



## Pose Estimation



# Programme (Day 1)

<b>Section 1:</b>	Background
<b>Section 2:</b>	What is Human Pose Estimation?
<b>Section 3:</b>	Publicly Available Human Pose Dataset
<b>Tea Break</b>	
<b>Section 4:</b>	Neural Networks (Brief)
<b>Section 5:</b>	Teachable Machines
<b>Lunch Break</b>	
<b>Section 6:</b>	Architectures for Action Estimation
<b>Section 7:</b>	Lab1: Pose Estimation with OpenPose
<b>Tea Break</b>	
<b>Section 8:</b>	Lab2: Pose Estimation with PoseNet
<b>Section 9:</b>	Lab3: Action Recognition with PoseNet?
<b>~ End ~</b>	

# Prerequisites

Familiar with

- Python Language
- Virtual Environment
- Anaconda
- Any text editor (Pycharm)



We explore different Neural Network Architectures but not specific to optimizing or hyperparameter tuning.



Google Drive



# Resources

---

- Goto <https://bit.ly/39jRQh2>
  - Download pdf (this presentation deck and activity sheets)
  - Links to other resources.

## Accounts needed:

- You will need a Google account.  
(Please sign up for one if you do not have it.)
- You will also need both Google Colab and Google Drive.



# Introduction of trainer

---



**Name**

Zack Toh

**Telegram**

@zacktohsh

**Email**

Zack\_toh@rp.edu.sg

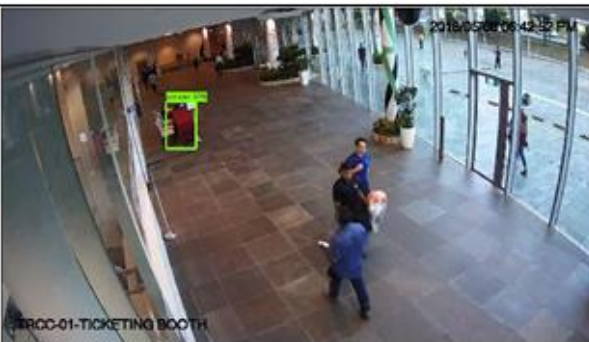


# Past Projects (Crowd Detection) (18RIG009)

## SSDv1



## SSDv2



## Yolo





# Deep Learning for Sports Tagging (19RBRF07)

Contextual awareness is the holy grail of computer vision. It aims at equipping a machine with the ability of deciphering what is happening in any given image. The automated tagging in sports is a new emerging area in sports industry. This project would contribute tremendously to the productivity and efficiency of sports tagging and dramatically reduce the labor needed in curating the sports statistic. The proposed prototype has high commercialization potential as shared by current sports tagging companies (e.g., Dartfish SA, HUDL and Prozone).



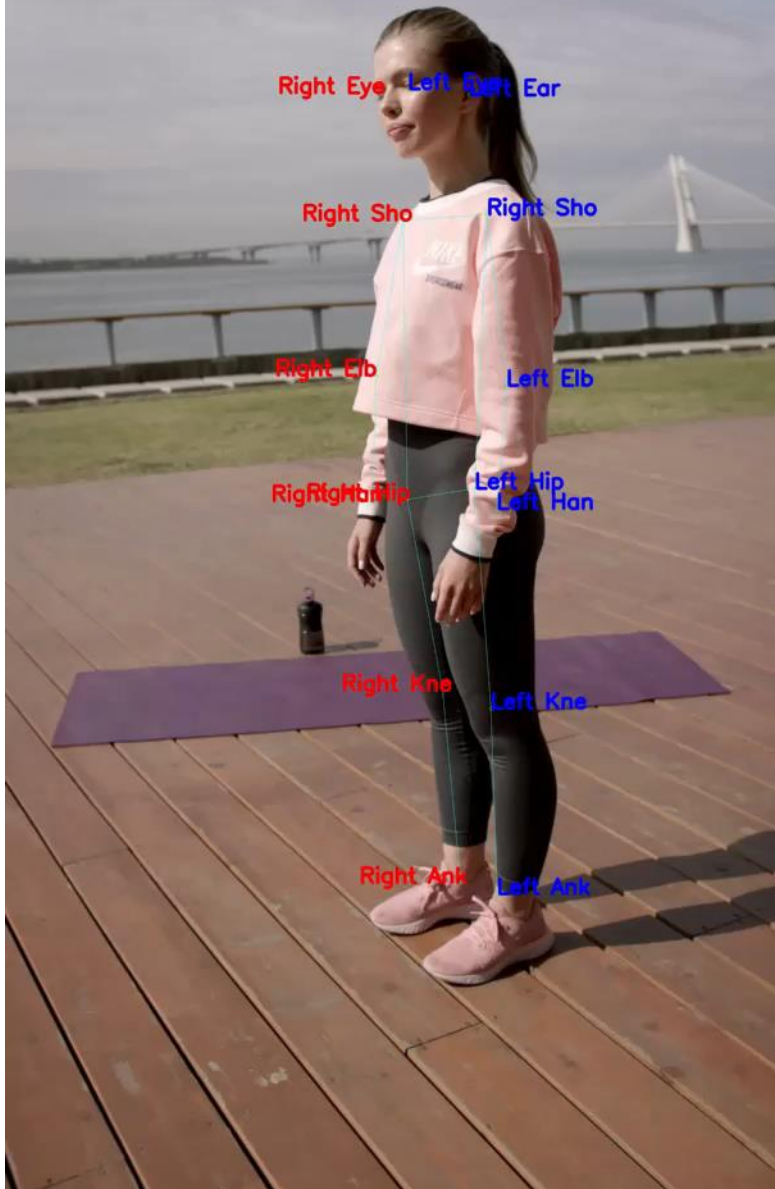
# Processed Throw-in





# Innovative Approach and Value (Current)





## Pose Estimation

Video obtained from: <https://www.pexels.com/>

Pose generated with Posenet



# Talk to flower

---

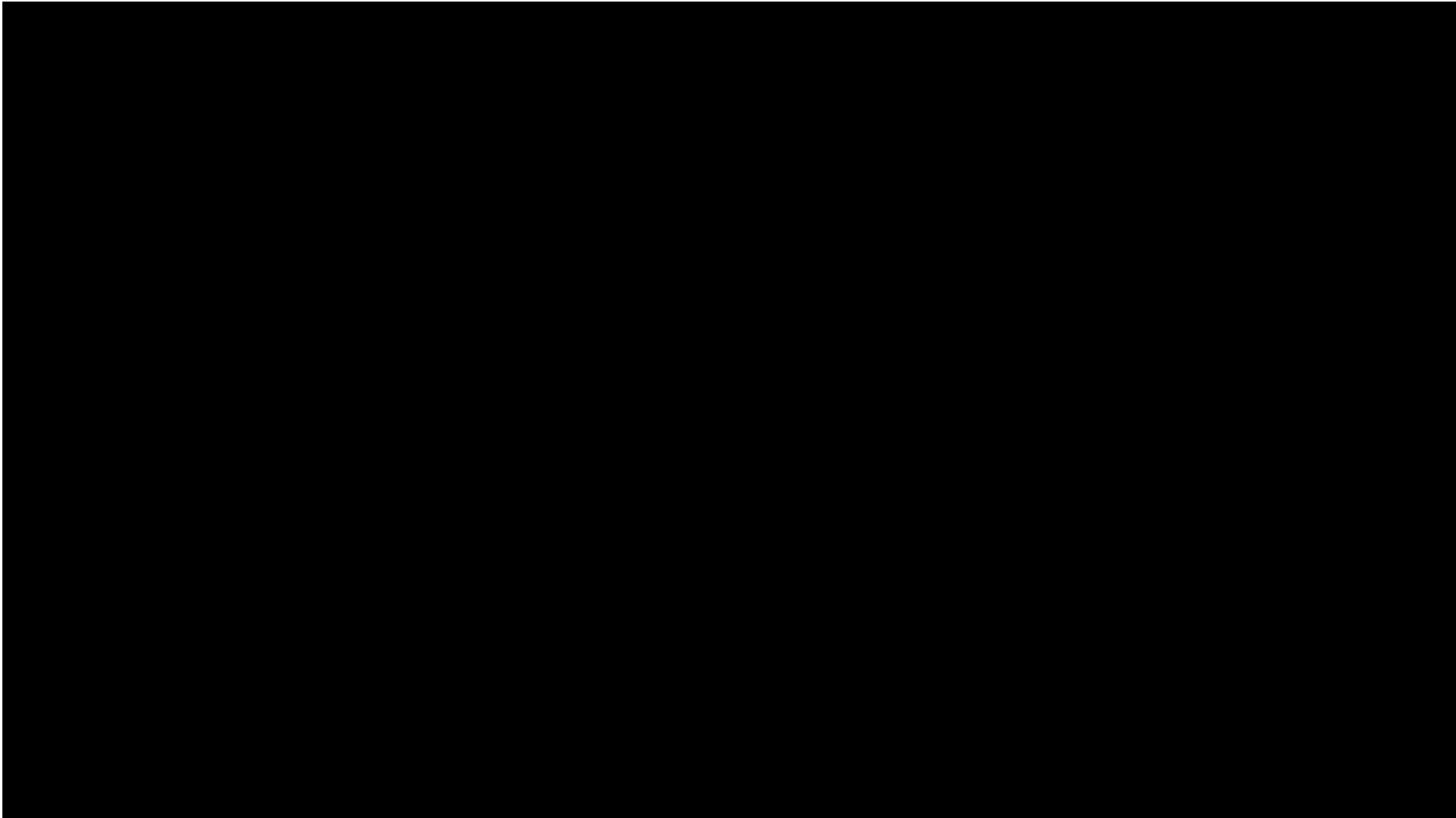


[https://www.youtube.com/watch?v=nsPQvZm\\_rgM](https://www.youtube.com/watch?v=nsPQvZm_rgM)



# Sci-fi Movies Coming True!

---



<https://www.youtube.com/watch?v=IG7DGMgfOb8>

# Background

---





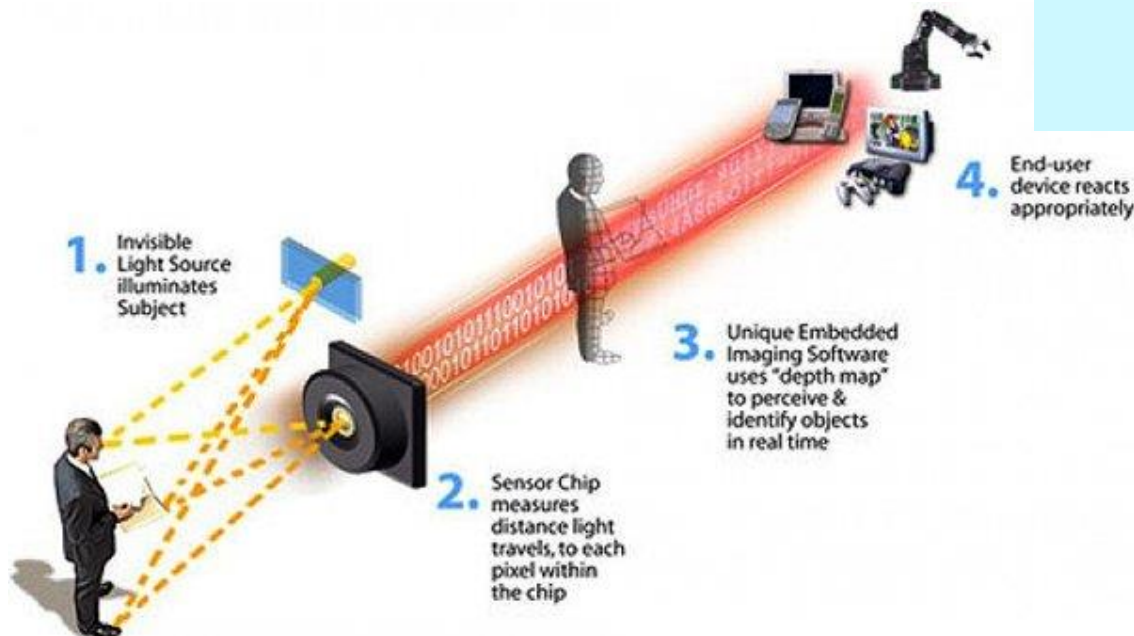
# Background

---

- Xbox Kinect was one of the first few technologies that was able to provide gamers with a full-body gaming experience.
- An Artificial Intelligence (AI) model was created based on a huge amount of motion videos.
- The AI model allows the Kinect to map the human to a skeletal structures to interpret the movements.



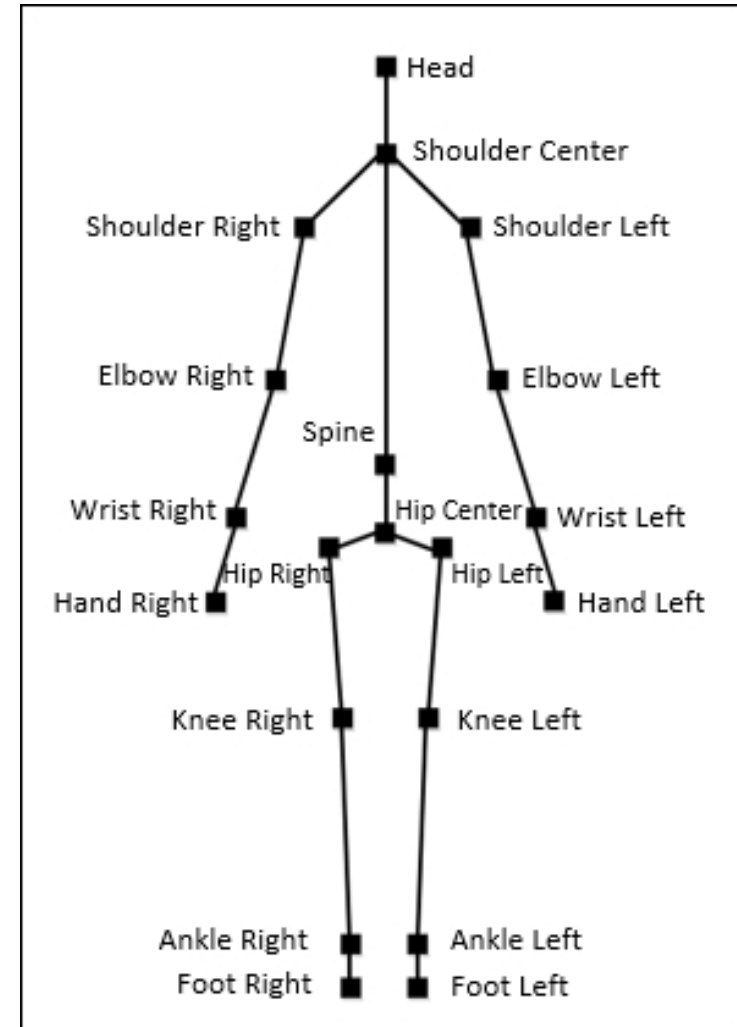
- 





# Background

- The Kinect for Windows SDK provides us with a set of APIs that allow easy access to the 20 skeleton joints.
- The SDK uses multiple channels to detect the skeleton.
- The default channel tracks all 20 skeletal joint positions.





Try this!

---

<https://storage.googleapis.com/tfjs-models/demos/posenet/camera.html>

# What is Human Pose Estimation?

---





# Human Pose Estimation

---

- Human pose estimation refers to the process of inferring poses in an image.
- How it is done is through the identification of a person's joint in an image or a video.
- This problem is also sometimes referred to as the localization of human joints.
- Pose estimation can be classified into different categories such as:
  - single pose estimation
  - estimating poses in an image with many people
  - estimating poses in crowded places
  - estimating poses in videos.



# Human Pose Estimation

---

- Pose estimation can be performed in either 3D or 2D. Some of the applications of human pose estimation include:
  - Activity recognition
  - Animation
  - Gaming
  - Augmented reality
- Approached to solving human pose estimation can take either of the following approaches:
  - top-down (human detection followed by key-point detection)
  - bottom-up (key-point detection followed by human detection)

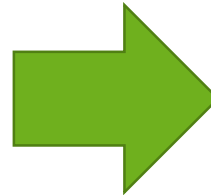


# Human Pose Estimation

- Top-down approach:



Object Detection

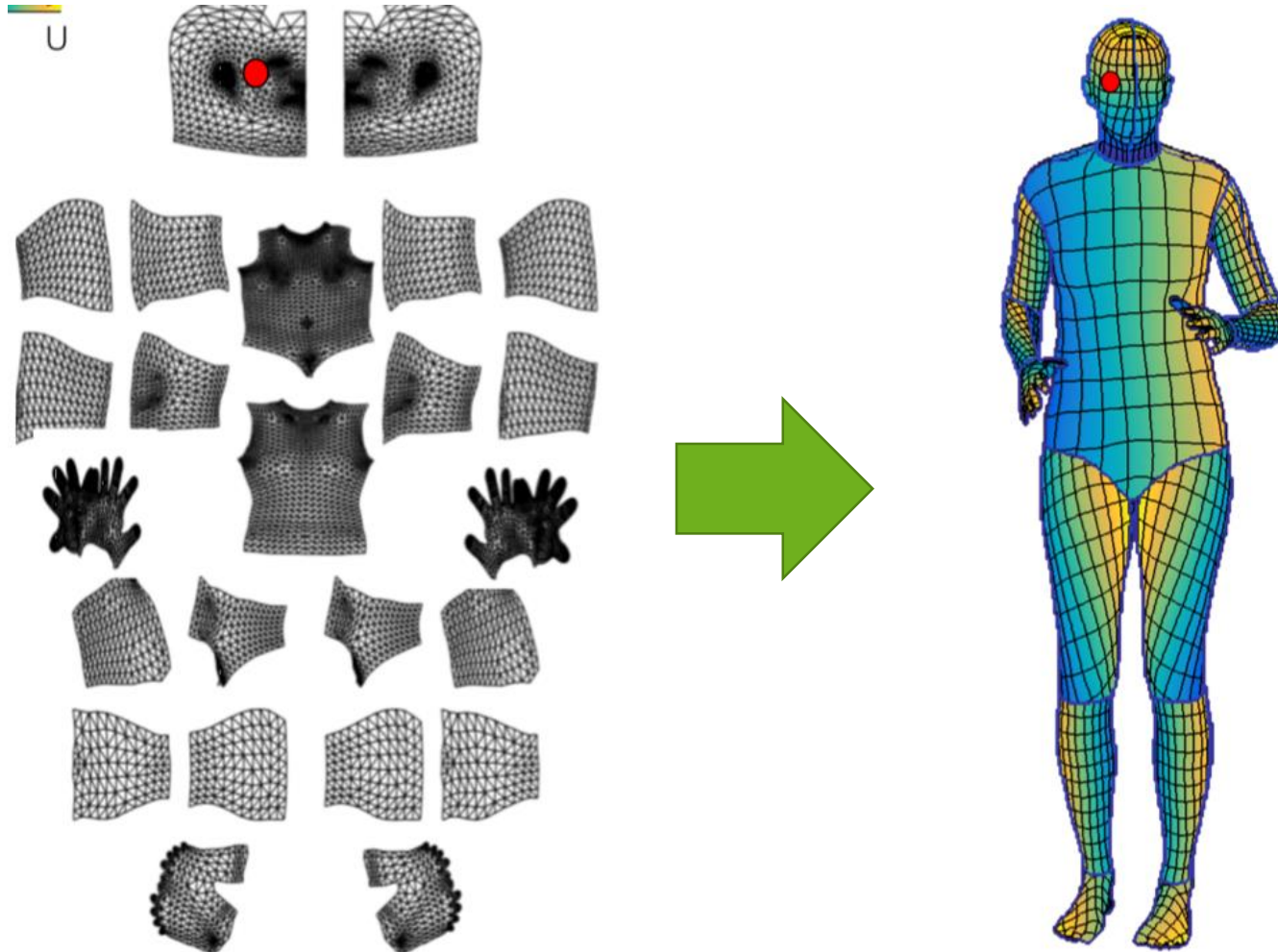


Joint detection



# Human Pose Estimation

- Top-down approach:



# Publicly Available Human Pose Dataset

---





# MPII Human Pose Dataset

- MPII is a human pose dataset that contains around 25K images containing over 40K people with annotated body joints.
- The dataset is made up of 410 human activities and each image is provided with an activity label.
- Annotate means the tagging of the identified human joint on the image with reference to the human.





# Leeds Sports Pose (LSP) Dataset

- The LSP dataset contains 2000 pose annotated images of mostly sports people.
- Each image has been annotated with 14 joint locations. Left and right joints are consistently labelled from a person-centric viewpoint.







# Frames Labeled In Cinema (FLIC) Dataset

- The FLIC dataset contains 5003 images curated from movies.
- The dataset was curated through human detection from the frames.
- Labelling of the frames were done through “crowdsourcing marketplace Amazon Mechanical Turk”, with 10 upper body joints.
- Severely non-frontal images are removed from the dataset.





# ***15 Mins Break***

[bit.ly/top10\\_2020](https://bit.ly/top10_2020)



# Neural Networks (Brief)

---





# Neural Networks

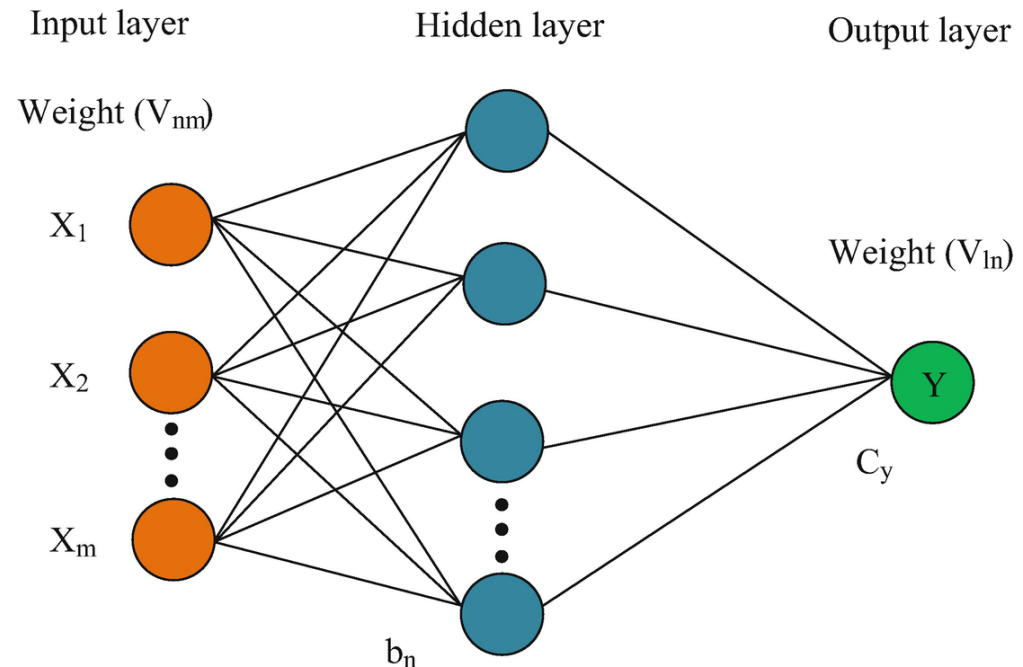
---

- So which neural network should we be using?
- Before we take a look at the answer, let us take a minute to look through the different neural network below:
- Artificial Neural Network (ANN)
- Convolution Neural Network (CNN)



# Artificial Neural Network (ANN)

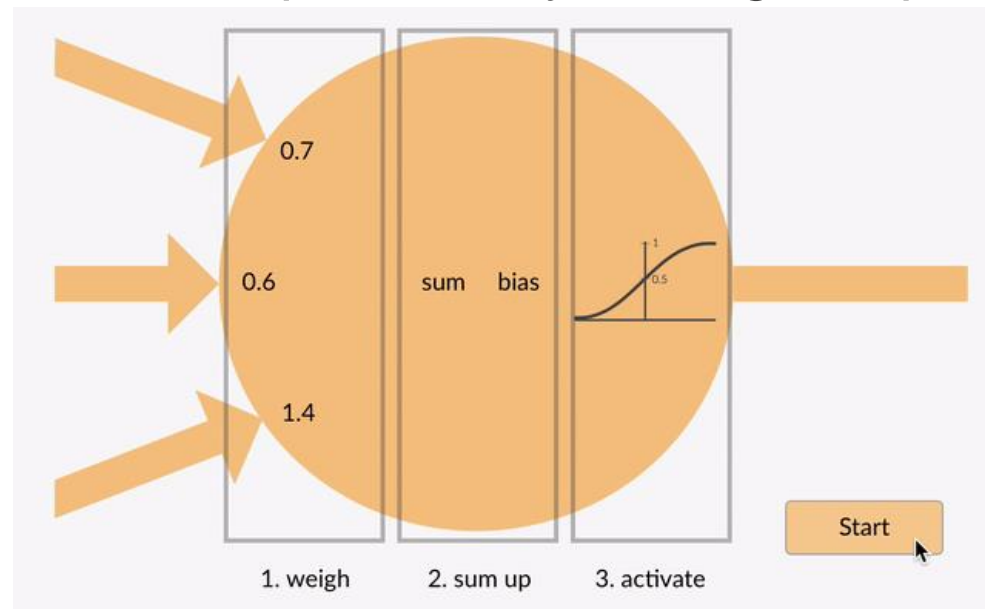
- Artificial Neural Network (ANN) typically comprise of an input layer, a hidden layer and an output layer.
- As how we process a single logistic regression, that can also be seen as a single neuron/perceptron in ANN.
- ANN is a feed forward network since data travels in a forward direction, from the input, through the hidden layers and to the output.





# Artificial Neural Network (ANN)

- How a typical neuron in ANN works is by summing the weights together with the bias and passing it through an activation function.
- The activation function is much like an on/off switch where it will turn each feature on or off.
- This allows for the network to learn the relationship between the input and the output thereby solving complex problems.





# Artificial Neural Network (ANN)

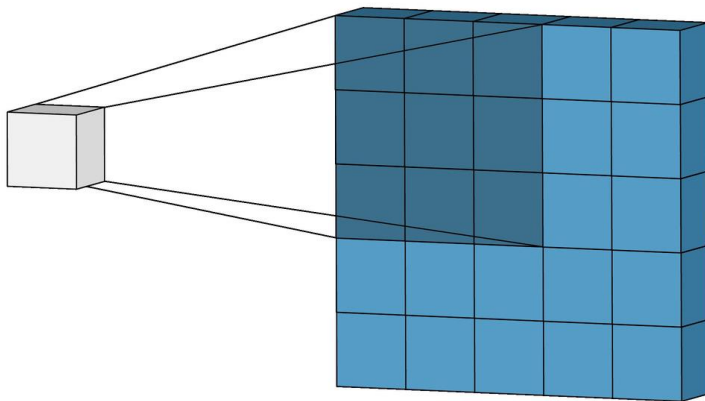
---

- As good as ANN may be, it has its own set of challenges too.
- If used for image processing, it will require the conversion of a 2D image into a single dimension vector for processing.
- This would result in a loss of spatial features of an image.
- Spatial features refer to the arrangement of pixels and the relationship between them in an image.
- In addition to the above, ANN is also unable to capture sequential information in the input data which is required for dealing with sequence data.



# Convolution Neural Network (CNN)

- CNN makes use of kernels to extract the key features from the input using the convolution operation. By doing this, it is able to retain the spatial features from an image.
- The spatial feature will help us in identifying the object accurately, the location of an object, as well as its relation with other objects in an image.





# Pooling Layer

- When features are identified in the output, these features tend to have a bit more details than required (high resolution).
- Being too specific in terms of the identified feature may result in the feature map as being too rigid unable to be generalized for detection at other location of the image.
- In order to solve this problem, the feature maps are down sampled to allow the feature map to be reusable across the whole image.

**Feature Map**

6	6	6	6
4	5	5	4
2	4	4	2
2	4	4	2

**Max  
Pooling**

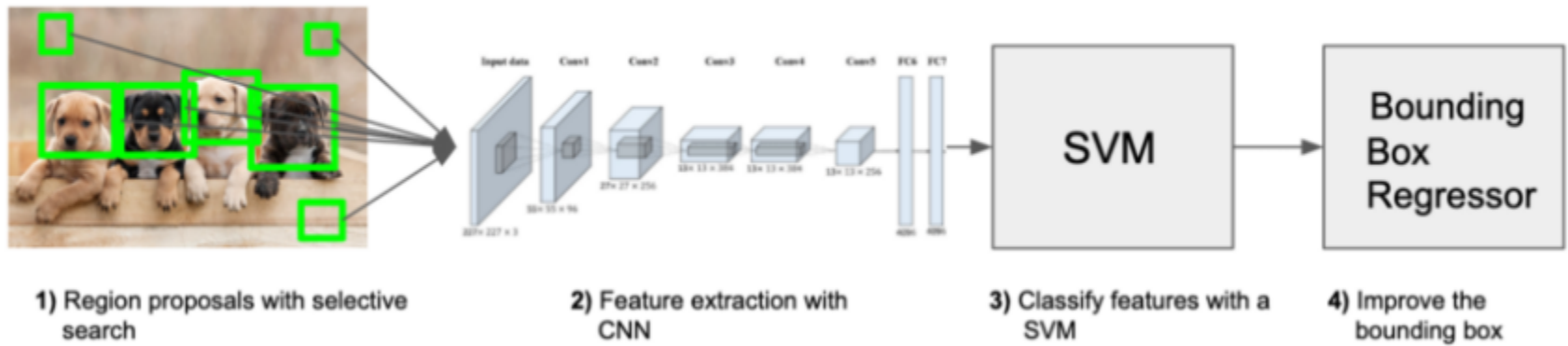

**Average  
Pooling**


**Sum  
Pooling**




# R-CNN

## R-CNN System



The R-CNN system





# Selective Search

---

Selective Search starts by over-segmenting the image based on intensity of the pixels using a graph-based segmentation method.



Input Image



Output Image

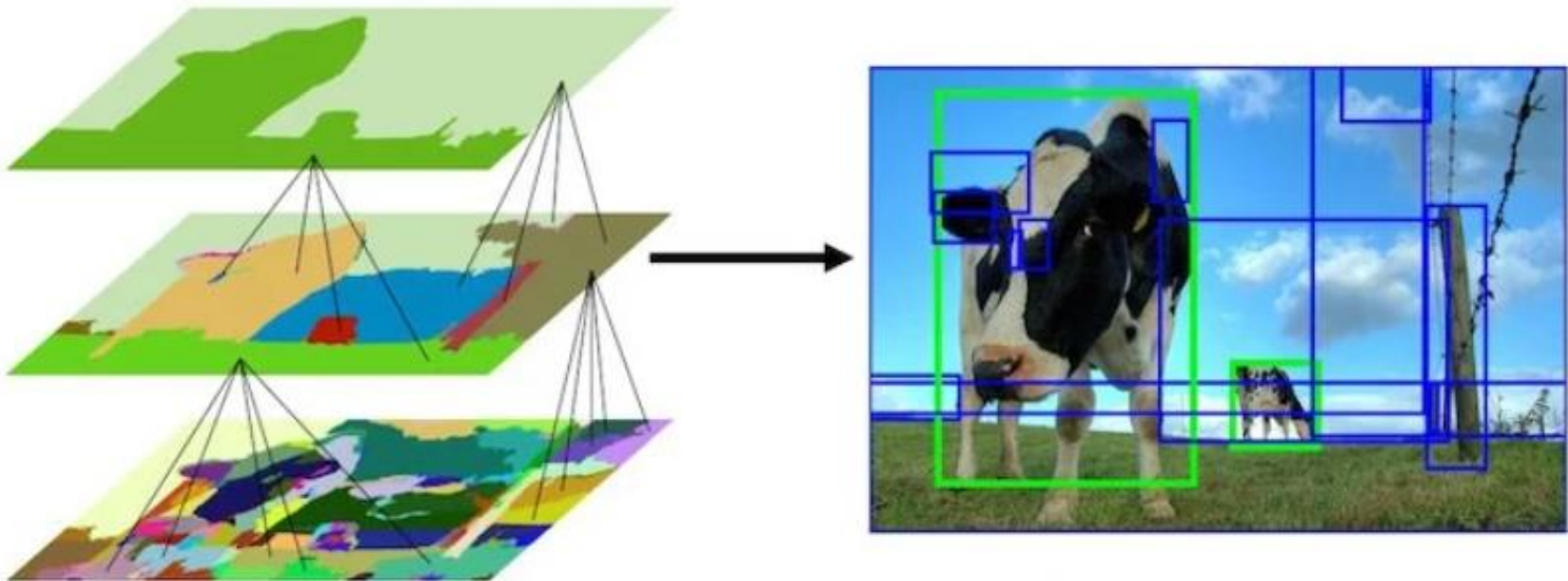


# Selective Search

1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1

Regions are grouped based on similarities:

- Color Similarity
- Texture Similarity
- Size Similarity
- Shape Compatibility
- Final Similarity

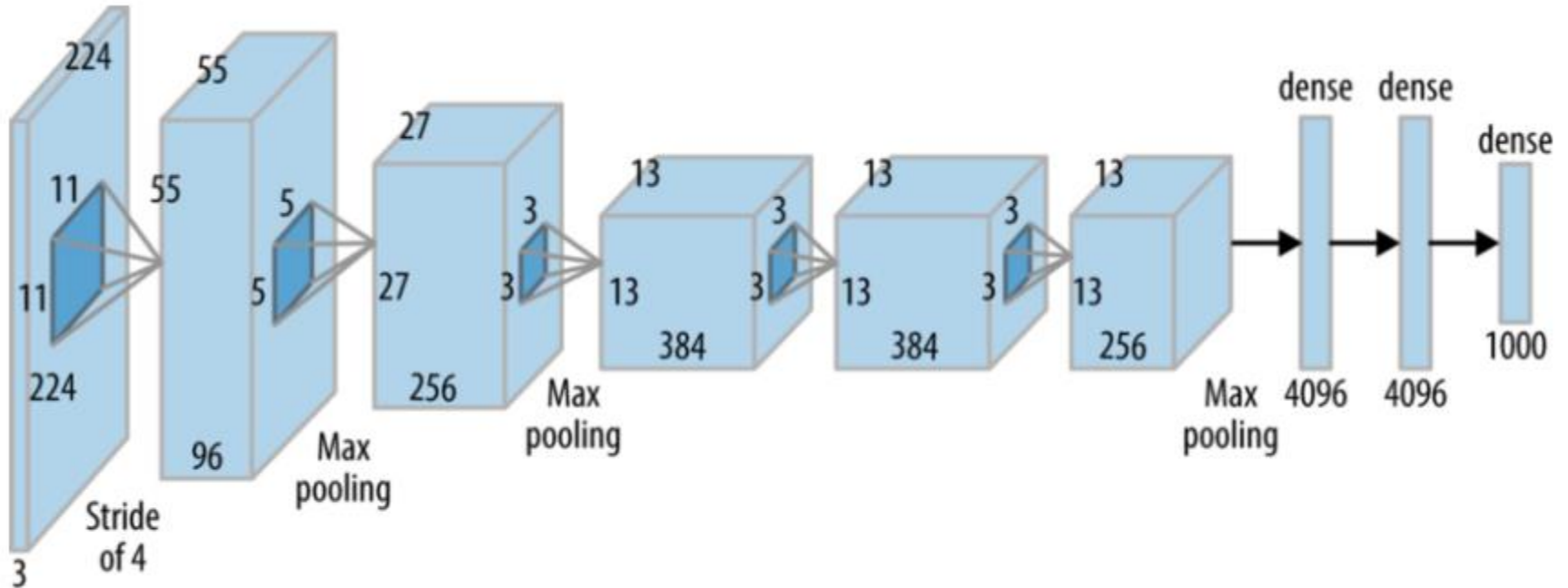








# R-CNN



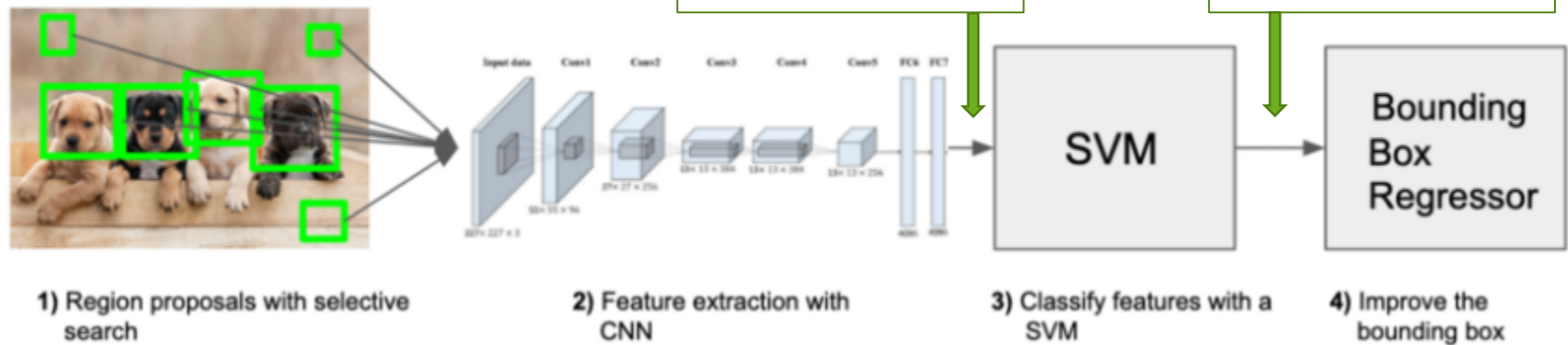
AlexNet

AlexNet used as a feature extractor.  
Images from the image proposals are resized  
before passing them through Alexnet.



# R-CNN

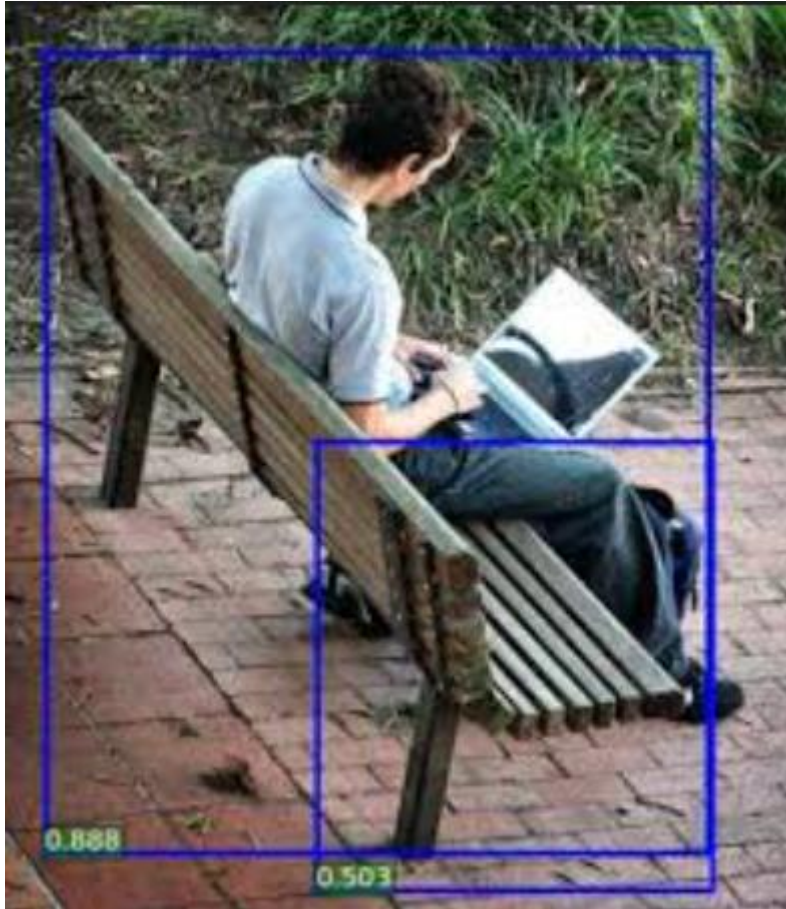
## R-CNN System



The R-CNN system



- At this point, there may still exist many bounding boxes for the same object.



Solution: Just choose the bounding box with the highest confidence and remove other overlapping bounding boxes.

---



- Could the same logic of choosing the bounding box with the highest confidence apply to the scenario here? What would happen?



Can we just choose  
the bounding box with  
the highest  
confidence?


---





# So what is the best solution?

- Calculate how much overlap exist between two bounding boxes.
- Remove bounding boxes only if there exist a significant amount of overlap between two bounding boxes.
- Solution: Greedy Non-maximum Suppression
- Calculation: Intersection over Union (IoU)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




# Bounding Box Regression

- Bounding Box Regression was also proposed by the original paper as the finishing touch to improve the average precision.
- This segment requires training too.
- How it is achieved is through the calculation of the loss.
- Loss is calculated based on the difference between the expected and predicted region.
- The loss will be back propagated through the fully connected layers to update the weights.
- With the new weights, forward propagation is performed to achieve a new and loss.
- The steps are repeated till minimum loss is achieved.





# R-CNN Problem

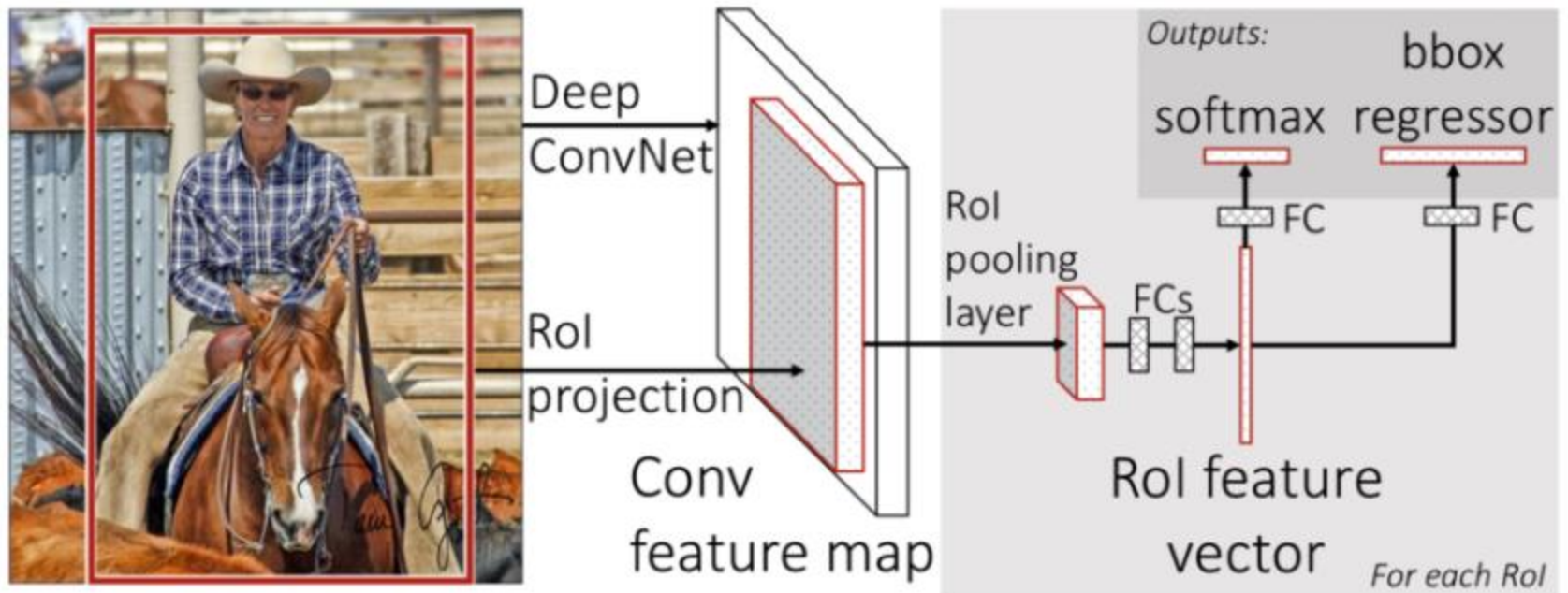
---

- Speed: feature map calculation – one forward pass per region proposal.
- Training: CNN (feature extractor), SVN and Bounding Box Regressor, each require training.
- Large memory requirement: every feature map of each region proposal is needed to be stored.



# Fast R-CNN

- To address the problem, the architecture is changed.





# Fast R-CNN

1. Process the image with CNN to produce a feature map.
2. For each region proposal (selective search), extract the corresponding feature map (region proposal feature map).
3. Region proposal feature map is resized to a fixed size using Region of interest pooling.

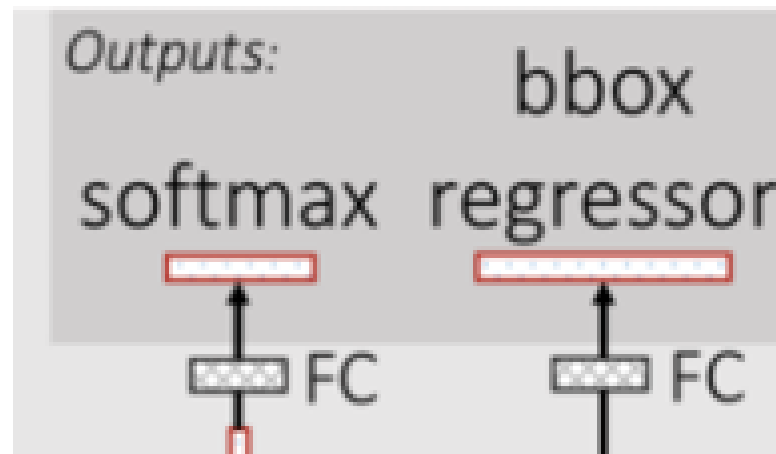
input

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91



# Fast R-CNN

4. The fixed size region proposal feature map is then flattened (from a three-dimensional layer into a single dimensional vector:  $3 \times 3 \times 3$  tensor  $\Rightarrow$  vector of size 27) into a feature vector.
5. The feature vector is then passed into a fully connected layer with two outputs:
  - Softmax classification layer
  - Bounding Box Regression





# Fast R-CNN Problem

---

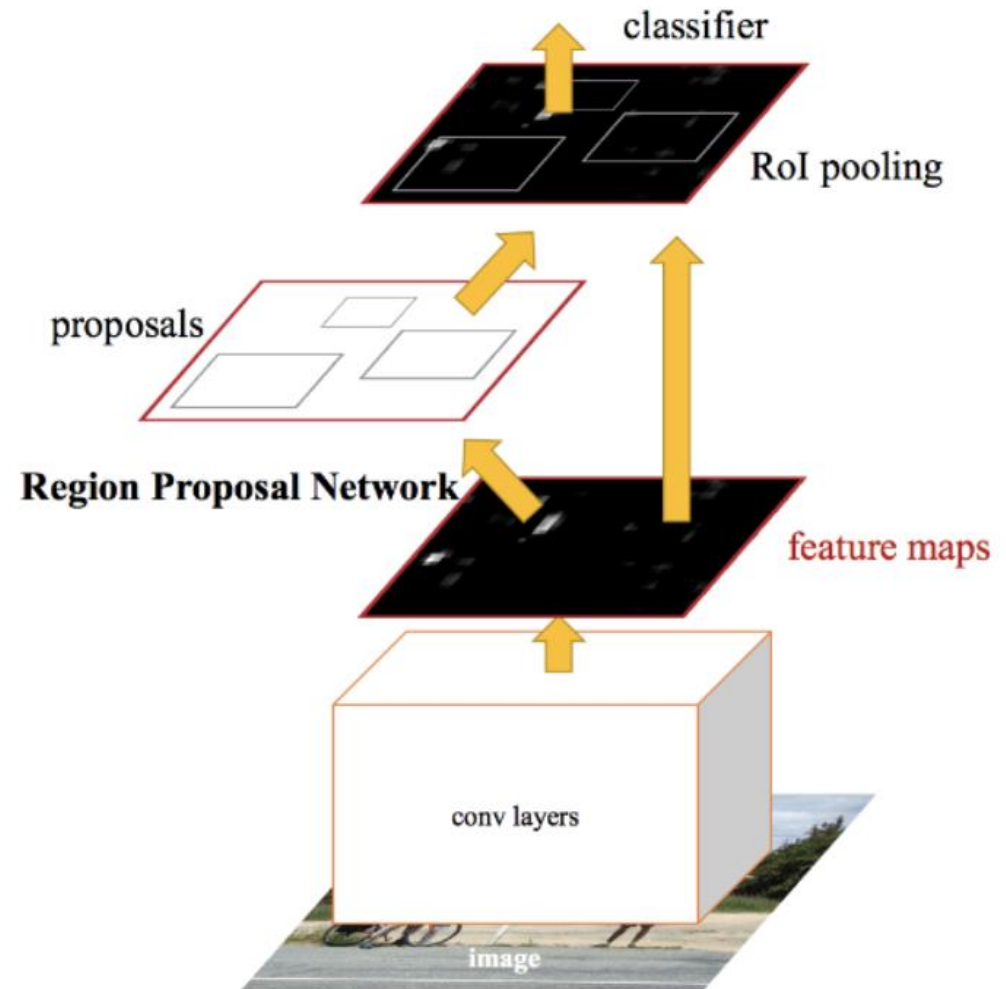
- Though Fast R-CNN is much faster than R-CNN, it is still not feasible in the real-world scenario given that it requires about 2 seconds to process an image for object detection.
- The main reason is because of the use of selective search in Region Proposal.





# Faster R-CNN

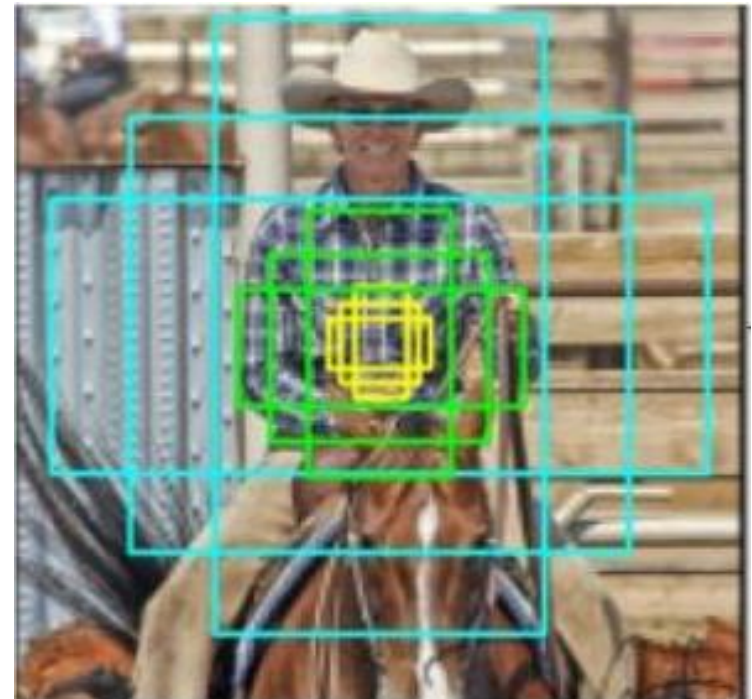
- The key difference between Faster R-CNN and Fast R-CNN is the change from the use of Region Proposal from selective search to Region Proposal Network (RPN).
- The other change is that the extracted feature map is being shared between RPN and the downstream detection network.





# Region Proposal Network (RPN)

- Output feature map from CNN is processed using RPN.
- RPN makes use of “Anchors” of various size and aspect ratio.
- Key is to identify if an object is present.
- Figure shows 9 possible anchors in three different aspect ratios and sizes.

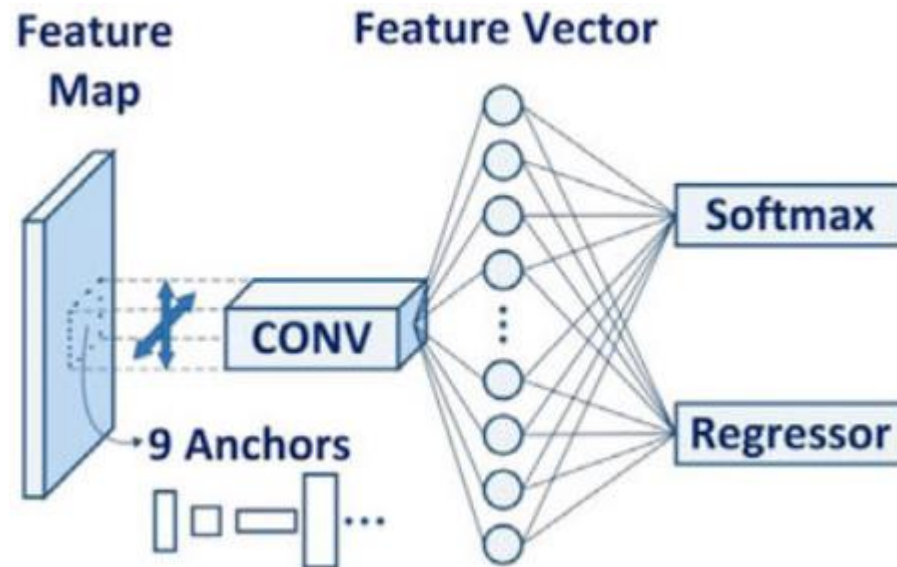


If an object is present, the anchor's coordinated will be refined to produce a region of interest.



# Region Proposal Network (RPN)

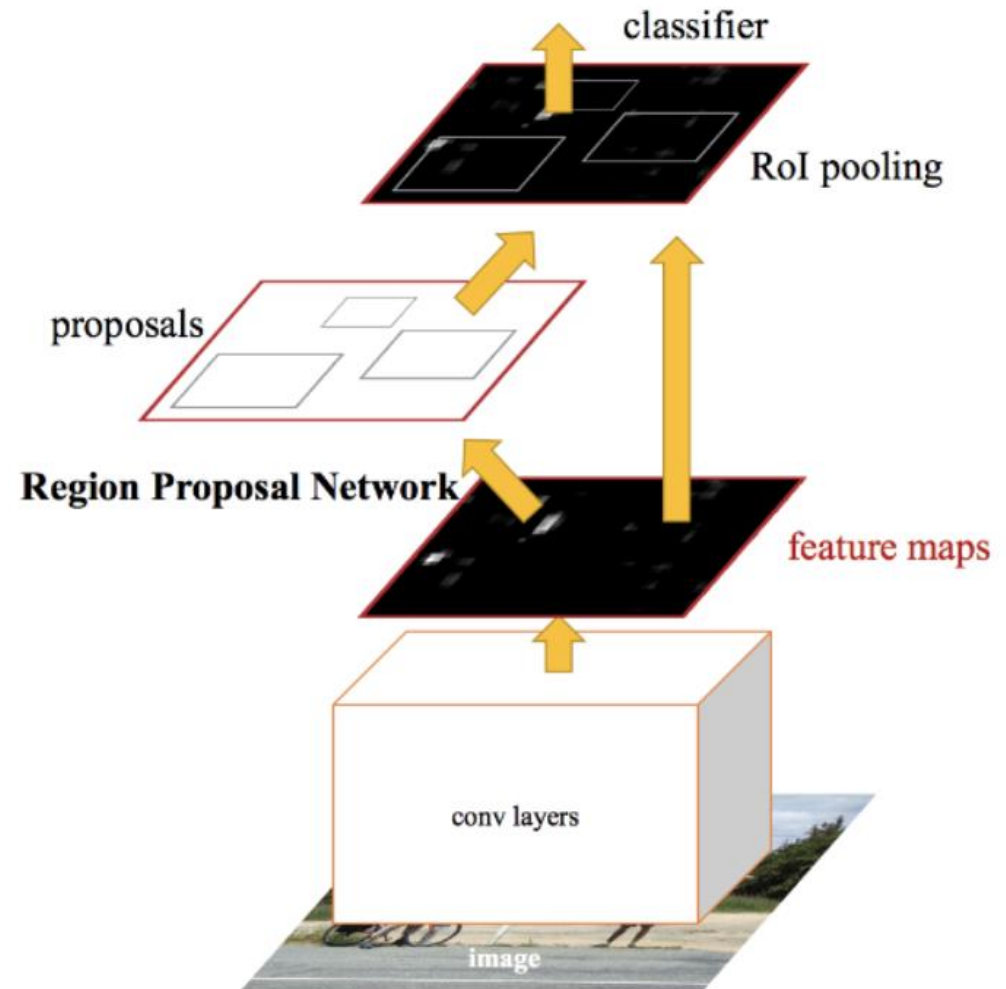
- RPN itself produces two outputs.
  - Classification (Softmax) – to determine if there is an object
  - Regression layer – the prediction of bounding box that best encase the object.





# Faster R-CNN

- The bounding box proposals from RPN are then passed back to the RoI pooling layer.
- Feature maps are produced from RoI and are passed to a fully connected later which has Softmax for classification of the object, and regression for the best bounding box coordinates.





# Faster R-CNN Problem

---

- The proposed algorithm still require thousands of passes through a single image to extract all the objects.
- Due to the linearity of the different modules, performance is as good as the performance of the previous module.



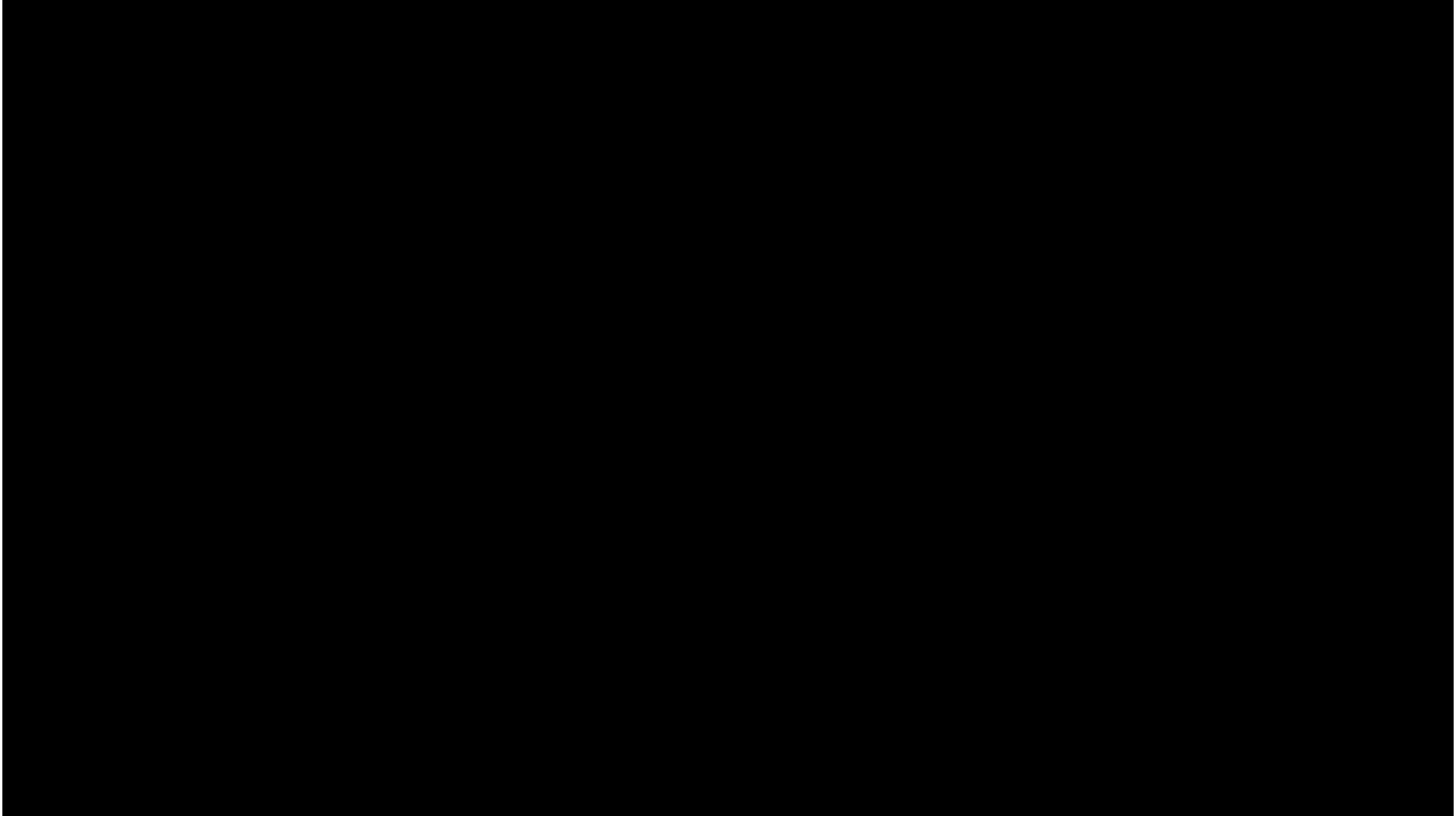
# Teachable Machine





# Teachable Machine

---





# Steps

1. Go to <https://teachablemachine.withgoogle.com/>

2. Click on “Get Started”



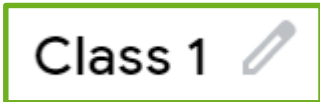
Get Started

3. Click on “Pose Project”



## Pose Project

4. Edit your labels

Class 1 

5. Click on either the “Webcam” or “Upload” to choose the source of your data.



Webcam



Upload

# 60 mins Lunch Break

## Some interesting videos

<https://www.youtube.com/watch?v=bmNaLtC6vkU>

[https://www.youtube.com/watch?v=Nnf8P5A\\_saE](https://www.youtube.com/watch?v=Nnf8P5A_saE)

## Gentle Reminder:

1. Get your google account ready.
2. Get your google drive and google colab account ready.

**LUNCH BREAK**



# Architectures for Pose Estimation

---

# Pose Estimation (Top-Down)

---

DensePose as an example



# DensePose (Facebook AI Research)



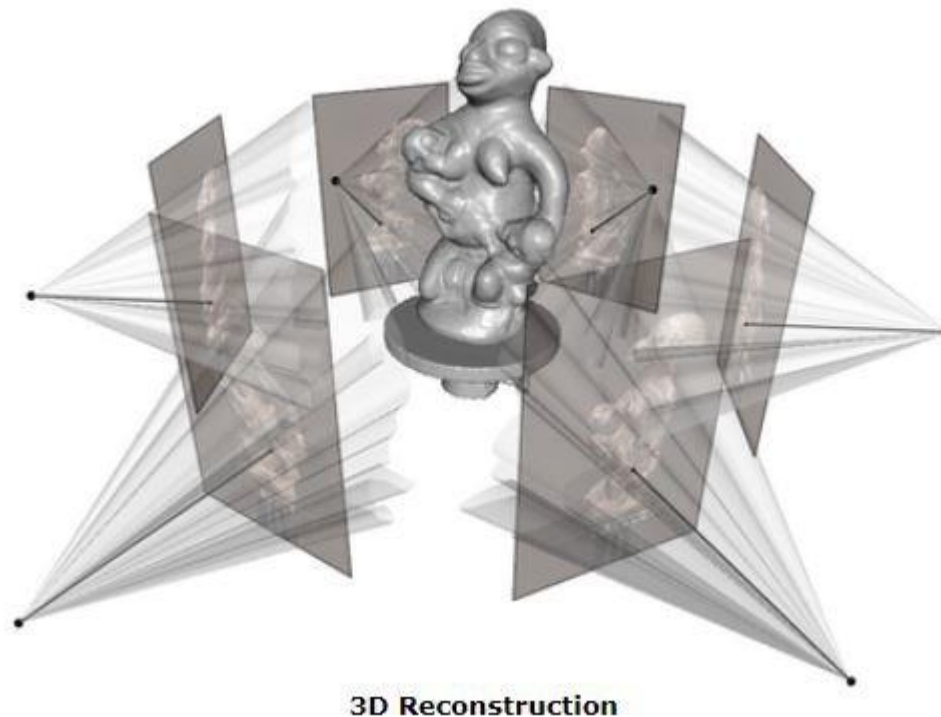
[https://www.youtube.com/watch?v=RQGuSJYo\\_Ds](https://www.youtube.com/watch?v=RQGuSJYo_Ds)





# 3D Imaging

- A typical 3D imaging is done through manipulating of a surface through rotations and it is often done with the aid of depth sensors.

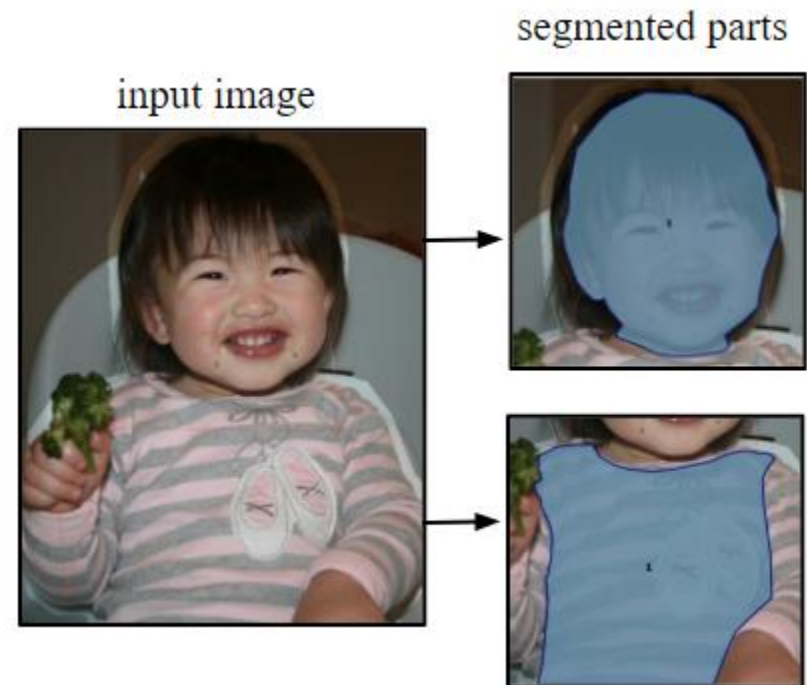






# DensePose (Facebook AI Research)

- Dense human pose estimation attempts to recreate a 3D human images.
- This is performed though the use of human annotators to map of all human pixels of an RGB image in a 2D image, to the 3D surface of the human body.
- The first step requires the system to identify the body region of the body parts.
- Following that, an estimation of the body part is performed.

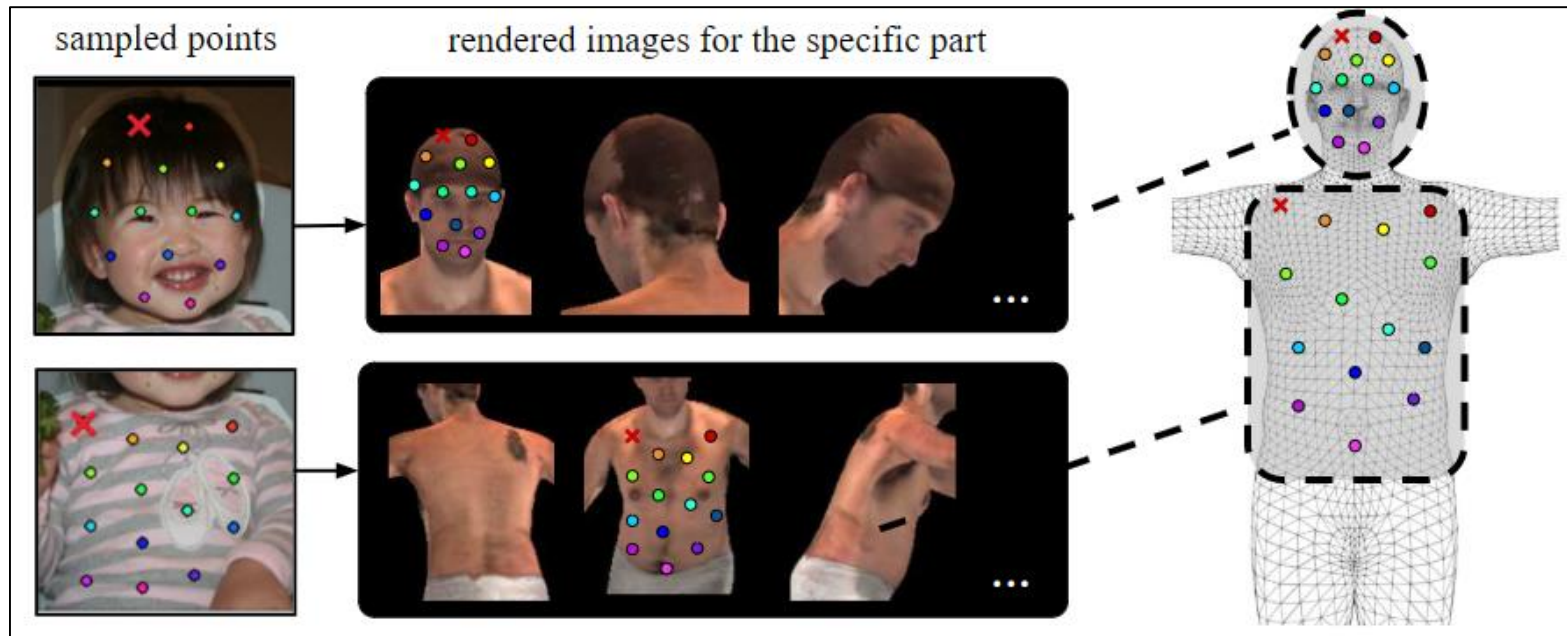




# DensePose

## (Facebook AI Research)

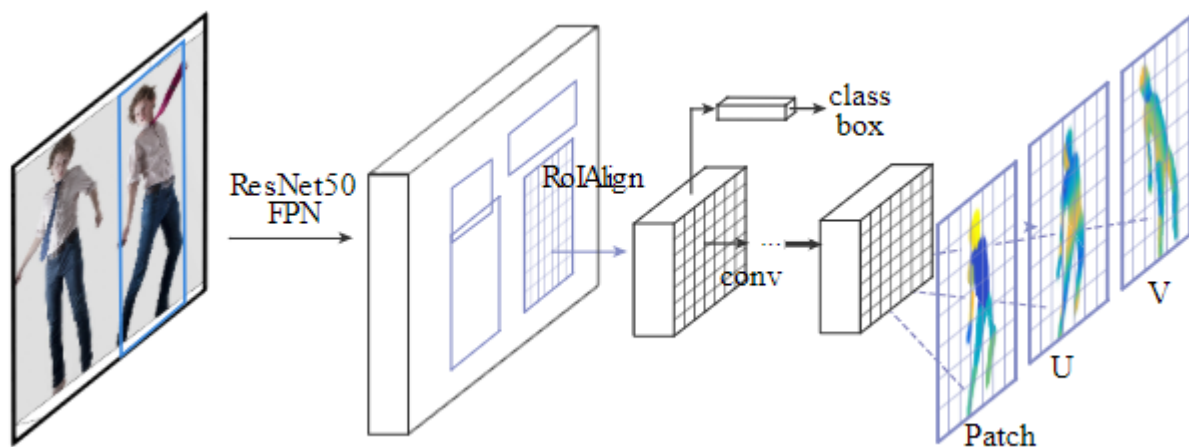
- The second step requires the sampling of regions using roughly equidistance points for each body part.
- The annotator will try to attempt to match these sampled points to the surface.
  - The process is simplified through providing of six pre-rendered views of the same body part.
  - Landmarks can be placed on these parts which will allow the annotator to identify the closest matching view. (best matched angle)





# DensePose-RCNN

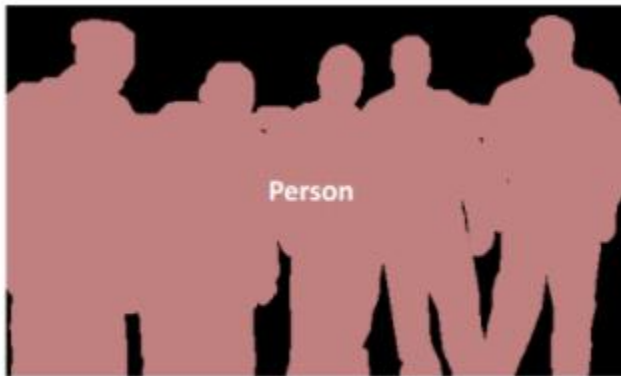
- The use of the following architectures were adopted:
  - Mask-RCNN
  - Feature Pyramid Network (FPN) features
  - ROI-Align pooling
- Following that, a fully-convolutional network is applied to classify the surface part.



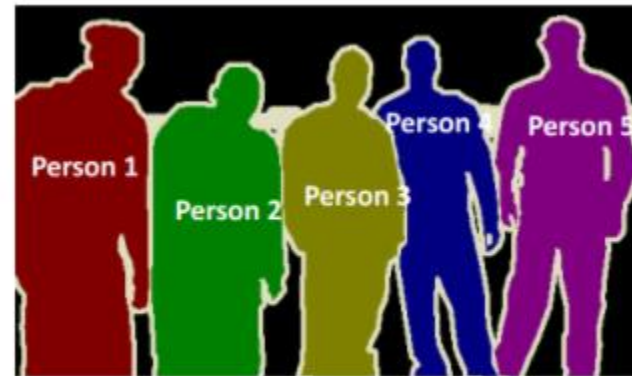


# Mask-RCNN

- Mask-RCNN is commonly used to perform instance segmentation.
- What instance segmentation typically does is that it identifies the individual pixels that belong to each person as seen below.
- This is in contrast to semantic segmentation where the identification of the individual pixels is performed by a cluster of objects.



Semantic Segmentation



Instance Segmentation



# Mask-RCNN

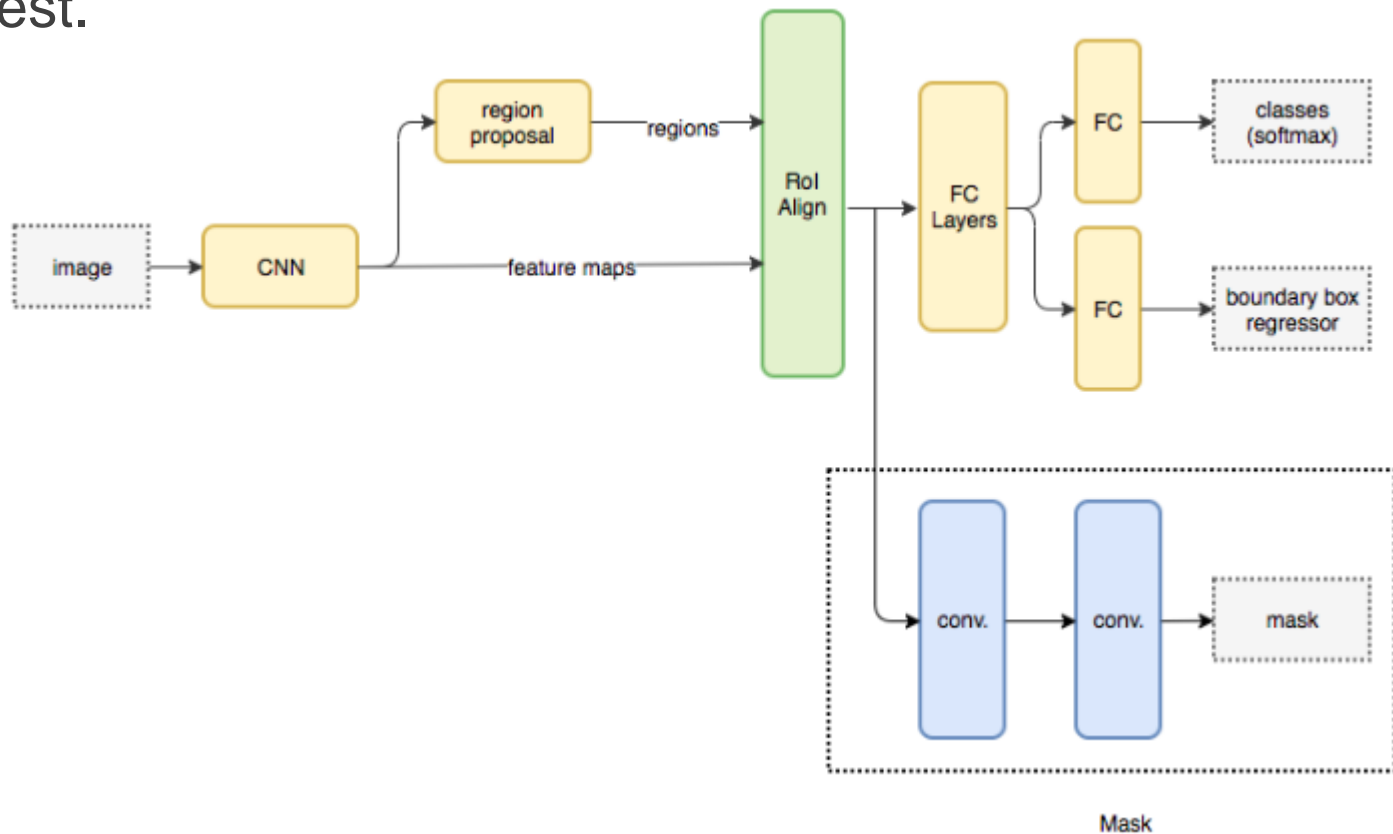
- In the typical object detection such as fast-RCNN, bounding box proposals are proposed in the region where there is an object and the most significant bounding box is chosen above all.
- What Mask RCNN does is that it performs pixel level segmentation on the object for object identification.





# Mask-RCNN

- The Mask-RCNN is typically achieved through the addition of convolution layers in order to produce the mask.
- The input of the convolution layer is the object that has been identified from a CNN in order to locate the object of interest.

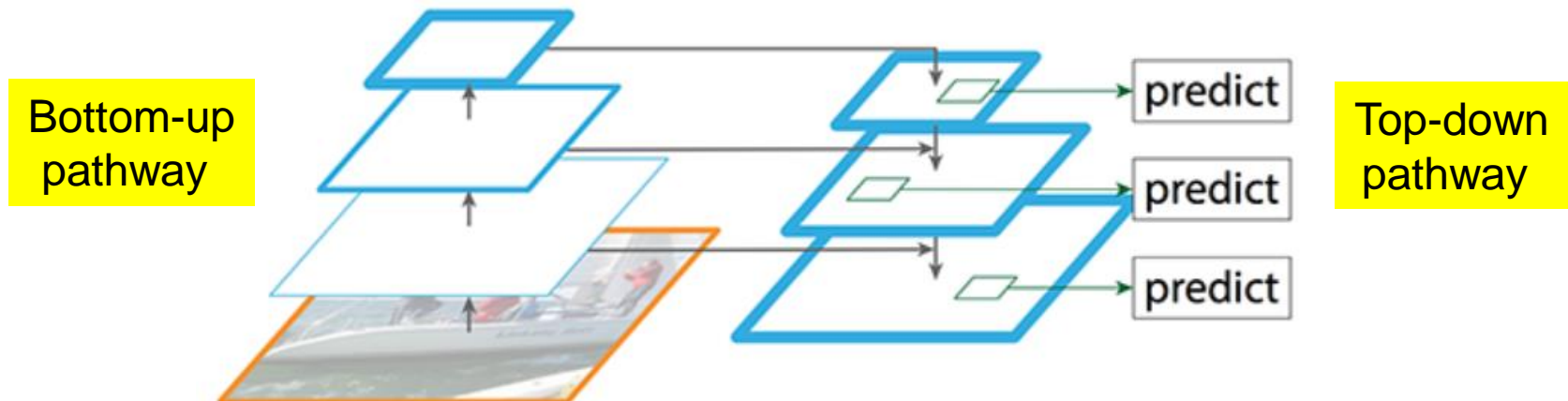






# Feature Pyramid Network (FPN)

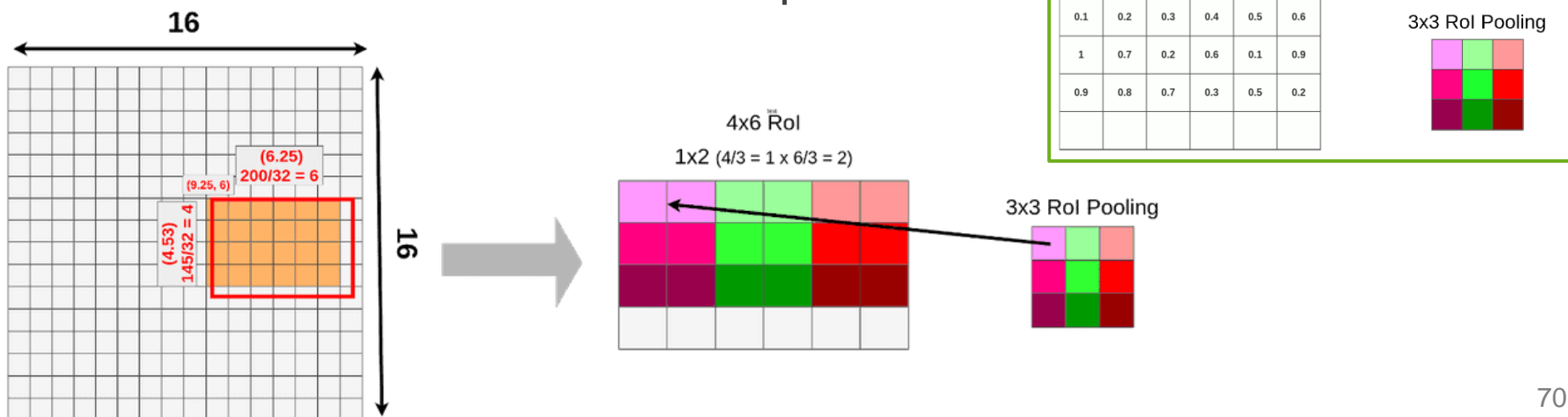
- The FPN consists of both a bottom-up and top-down pathway.
- The bottom-up pathway is the standard CNN backbone for feature extraction and each stage is defined as each level of the pyramid.
- The output of the top layer of the bottom-up pathway will be used as the input for the top-down pathway.
- The top-down pathway will then upscale the image at each pyramid level. This process is able to help enhance the image through reducing any pixilation effect that may exist in an image.
- Each pyramid level connects the feature map for both the bottom-up and top-down pathway together.
- The bottom-up pathway produces a more accurate spatial accuracy while the top-down pathway produces a more accurate semantic accuracy.
- The combination of these two pathways in FPN will allow for a more accurate detection.





# ROI Align

- ROI (Region of Interest) Align is to help to improve the accuracy of the ROI pooling.
- ROI pooling is predominantly used in Fast-RCNN and Faster-RCNN mainly for the purpose of object detection.
- But as the ROI pooling may often identify only part of the object, while sufficient for object detection, is inaccurate for object mask detection.
- ROI Align solves this problem as it is able to identify the whole area of the feature map.



# Pose Estimation (Bottom-Up)

---

OpenPose as an example



# OpenPose (Background)

---

- OpenPose library is an open-source real-time human pose estimation algorithm.
- It is able to perform multi-person 2D pose detection, including body, foot, hand, and facial key points.
- It is developed by researchers at the Carnegie Mellon University and some of the authors of this paper are from IEEE.
- The library is built with the help of COCO and MPII dataset.
- The method uses nonparametric representations known as Part Affinity Fields (PAFs)., to learn to associate body parts with individuals in the image

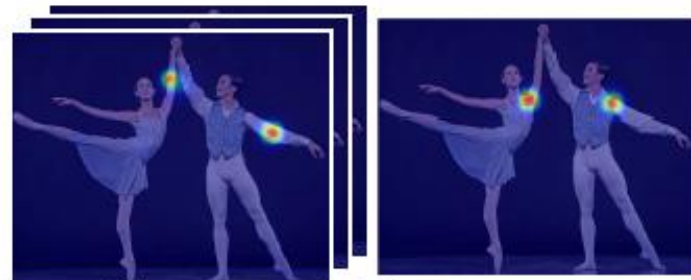


# OpenPose (Method)

1. Whole image (a) is passed through a convolution network.
2. Confidence map (b) of the detected body parts are produced.
3. Part Affinity Fields (PAFs), (c), a 2D vector field, is produced that relates the degree of association between parts.



(a) Input Image



(b) Part Confidence Maps



(c) Part Affinity Fields



# OpenPose (Method)

- Both the confidence maps and the PAFs are parsed by greedy inference (d).
- The output the 2D key points for all people in the image is then produced (e).



(d) Bipartite Matching

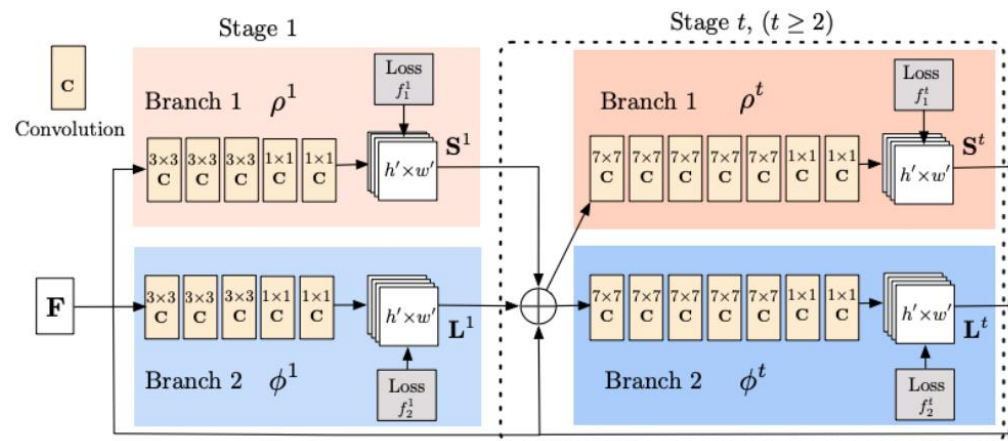


(e) Parsing Results



# OpenPose (Explained)

1. An RGB image is fed as input into a “two-branch multi-stage” CNN.
  - a) The first branch (beige) predicts the confidence maps (S) of different body parts location.
  - b) The second branch (blue) predicts the part affinity fields (L).
2. The second stage requires the input of the original image feature (extracted through CNN) that is concatenated with both (S) and (L).
3. Each stage will produce a more refined prediction and the process is repeated for  $t = 6$ .





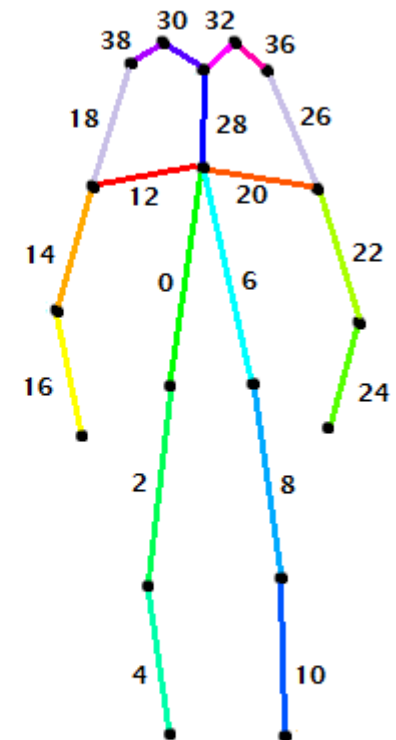


# Parts and Pairs

- Part - an element of the body  
(Example: neck, left shoulder or right hip)
- Pair – the connection between two parts.
- (Example: nose and the left eye, ears and shoulders)



Parts



Pairs



# Lab1: Pose Estimation with OpenPose

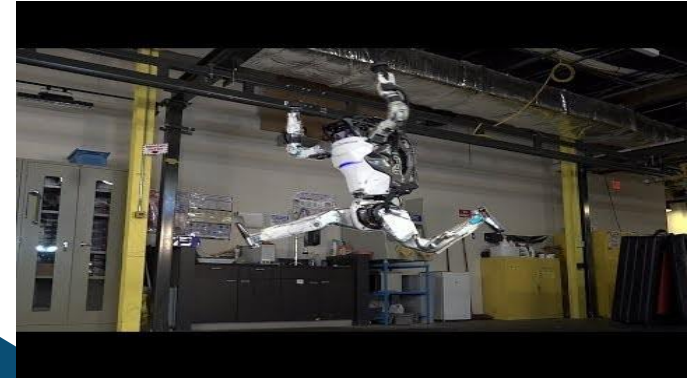


<https://youtu.be/ooqiY7hFWE8>



# ***15 Mins Break***

[https://youtu.be/\\_sBBaNYex3E](https://youtu.be/_sBBaNYex3E)



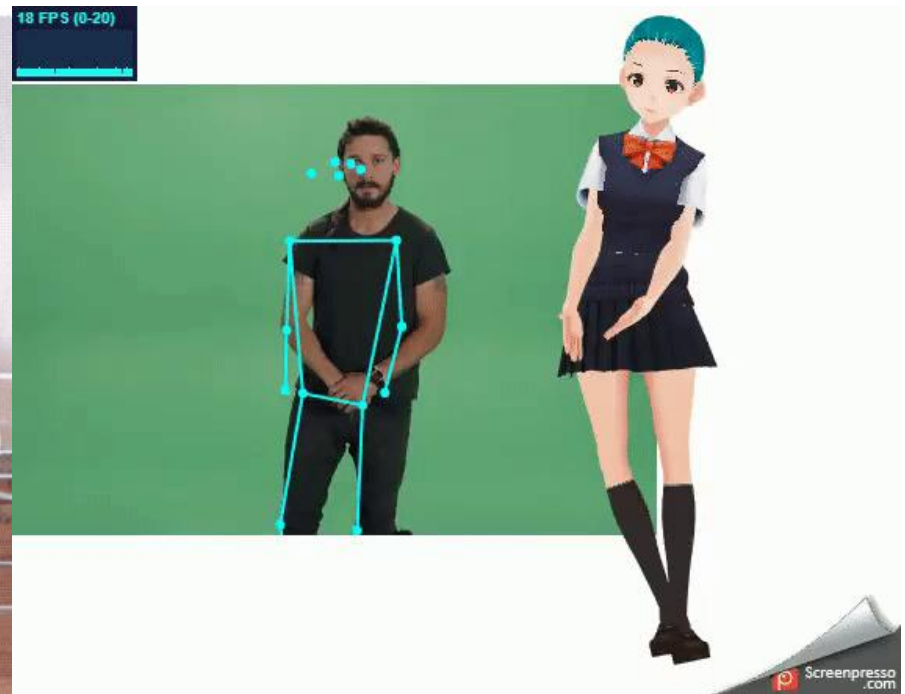


# PoseNet



# PoseNet

- How PoseNet works is that it first input an RGB image through a CNN.
- A pose algorithm is then used to decode the pose, pose confidence scores, keypoint positions and keypoint confidence scores.





# PoseNet

- The parameters that PoseNet first identifies the existence of a pose in a detected human from an image as the pose confidence score.
- Together with the pose confidence score, PoseNet also identifies 17 keypoints in a human joint/body part.
- These 17 keypoints are represented by both the spatial location together with the confidence.
- The confidence score from both the keypoint and the pose confidence score have a range between 0.0 and 1.0.





# Lab2: Pose Estimation with PoseNet





# Lab3: Action Recognition with PoseNet?



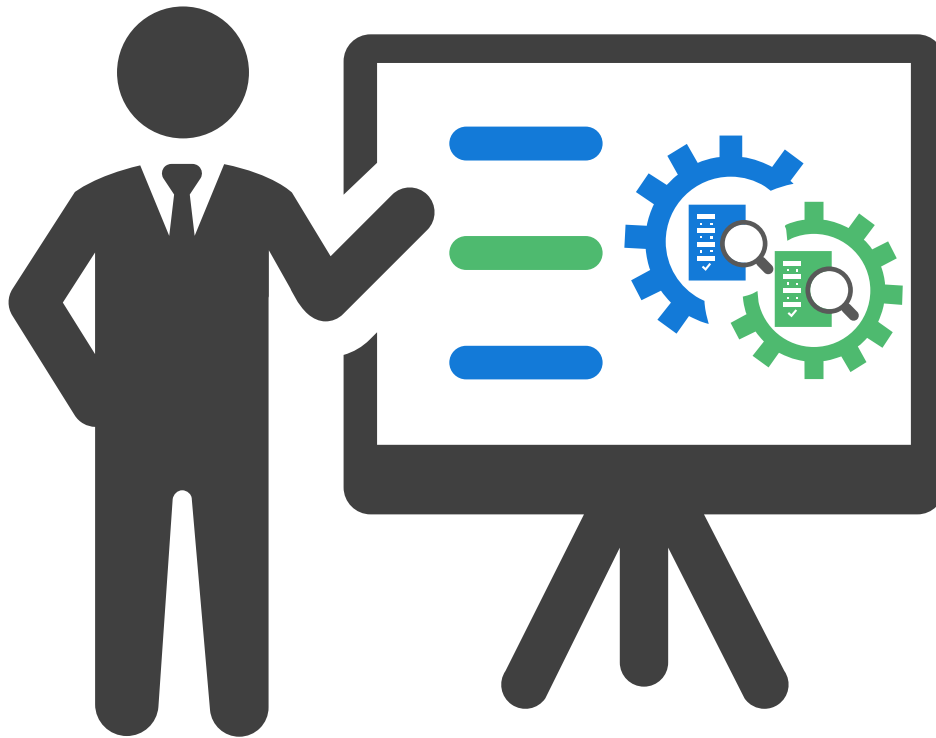
# References

---

- <https://www.kdnuggets.com/2019/08/2019-guide-human-pose-estimation.html>
- <https://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html#:~:text=Hardware,body%2Dtype%20and%20facial%20features.>
- [https://subscription.packtpub.com/book/game\\_development/9781849692380/6/ch06lvl1sec50/skeleton-tracking-with-the-kinect-sdk](https://subscription.packtpub.com/book/game_development/9781849692380/6/ch06lvl1sec50/skeleton-tracking-with-the-kinect-sdk)
- <https://medium.com/analytics-vidhya/understanding-openpose-with-code-reference-part-1-b515ba0bbc73>
- <http://densepose.org/>
- <https://arvrjourney.com/human-pose-estimation-using-openpose-with-tensorflow-part-2-e78ab9104fc8>
- <https://github.com/ildoonet/tf-pose-estimation>
- <https://github.com/CMU-Perceptual-Computing-Lab/openpose>
- <https://medium.com/@tibastar/mask-r-cnn-d69aa596761f>
- <https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c>
- <https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610>
- <https://www.youtube.com/watch?v=XGi-Mz3do2s>
- <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>



# Summary



Email  
Zack\_toh@rp.edu.sg

Telegram  
@zacktohsh

Source code:



Thank you!