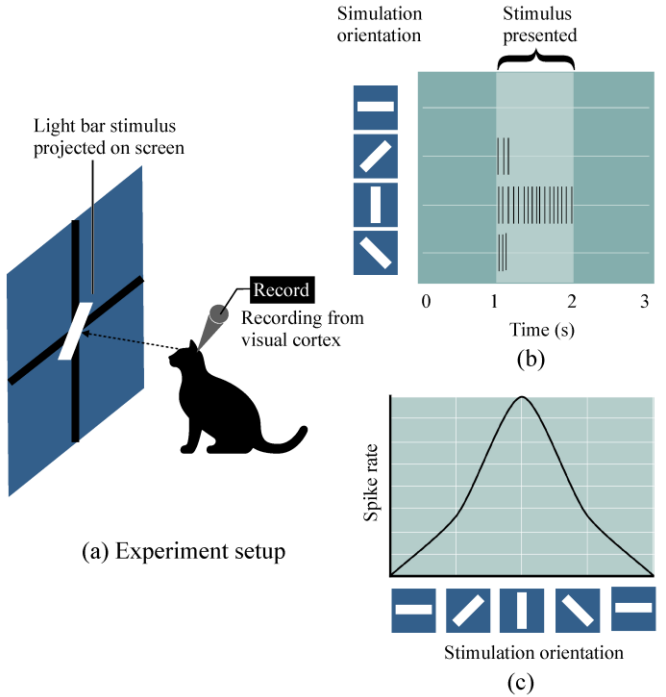
Introduction to Computer Vision and Object Detection (YOLO)

Motivation

- Hubel and Weisel's research discovered that neurons in the primary visual cortex respond to oriented edges



Human Vision

Input Image

Imaging

Process the Input
(Brain)

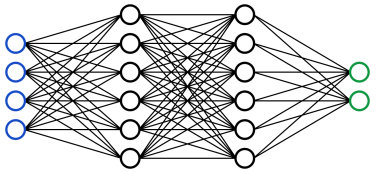
It’s your cat
(hiding under the bed)
(afraid of lightning)

Output

Computer Vision

Input Image

Imaging



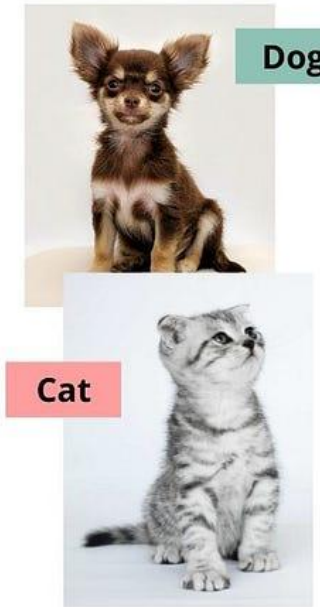
Process the Input
(CPU or GPU)

Cat

Output

Few Types of Computer Vision

Image Classification

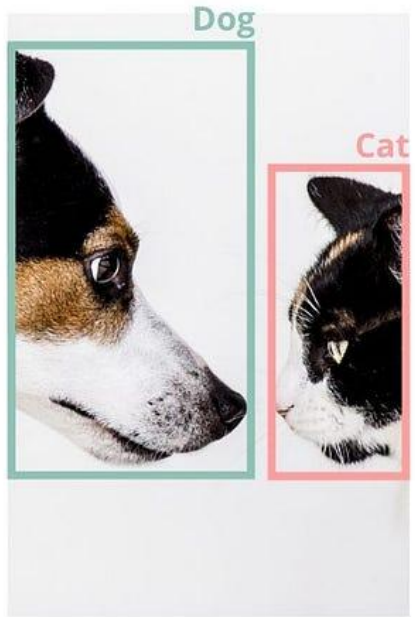


Classify the subject

Object Localisation

Find position of the subject

Object Detection



Classify all objects
in the image

Image Segmentation

Dog, Cat

Separate different sections
in the image

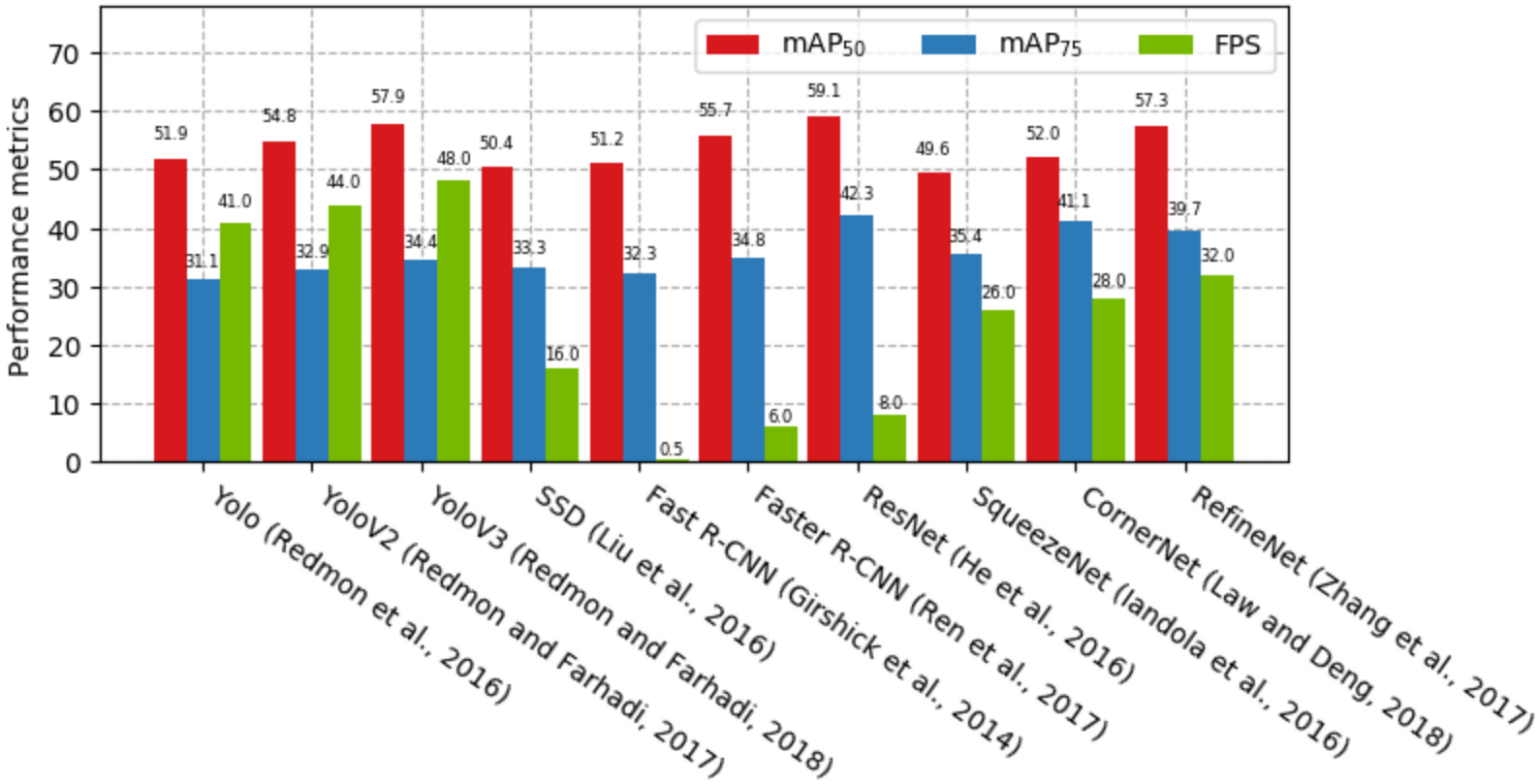
Other types include: Facial recognition, expression detection, image restoration etc.

Image Source: medium.com (Ekaterina Zagarniuk)

Different Computer Vision Models

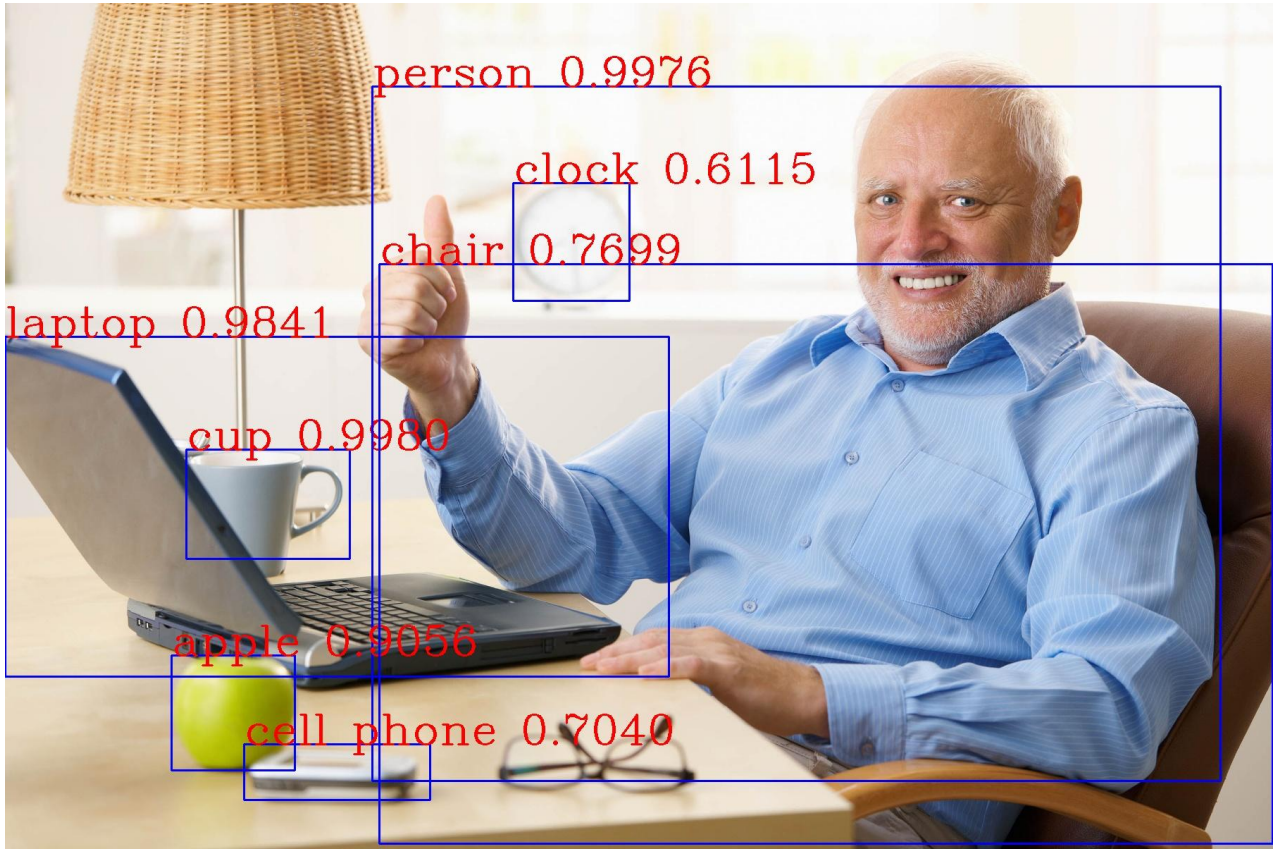
- YOLO (You Only Look Once): Speed and accuracy in object detection tasks
- VGG (Visual Geometry Group): Commonly used for image classification tasks
- ResNet (Residual Network): Often used for image classification, object detection, and image segmentation tasks
- Inception (GoogLeNet): Good for image classification tasks and has been influential in the development of other architectures.
- MobileNet: A lightweight convolutional neural network architecture, suitable for image classification and object detection on mobile devices.

Why we chose YOLO?



Currently, YOLO has the best trade off between speed and accuracy when it comes to object detection

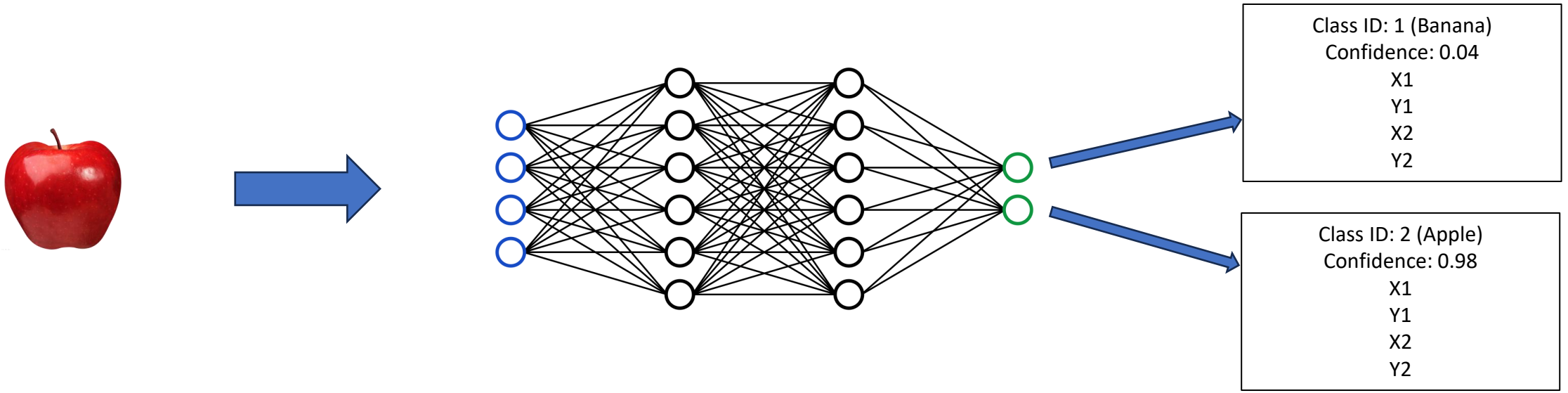
Object Detection - YOLOv3 - Example



Hide the Pain Harold

You can see lower confidence values for blurred out objects (because of bad edge detection)

How does it work?



You can search for a pretrained model on GitHub, clone the repository and start using it right away to detect objects

- You input the image to the model, and it will output 5 values
  - Confidence value
  - Depending upon the confidence value of the final neuron, map it to its class (Eg. Apple)
  - Coordinates (center of bounding box): X,Y
  - Width and Height of the bounding box: W, H

Under the hood

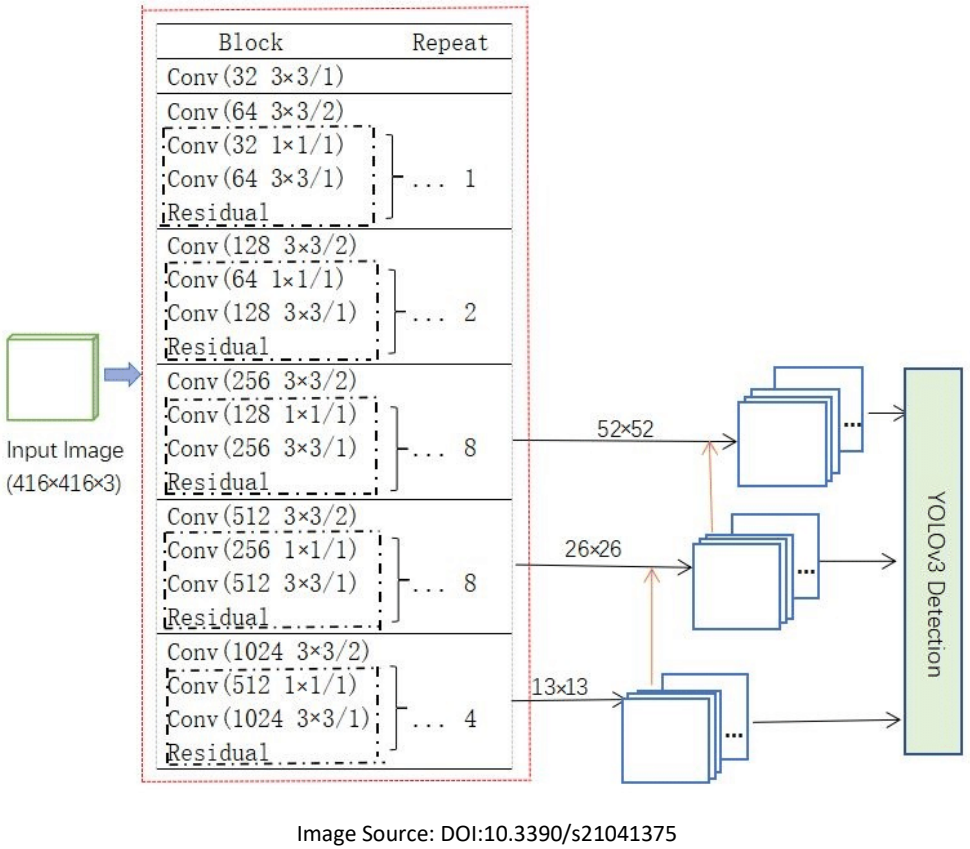
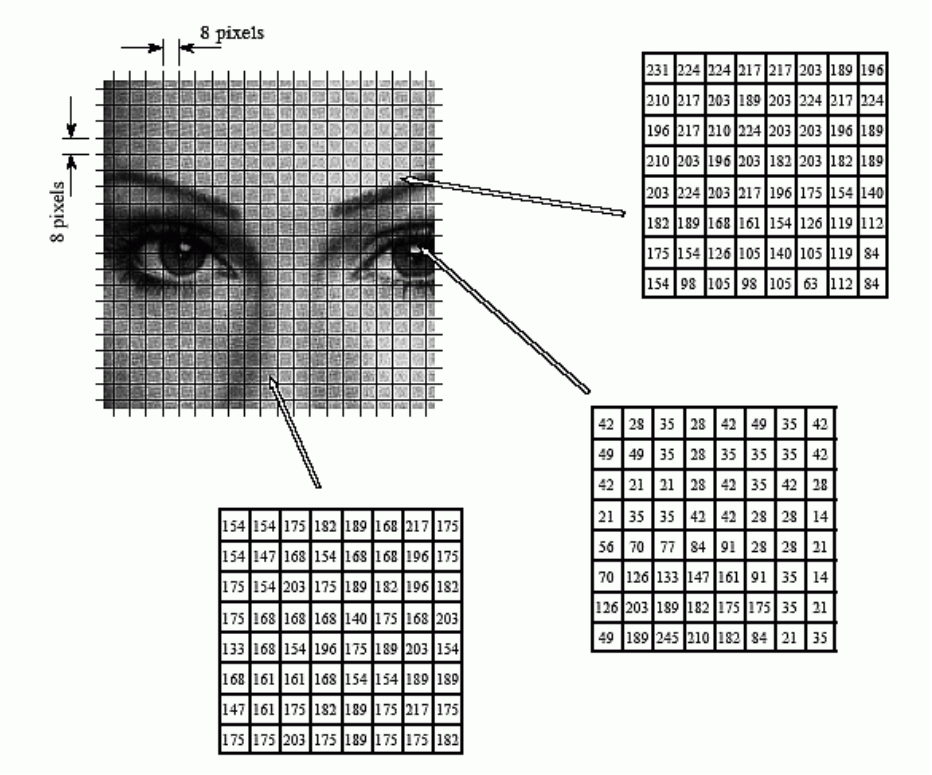


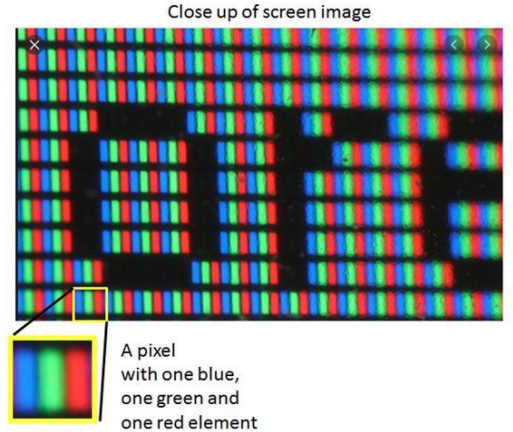
Image Source: DOI:10.3390/s21041375

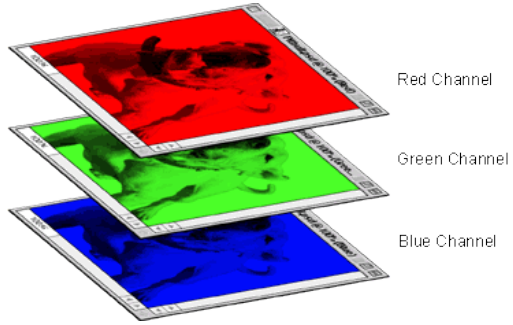
Layers used:
Convolutional
Batch Normalization
Leaky ReLU Activation
Fully Connected (Removed after YOLOv1)
YOLO Layer (YOLOv2 and above)

It's a lot to take in, what do these different layers even mean?

Input Layer







- An image is just a bunch of numbers
- A greyscale image will have just a single set of values, which would be brightness, like in the first example
- But a color image will have three values, one for red, one for green and one for blue. We can separate those into different channels and then input them into the neural network

Convolution layer

- Convolutional Layers are used to extract features (called feature maps) from our input image
- The first few layers are used to extract low level features such as edges, corners, outlines etc.
- The deeper layers are used to extract intricate features like eyes, ears, mouth nose etc.

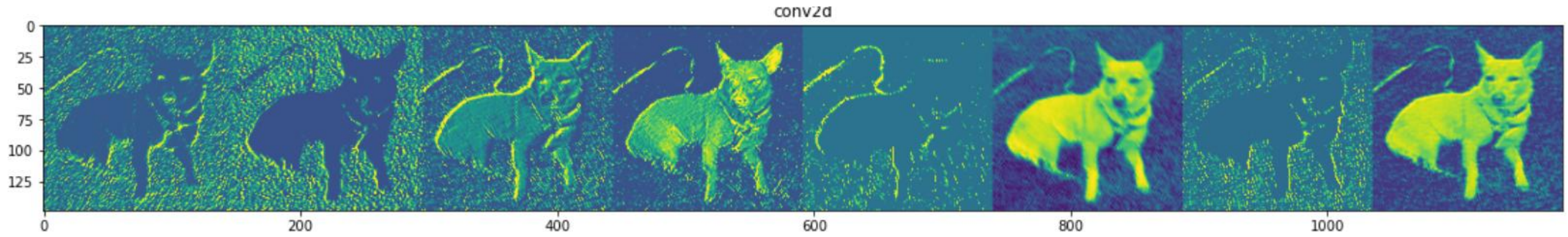
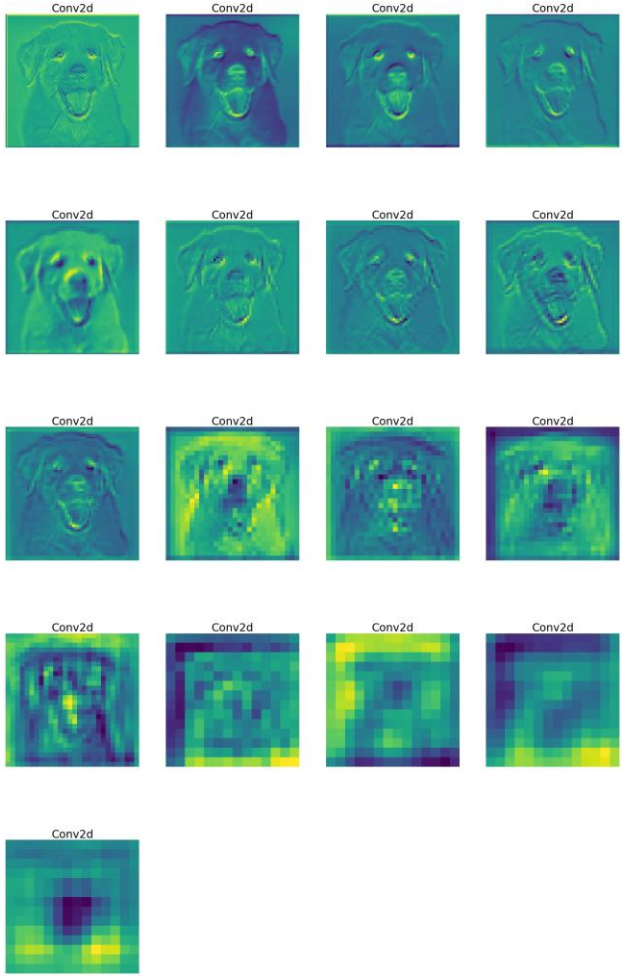


Image Source: analyticsvidhya.com

Convolutional Layers Example

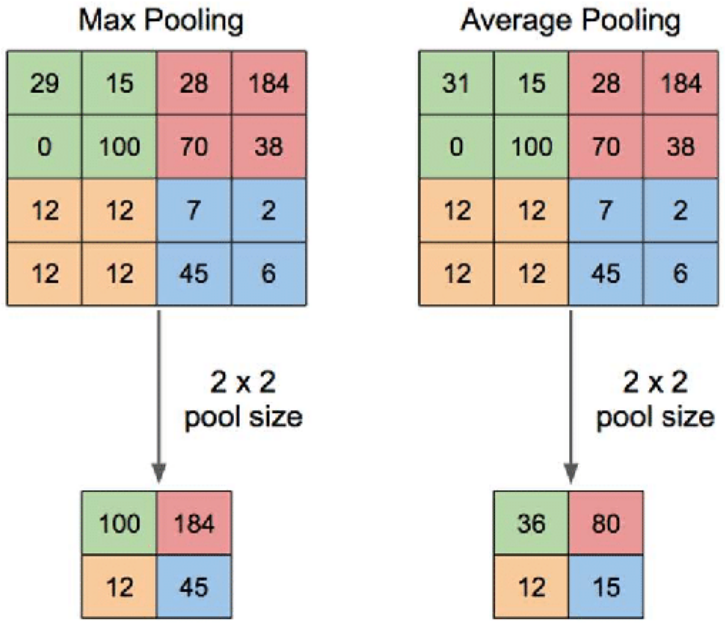
Input Image

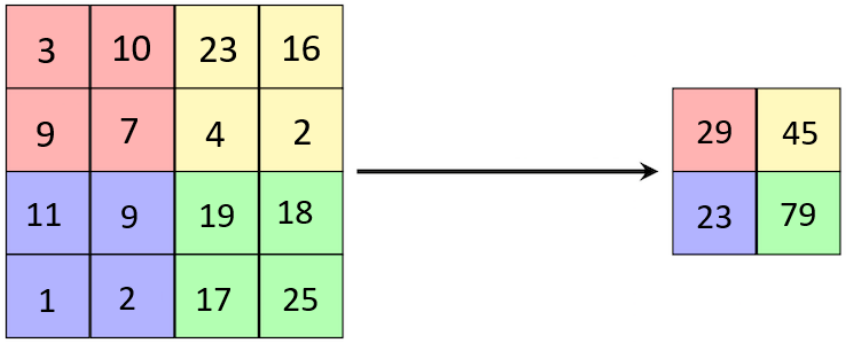


Feature Maps

YOLO uses convolution instead of pooling

We reduce the size of the images to extract smaller details

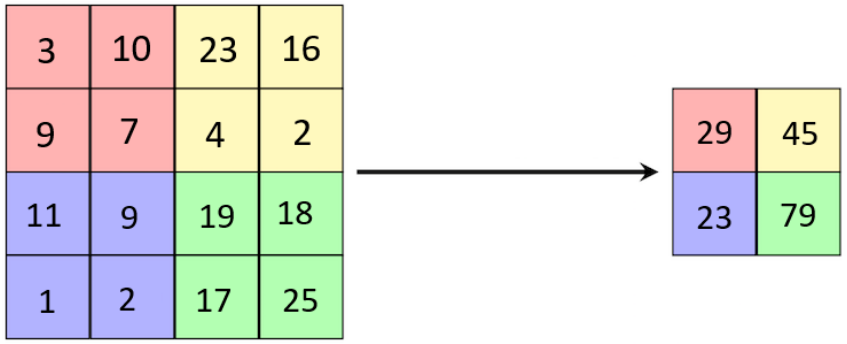




Convolution (Stride = 2)

Max Pooling and Average Pooling may lose low level features
Which is why YOLO uses Convolution Layers to retain this information

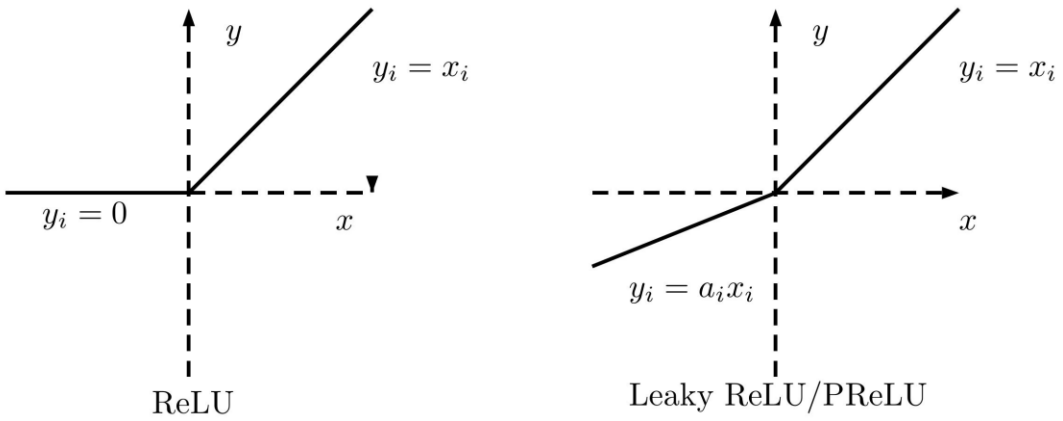
Batch Normalization Layer



Convolution (Stride = 2)

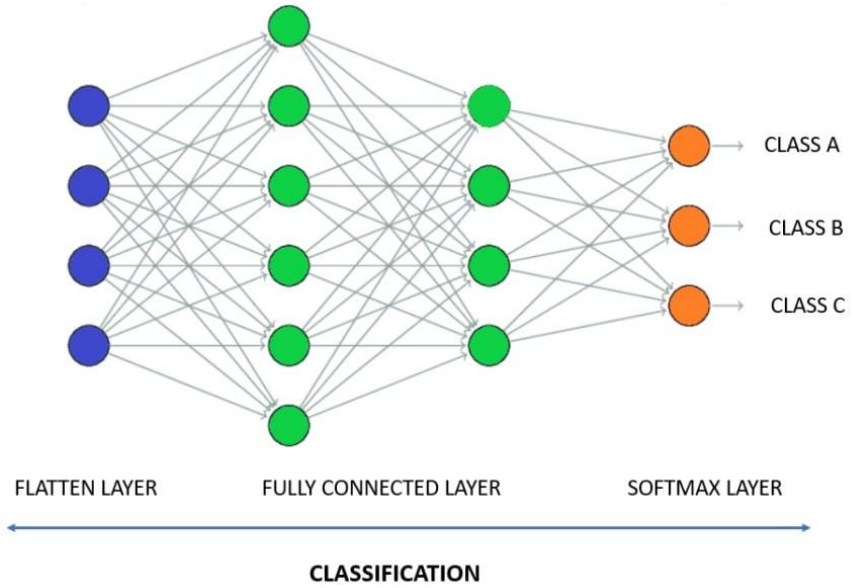
- Look at our previous convolution example
- The size of our numbers will keep on increasing with every convolution operation and can cause instability in our network
- YOLO implements a batch normalization layer after every convolution layer to normalize the all the outputs
- It also helps with training and other things. To simply put, smaller numbers, easier to work with

Leaky ReLU Layer



- YOLO uses the Leaky ReLU activation layer after each batch normalization layer
- The goal is to introduce non-linearity, eg: Leaky ReLU (−2)= max(0.1 × −2 , −2) = max(−0.2 , −2) = −0.2
- This means, the higher the value, the more the activation of the neuron

Fully Connected Layers



After the final convolutional layer, we flatten (convert all our outputs into a single dimension array) and use them as input to Fully Connected layers where each neuron is connected to each other neuron
Depending upon the final values of the neurons, a large output means those particular edges for that particular class were detected and there's a higher chance this is the object

Fully Connected Layers were removed after YOLOv1
But they are still used in almost all the other Convolutional Neural Networks

Detection (YOLO Layer)

So how does YOLOv3 detect objects?

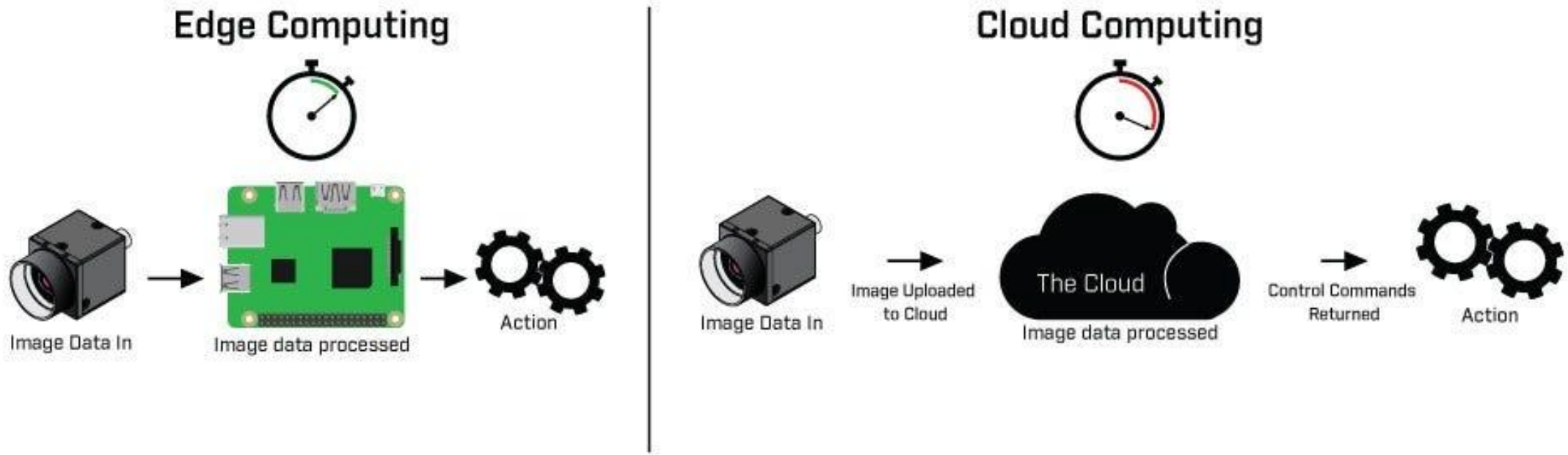
- Detections in YOLOv3 occur at layer 82, 94 and 106 which are called as “YOLO” layer
- Each cell of the produced feature map can detect objects
- The job of the YOLO layer is to find the anchor point of the detected object which is done by choosing the cell with the highest confidence value
- Since detections are done at 3 layers, we can use the detection with the highest confidence value

Detection Example

Edge Computing

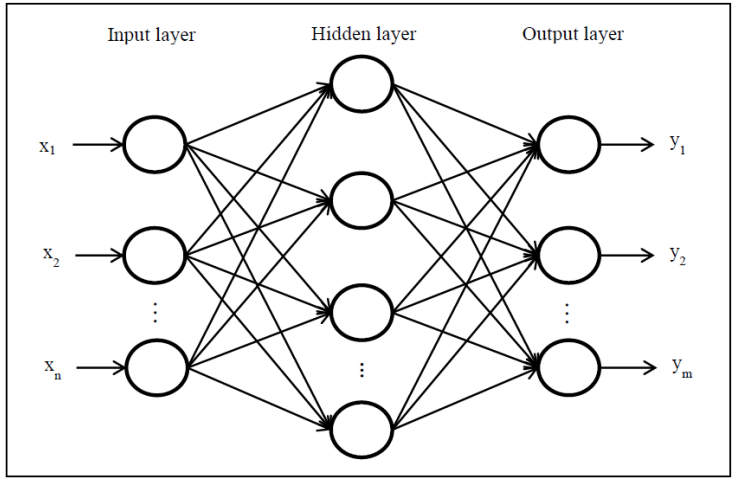
- Edge computing refers to the devices that process data on the far edge network cloud
- Eg. Security cameras, mobile phones, sensors
- In a lot of cases, transmitting all the data from the edge devices to the server for processing can be slow and take a lot of time
- Instead, we could deploy edge devices such as Raspberry Pis and NVIDIA Jetsons that are small but have large computing capabilities to process the data on the

Edge Computing

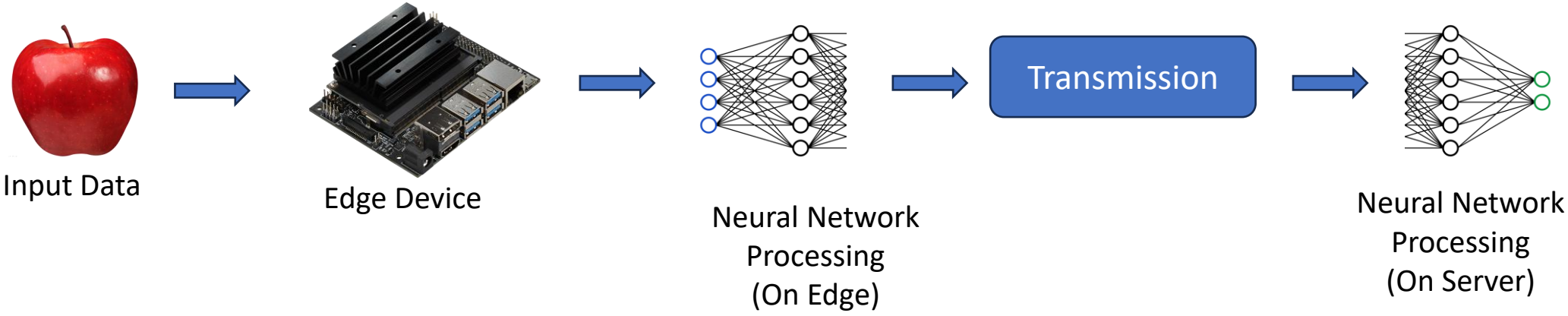


Model Partitioning

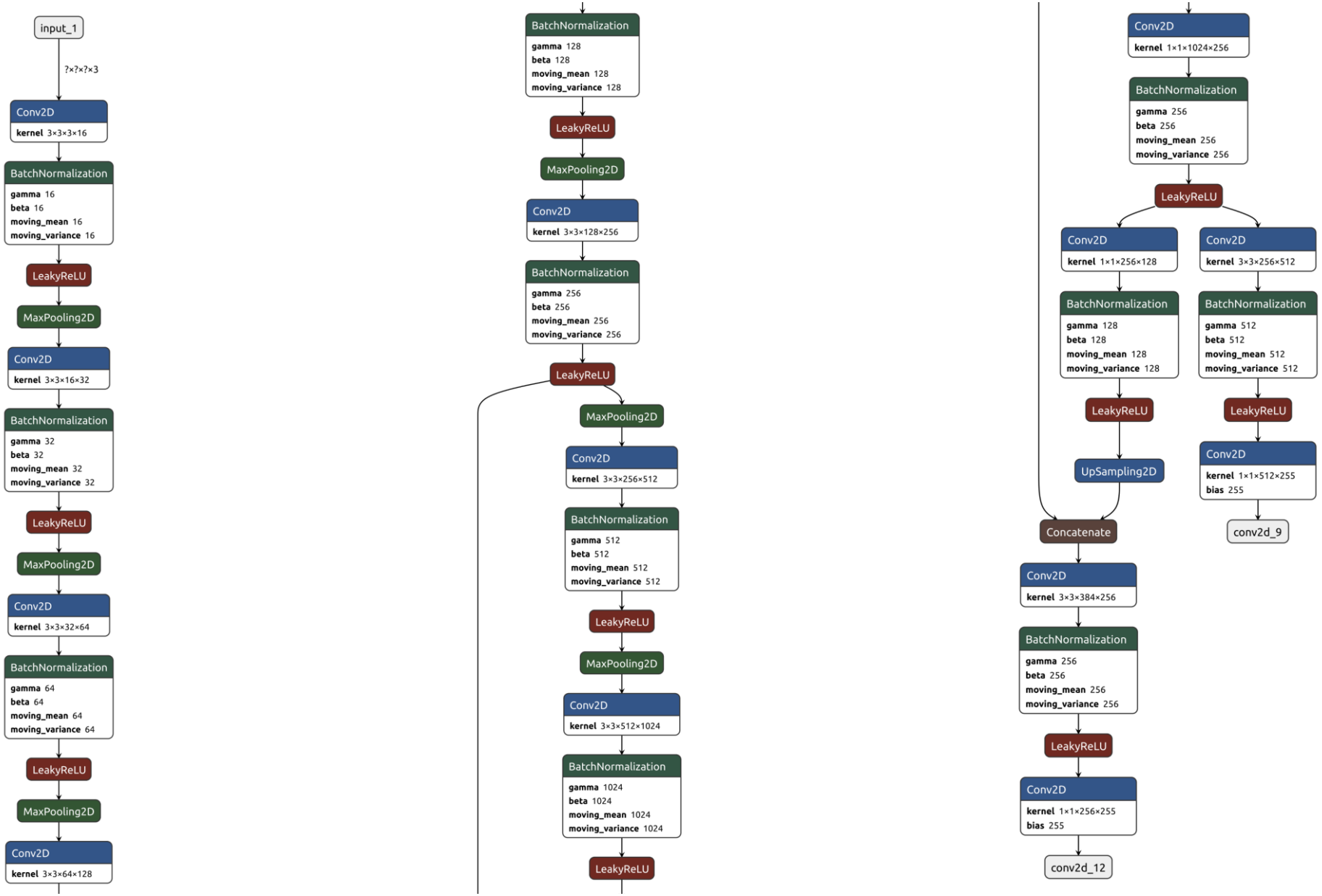
- Neural Networks are usually feed forward
- Meaning, data only flows in one direction
- We can leverage this to partition our neural network
- Example: YOLOv3-tiny can be partitioned at first 18 layers because those layers are feed forward



Model Partitioning Example



YOLOv3-tiny Model Structure



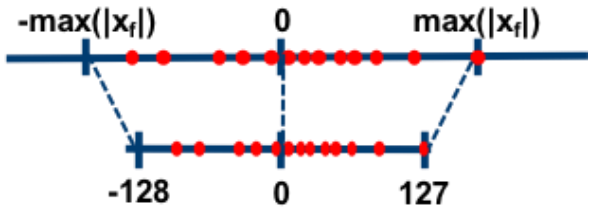
Quantization

- Quantization is the process of mapping a large set of values to a smaller set of values
- Eg. We have three values {5000, 2500, 10000} which can be quantized to 8 bit unsigned integers as {128, 63, 255}
- Or as 8 bit signed integers as {0, -64, 128}

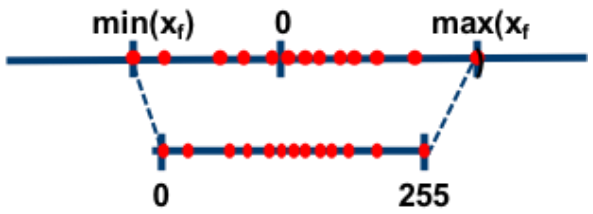
- If we have thousands of values, saving them as 8bit integers (instead of 32bit Floats) will taking less space
- Note: The accuracy of your neural network will go down but the effect is minimal

Quantization Types

- Symmetric Quantization: Signed numbers



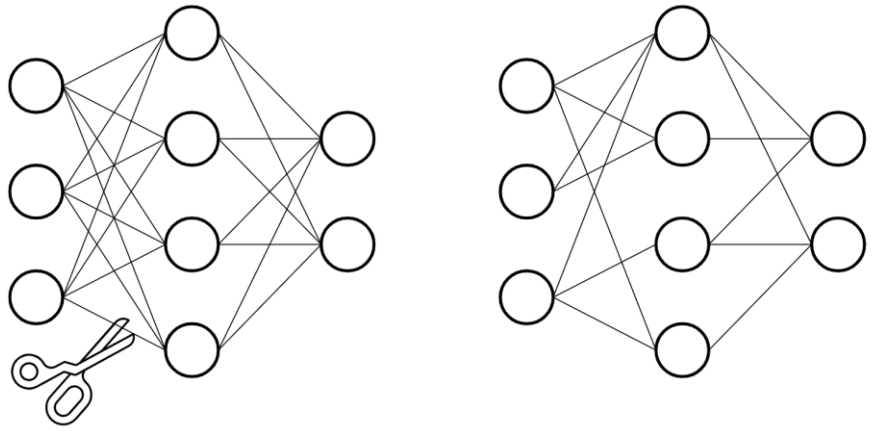
- Asymmetric Quantization: Unsigned numbers



Note: Type of Quantization used does not affect accuracy since the range of numbers is the same

Model Pruning

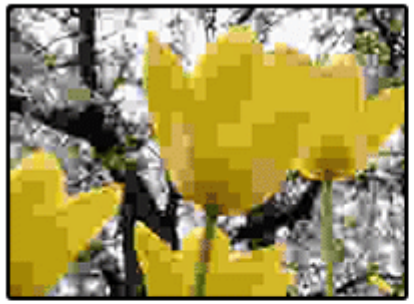
- Remove certain connections or neurons which’s weights contribute little or nothing to the neural network
- This can help reduce the size and complexity of the neural network



Compression

- After quantization, the size of the intermediate result may not be small enough for transmission to justify transmission
- We can try to further compress the intermediate result using different compression techniques, one common one being JPEG compression
- The time to compress and then transmit the intermediate result should be less than the time it takes the time to transmit the intermediate result without any compression

JPEG Compression Example



90%

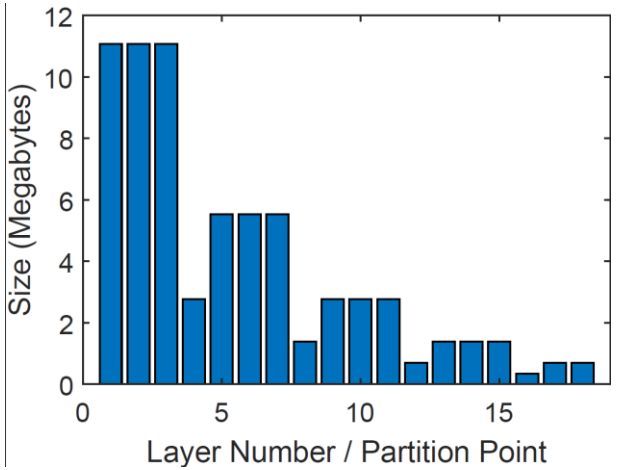
50%

10%

0%

Why partition, quantize and compress?

- When data flows through a neural network, the size decreases



- We can partition the neural network and process a part of it on the edge device, then transmit the smaller size intermediate result and then process the rest of it on an edge server

Extra Resources

Vsauce - The Stilwell Brain

https://www.youtube.com/watch?v=rA5qnZUXcqo

(No math, just how the brain and neural networks work)

3Blue1Brown - What is a neural network

https://www.youtube.com/watch?v=aircAruvnKk

(Simple Number Detection Example)

Visualize CNN (Tiny VGG)

https://poloclub.github.io/cnn-explainer/