SUMMARY OF COURSERA COURSE

CONVOLUTIONAL NEURAL NETWORK

RATINGS: 5/5

WEEK 1 – FOUNDATIONS OF CONVOLUTIONAL NEURAL NETWORKS

UNIT 1: Computer Vision

- Computer vision has been advancing rapidly thanks to Deep Learning
- Advance in Computer Vision is leading to more inventions
- Computer Vision Problems: Image Classification, Object Detection, Neural Style Transfer (combining images into one)
- In CV the input features can be very large, the computational cost is much
- To help with this we need to implement Convolutional Neural Network (CNN)

UNIT 2: Edge Detection Example

- The convolution operation is one of the fundamental building blocks of a CNN
- Convolution: it applies a "filter" (kernel) to the image
- Most Deep Learning Frameworks that have good support for CV have a function for "convolution"
- A vertical edge is a 3x3 region where there are bright pixels on the left and dark pixels on the right (based on the example filter applied)

UNIT 3: More Edge Detection

- A horizontal edge is a 3x3 region where there are bright pixels on the left and dark pixels on the right (based on the example filter applied)
- Different filters help you to find the vertical and horizontal edges
- "sobel filter", "scharr filter"
- We could learn the values of the filter by treating the values as parameters to be learned by algorithm, this is one of the most powerful ideas in Deep Learning

UNIT 4: Padding

- One modification to the basic convolutional operation in order to build Deep NN is "padding"
- (n x n: input) convolve (f x f: filter) = (n-f+1 x n-f+1: output)
- Problems with basic convolution: shrinking output, throwing away info from edge
- "Padding" the image before Convolution helps to solve these problems
- Valid Convolution: No padding
- Same Convolution: padding is applied do that the output size is same with the input size
- Dimensions of filter are usually "odd" in CV

UNIT 5: Strided Convolutions

- Strided Convolution is another piece of the basic building block of convolutions as used in CNN
- It determines the number of step the "filter" takes on the input images
- (n x n: input) convolve (f x f: filter), using padding = p, stride = s, output = ((n+2p-f)/s)+1 **x** ((n+2p-f)/s)+1
- We don't bother flipping the "filter" in Deep Learning
- Mathematician call it "cross-correlation" and call it "convolution" only if we flip the "filter" before applying it

UNIT 6: Strided Convolution Correction

UNIT 7: Convolutions over volumes

- To convolve 3D input we also use a 3D "filter" which gives a 2D output
- The "filter" is expected to have the same number of channels with the input
- We can use multiple "filters" we will just stack the output of each filter together
- (n x n x nc: input) convolve (f x f x nc: filter), using padding = p, stride = s, output = ((n+2p-f)/s)+1 **x** ((n+2p-f)/s)+1 x numberofchannels
- People refer to "channel" also as "depth"

UNIT 8: One Layer of a Convolutional Network

- Convolution does the linear function
- The number parameter remains dependent on the filter no matter the size of the input
- Number of filter determine the number of channels of the output

UNIT 9: Simple Convolutional Network Example

- Flatten/unroll our output dimensions and feed it to our model to make prediction for our output networks
- Most of work in CNN is selecting hyper parameters like stride, padding, no of filters
- The number of channels usually increase as we go deeper in our CNN
- Types of Layer in a CNN: Convolution, Pooling, Fully Connected (FC)
- You can build a CNN with only Convolution layers

UNIT 10: Simple CNN Example Correction

UNIT 11: Pooling Layers

- CNN often use "pooling layers" to reduce the size of the representation, to speed the computation
- "Max Pooling": takes the max value in a "region" determined by its hyper parameters "f" similar to filter size and the "s" stride, it has no parameters to learn
- The "max pooling" is done independently on each "channel"

- "Average Pooling": takes the average value, it is used less often compared to "max pooling"
- Hyper parameters values of f=2/3 and s=2 are usually used
- Number of input channel is equal to the number of output channel

UNIT 12: CNN Example

- The "Convolution + Pooling layers" are referred to as a "layer", sometimes they are referred to as different layers
- "Fully Connected Layer": is similar to a single layer of a neural network
- "Conv > Pool > Conv > Pool > FC > FC > FC > Softmax": is common arrangement for CNNs
- The activation size usually drops as it goes deeper in CNN

UNIT 13: CNN Example Correction

UNIT 14: Why Convolutions

- Advantages of using "Convolution Layers" are: Parameter Sharing and Sparsity of connections
- Parameter Sharing: a feature detector (such as vertical edge detector) that is useful in one part of the image is probably useful in another part of the image, thus reducing the number of parameters
- Sparsity of connections: in each layer, each output value depends only on a small number of inputs, makes it more robust to capture "translation invariance"
- Most people just use an Architecture someone used/built and apply it

UNIT 15: Why Convolutions Corrections

UNIT 16: Quiz

UNIT 17: Programming Assignment – Step by Step

UNIT 18: Programming Assignment – Application

UNIT 19: Yann LeCun Interview

- In 2012 during a workshop a lot of people were convinced about CNN
- You can learn a lot of online resources
- Make yourself useful, contribute to open source

WEEK 2 – DEEP CONVOLUTIONAL MODELS: CASE STUDIES

UNIT 1: Why look at case studies?

- Looking at case studies helps to gain intuition
- Read some Computer Vision papers

UNIT 2: Classic Networks

- Classic NN: LeNet-5, AlexNet, VGGNet
- "LeNet-5": it was trained to recognized hand digit, it used 5x5 filters, it used convolution and pooling simultaneously, then Fully Connected Layer, it had about 60,000 parameters, it used sigmoid/ tanH activation functions as we go deeper into the NN the width and height reduced but the channel increased
- LeNet-5 paper: "LeCun et all, 1998", focus on section 2 and 3
- "AlexNet": similar to LeNet-5 but much bigger, it had about 60 million parameters, it used ReLu activation functions, "Krizhevsky et al, 2012", this paper helped Deep NN to breakout
- "VGG-16": "Simonuyan & Zisserman 2015", is a very deep NN, it has 16 layers that have weights, it has about 138 million parameters, the NN is relatively uniform, it does almost has well as VGG-19

UNIT 3: ResNets

- Very deep NN are difficult to train because of vanishing and exploding Gradient Descent
- Resident NN help to fix this and allow us to train Deep NN (>= 150 layers)
- Residual NN use residual blocks which operate on "shortcuts/ skip connection" which are applied after the linear function before the non-linear activation functions
- In reality our training error get worse for very Deep NN but with ResNet we can train deep NN without worrying about this
- "He et al, 2015"

UNIT 4: Why ResNets Work

- Doing well on a train is a prerequisite to doing well on our dev/test set
- Adding "residual block" to a NN does not hurt the NN
- Identity function is easy for Residual Block to learn
- In ResNets we use a lot "same convolutions" so that dimensions can be preserved to allow shortcuts, skip connections to work effectively
- To turn a plain NN to a ResNets we adding "shortcuts/ skip connections" to form residual blocks

UNIT 5: Networks in Networks and 1 x 1 Convolutions

- One by one convolution is one of the ideas that really help in terms of designing content architectures, it simply adds non-linearity to the network, it is useful for Inception Networks
- "Lin et al, 2013", it is also called Network in network
- If you want shrink the number of channel you can apply a 1 x 1 convolution
- Pooling layers are used to shrink width and height

UNIT 6: Inception Network Motivation

- "Szegedy et al, 2014"
- Instead of wondering of to use Convolution, type of filter, pooling layer, the Inception Network does them all and concatenate the results in Inception Block/Module
- To use "max pool" in Inception Block/Module you need add padding in order to make the dimension match
- We use 1 x 1 convolution to shrink volume (Bottleneck Layer) before applying our desired filter, it helps to reduce computational cost

UNIT 7: Inception Network Motivation Correction

UNIT 8: Inception Network

- An Inception Network simply a lot of Inception Block/Module repeated
- "gooLeNet Network"
- The movie "Inception" was a motivation for the Inception with the quote "we need to go deeper"

UNIT 9: Using Open-Source Implementation

- It is sometime difficult to replicate someone else work from only the Research Paper
- But it is easy to implement if it in Open Source on GitHub
- Starting with Open Source is a better/faster way to start on a projects

UNIT 10: Transfer Learning

- If you are building a CV application rather than training the ways from scratch, you often make much faster progress if you download ways that someone else has already trained on the network architecture and use that as pertaining and transfer that to a new task that you might be interested in
- Download Open Source Implementation (both code and weight) use as initialization
- Most framework have capability for Transfer Learning like "freezing some layers from being retrained", you usually have to change the output unit to your task
- You can pre-compute or save to disk
- If you have small dataset freeze most layers and train less layers, but if you have more data freeze less layers and train more layer
- It is very worth considering

UNIT 11: Data Augmentation

- Most CV task could use more data, so Data Augmentation is one of the techniques that is often used to improve the performance
- Common Augmentation Method: Mirroring, Random Cropping, Rotation, Sheering, Local Warp, Color Shift (distorting the color channels)
- PCA is used for color distortion details in the "AlexNet Paper", also called "PCA color augmentation"

- It is good practice to use Multi thread processing, one to load data and apply distortion another for the training
- Data Augmentation also takes some parameters

UNIT 12: State of Computer Vision

- Decent amount of data for Speech Recognition, but CV seems to still need more data cause of the Architecture
- Less data for Object Detection than for Image Recognition
- With large data you can implement simpler algorithms and less hand engineering
- With little data you have to implement more hand engineering of features/network architecture/other components ("hacks")
- Transfer learning helps a lot
- Tips for doing well on benchmarks/ winning competitions: Ensemble (average their outputs, it is computationally expensive), Multi-crop at test time (Run classifier on, multiple versions of test images and average results)
- He does not use them when building for production
- Use architectures of networks published in the literature, use open source implementations if possible, use pertained models and fine-tune on your dataset

UNIT 13: Quiz

UNIT 14: Programming Assignment (Optional) – Keras

- Keras is a high-level framework, it is higher than Tensorflow
- However, Keras is more restrictive than lower-level frameworks like Tensorflow, so some very complex models can only be implemented in Tensorflow and not Keras
- Keras will work fine for many common models
- 4 steps in Keras: Create > Compile > Fit/Train > Evaluate/Test

UNIT 15: Programming Assignment – Residual Networks

WEEK 3 – OBJECT DETECTION

UNIT 1: Object Localization

- Objection detection is one of the areas of CV that is exploding
- In "Classification with Localization" there is only one objects to classify and localize but in "Detection" we can have multiple objects
- Modify the NN to output a "bounding box" (bx, by, bw, bh)

UNIT 2: Landmark Detection

- You can make a NN to output just the position of certain points (landmark) on an image (like the corner of the eyes)

- Landmark detection is a key building block for emotion detection, face filters common to social media platforms, detecting posture
- Landmark points must be consistent throughout all the images

UNIT 3: Object Detection

- Sliding Windows Detection: take part of the image (sliding window) and pass it to a CNN trained on closely cropped image to detect object, this might be repeat for increasing sliding window size
- It is very computational expensive, and if you increase the stride to reduce the computational cost but it will hurt performance
- This computational expense can be fixed

UNIT 4: Convolutional Implementation of Sliding Windows

- We can view a Fully Connected layers as "1 x 1 x #rows" by implementing them with filters
- "Sermanet et al, 2014"
- We do a convolution implementation of Sliding Windows to save computational cost, it combines all the computations into one process instead of doing it sequentially and output result for each sliding window

UNIT 5: Convolution Implementation of Sliding Windows CORRECTION

UNIT 6: Bounding Box Predictions

- Convolutional implementation of sliding window is more computational efficient but it does not quite output the most accurate bounding boxes
- "YOLO Algorithm": "You only look once", Redmon et al, 2015, it helps to fix this problem of inaccurate bounding boxes
- It divides the image into grid, then carries out sliding window on each grid
- With YOLO algorithm the NN outputs more accurate bounding box
- We might need to use a finer grid to prevent a grid from having multiple objects in a grid
- The YOLO algorithm also uses a "convolutional implementation", making it run quite fast
- "bx, by, bw, bh" are specified relative to the Grid cell the object is located so bx and by have to be between 0 and 1, but bw, bh can be greater than 1
- YOLO paper might be a bit complex

UNIT 7: Intersection Over Union

- "Intersection Over Union" function can be used to evaluate our object detection algorithm; it computes the intersection over union of two bounding boxes, evaluating object localization
- The algorithm is working well if the IOU >= 0.5

UNIT 8: Non-max Suppression

- One of the problems of Object Detection is that our algorithm may find multiple detections of the same objects, "Non Max Suppression" helps to prevent this by making the algorithm detect an object just once
- Non Max Suppression helps to clean up multiple detection, discard all boxes <= a certain Pc threshold > checks and select the box with highest Pc (probability of object) > suppress all those that have high IOU relative to the selected box > then select the box with the highest Pc

UNIT 9: Anchor Boxes

- One of the problems with Object detection so far is that each grid cell can detect only one object
- Using the idea of "Anchor Boxes" helps the grid cell to detect two objects
- Associate the predictions to Anchor Boxes
- Check which of the anchor boxes in a grid cell has a higher IOU, to determine the object type
- If two objects with same anchor boxes are found in the same Grid Cell the algorithm will not work well
- Sometimes people choose their anchor boxes by hand, you can use some algorithms

UNIT 10: YOLO Algorithm

- Our label "y" = grid size x grid size x no of anchor boxes x (5 + no of classes)
- For the example above: Input Image > CNN > Output (of shape y above)
- Outputting the Non-Max suppressed outputs: for each grid cell get 2 predicted bounding boxes > Get rid of low probability predictions > for each class use non-max suppression to generate final predictions
- It is one of the most effective Objection Detection algorithms

UNIT 11: YOLO algorithm Correction

UNIT 12: Region Proposals (Optional)

- Regions Proposals are becoming influential in CV, he rarely uses it
- Girshik et al, 2013
- R-CNN: Region with CNN, tries to pick some regions on the image to run CNN on instead of running it on the whole image
- It uses a "segmentation algorithms" to determine regions to run CNN on
- The original R-CNN was quite slow but with the use of Convolution implementation it was made to work faster
- "Girshik, 2015 Fast R-CNN"; "Ren et al, 2016 Faster R-CNN"

UNIT 13: Quiz

UNIT 14: Programing Assignment – Car Detection

WEEK 4 – SPECIAL APPLICATIONS: FACE RECOGNITION AND NEURAL STYLE TRANSFER

UNIT 1: What is Face Recognition

- Face Verification: Input > Image, name/ID, Output > whether the input image is that of the claimed person
- Face Recognition: has a database of K persons, Input an image, Output ID if the image is any of the K persons

UNIT 2: One Shot Learning

- It is one of the challenges of face recognition, it means you have to recognize a person with just one single image
- To solve one shot learning problems you need to learn a "similarity" function, "degree of difference between images"

UNIT 3: Siamese Network

- "Tiagman et al, 2014"
- Comparing the "encoding of input images" is usually called Siamese Network
- If the images are the same person you want the "distance between the encodings" to be small and vice versa

UNIT 4: Triplet Loss

- One way to learn the parameters of the NN so that it gives you a good encoding for your pictures of faces is to define an applied GD on the triplet loss function
- "Schroff et al, 2015"
- You be looking at an "Anchor", "Positive", "Negative" images at once, you want the encoding between anchor and positive to be small, and you want it to be less than/equal to the encoding between anchor and negative
- You need a training dataset that has different picture of the same person so we can have pairs of Anchor and Positive images
- Choose triplets that are hard to train on
- It is sometimes useful to use someone pre trained model

UNIT 5: Triplet Loss Correction

UNIT 6: Face Verification and Binary Classification

- Alternative to triplet loss is to train the face verification system as a binary classification problem
- So you create a data set of two images instead of three like in triplet loss and output if they are the same or not

UNIT 7: Face Verification and Binary Classification Correction

UNIT 8: What is Neural Style Transfer

- Neural style transfer allows you to generate your own artwork

UNIT 9: What are deep ConvNets Learning?

- "Zeiler and Fergus, 2013"
- In shallow layers the units extract simple features from the image (edges, color) but the deeper we go the more complex features are extracted from the image

UNIT 10: Cost Function

- Cost function for a Neural Style Transfer
- A cost function that measures how good the "generated image" is
- "Gatys et al, 2015"
- Updating the pixel values of the generated image through gradient descent which tries to minimize the cost function

UNIT 11: Content Cost Function

- Find the similarity between the content image and the generated
- It computes the elementwise sum of square difference between the activation of the content and the activation of the generated image

UNIT 12: Style Cost Correction

UNIT 13: Style Cost Function

- "Gatys et al, 2015"
- Style is defined as correlation between activations across channels
- Compute style for both style image and the generated image
- Style matrix are also called the "Gram Matrix"
- It is best to take style cost function across multiple layers

UNIT 14: 1D and 2D Generalizations

- Most of the ideas we have learnt so far can be applied to 1D and 3D data
- Example of a 1D data is an EKG of someone's Heart Beat
- For most 1D data we use "Recurrent NN"
- CT and CAT Scan produces 3D Images showcasing different slices of the human body
- If the input is 1D/3D the filter is also 1D/3D
- Remember the channels of the filter must match does of the input image
- Vast majority of data in CNN are 2D

UNIT 15: Quiz

UNIT 16: Programming Assignment – Art generation with Neural Style Transfer

UNIT 17: Programming Assignment – Face Recognition