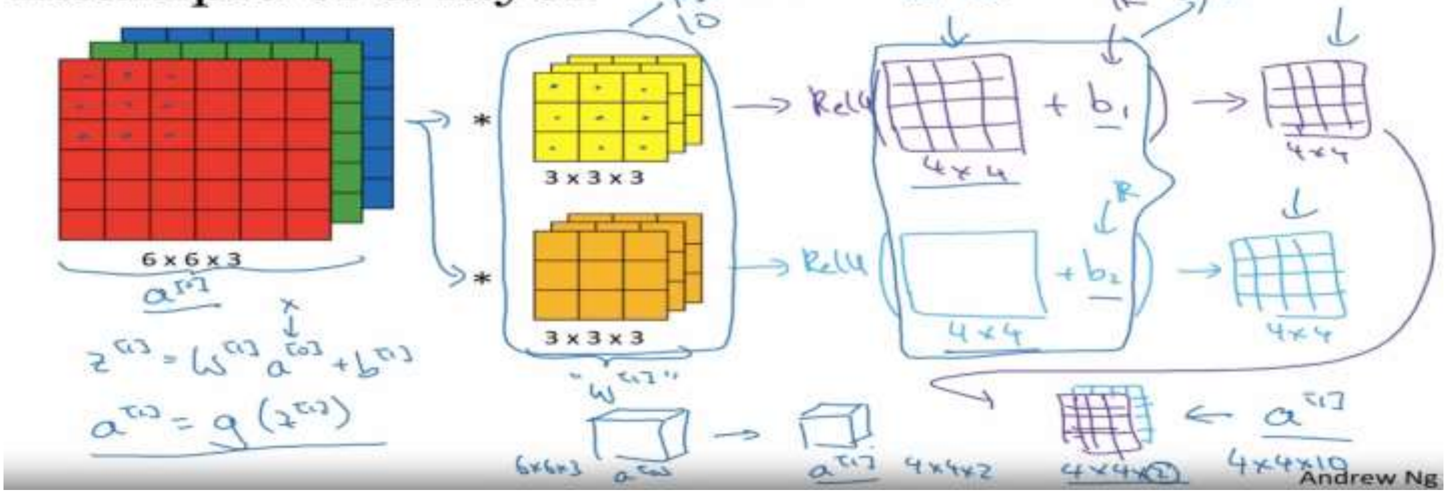


104.3 - Convo on RGB Images using multiple filters



104.4 - Single layer of a C.N.N 1

Example of a layer



104.4 - Single layer of a C.N.N 2 notations

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size
$p^{[l]}$ = padding
$s^{[l]}$ = stride
$\boxed{n_c^{[l]}}$ = number of filters
Each filter is: $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}}$
Activations: $a^{[l]} \to n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$
Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
bias: $n_c^{[l]}$ — $(1,1,1,n_c^{[l]})$ ← #filters in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times \boxed{n_c^{[l-1]}}$
Output: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$

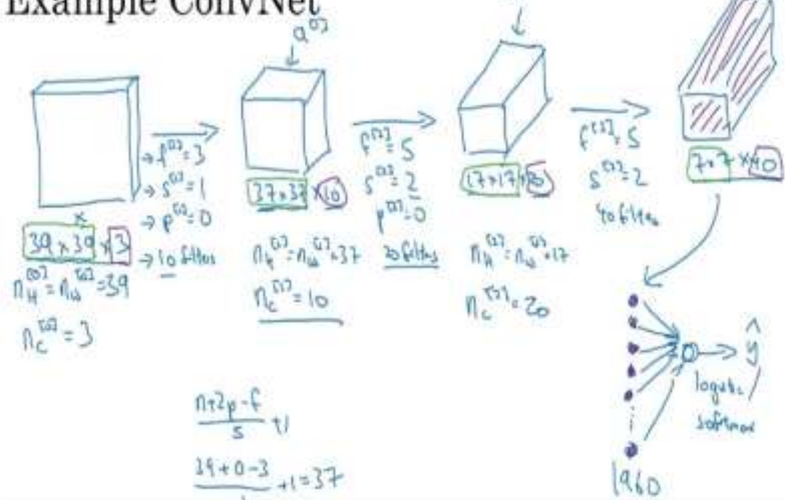
$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$A^{[l]} \to m \times \underbrace{n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}}$

$n_c \times n_H \times n_W$

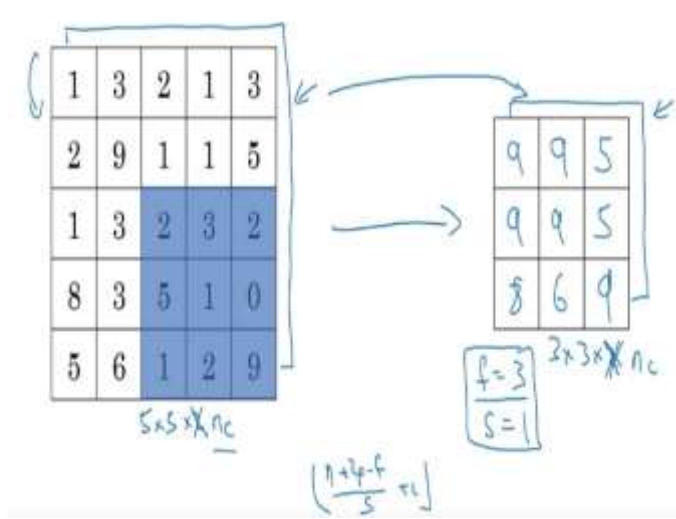
106.1 Simple C.N.N example

Example ConvNet



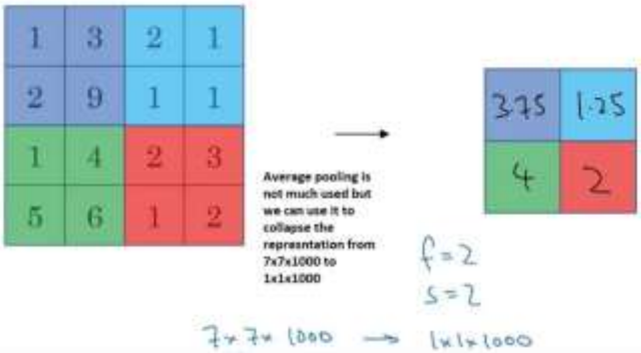
108 - Max pooling

Pooling layer: Max pooling



108.1 - average pooling 1

Pooling layer: Average pooling



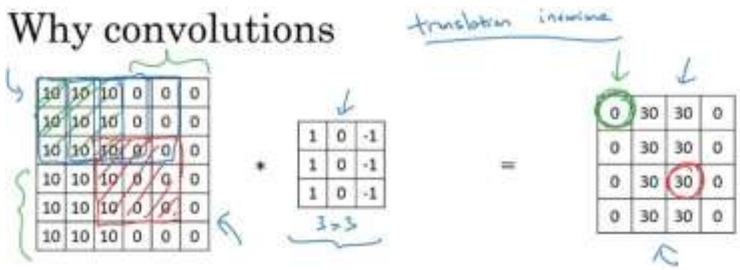
109 - N.N example

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	456
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	2,416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,120
FC4	(84,1)	84	10,164
Softmax	(10,1)	10	850

109.1

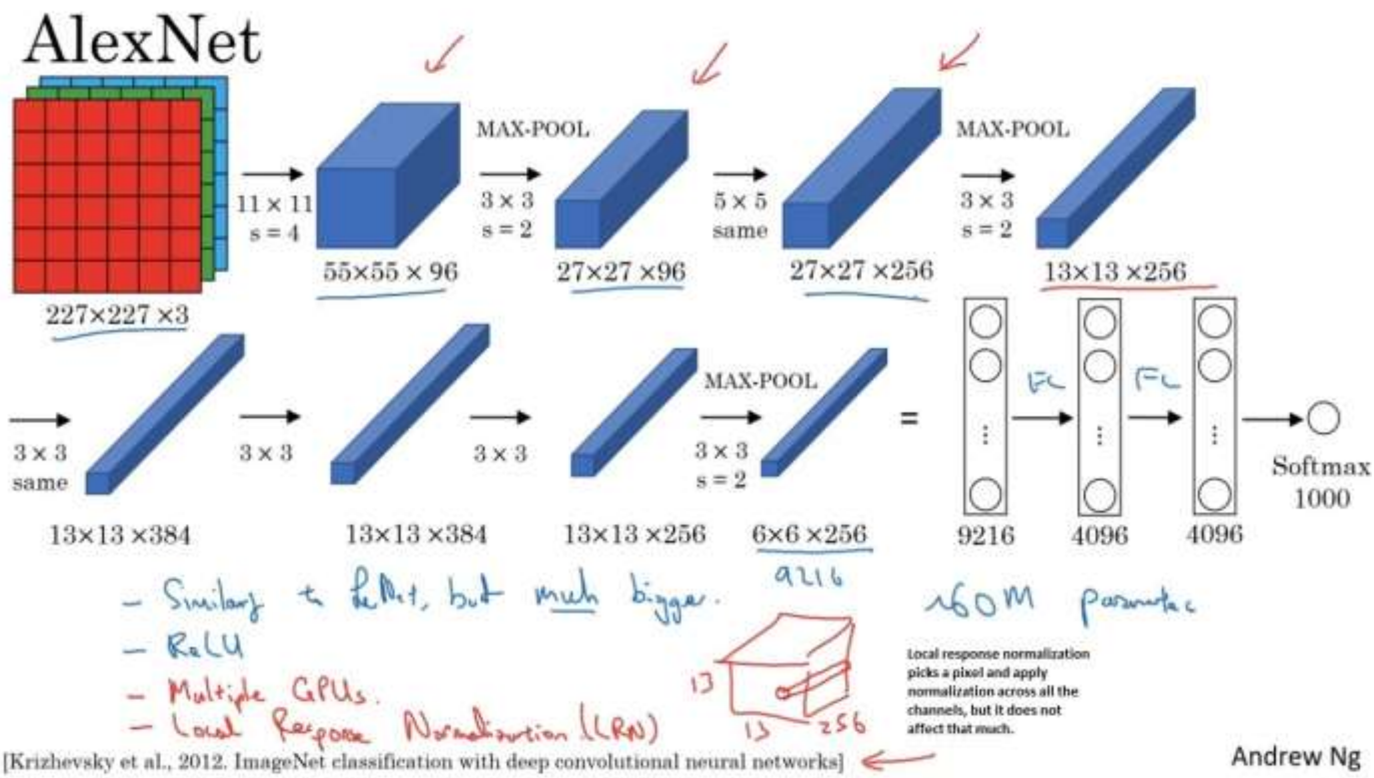
Why convolutions



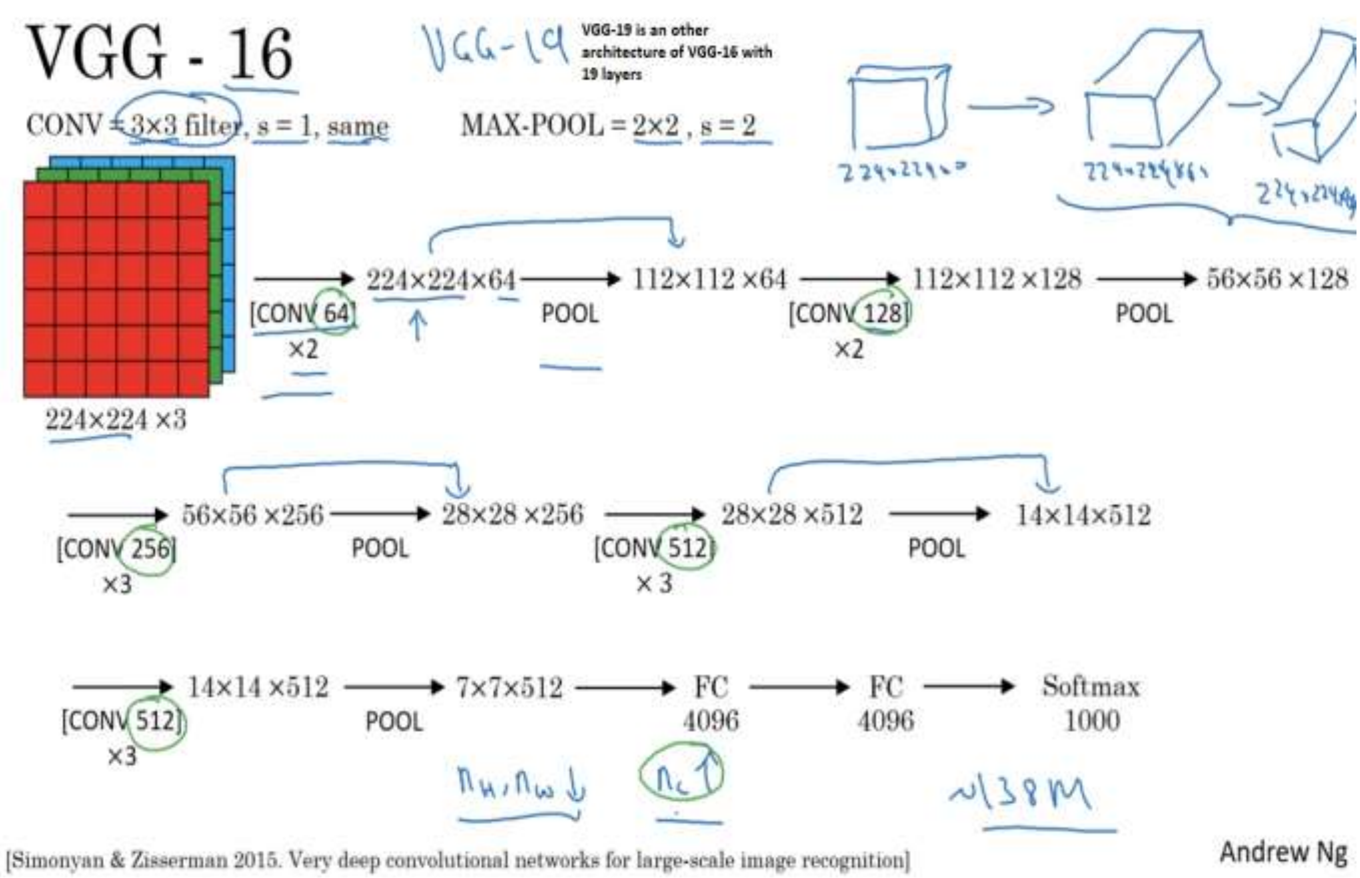
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

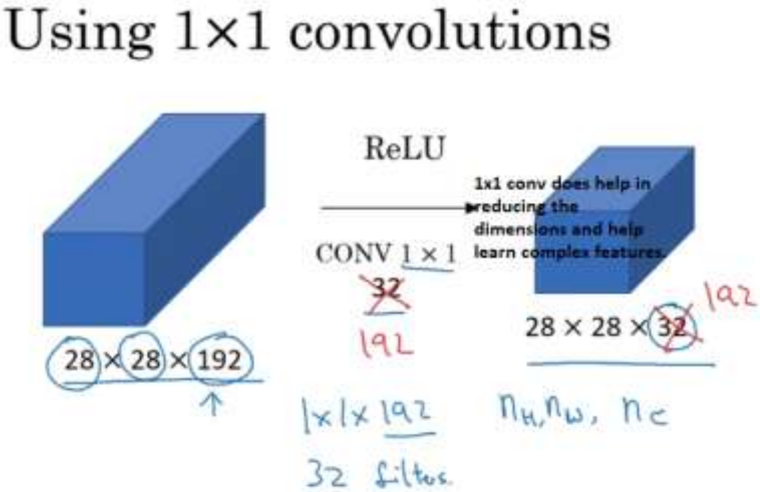
113 – alexNet



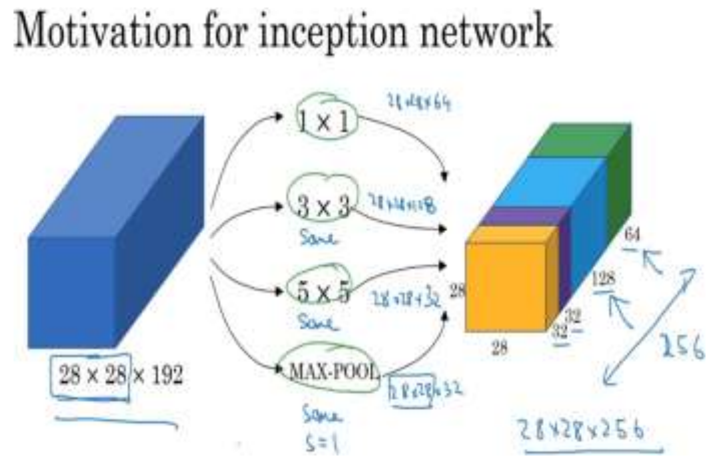
115 - VGG-16



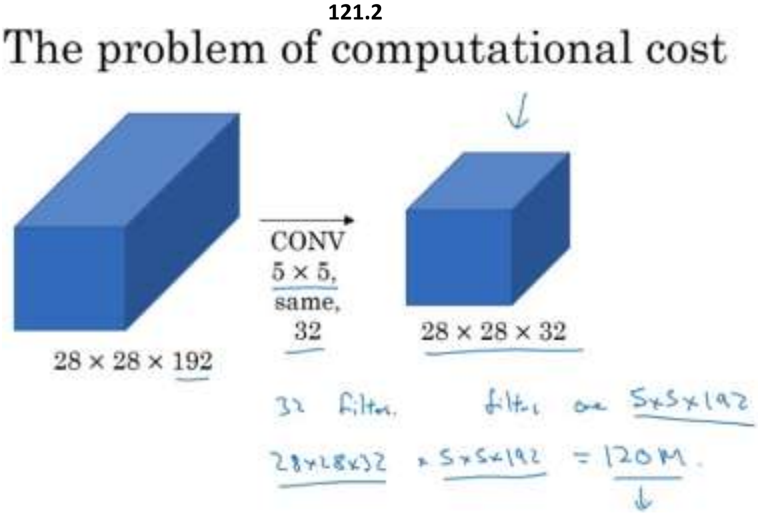
119



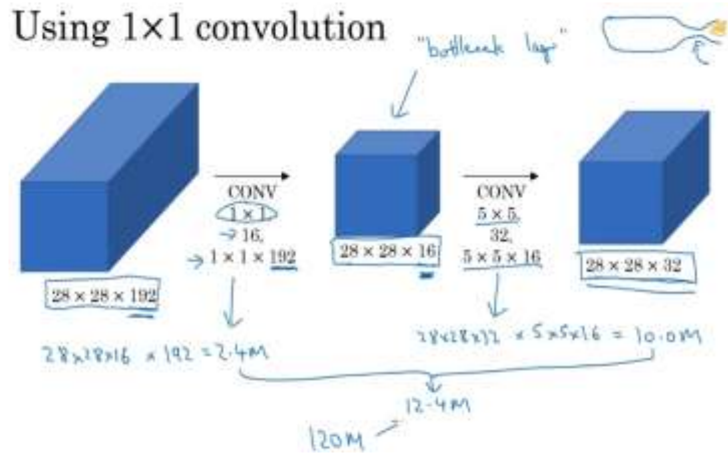
121.1



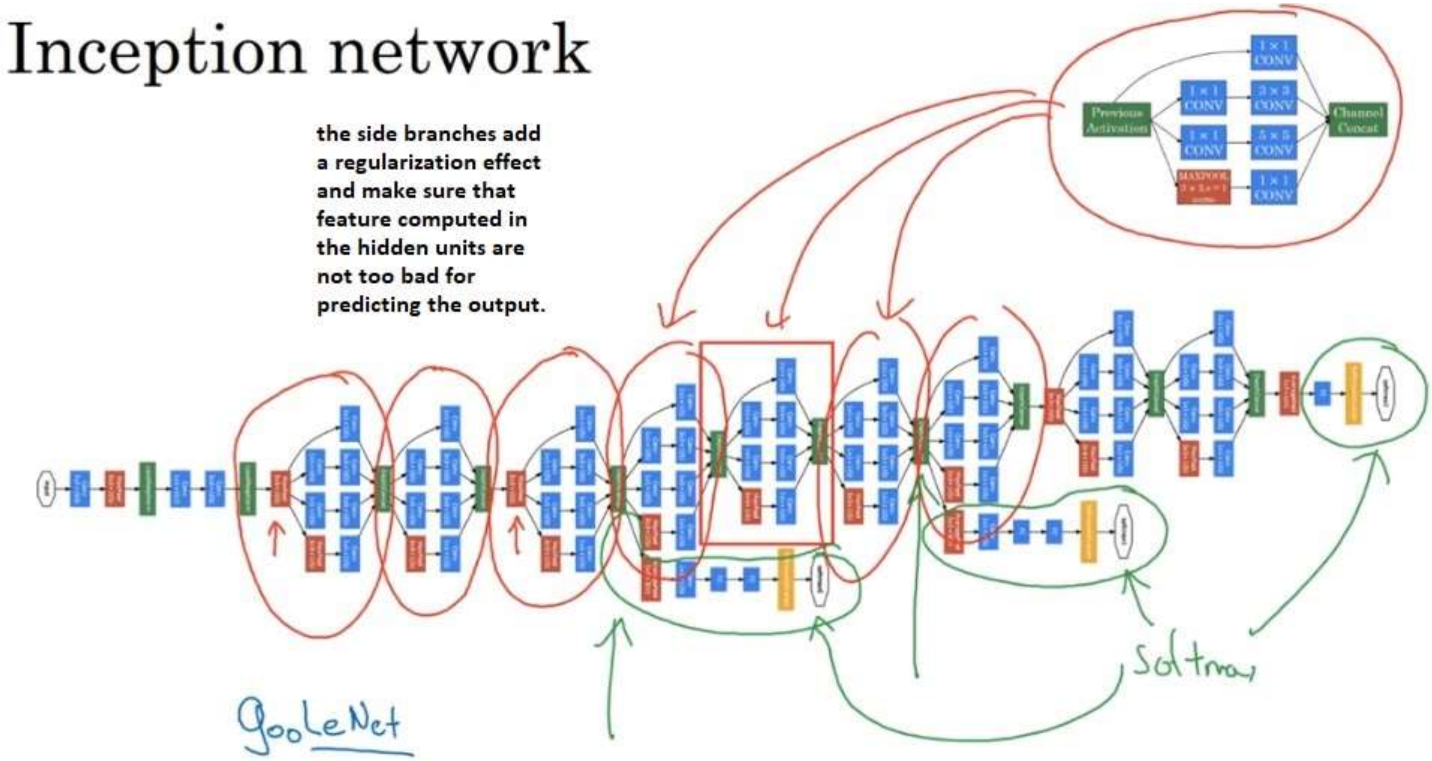
121.2



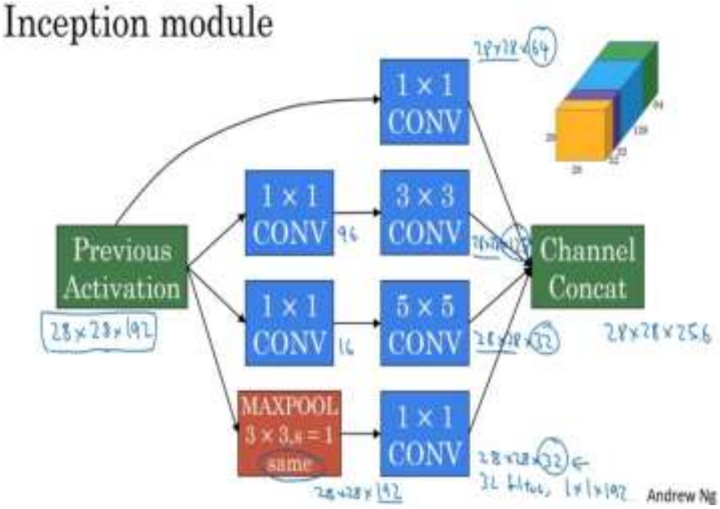
121.3



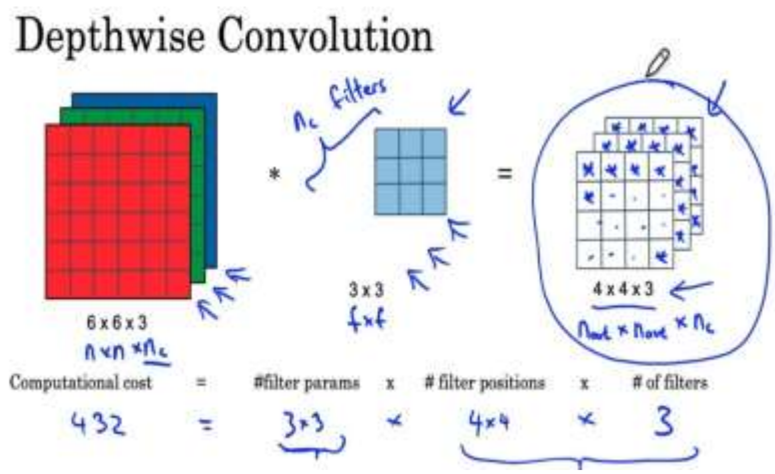
122.1



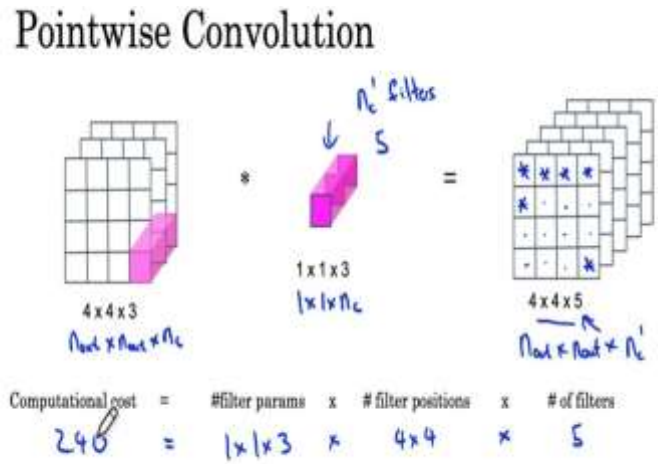
122



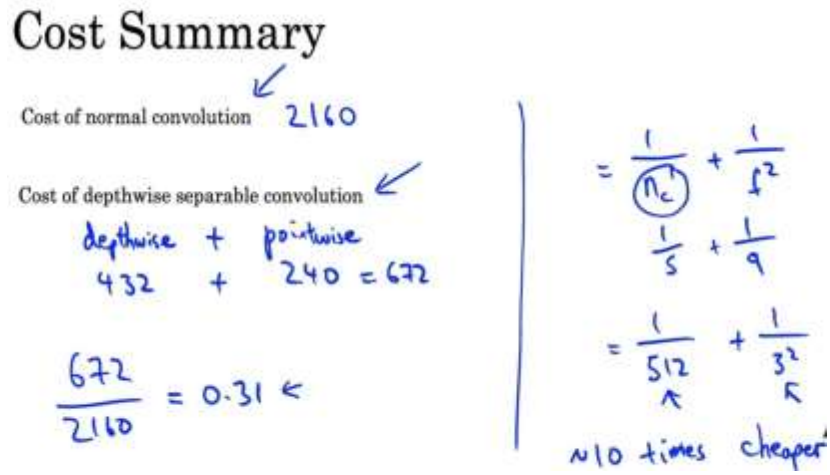
123.1



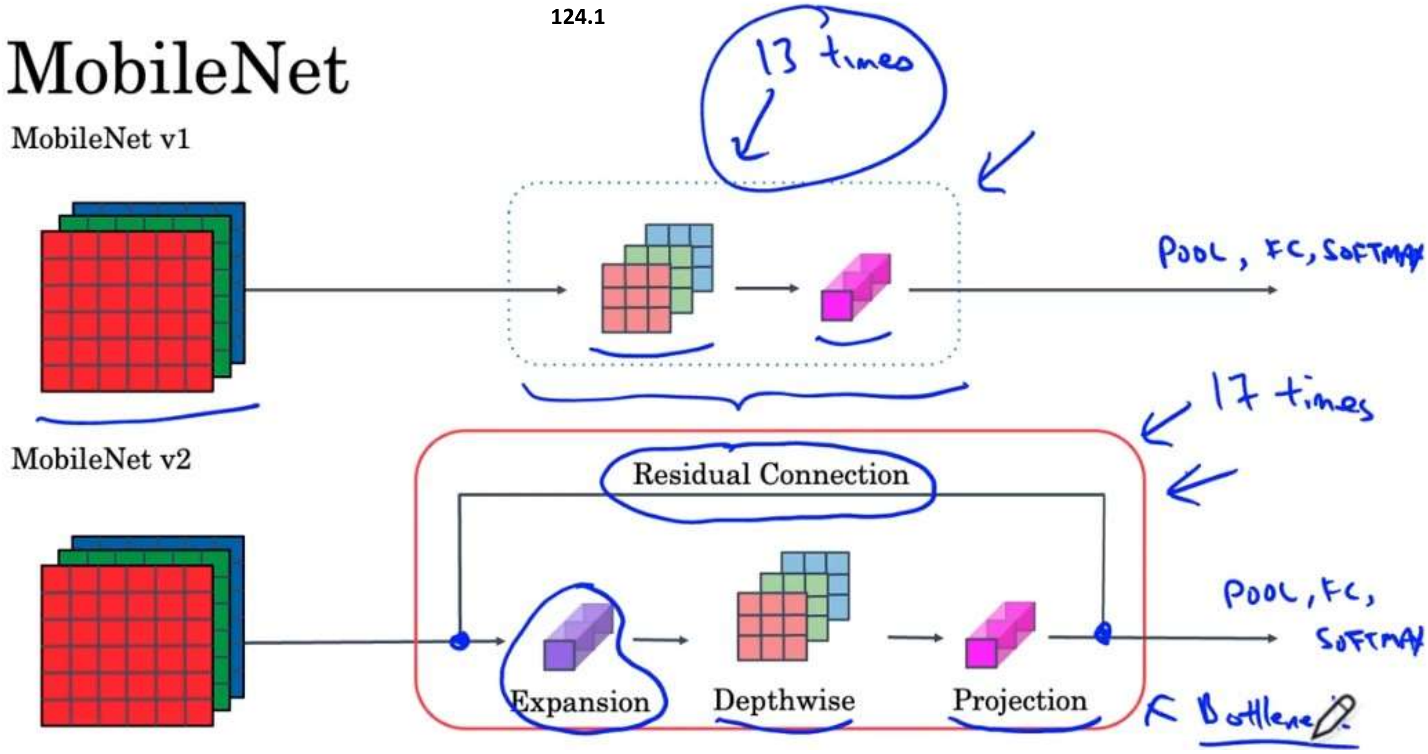
123.2



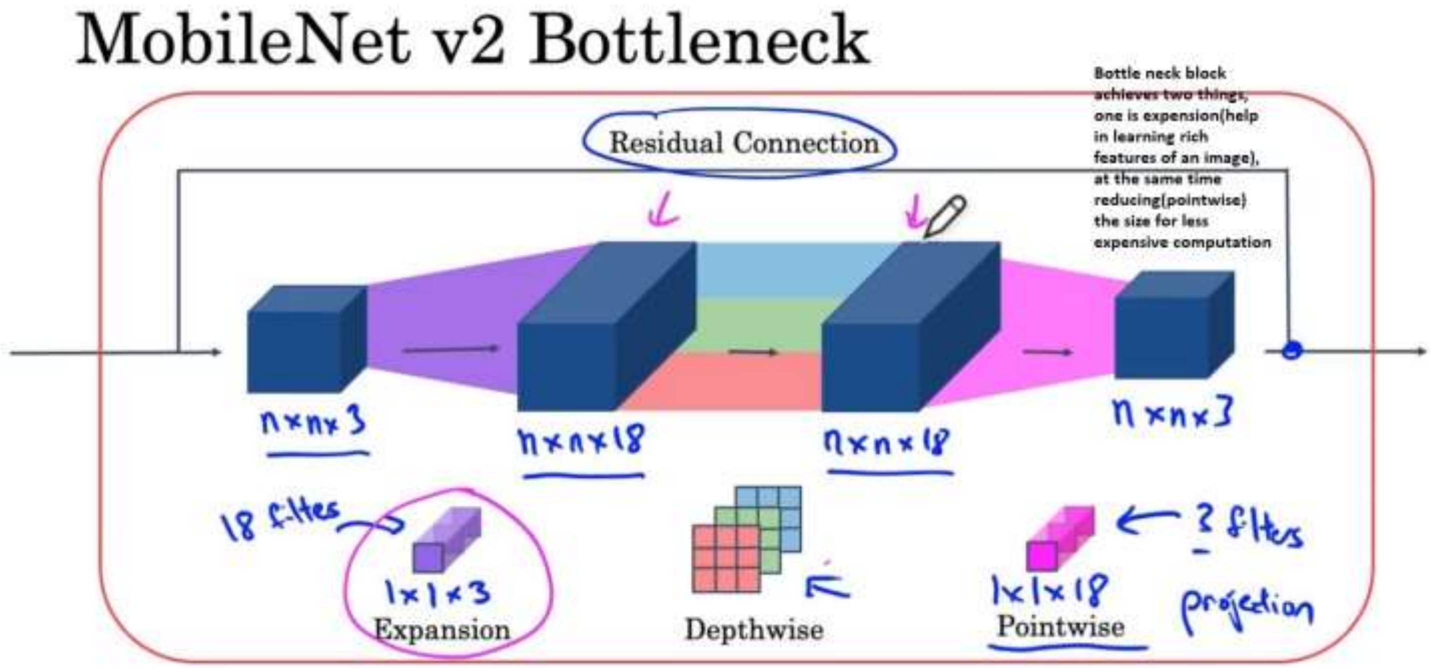
123.3



124.1

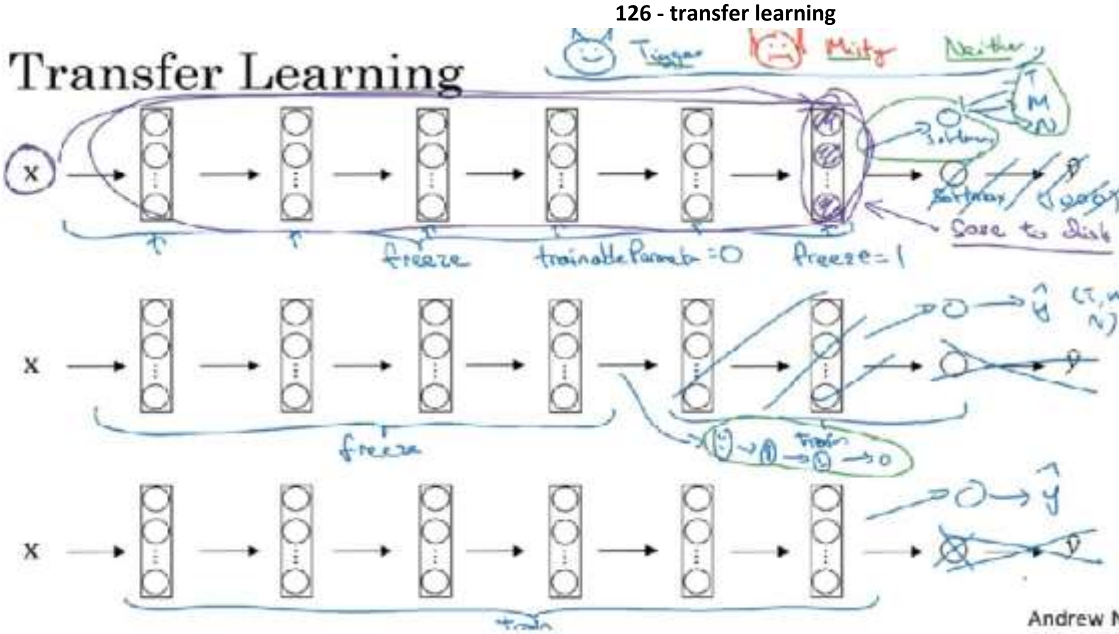


124.2



Bottle neck block achieves two things, one is expension(help in learning rich features of an image), at the same time reducing(pointwise) the size for less expensive computation

126 - transfer learning

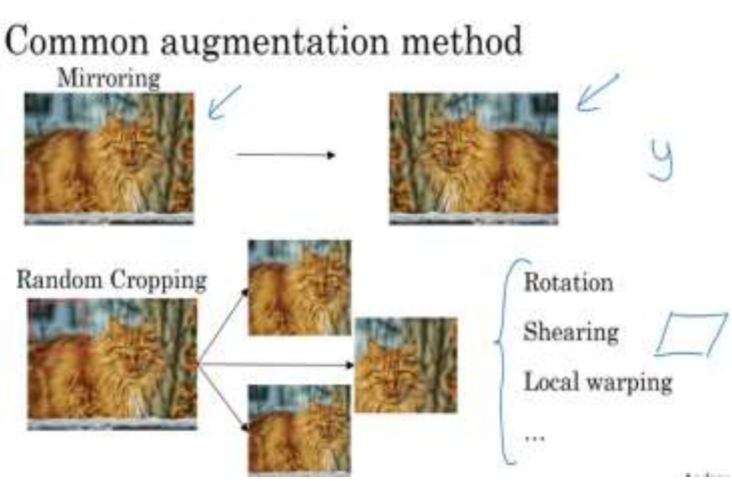


if u hv small dataset then download trained model from github(or imagenet) freeze, weighted layers and just remove output layer and apply softmax with your new output

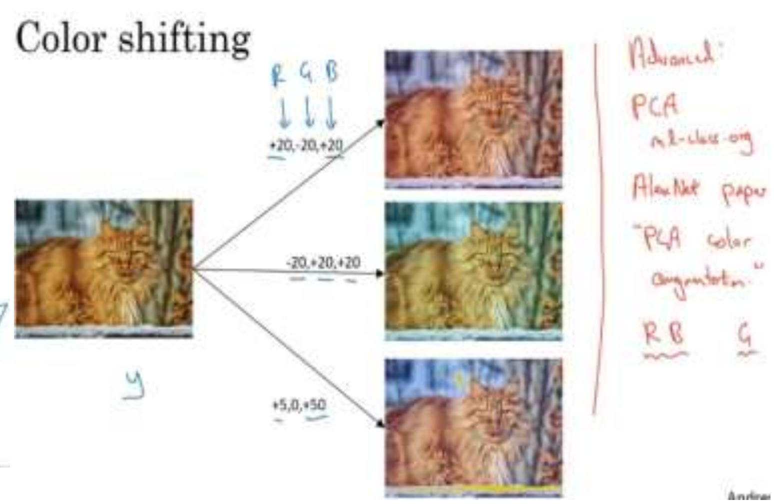
if u hv a bit large dataset then download trained model from github(or imagenet) freeze initial , weighted layers and train later layers and just remove output layer and apply softmax with your new output

if u hv a very large dataset then download trained model from github(or imagenet) use all the wieghts as intialization and just remove output layer and apply softmax with your new output

127.1

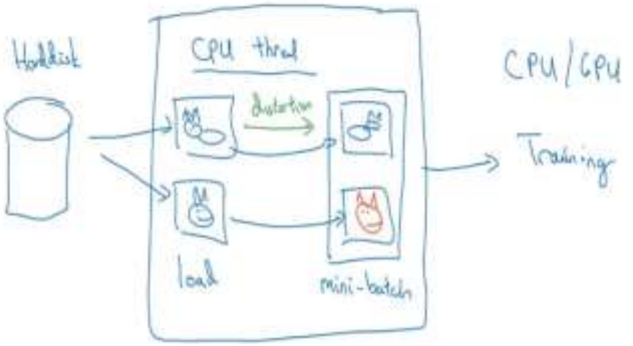


127.2



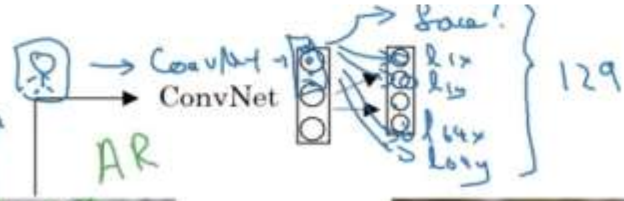
127.3

Implementing distortions during trainin



130

Landmark detection



b_x, b_y, b_h, b_w

we can label the points in order to identify the landmarks, whether the landmarks belong to face or legs or arms etc

Facial Landmark Detection is a computer vision task that involves detecting and localizing specific points or landmarks on a face, such as the eyes, nose, mouth, and chin. The goal is to accurately identify these landmarks in images or videos of faces in real-time and use them for various applications, such as face recognition, facial expression analysis, and head pose estimation.

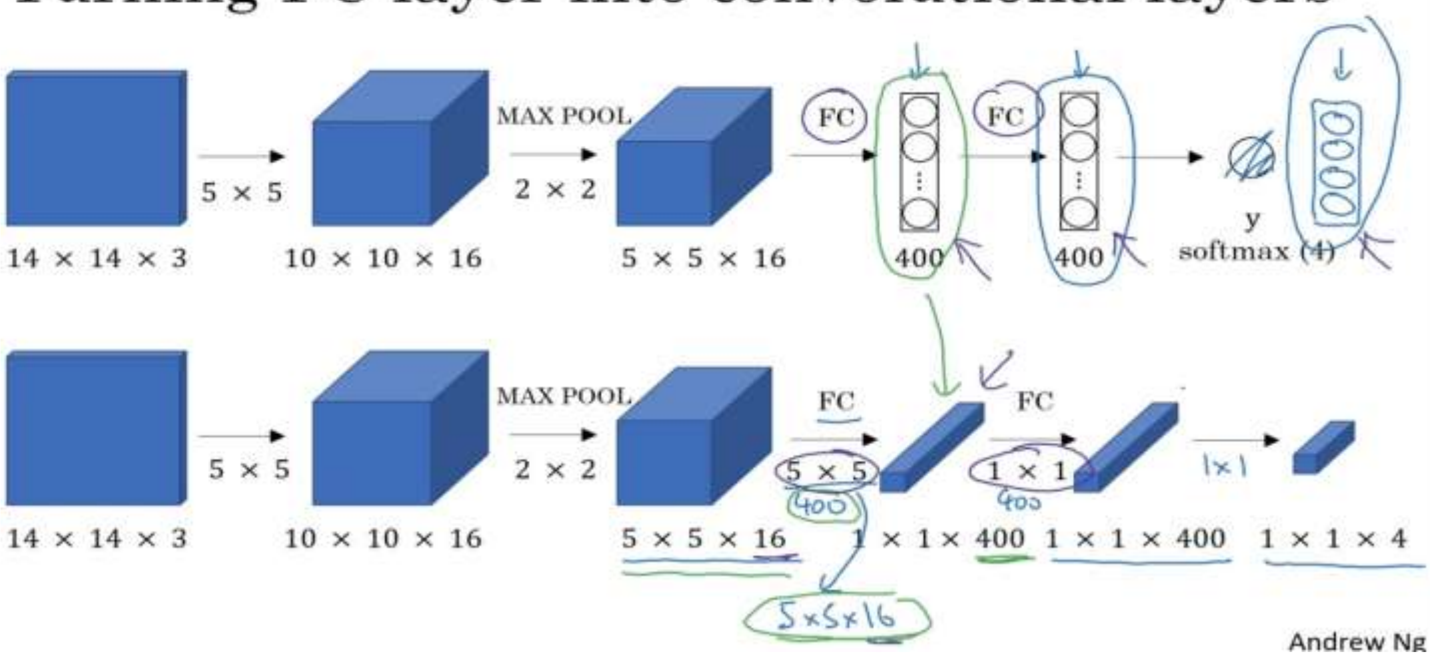
$l_{1x}, l_{1y},$
$l_{2x}, l_{2y},$
$l_{3x}, l_{3y},$
l_{4x}, l_{4y}
⋮
l_{64x}, l_{64y} } x, y

$l_{1x}, l_{1y},$
⋮
l_{32x}, l_{32y}

132.1

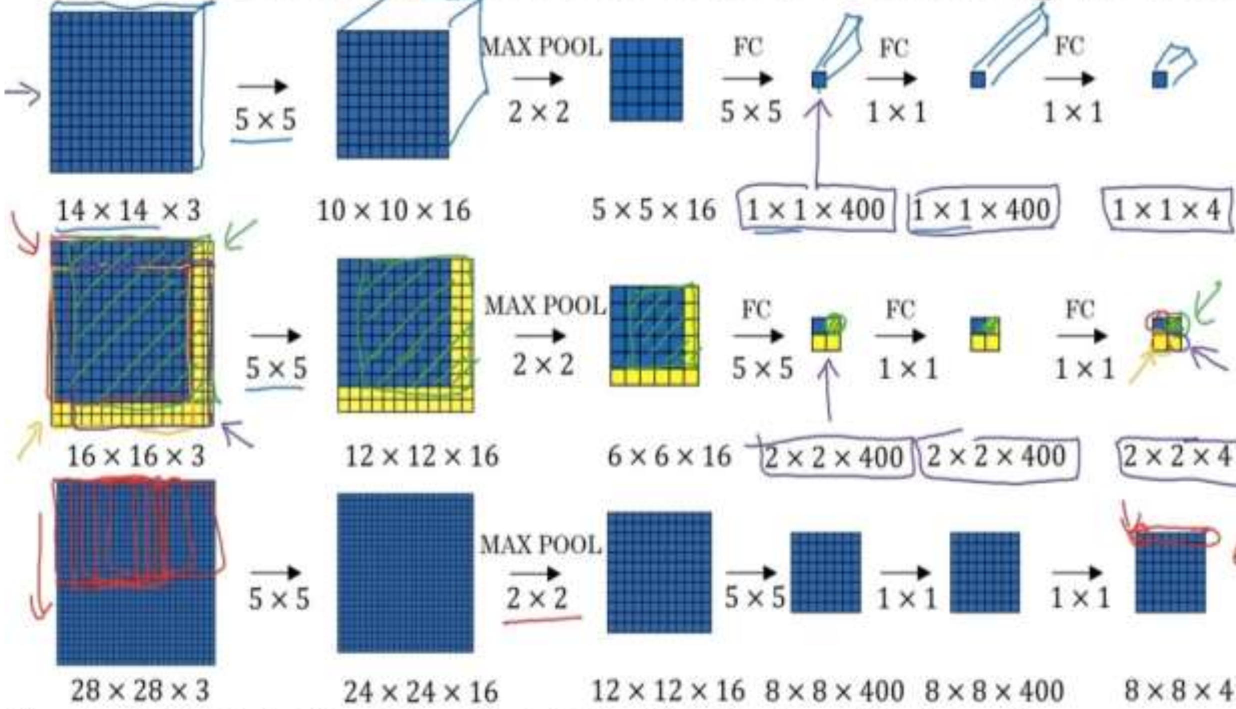
And

Turning FC layer into convolutional layers



132.2

Convolution implementation of sliding windows



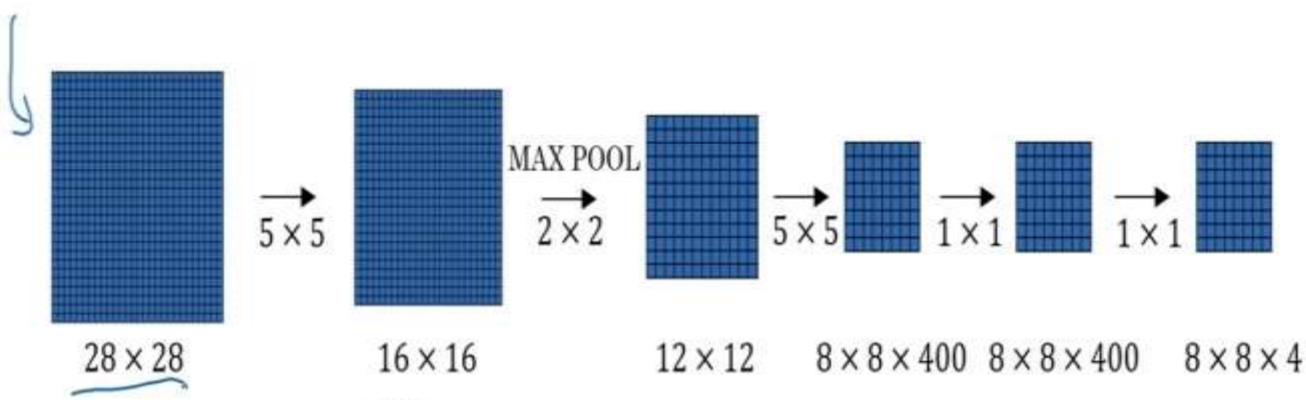
Lets say your test set image is 16x16x3, so the blue box(14x14x3) will convolve on the 16x16x3 image with stride of 2 and return classification 0 or 1. The resultant image will again convolve with 10x10x16 with stride of 2 and soo on

so what this process does is instead of forcing u to run forwardprop on 4 subsets of the input image independently, instead it combines all 4 into 1 form of computation & shares alot of computation in the region of the image that are common.

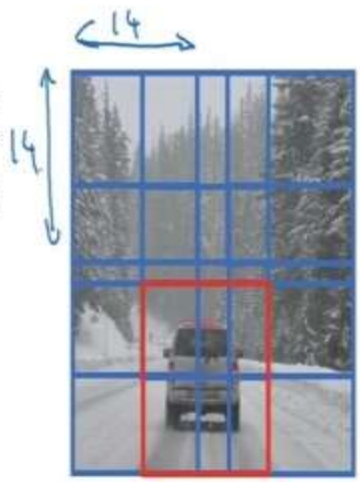
[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

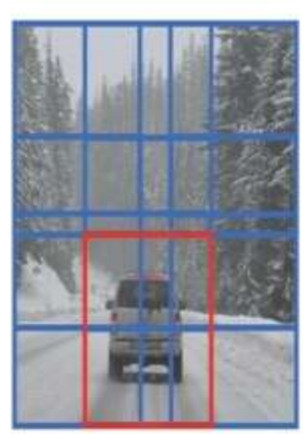
132.3

Convolution implementation of sliding windows



Previously what we did was take , let's say a kernel of 14x4 and slide this onto the picture one by one , and that makes the computation expensive and we observed that the majority of the computation caused by moving kerne was duplicative.

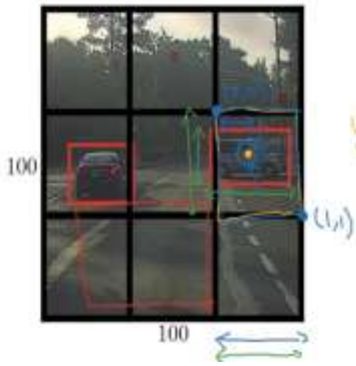




Now with the help of convNet on 132.2 we can directly pass the image in the network and the picture gets detected.

133.2

Specify the bounding boxes



$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

0.4, 0.3 } between 0 and 1

0.9, 0.5 } could be > 1

133.3

Here's how YOLO works in more detail:

Input image: YOLO takes an input image of any size and divides it into a grid of cells.

Anchor boxes: Each grid cell predicts a fixed number of anchor boxes, which are predefined boxes of different shapes and sizes. Each anchor box is represented by its center coordinates, width, height, and a confidence score, which represents the probability that the box contains an object.

Class probabilities: For each anchor box, YOLO predicts class probabilities for each object class that the model has been trained to detect. These probabilities represent the probability that the object inside the box belongs to a particular class.

Non-max suppression: After prediction, YOLO applies non-maximum suppression (NMS) to remove overlapping bounding boxes and keep only the most confident predictions.

Output: The final output of YOLO is a set of bounding boxes and class probabilities for each object detected in the input image.

134.1

Evaluating object localization

Intersection over Union (IoU)

$$= \frac{\text{size of}}{\text{size of}}$$

"Correct" if IoU ≥ $\boxed{0.5}$

0.6

IOU gives accurate bounding box, if IOU is 1 then the bounding box is 100% accurate.

More generally, IoU is a measure of the overlap between two bounding boxes.

134.2

Non-max suppression algorithm

19× 19

Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$ $\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$

Discard all boxes with $p_c \le 0.6$

While there are any remaining boxes:

- Pick the box with the largest p_c. Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

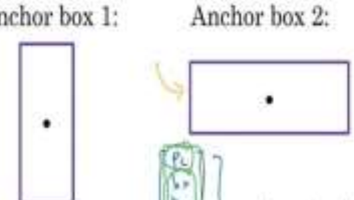
Andrew Ng

135.1

Overlapping objects:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1: Anchor box 2:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix}$$

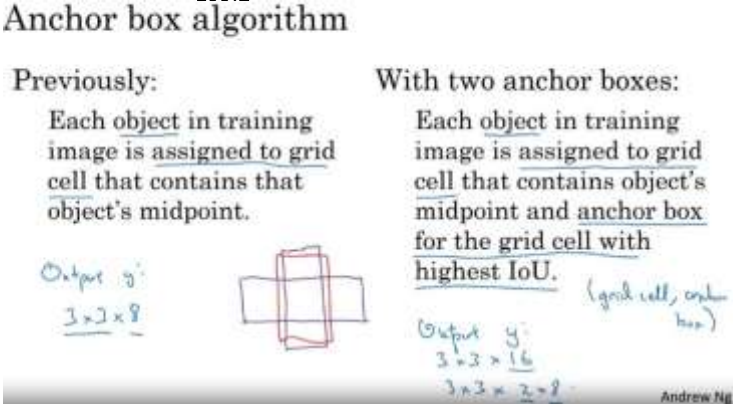
anchor box 1

anchor box 2

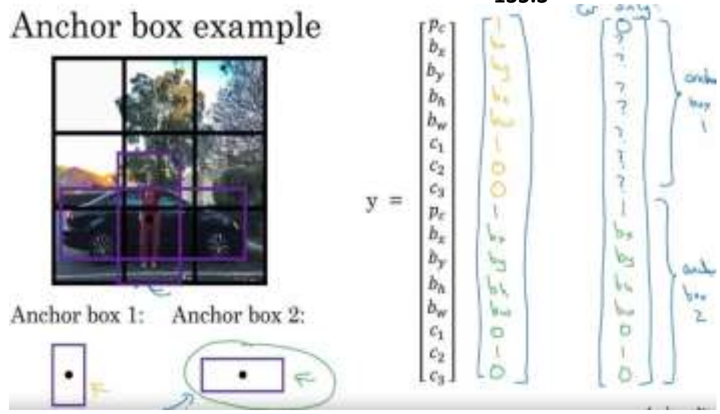
[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

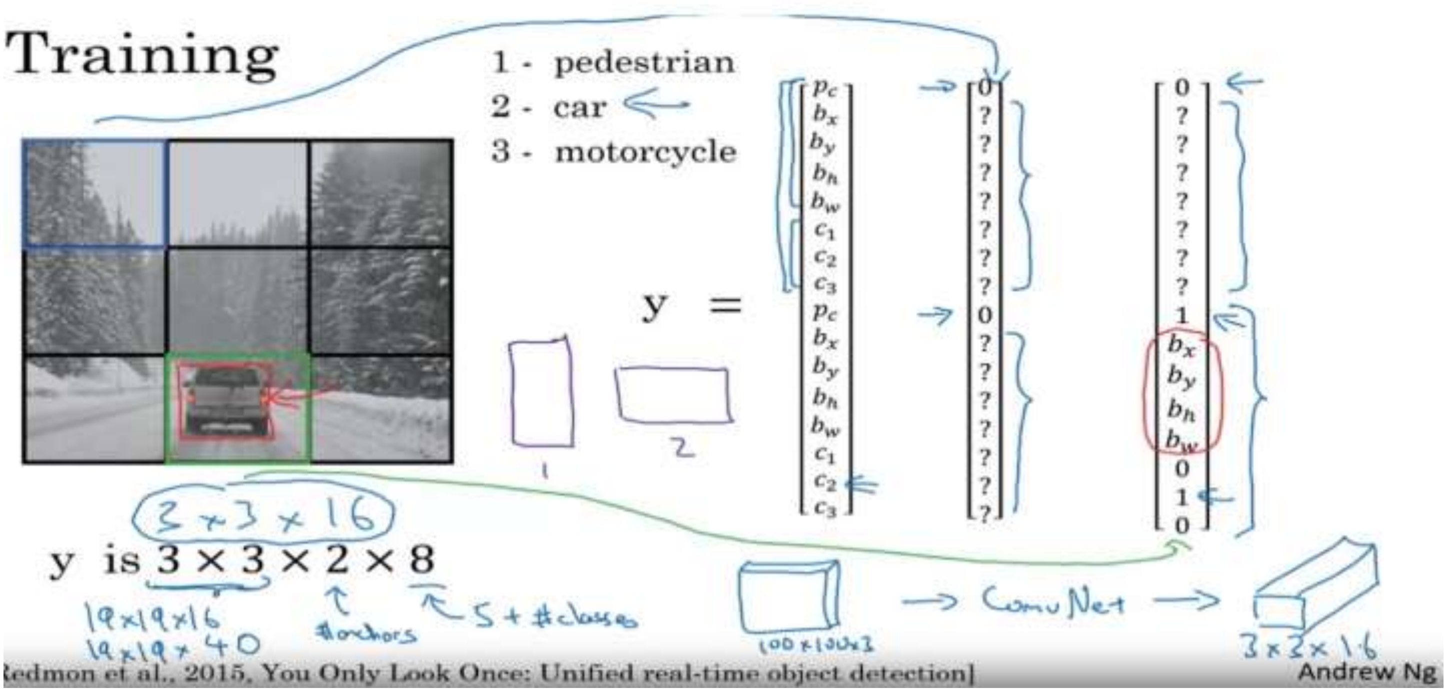
135.2



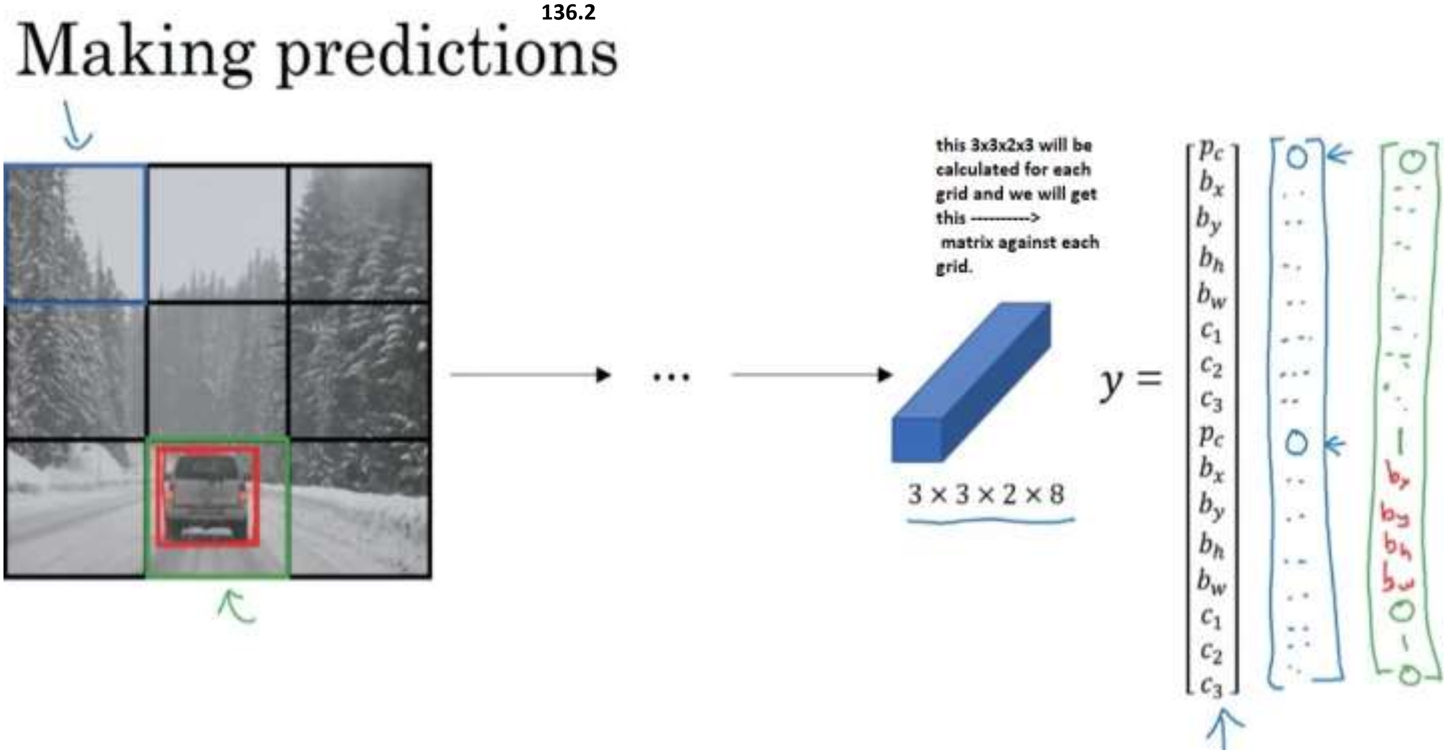
135.3



136.1



136.2



Outputting the non-max supressed outputs

136.3

- For each grid call, get 2 predicted bounding boxes.
  Each grid will have 2 anchor boxes of different sizes.

- Get rid of low probability predictions.

- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

136.4

Here's how RCNN works in more detail:

Region proposal: The first step in RCNN is to generate a set of region proposals using a separate algorithm, such as Selective Search. Selective Search combines low-level and high-level image features to identify regions that are likely to contain objects.

Feature extraction: For each proposed region, RCNN uses a deep convolutional neural network (CNN) to extract a fixed-length feature vector. This feature vector captures the salient features of the proposed region.

Classification: The feature vector is then fed into a set of fully connected layers, which are trained to classify the object in the proposed region into one of several classes. These classes typically correspond to different object categories, such as "car", "person", "dog", etc.

Bounding box regression: In addition to classifying the object, RCNN also predicts the object's bounding box coordinates (i.e., the object's location and size within the proposed region) using a separate set of fully connected layers.

Non-maximum suppression: After classification and bounding box regression, RCNN applies non-maximum suppression (NMS) to remove overlapping bounding boxes and keep only the most confident predictions.

Output: The final output of RCNN is a set of bounding boxes and class probabilities for each object detected in the input image.

136.5

Region proposal: R-CNN

Conlvolving an image is an expensive task, so what R-CNN does is it applies the segmentation algorithm and creates multiple segments of an image, which makes it easier to identify objects since it makes multiple blobs which reduces the effort to find objects

Segmentatial algorthm

$\sim 2{,}000$

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation] Andrew Ng

Faster algorithms

136.6

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

Proposal generation: Like other region-based methods, Fast R-CNN relies on a separate algorithm to generate region proposals, such as Selective Search, which can be time-consuming.

Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN: Use convolutional network to propose regions.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks] Andrew Ng

137.1

Per-pixel class labels

1. Car
2. Building
3. Road

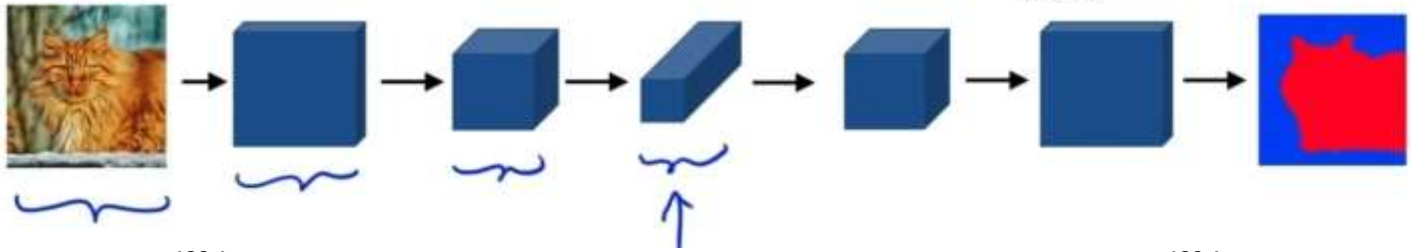
Segmentation Map

137.2

Deep Learning for Semantic Segmentation

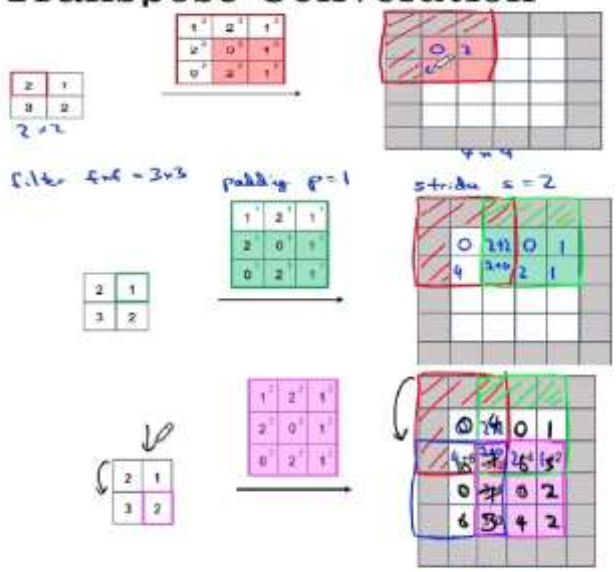
So when we pass the image through N.N the normally the image dimensions gets smallar but here the dimensions get bigger(means as we go deeper into the N.N the height and width will go back and # of channels will decrease) so they can gradually blow it back-up to a full-size image, which is a size we want for the output.

All this is done using transpose convolution

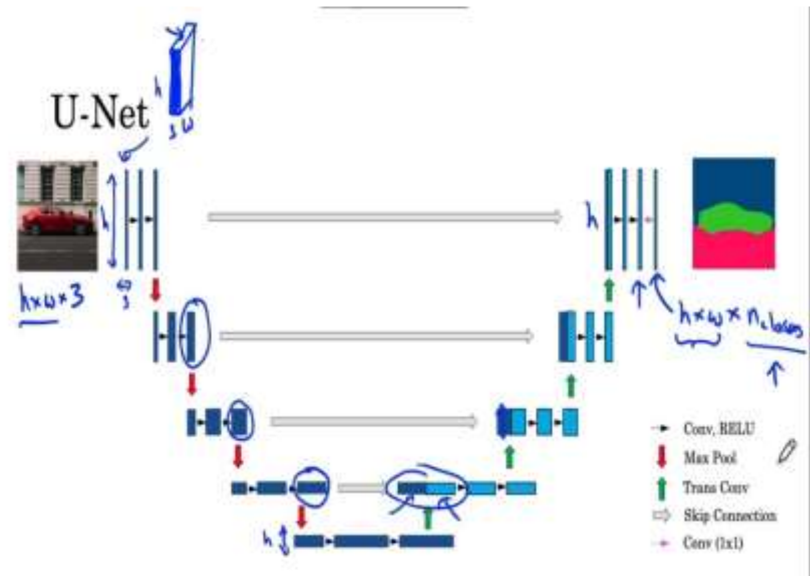


138.1

Transpose Convolution



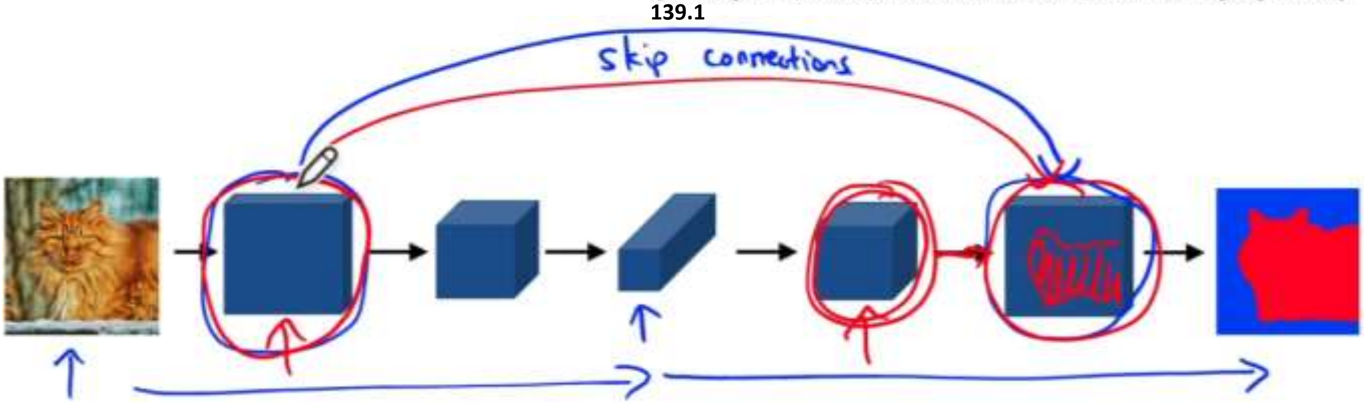
139.1



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Andrew Ng

139.1



By reducing the channels we againmake picture bigger using transpose convolution and by skipping connection we are making sure that N.N knows where in the original image the cat located.

144

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly, $d(A,P) + \alpha \le d(A,N)$ is easily satisfied.

$\|f(A)-f(P)\|^2 + \alpha \le \|f(A)-f(N)\|^2$

Choose triplets that're "hard" to train on.

$d(A,P) + \alpha \le d(A,N)$

$d(A,P) \approx d(A,N)$

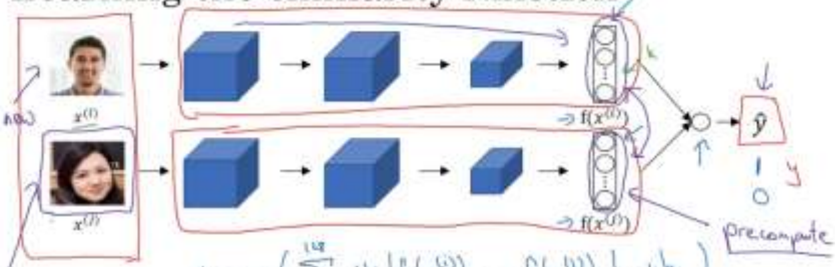
Face Net

Deep Face

Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

145.1

Learning the similarity function



$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

$$\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k} \qquad \chi^2$$

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

145.2

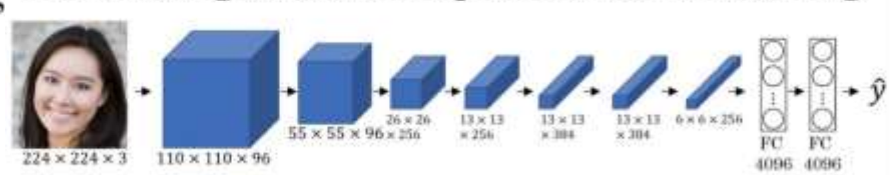
Face verification supervised learning

x	y
	1 "Same"
	0 "different"
	0
	1

Taigman et. al., 2014. DeepFace closing the gap to human level performance]

146.1

Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

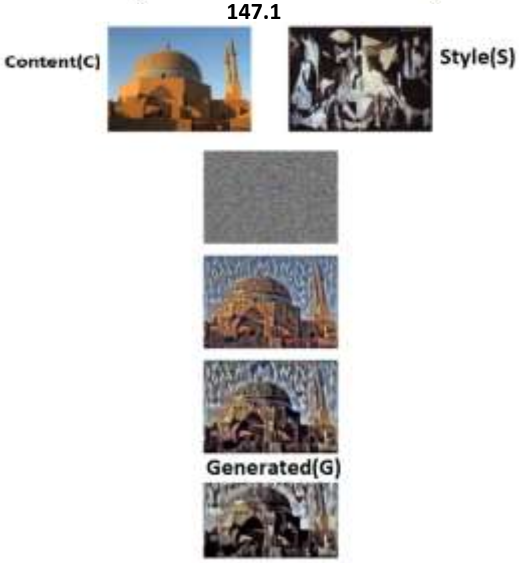
Andrew Ng

146.2

Visualizing deep layers

Layer 1   Layer 2   Layer 3   Layer 4   Layer 5

147.1



147.2

Content cost function

$$J(G) = \alpha\, J_{content}(C,G) + \beta\, J_{style}(S,G)$$

- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

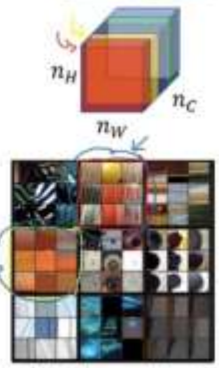
$$J_{content}(C,G) = \frac{1}{2}\left\| a^{[l](C)} - a^{[l](G)} \right\|^2$$

If J(C,G) is greater , this reprsents that the generated image will be more similar to content image, and if J(S,G) is greater then this reprsents that the generated image will be more similar to Style image, the hyperparameters are there to incentivize these J(C,G) and J(S,G).

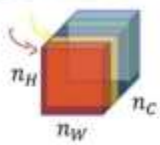
148.1

Intuition about style of an image

Style image



Generated Image



Correlated?
Uncorrelated

Correlated mean that wherever there is vertical subtle texture in 1,2, the 2x1 will also have it.

Uncorrelated mean that wherever there is vertical subtle texture in 1,2, the 2x1 will not have it.

[Gatys et al., 2015. A neural algorithm of artistic style]

148.3

Style cost function

$\| G^{[l](S)} - G^{[l](G)} \|_F^2$

$$J^{[l]}_{style}(S,G) = \frac{1}{(2n_H^{[l]}n_W^{[l]}n_C^{[l]})^2}\sum_k\sum_{k'}\left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'}\right)^2$$

$$J_{style}(S,G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S,G)$$

$$J(G) = \alpha J_{content}(C,G) + \beta J_{style}(S,G)$$

G

148.2

Style matrix

H W C

Let $a^{[l]}_{i,j,k}$ = activation at (i,j,k). $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

n_c

$G^{[l]}_{kk'}$

$k = 1, \dots, n_c^{[l]}$

$$\rightarrow G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}}\sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk}\, a^{[l](S)}_{ijk'}$$

this is done for both style(S) and generated(G) image.

"Gram matrix"

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H}\sum_{j=1}^{n_W} a^{[l](G)}_{i,j,k}\, a^{[l](G)}_{i,j,k'}$$

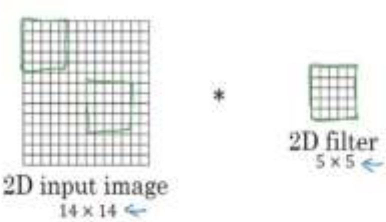
Calculates how correlated the activations are in channel k and k' them compare them.

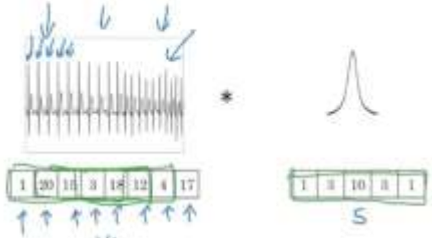
$$J^{[l]}_{style}(S,G) = \| G^{[l](S)} - G^{[l](G)} \|_F^2$$

$$= \frac{1}{(2n_H^{[l]}n_W^{[l]}n_C^{[l]})^2}\sum_k\sum_{k'}\left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'}\right)^2$$

149.1

Convolutions in 2D and 1D





14×14×3 * 5×5×3

→ 10×10×16

10×10×16 * 5×5×16

→ 6×6×32

14×1 * 5×1

→ 10×16

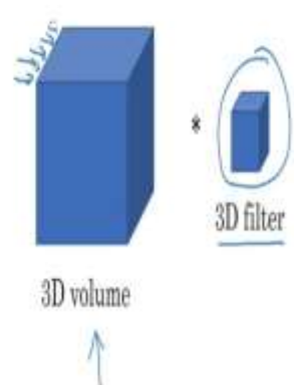
10×16 * 5×16

→ 6×32

Andrew

149.2

3D convolution



n_c

14×14×14×1

* 5×5×5×1

16 filters

→ 10×10×10×16

* 5×5×5×16

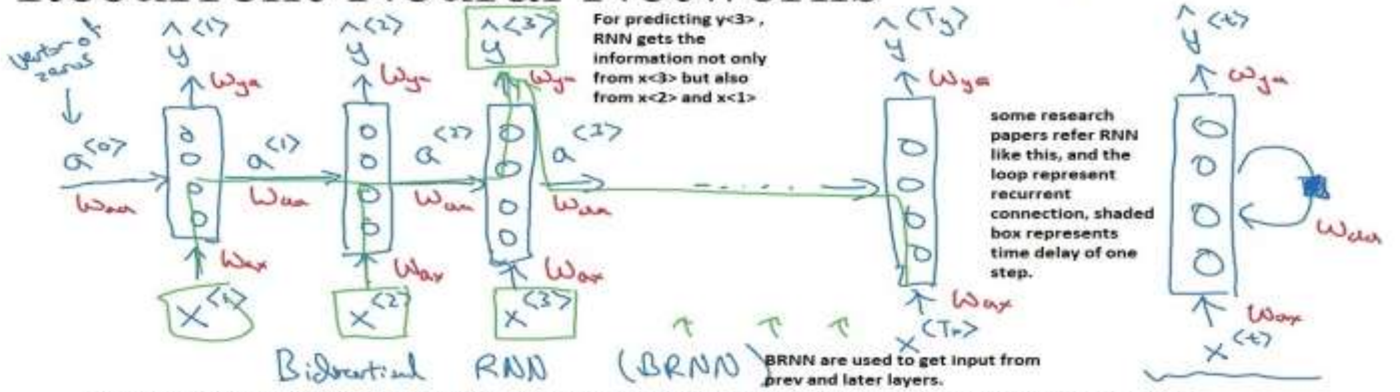
32 filters

→ 6×6×6×32

154.1

Recurrent Neural Networks

$T_x = T_y$



Downside is that RNN only uses the info that is in the erlier in the seq to predict y<3> , it does not use info from x<4>,x<5> and so on

Below we can see given the first three words it not possible to predict whether teddy is part of a person's name. In the first eg it is , in the 2nd it is not.

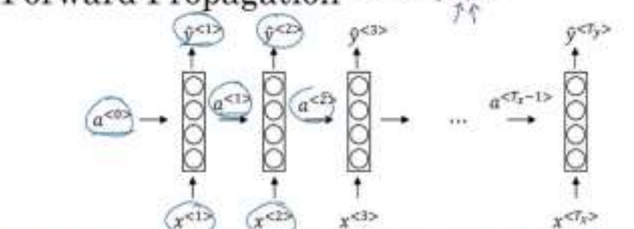
He said, "Teddy Roosevelt was a great President."

He said, "Teddy bears are on sale!"

155.1

Forward Propagation

$a \leftarrow W_{ax} x^{<t>}$



$a^{<0>} = \vec{0}$

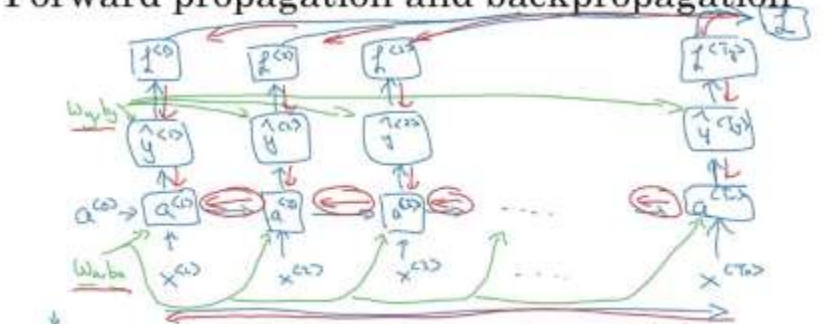
$a^{<1>} = g_1(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a) \leftarrow \tanh / \text{ReLU}$

$\hat{y}^{<1>} = g_2(W_{ya} a^{<1>} + b_y) \leftarrow \text{Sigmoid}$

$$\boxed{\begin{aligned} a^{<t>} &= g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a) \\ \hat{y}^{<t>} &= g(W_{ya} a^{<t>} + b_y) \end{aligned}}$$

155.3

Forward propagation and backpropagation



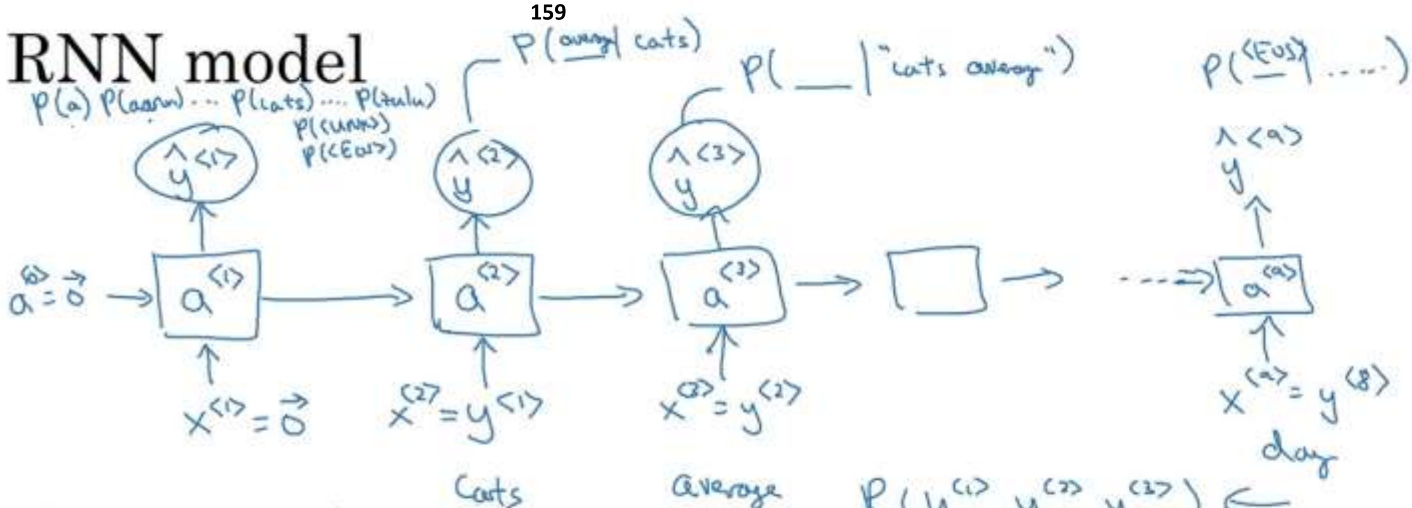
$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1-y^{<t>}) \log (1-\hat{y}^{<t>})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Backpropagation through time

159

RNN model



Cats average 15 hours of sleep a day. <EOS>

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

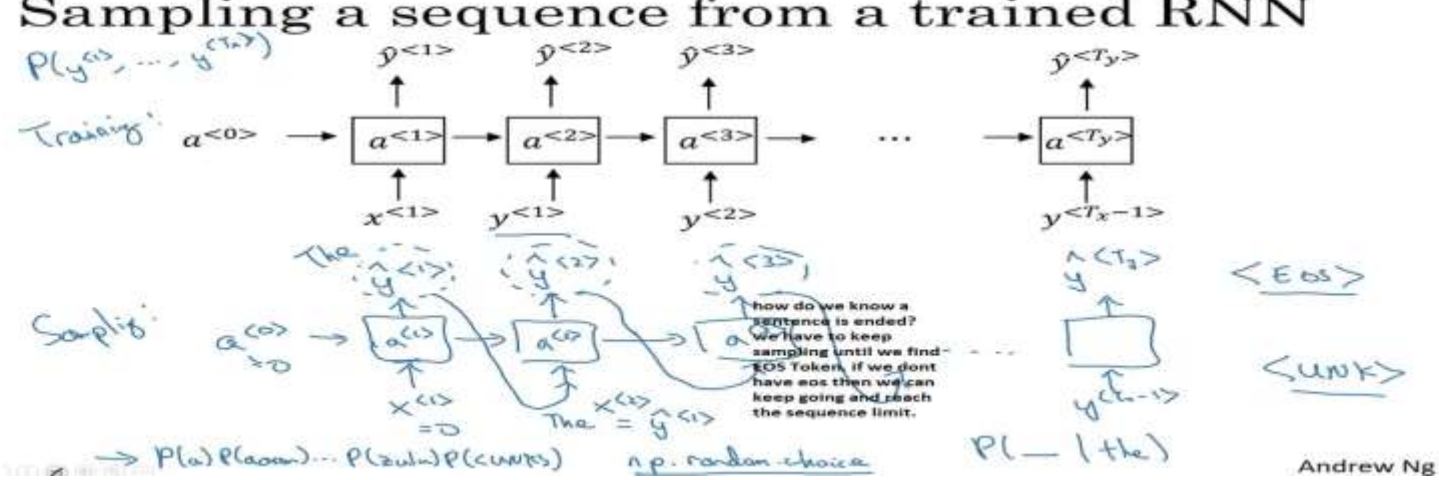
$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

$P(y^{<1>}, y^{<2>}, y^{<3>})$

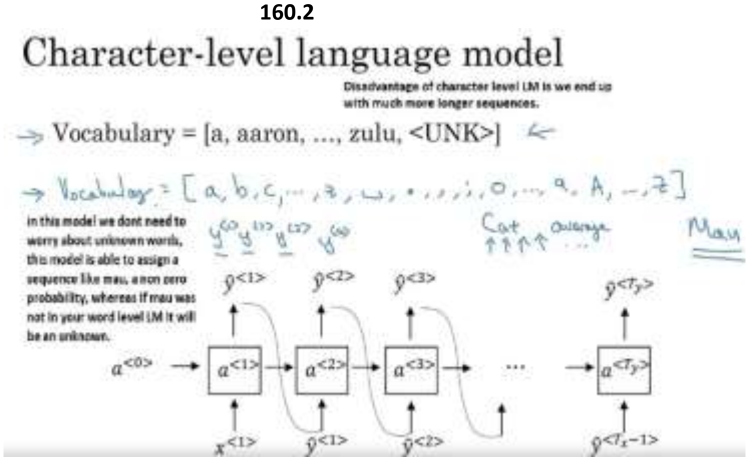
$= P(y^{<1>})\, P(y^{<2>} | y^{<1>})\, P(y^{<3>} | y^{<1>}, y^{<2>})$

Andrew Ng

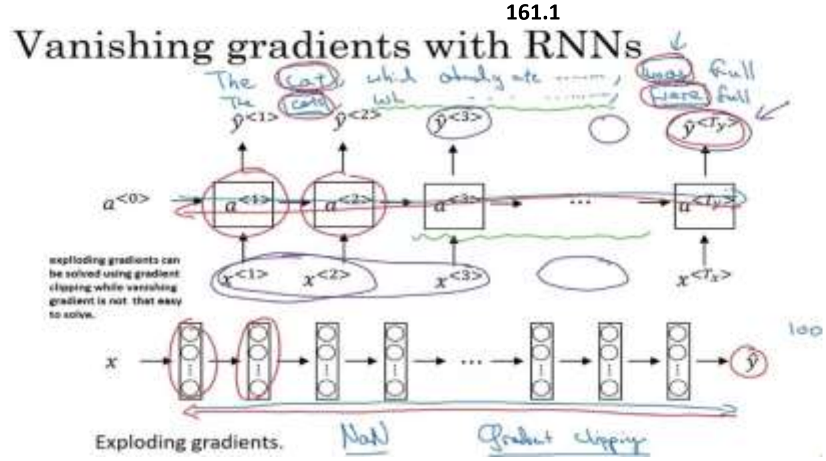
160.1



160.2

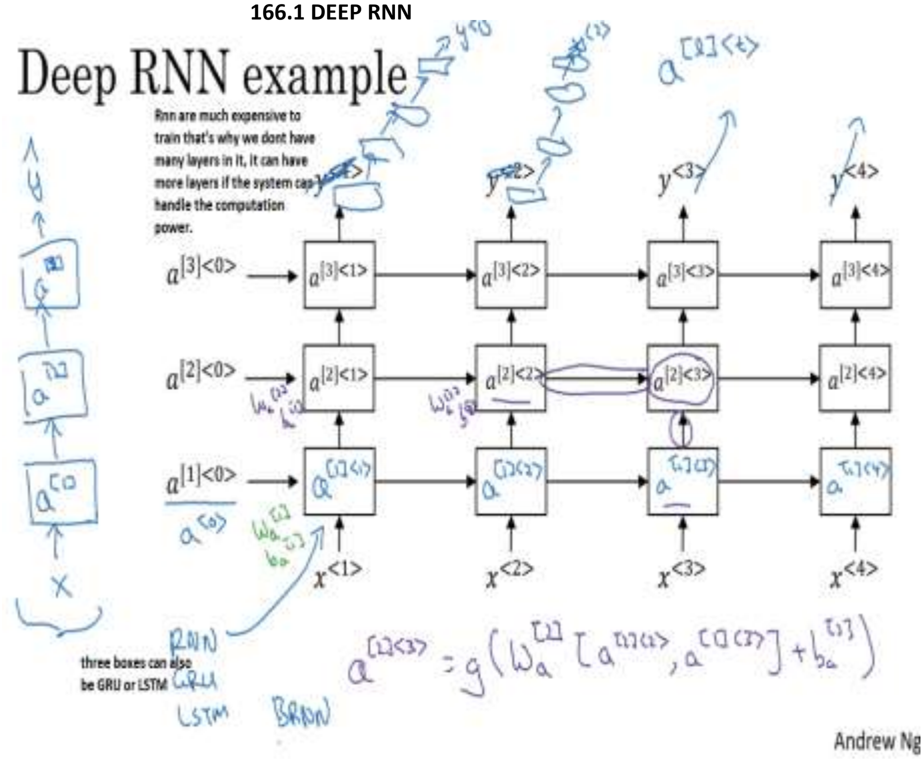


161.1



166.1 DEEP RNN

166BRNN



167.1

Word representation

V = [a, aaron, ..., zulu, <UNK>] $|V| = 10{,}000$

1-hot representation

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
O_{5391}	O_{9853}				

I want a glass of orange juice

I want a glass of apple ?

A model doesnt know that apple and orange juice are similar, because if you take the product of any two vectors, it will be 0. So it does not know that somehow apple and orange are similar category.

167.2

Featurized representation: word embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
size, cost, alive, verb	⋮	⋮				
	e_{5391}	e_{9853}				

300

In this way we assign different features to different words based on numbers, upon processing this whole vectorized feature representation, the model can see that in the food category orange and apple are correlated.

I want a glass of orange juice.

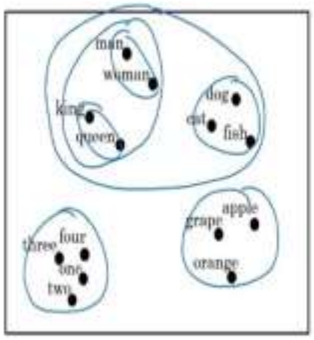
I want a glass of apple juice.

Andrew Ng

167.3

Visualizing word embeddings

The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function.



t-SNE

→ 300 D ↓ 2 D

its called embedding because each word is embedded in the space.

[van der Maaten and Hinton., 2008. Visualizing data using t-SNE] Andrew Ng

168.1

Transfer learning and word embeddings

A
1. Learn word embeddings from large text corpus. (1-100B words)

   (Or download pre-trained embedding online.)

B
2. Transfer embedding to new task with smaller training set. (say, 100k words) → 10,000 → 300

3. Optional: Continue to finetune the word embeddings with new data.

168.2

Analogies

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
	e_{5391} e_{man}	e_{woman}				

we can make model learn analogies using word embeddings by subtracting the respective vectors, just like we did with man vs woman and king vs queen, the goal is to balance the eqn, if the eqn gets balanced then its a sign that we have to use that word.

Man → Woman as King → ? Queen

$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

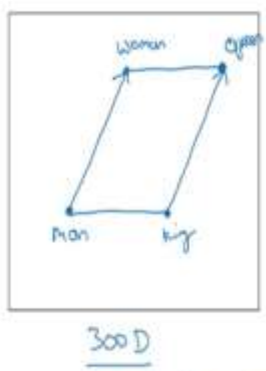
$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$e_{man} - e_{woman} \approx e_{king} - e_{?}$ e_{queen}

[Mikolov et. al., 2013, Linguistic regularities in continuous space word representations] Andrew Ng

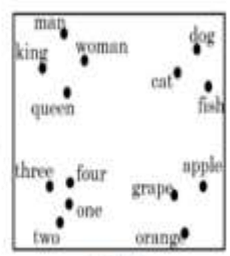
169.1

Analogies using word vectors



$300D \to 2D$

similarity function is used in order to find the word for analogy, some resach papers got the accuracy btw 30 to 75%. Most commonly used similarity function is cosine similarity.



$e_{man} - e_{woman} \approx e_{king} - e_{?}\ e_w$

Find word w: $\arg\max_w \ sim(e_w, e_{king} - e_{man} + e_{woman})$

30 - 75%

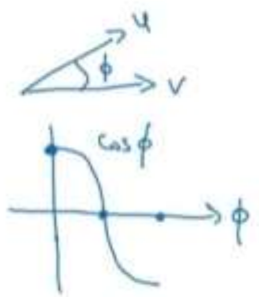
Andrew Ng

169.2

Cosine similarity

$$\text{sim}(u,v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

it's called cosine because of the cosine angle btw two vectors u and v

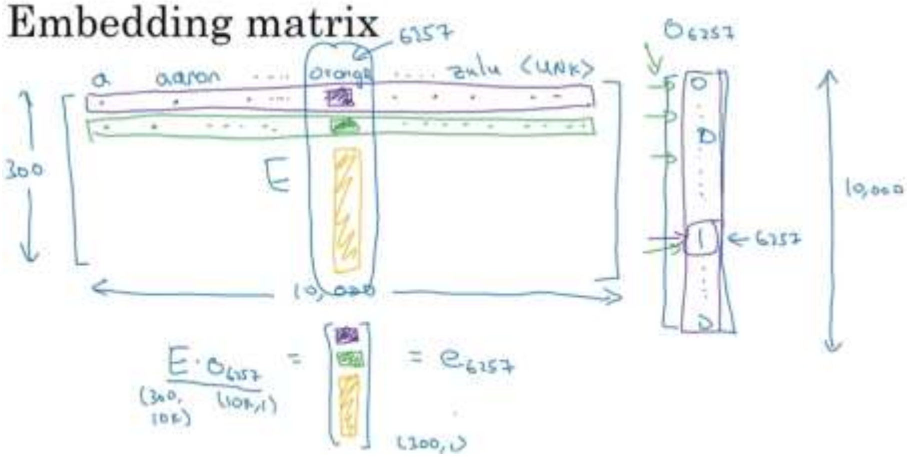


$\|u-v\|^2$

we can also use squared difference similarity, but it is more like dissimilarity but it works.

169.3

Embedding matrix



$E \cdot o_{6257} = e_{6257}$

(300, 10k) (10k, 1) (300, 1)

170.1

Neural language model

I	want	a	glass	of	orange	juice .
4343	9665	1	3852	6163	6257	

$e_{4343} = E\, o_{4343}$

apple juice.

~~I $o_{4343} \longrightarrow E \longrightarrow e_{4343}$~~

~~want $o_{9665} \longrightarrow E \longrightarrow e_{9665}$~~

a $o_1 \longrightarrow E \longrightarrow e_1$

glass $o_{3852} \longrightarrow E \longrightarrow e_{3852}$

of $o_{6163} \longrightarrow E \longrightarrow e_{6163}$

orange $o_{6257} \longrightarrow E \longrightarrow e_{6257}$

E here will be similar

in this case if we have total 6 words and the embedding is of 300 words then we will have 6x300 = 1800 word vector.

softmax 10,000

$W^{[1]}, b^{[1]}$ $W^{[2]}, b^{[2]}$

here we can specify how many embedding words we need to predict "juice", it can be all six words or initial 4 words, acc. to this the vector shape is determined

~~1800~~ 1200

[Bengio et. al., 2003, A neural probabilistic language model]

Andrew Ng

170.2

Other context/target pairs

I want a glass of orange juice to go along with my cereal.

context target

Context: Last 4 words.

we can specify 4 words from L & R.

- 4 words on left & right — a glass of orange ___ to go along with
- Last 1 word — orange ___
- Nearby 1 word — glass ___

We can specify orange to identify juice

we can identify one nearby word, such as glass, which refers to juice if we think abt it

skip gram

178.1

Model

$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} \underbrace{f(X_{ij})}_{\text{weighting term}} (\theta_i^T e_j + b_i + b_j' - \log X_{ij})^2$$

"$\theta_i^T e_j$"

$f(X_{ij}) = 0$ if $X_{ij} = 0$. "$0 \log 0$" $= 0$

this, is, of, a, ...

durian

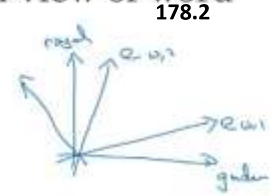
θ_i, e_j are symmetric

$$e_w^{(final)} = \frac{e_w + \theta_w}{2}$$

178.2

A note on the featurization view of word embeddings

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)
Gender	−1	1	-0.95	0.97
Royal	0.01	0.02	0.93	0.95
Age	0.03	0.02	0.70	0.69
Food	0.09	0.01	0.02	0.01



$$\text{minimize } \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i - b_j' - \log X_{ij})^2$$

$(A\theta_i)^T(A^{-T} e_j) = \theta_i^T \cancel{A^T A^{-T}} e_j$

Andrew Ng

179.1

Sentiment classification problem

$x \longrightarrow y$

x	y
The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆

10,000 → 100,000 words

179.2

Simple sentiment classification model

The	dessert	is	excellent	★★★★☆
8928	2468	4694	3180	

The $o_{8928} \longrightarrow E \longrightarrow e_{8928}$

desert $o_{2468} \longrightarrow E \longrightarrow e_{2468}$

is $o_{4694} \longrightarrow E \longrightarrow e_{4694}$

excellent $o_{3180} \longrightarrow E \longrightarrow e_{3180}$

Avg → O softmax → $\hat{y}$ 1-5

300D

300 Dimensional

This approach lacks where given a sentence like this one that has mulitple "good" in it, the model end up predictiing it to be 5 star.

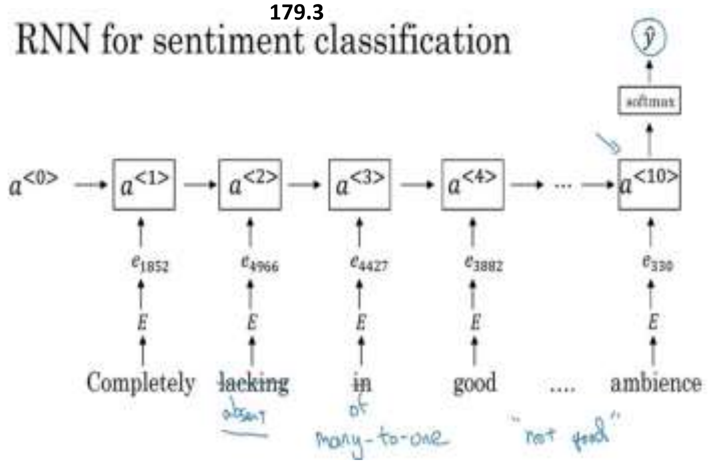
300D

100 B words

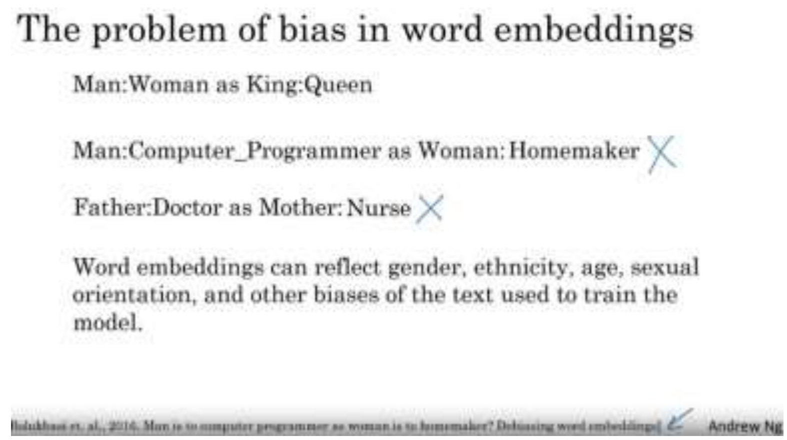
"Completely lacking in good taste, good service, and good ambience."

Andrew

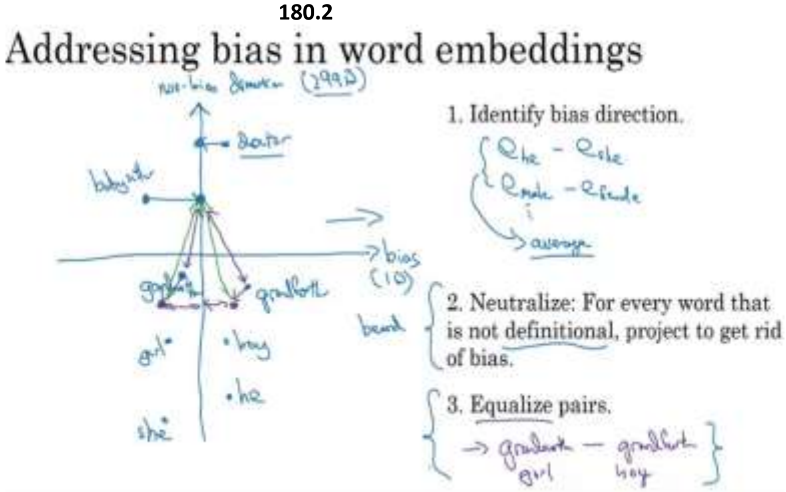
179.3



180.1



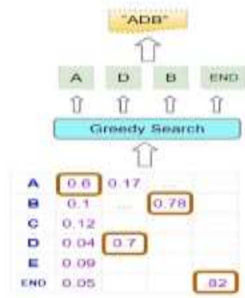
180.2



184.1

Greedy Search

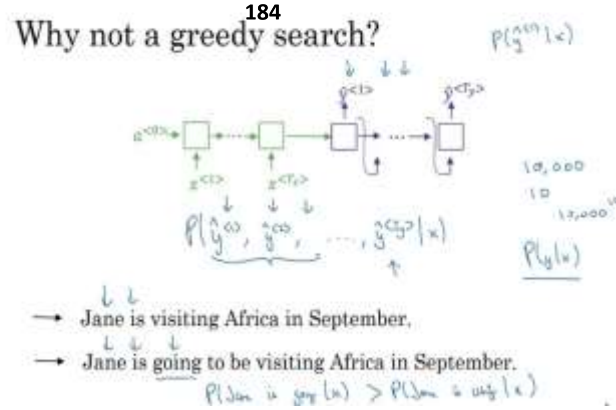
A fairly obvious way is to simply take the word that has the highest probability at each position and predict that. It is quick to compute and easy to understand, and often does produce the correct result.



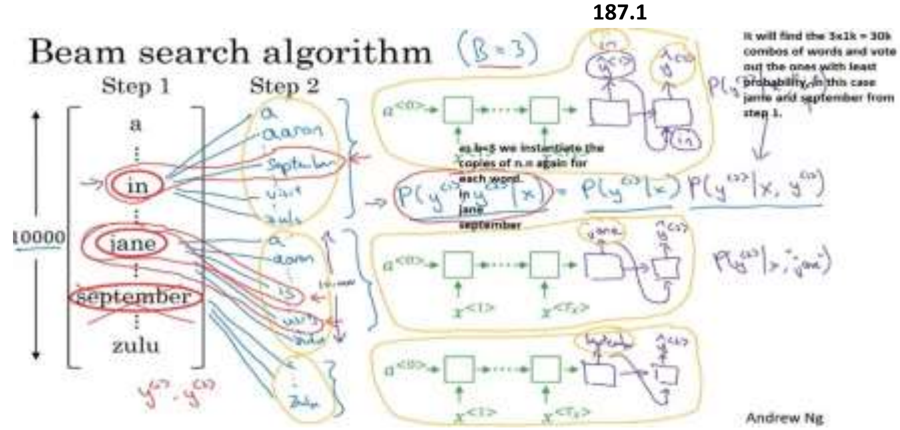
Greedy Search (Image by Author)

In fact, Greedy Search is so easy to understand, that we don't need to spend more time explaining it 😀. But can we do better?

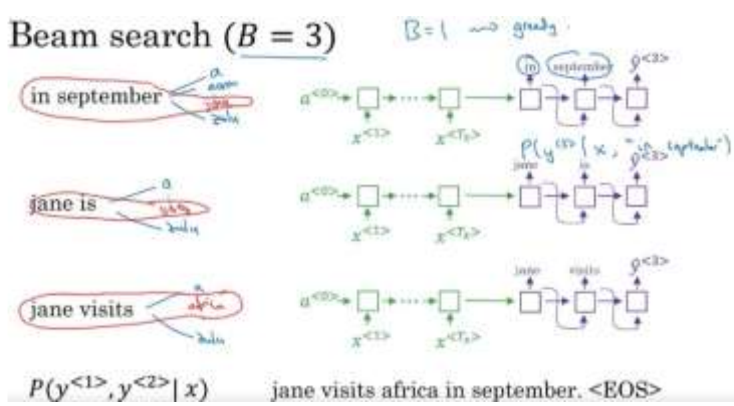
184



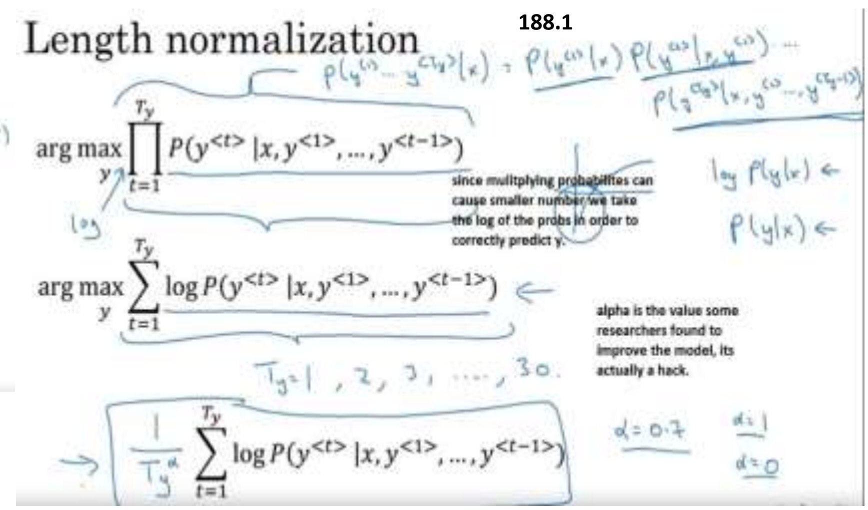
187.1



187.2



188.1



188.2

Beam search discussion

large B: better result, slower
small B: worse result, faster

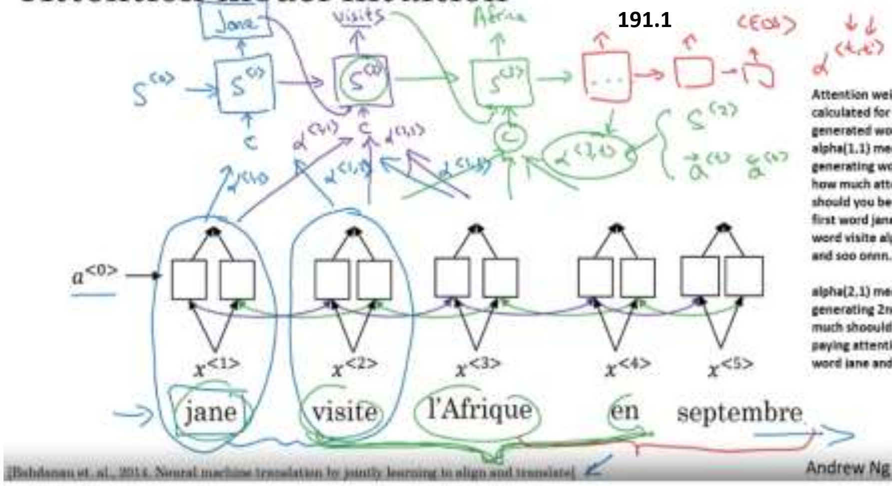
Beam width B?

1 → 10, 100, 1000 → 3000

Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find exact maximum for $\arg\max_y P(y|x)$.

191.1

Attention model intuition



Attention weights are calculated for each generated word. alpha(1,1) means while generating word jane how much attention should you be paying on first word jane. second word visite alpha(1,2) and soo onnn.

alpha(2,1) means while generating 2nd word how much shoould u be paying attention to first word jane and so on

188

Example

→ RNN
→ Beam Search

BT

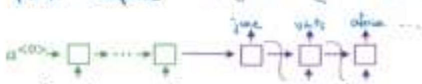
Jane visite l'Afrique en septembre.

To overcome this problem we can calculate y* and y^ separately and compare their probabilities, select the one with higher prob

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. $(\hat{y})$

RNN computes $P(y^*|x) \gtrless P(\hat{y}|x)$



Error analysis on beam search

$P(y^*|x)$
$P(\hat{y}|x)$

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. $(\hat{y})$

Case 1: $P(y^*|x) > P(\hat{y}|x)$ $\arg\max_y P(y|x)$

Beam search chose $\hat{y}$. But y^* attains higher $\boxed{P(y|x)}$.

Conclusion: Beam search is at fault.

Case 2: $P(y^*|x) \leq P(\hat{y}|x)$

y^* is a better translation than $\hat{y}$. But RNN predicted $\boxed{P(y^*|x)} < P(\hat{y}|x)$.

Conclusion: RNN model is at fault.

Error analysis process

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	B
...	...	—	—	R
		—	—	R
				R
				⋮

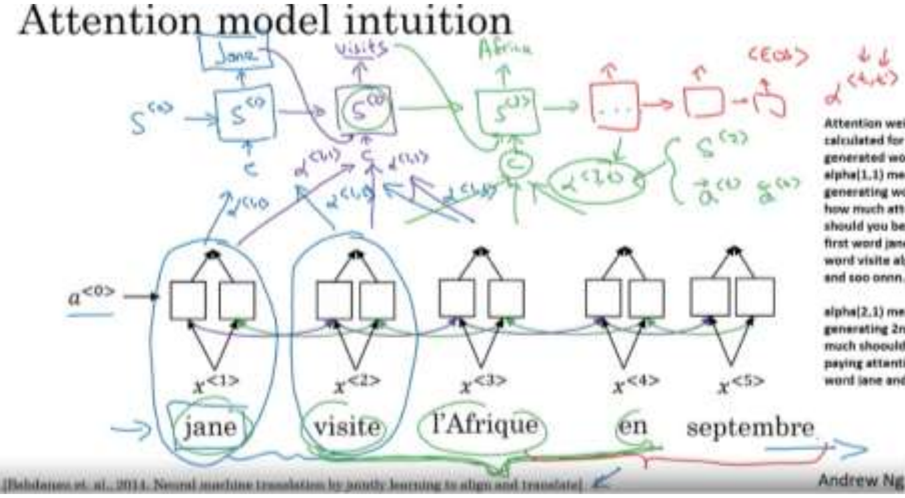
if most errors are by beam , then we can try increasing b value and tune other hyperpameters.

if RNN is making more errors then we can add regularization or get more training data and soo

Figures out what fraction of errors are "due to" beam search vs. RNN model

191.1

Attention model intuition

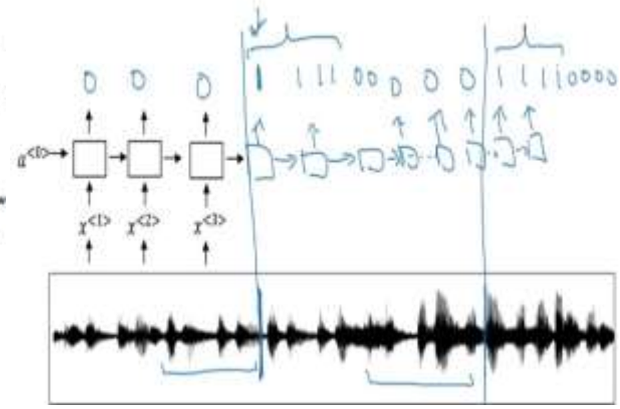


Attention weights are calculated for each generated word. alpha(1,1) means while generating word jane how much attention should you be paying on first word jane. second word visite alpha(1,2) and soo onnn.

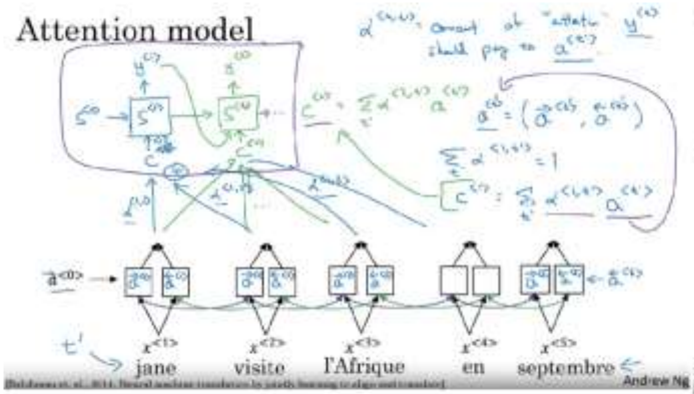
alpha(2,1) means while generating 2nd word how much shoould u be paying attention to first word jane and so on

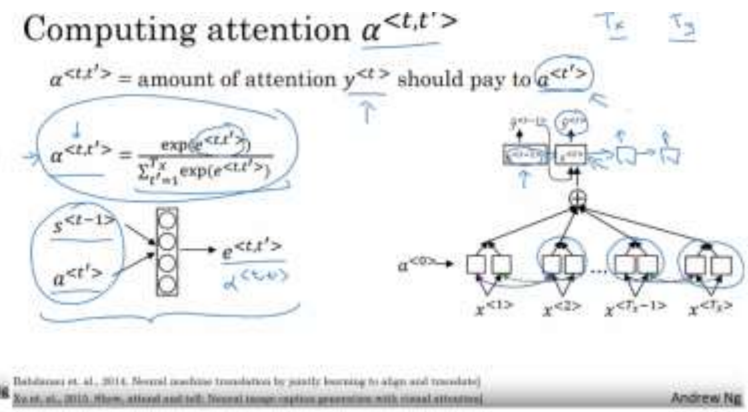
194

Trigger word detection algorithm

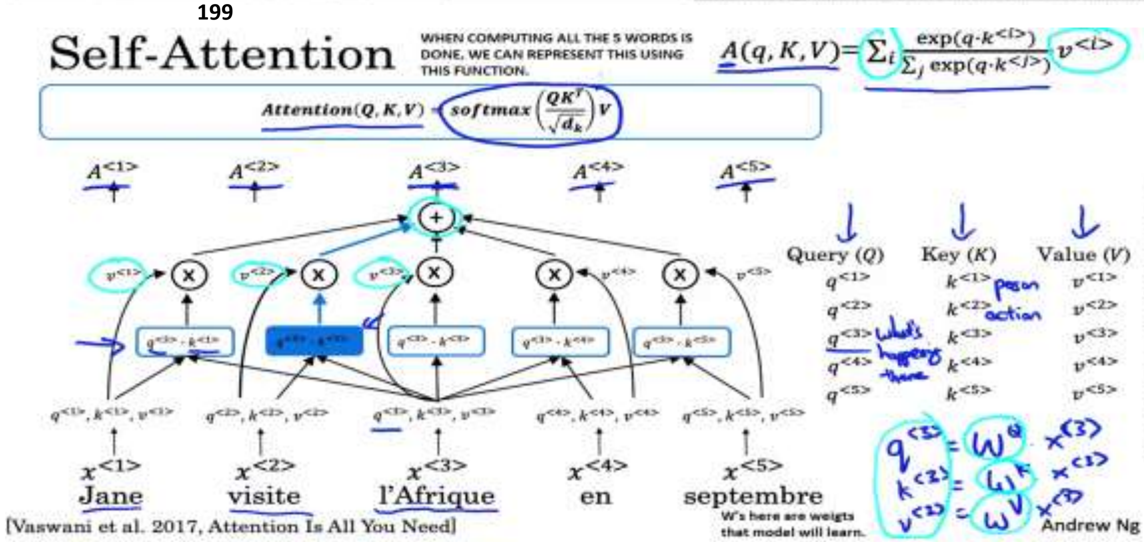


191.2



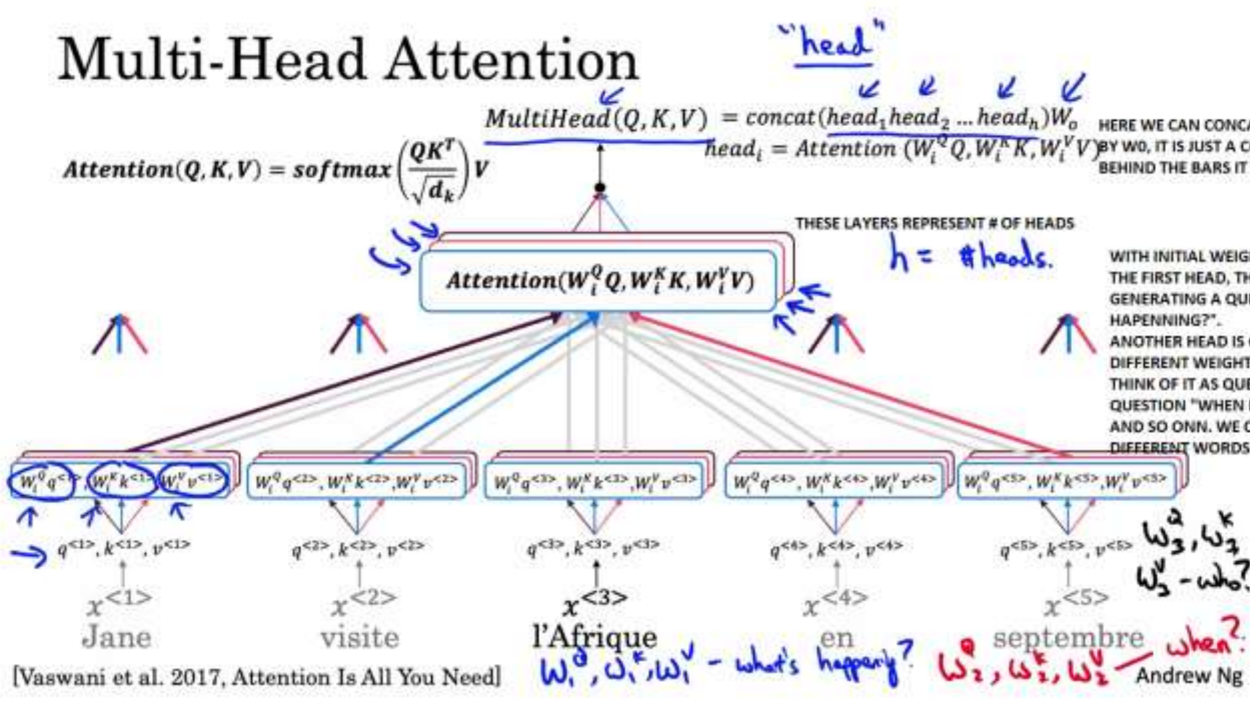


199



FOR WORD L'AFRIQUE THE MODEL WILL FIRST GENERATE A QUERY<3> "WHAT'S HAPPENING THERE". THEN THE QUERY IS MULITIPLIED BY EACH WORD'S k VALUE. IF k<1> IS REPRESENTING "PERSON" AND k<2> IS REPRESENTING "ACTION" THEN WHEN q<3> IS MULITPLIED BY k<1>, THEN WE WILL FIND THAT K<2> HAS THE LARGEST VALUE AS COMAPRED TO OTHERS.
AFTER THIS WE TAKE THE SOFTMAX VALUES AND MULTIPLY IT WITH THE PRODUCT OF q & k.
THEN WE SUM UP ALL THE VALUES THAT WE GOT BY MULTIPLYING THE q & k AND THAT WILL BE OUT A<3> WHICH IS EQUIVALENT TO A(q<3>,K,V)

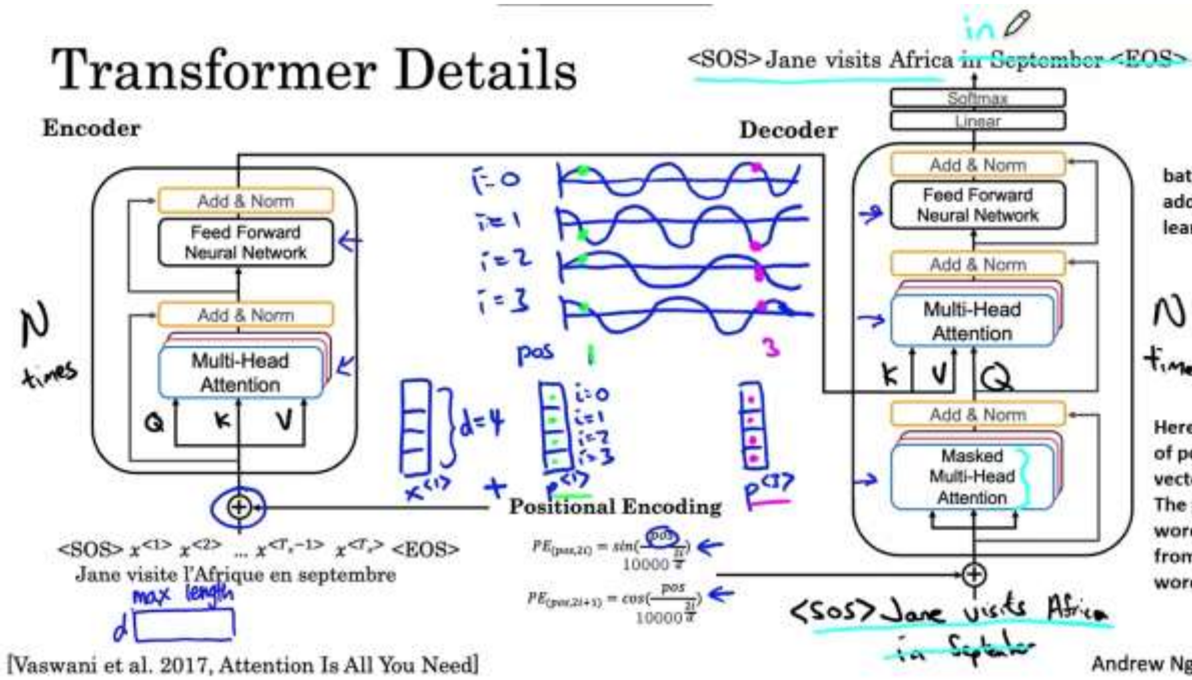
200



HERE WE CAN CONCATENATE AND MULTIPLY BY W0, IT IS JUST A CONCEPTUAL WAY, BEHIND THE BARS IT IS COMUTING PARALLEL

WITH INITIAL WEIGHTS WE WILL CALCUALTE THE FIRST HEAD, THINK OF IT AS QUERY GENERATING A QUESTION "WHAT'S HAPENNING?".
ANOTHER HEAD IS CALCULATED USING DIFFERENT WEIGHTS LEARNED BY MODEL, THINK OF IT AS QUERY GENERATING A QUESTION "WHEN IT IS SMTH HAPENING?" AND SO ONN. WE CALCUALTE THIS FOR DIFFERENT WORDS.

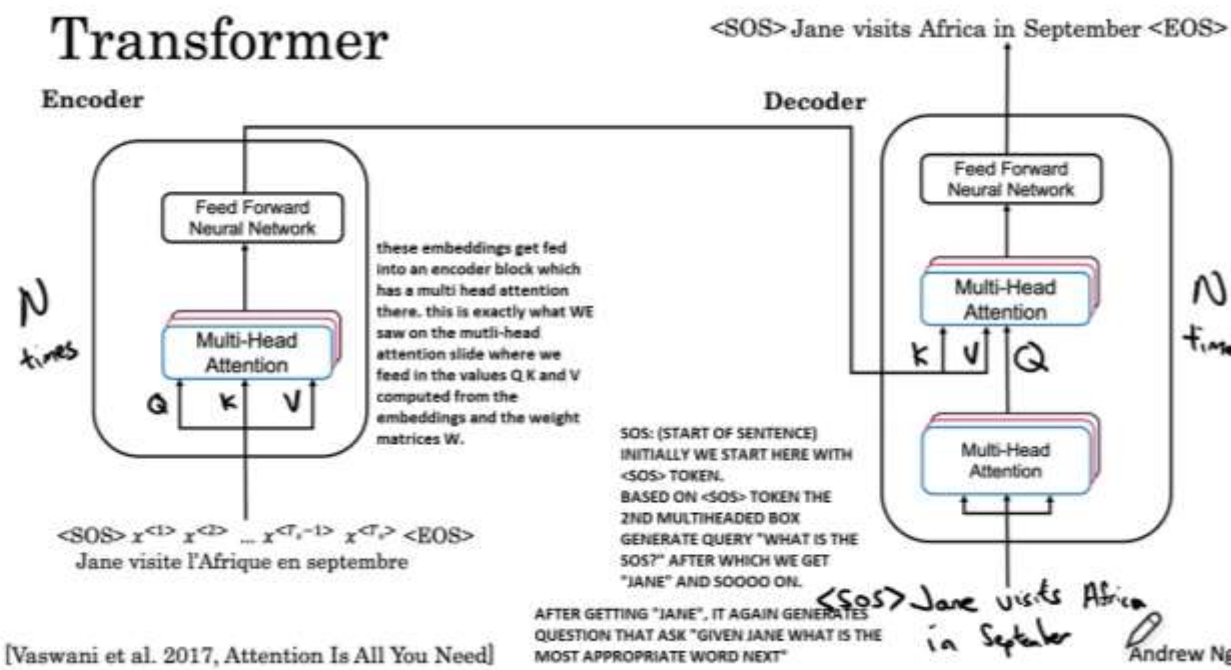
201.1



batch norm layer is added to speed up learning

Here pos is position , i is index of pos, d is the embedded vector.
The positional vector p<1> of word "JANE" will be different from positional vector p<3> of word "L'AFRIQUE".

201



these embeddings get fed into an encoder block which has a multi head attention there. this is exactly what WE saw on the mutli-head attention slide where we feed in the values Q K and V computed from the embeddings and the weight matrices W.

SOS: (START OF SENTENCE)
INITIALLY WE START HERE WITH <SOS> TOKEN.
BASED ON <SOS> TOKEN THE 2ND MULTIHEADED BOX GENERATE QUERY "WHAT IS THE SOS?" AFTER WHICH WE GET "JANE" AND SOOOO ON.

AFTER GETTING "JANE", IT AGAIN GENERATES QUESTION THAT ASK "GIVEN JANE WHAT IS THE MOST APPROPRIATE WORD NEXT"