**STUDY ON YOLO ARCHITECTURE:**

**YOLO:**

YOLO, short for You Only Look Once is a convolutional neural network architecture designed for the purpose of object detection.

Ref.: <https://medium.com/adventures-with-deep-learning/yolo-v1-part-1-cfb47135f81f>

**CONVOLUTIONAL NEURAL NETWORK (CNN/ ConvNet):**

1. A CNN/ ConvNet is a Deep Learning algorithm which can take in an input image, assign importance (**learnable weights and biases**) to various aspects/objects in the image and be able to differentiate one from the other.
2. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.
3. A CNN sequence:



1. INPUT image: we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. These images can also be Grayscale, RGB, HSV, CMYK, etc. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.
2. Convolution + RELU Layer : The Kernel: Say the image I is of Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB) with Kernel/Filter, K which is 3x3x1 matrix and the Kernel shifts 9 times because of Stride Length = 1 , every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.
3. The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset.
4. There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.
5. Pooling Layer: The Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.
6. Flatten: After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.
7. Classification by Fully Connected Layer : The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique(multiclass).

Ref.: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

**OBJECT DETECTION:**

1. Object detection is the problem of localization and classifying a specific object in an image which consists of multiple objects.
2. We frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Ref.: Paper: <https://arxiv.org/pdf/1506.02640.pdf>

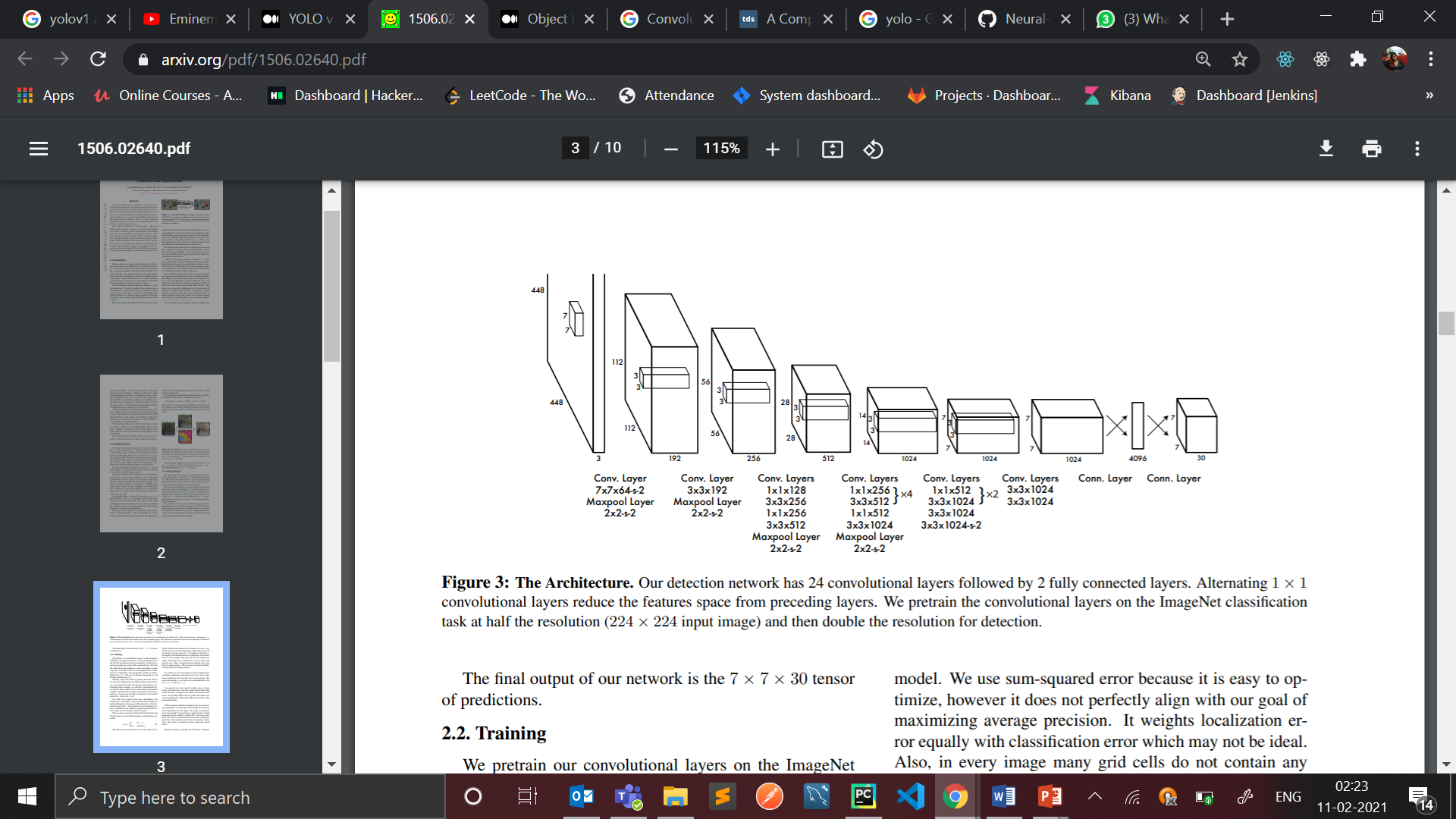
**TRADITIONAL OBJECT DETECTION:**

1. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image.
2. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image.
3. Approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene.
4. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

**WHY YOLO?**

1. YOLO reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.
2. YOLO is simple. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.
3. YOLO is extremely fast. Since we frame detection as a regression problem we don’t need a complex pipeline. We simply run our neural network on a new image at test time to predict detections.
4. YOLO achieves more than twice the mean average precision of other real-time systems.
5. YOLO performs the convolution on the whole image rather than sections of it due to which it encodes contextual information about the classes and their appearances. It makes less mistakes in predicting background patches as objects as it views the entire image and reasons globally rather than locally.
6. YOLO learns generalizable representations of objects due to which it can be applied to new domains and unexpected inputs without breaking.
7. YOLO performs unified detection by unifying the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

**NETWORK DESIGN:**



1. YOLO is implemented as a convolutional neural network and evaluated on the PASCAL VOC detection dataset.
2. The input is 448 x 448 image and the output is the class prediction of the object enclosed in the bounding box.
3. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.
4. The network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, and uses 1 × 1 reduction layers followed by 3 × 3 convolutional layers.
5. First 20 convolutional layers with leaky ReLU activation followed by an average pooling layer and a fully connected layer is pre-trained on the ImageNet 1000-class classification dataset.
6. The pretraining for classification is performed on dataset with resolution 224 x 224.
7. The layers comprise of 1x1 reduction layers and 3x3 convolutional layers
8. Last 4 convolutional layers with leaky ReLU activation followed by 2 fully connected layers are added to train the network for object detection
9. Object detection requires more granular detail hence the resolution of the dataset is bumped to 448 x 448.
10. The final layer predicts the class probabilities and bounding boxes. The final layer uses a linear activation.

**WORKING:**

1. The input image fed to the network is divided into a grid of dimension say S x S. Each grid cell is responsible for identifying whether it contains the center of an object belonging to any of the 20 classes. If the grid cell contains the center of the object it predicts say B bounding boxes to enclose the object. In addition to producing B bounding boxes, every grid cell is also responsible for producing confidence scores for each bounding box. Bounding boxes are a mechanism for localising objects in an image or video.
2. pc = Po x IOU = Confidence score of the box: Confidence score of a bounding box tells us how confident is the model in predicting the object and how accurate it thinks the bounding box that it created is. ‘pc’ is computed by multiplying the probability of the object being present in the box(Po) and the intersection over union (IOU) between the predicted box and the ground truth.
3. bx and by = coordinates of the centre of the object: The center of the object detected by the grid cell has an x and y coordinate measured from the top left corner of the grid cell which is the origin.
4. bh and bw = height and width of the bounding box: The height and width of the predicted bounding box by the grid cell mesured from the origin of the cell.
5. Probabilities of the type of object present in the bounding box denoted as Pr(class|object).
6. Grid cell can produce ‘C’ class probabilities as the network is trained to identify these ‘C’ classes however only 1 set of class probabilities is predicted per cell.
7. It cannot predict multiple objects eg a grid cell cannot predict to have a cat and a dog in it at the same time. This is one major limitation of YOLO v1. It is unable to localise and identify more than 1 object per grid cell.
8. The reason that a grid cell only predicts 1 class probability is due to the fact that fully connected layers are used to regress bounding boxes. One regression header can only regress a bounding box for 1 class object at a time. This means that a model using only 1 fully connected layer (1 regression head) will be unable to produce accurate results if there exists multiple objects of different classes anywhere in the image. The single layer will struggle to regress the bounding box due to interference of the two distinct class objects trying to use the same layer.
9. This problem can be partially fixed by training 1 regressor head for every class that means 20 regressor heads for 20 classes. This will direct each class object to a fully connected layer each. Thus, this does solve the problem of interference faced by a single fully connected layer due to distinct class objects.
10. There still exists the problem of interference due to multiple instances of the same class for example, if the image has many dogs. Since, the class dog is assigned only 1 fully connected layer to regress the bounding box, itwill face problems as many dogs will try to share the same layer.
11. This still leaves us with the problem of spatial interference of boxes for multiple objects of the same class which reduces accuracy for localisation. The concept of anchor boxes is used to predict multiple objects per grid cell. This is used by YOLO v2.
12. The solution is to train each fully connected layer regression head to consider only a limited region in the image rather than considering the entire image which is the cause of interference. To achieve this anchor boxes are used. When there is a high IOU of the ground truth bounding box and an anchor box, the regression head associated with that anchor box is given the responsibility of regressing the final bounding box. Hence, the anchor box limits the region considered by a regressor head. This results in less interferences due to objects present outside the region. There still exists the problem of detecting multiple objects of the same class with very high overlap with each other.

**LOSS FUNCTION:**

1. The function is a composition of multiple SSEs. During training, this loss function is optimized to improve the predictions of the network. SSE has a benefit over other loss functions as it is easier to use and optimize.
2. Differential weights are used for confidence predictions from boxes that contain objects and boxes that are empty during training. This means that the loss function only penalizes classification error for the boxes that contain an object within themselves. It doesn’t penalize empty boxes. Thus, the conditional probability of an object being present in a box plays an important role in deciding which bounding boxes factor in the loss function.
3. The loss function weighs both the localization and classification errors equally which is not ideal.
4. The loss function equally weighs the errors in large bounding boxes and small bounding boxes. This is because the sum of squared errors penalizes the deviation independent of the size of the predicted bounding box.