

Weapon Detection using Deep Learning Architecture

By

Aayush Kubba (A18001)

Kamalendu Das (A18011)

Mayank Verma (A18014)

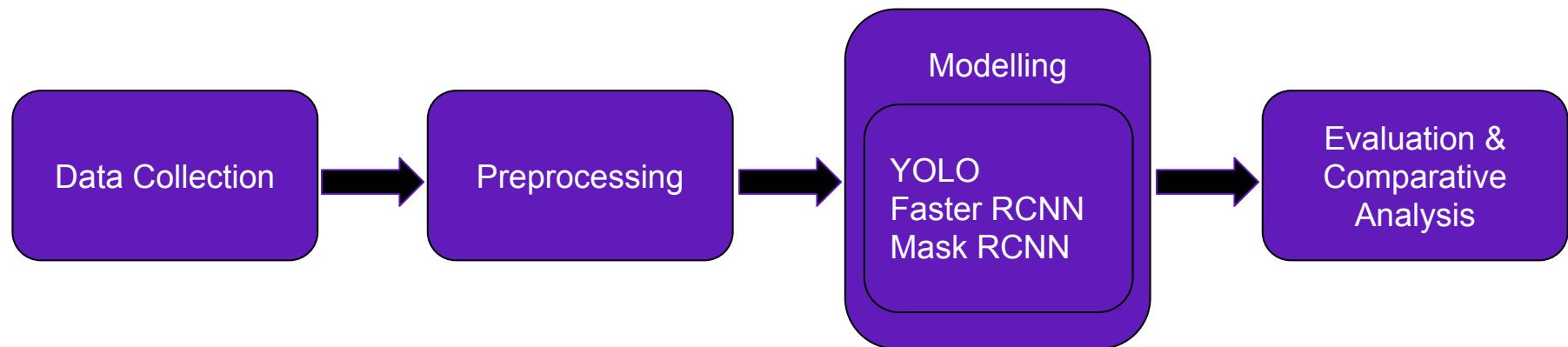
Shweta Mayekar (A18026)

Souvik Jana (A18028)

Problem Statement

Detection of Guns using different Object Detection Deep Learning Techniques - Faster RCNN and Mask RCNN.

Methodology



Data Collection

- Handgun Datasets from for all 3 approaches -
<https://sci2s.ugr.es/weapons-detection>
 - 1) Collected 3000 images for object detection with YOLO and Faster RCNN
 - 2) Created 1751 annotated images for object detection in Mask RCNN
 - 3) 3405 images for classification task using Transfer learning to reduce false positives
- ImageNet Database
- COCO Dataset

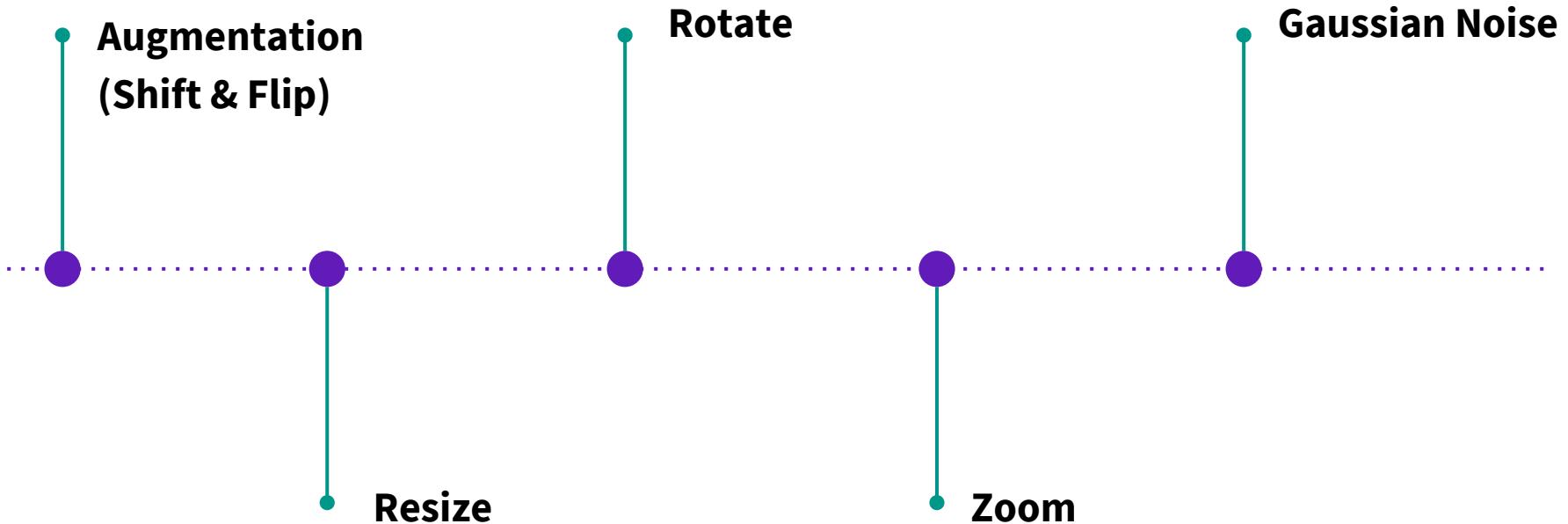
Annotation of Images for YOLO & Faster RCNN



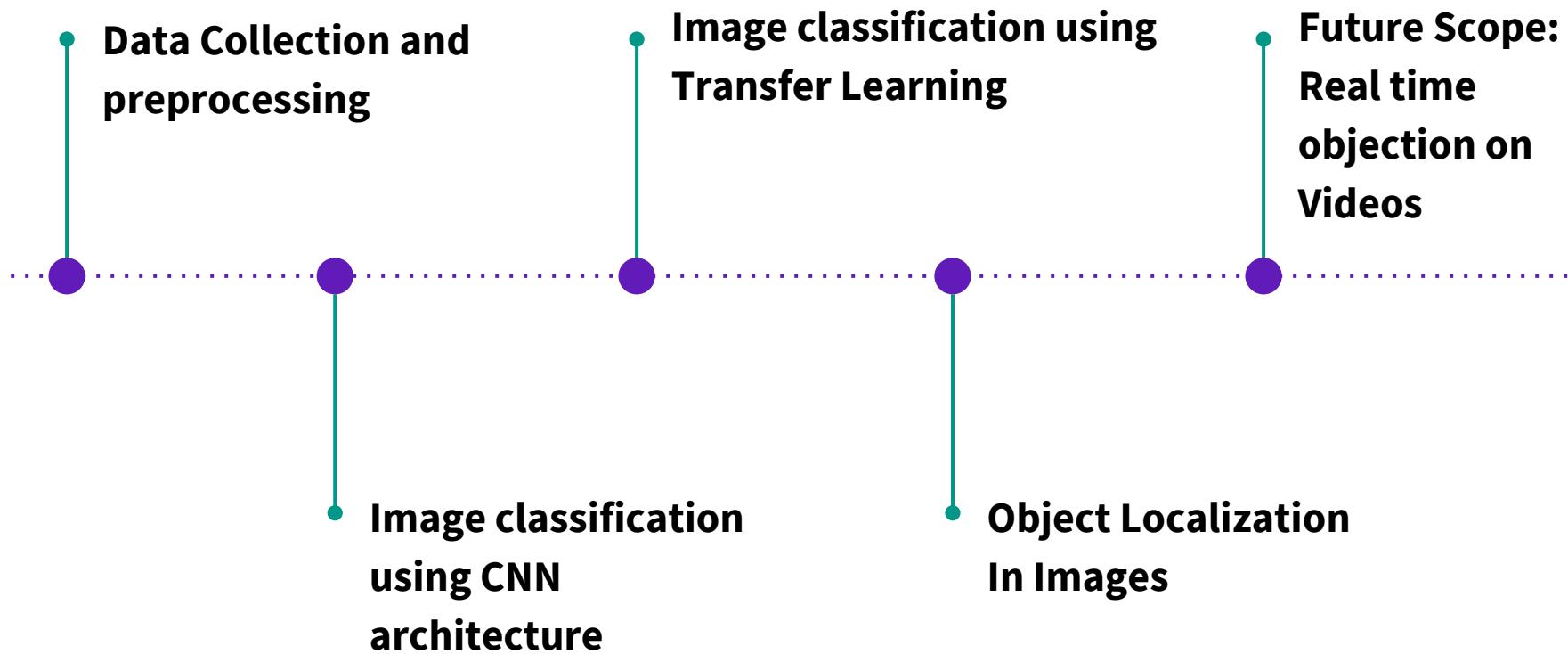
Annotation of Images for Mask RCNN



Pre-processing Images



MODELLING ROADMAP

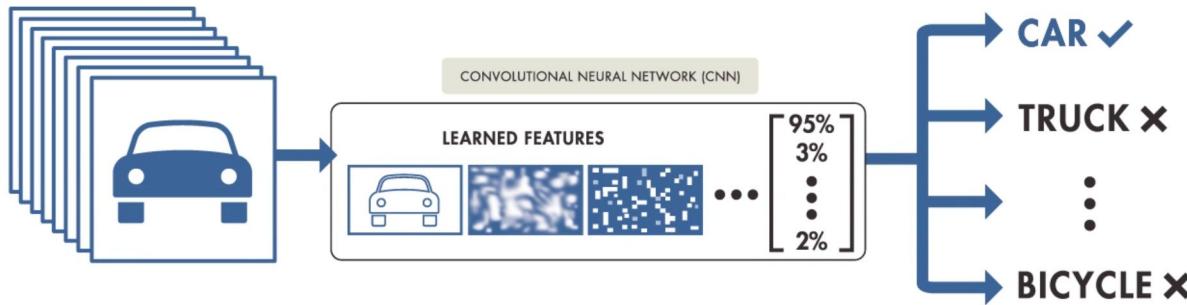


Transfer Learning

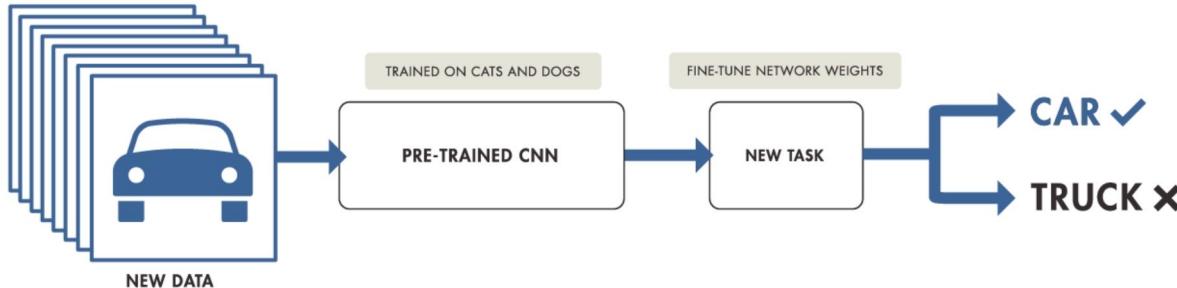
1. Transfer learning is a popular method in computer vision because it allows us to **build accurate models in a timesaving way** (Rawat & Wang 2017).
2. In computer vision, transfer learning is usually expressed through the use of **pre-trained models**.
3. Due to the high computational cost of training complex models, it is common practice to import and use models from published literature (e.g. VGG, Inception, MobileNet)

Concept of Transfer Learning

TRAINING FROM SCRATCH



TRANSFER LEARNING

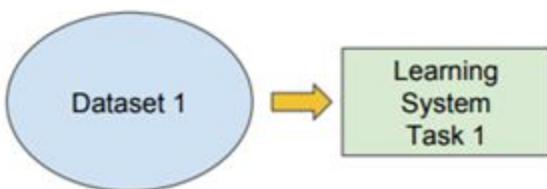


Traditional ML

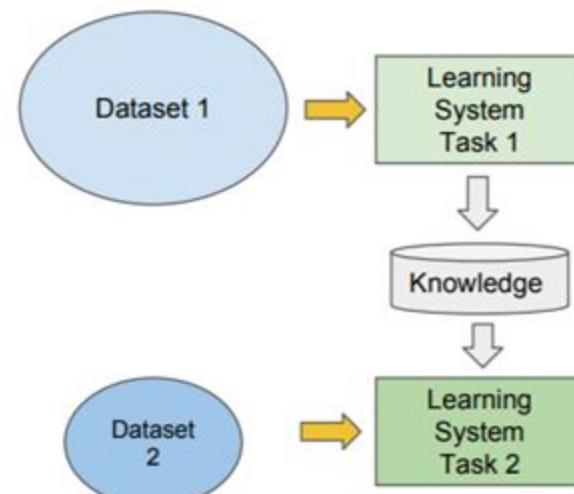
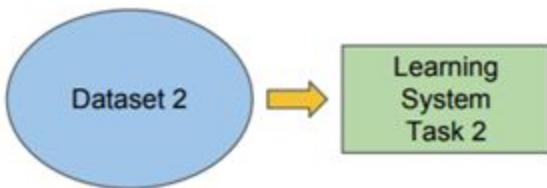
vs

Transfer Learning

- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data



Repurposing a pre-trained model

We fine-tune our model according to one of these strategies:

1. Train the entire model with the pre-trained models architecture(Computationally expensive and data issues)
2. Train some layers and leave the others frozen.

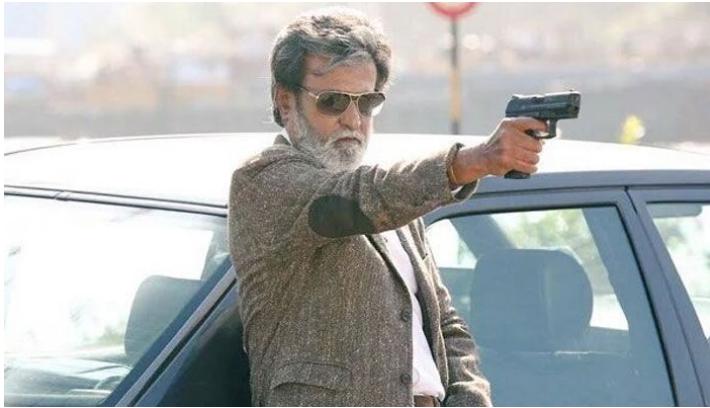
Image Classification Statistics

Model	Data Augmentation	Test Accuracy	Comments
CNN(Build from scratch)	No	0.66	
CNN(Build from scratch)	Yes	0.67	No significant boost
Pre-trained Model on Imagenet(Resnet,Inception V3, MobileNet)	No	0.87	
Pre-trained Model on Imagenet(Resnet,Inception V3, MobileNet) (With RMSprop and categorical_crossentropy as loss function)	Yes	0.91	Significant boost from original architecture

Image classification to object detection

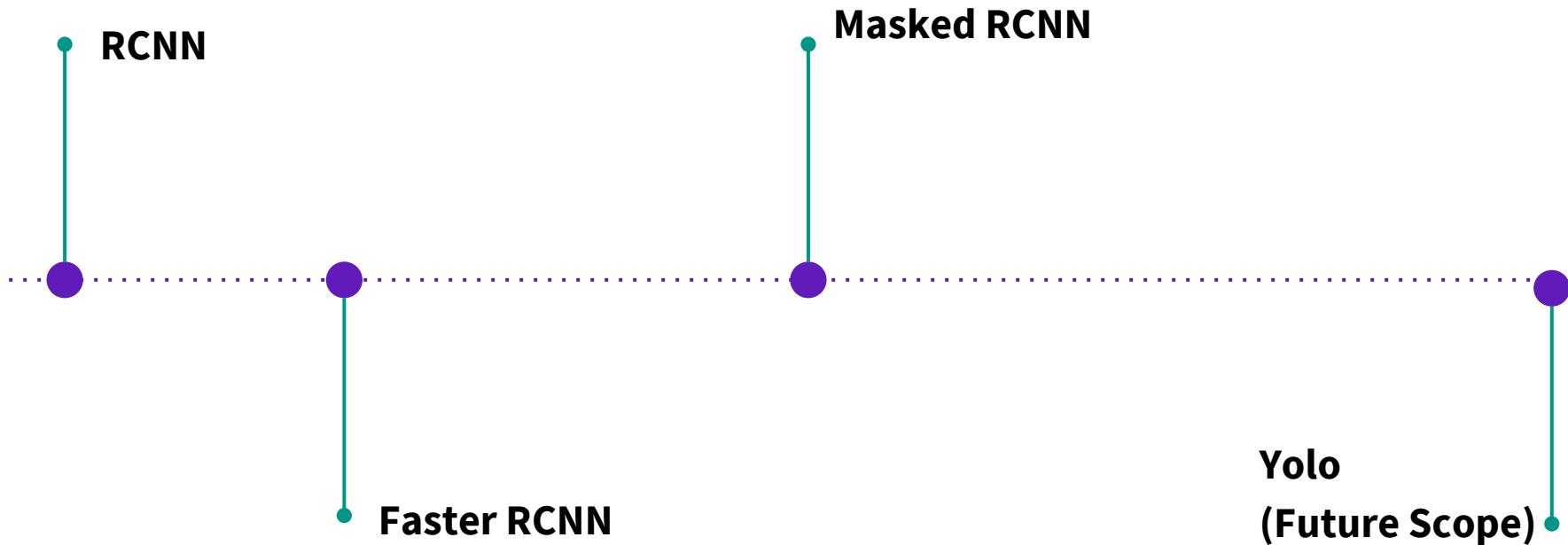
In Object detection, a bounding box is drawn around the object of interest to locate it within the image.

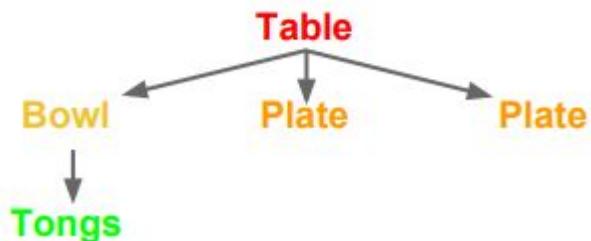
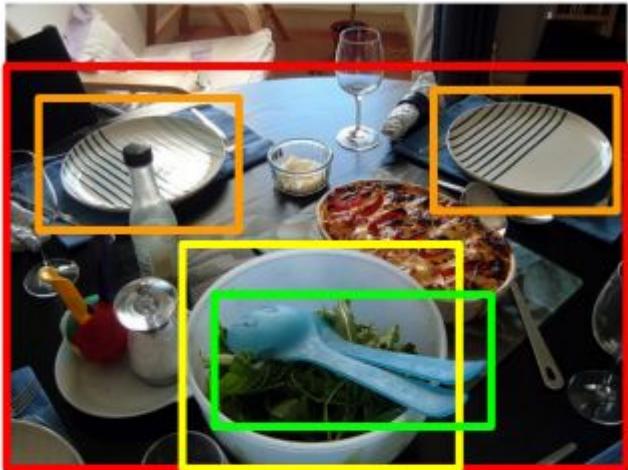
Why can't we just use CNN?



Many bounding boxes representing different objects of interest within the image which may not be known beforehand i.e. length of output layer is variable.

Object Detection ROADMAP





Sub-Segmentation maintaining Hierarchical Representation

1. Maintain Hierarchy
2. Generate many regions each of which belongs to at most one object.

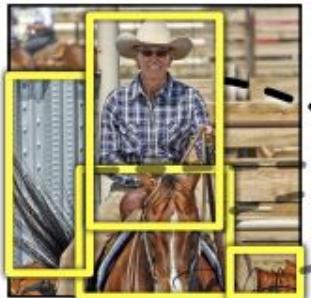
Detect Multiple Overlapping Object

R-CNN

R-CNN: *Regions with CNN features*

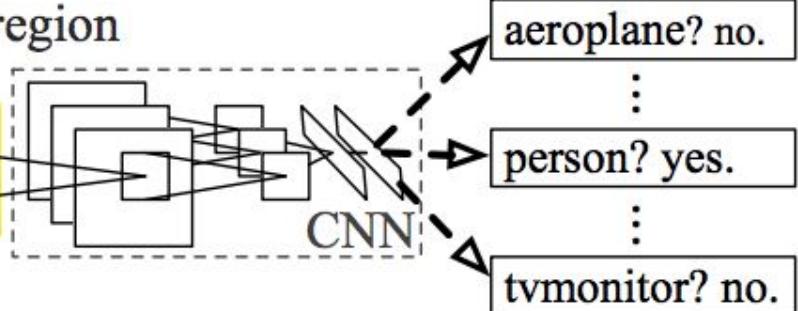


1. Input image



2. Extract region proposals (~2k)

warped region



3. Compute CNN features

4. Classify regions

Problems faced while building R-CNN

- It took a huge amount of time to train the network as it classified 2000 region proposals per image.
- Couldn't be scaled as it took around 47 seconds for each test image.
- The selective search algorithm(Greedy Approach) is a fixed algorithm. Hence, no learning was happening at that stage and it lead to the generation of bad candidate region proposals and hence bad object detection.

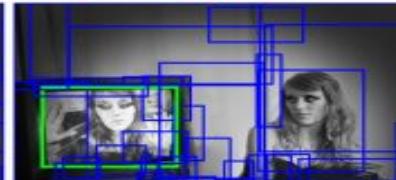
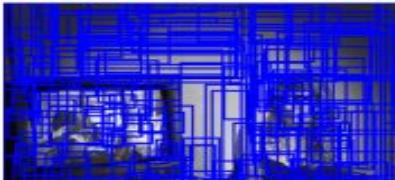
Selective Search

1. Idea: Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.
2. Recursively combine similar regions into larger ones.
Greedy algorithm:
 - From set of regions, choose two that are most similar.
 - Combine them into a single, larger region.
 - Repeat until only one region remains. This yields a hierarchy of successively larger regions.

Step 3: Use the generated regions to produce candidate object locations.



Input Image



Faster RCNN

**Concept of
Region
Proposal
Network(RPN)
& fixing time
complexity**

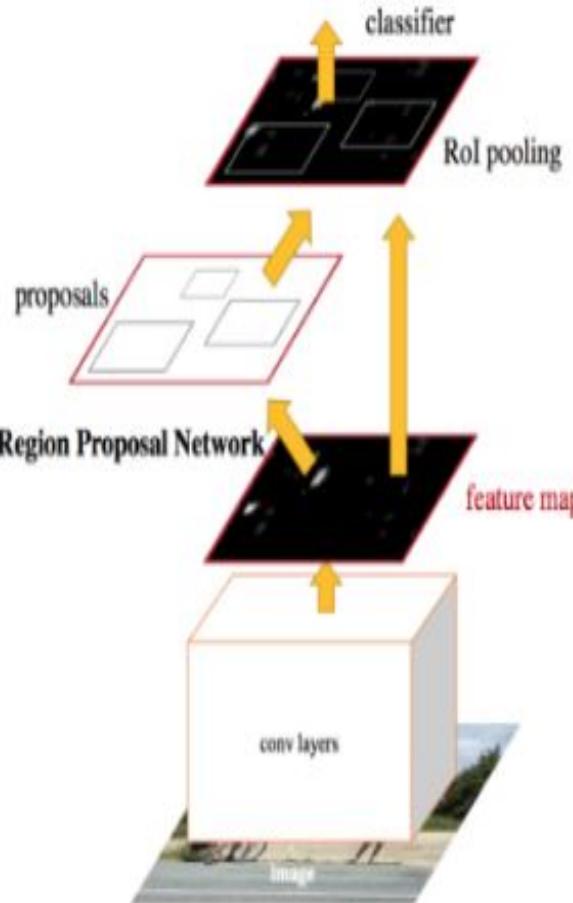


Fig. 7. An illustration of Faster R-CNN model. (Ima

- 3 things we have changed:
 1. CNN at first
 2. Instead of S.S use RPN
 3. Then ROI pooling.
- Train RPN:
 1. Each feature map creates anchors.
 2. Anchors have 3X3 boxes of diff ratios(1:1, 1:2, 2:1)
 3. Sampling, non-max-suppression used to reduce no of boxes.
 4. ROI based cropping pooling.

Faster RCNN

Detail
Architecture

1. Taken ResNet50 architecture at backend. With random initialization.
2. Train Samples: 2514. Val Sample: 486
3. Less amount of data on single class.
4. Optimised with Adam lr = $\exp(-5)$.
5. RPN Train: Positive sample IOU >0.7. Negative sample <0.3.
6. Threshold for bbox= 0.8
7. Regularizer prevented overfitting and false positive.
8. But less accuracy for different orientation of images.
9. Bounding box regression performance not well.
10. Time required for each image to read and write back to drive is 0.8 sec to 2.5 sec.

pistol: 97



pistol: 89



Detecting gun along with hand posture.
Good or Bad??

pistol: 88

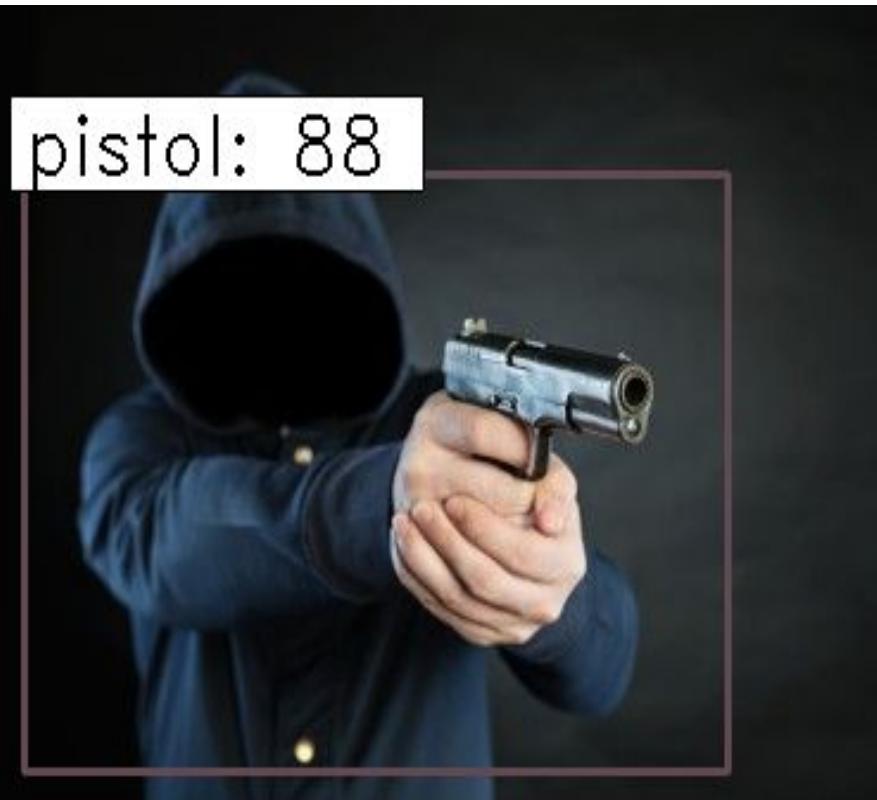


Further can be worked on roi classification. And non
max suppression.

Adam exp(-4) to exp(-5). Thresold 0.7



Interference of background



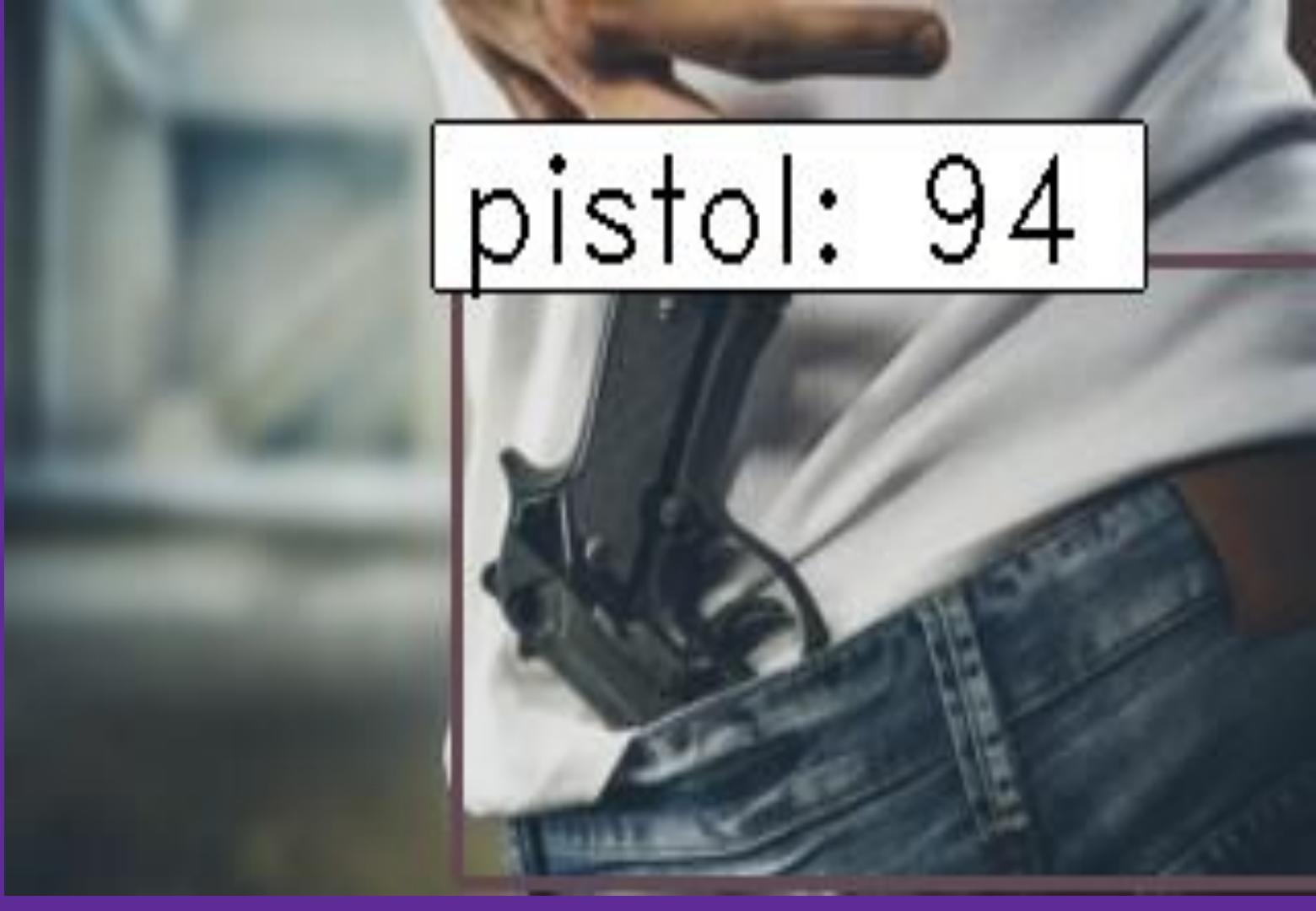


Just can't Hide

pistol: 92



pistol: 94



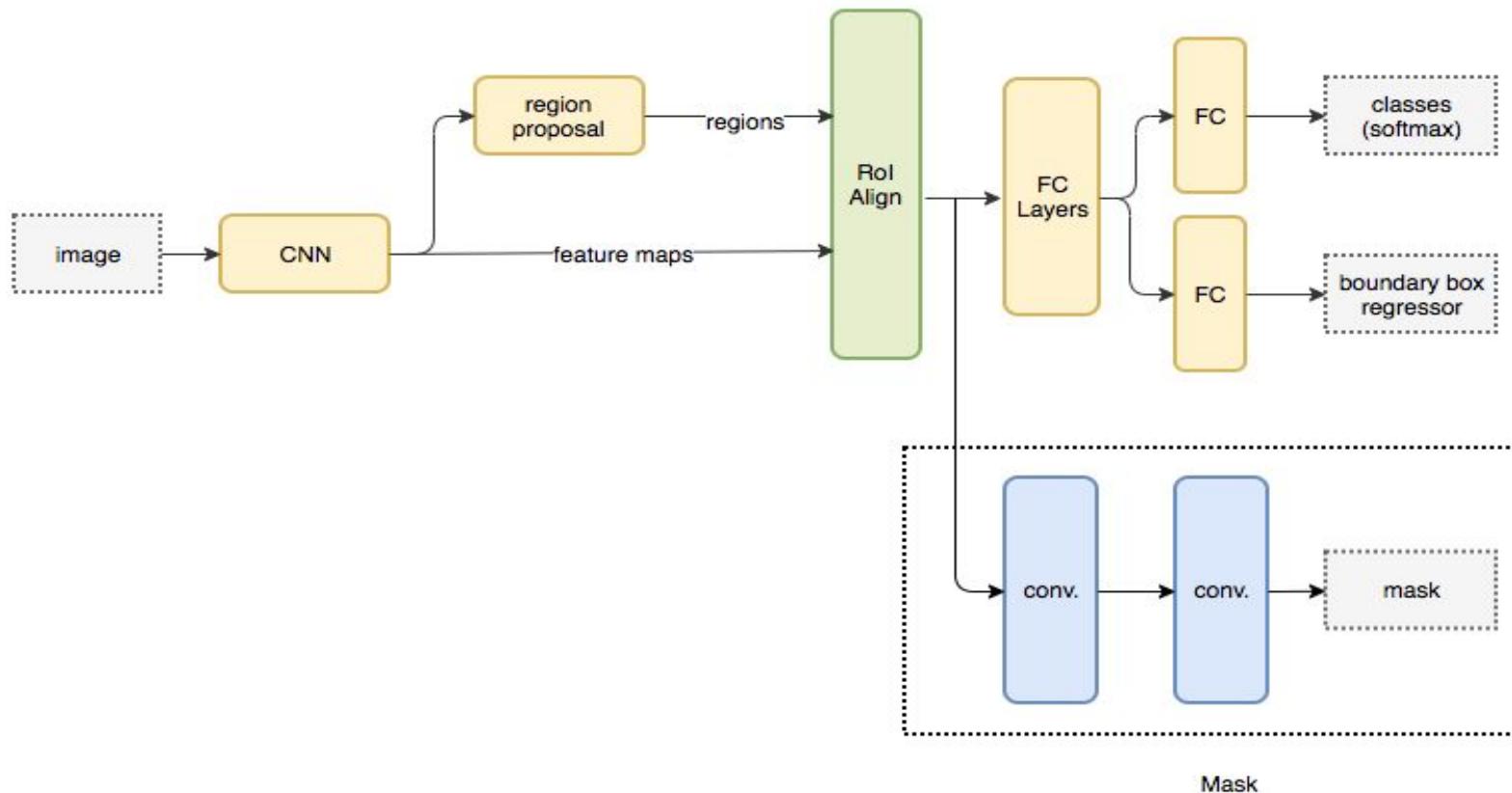
Mask RCNN

Official Paper published by
Facebook AI Research(FAIR) on
Jan 2018

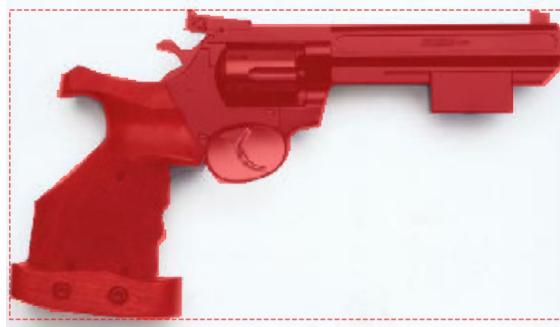
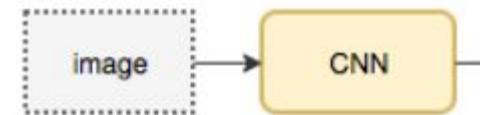
Best results till date on COCO
object detection challenge

Combines object detection with
image segmentation

Architecture of Mask RCNN



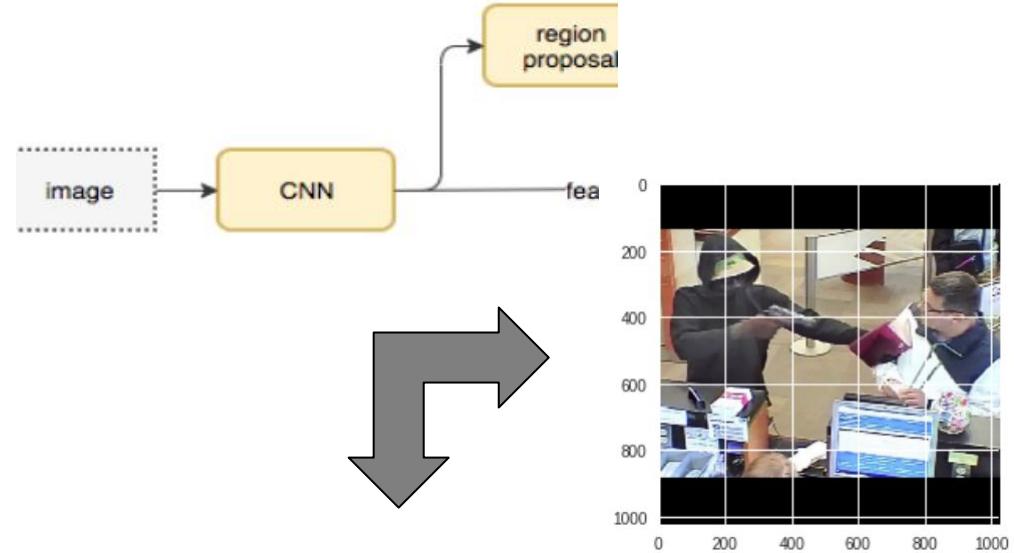
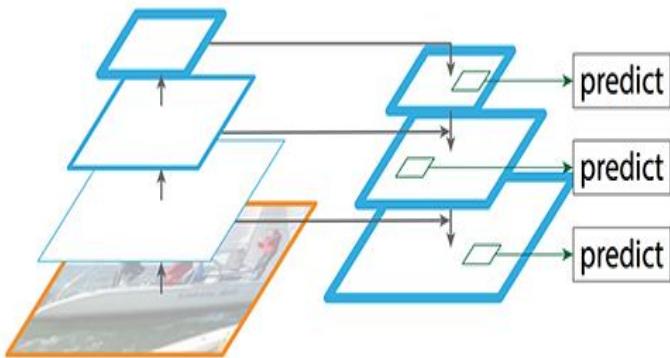
Inspecting Data for Mask RCNN



The ground truth for the Bounding Box and the Semantic Segmentation is created from annotated images.



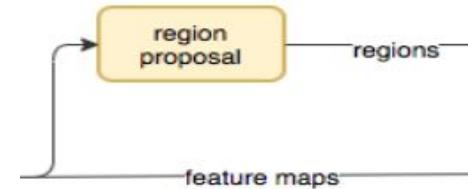
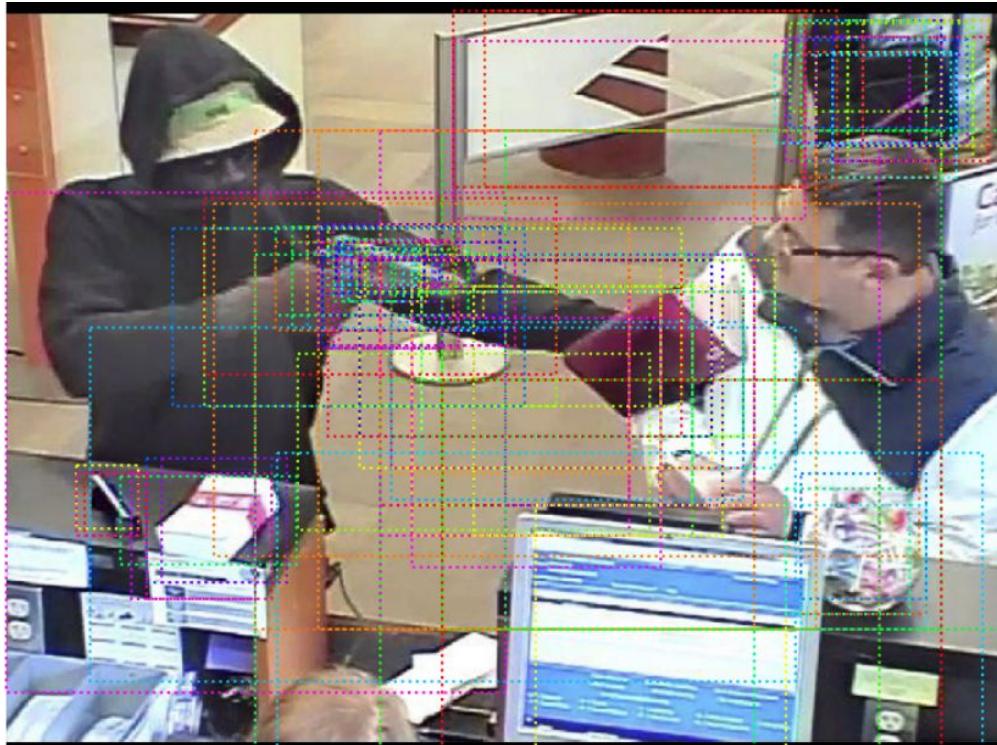
The FPN network



Feature Pyramid Network (**FPN**) is a feature extractor designed for such pyramid concept. It generates multiple feature map layers (**multi-scale feature maps**) with better quality information for **small** object detection.



Region Proposals Generation

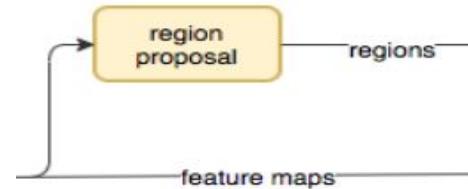
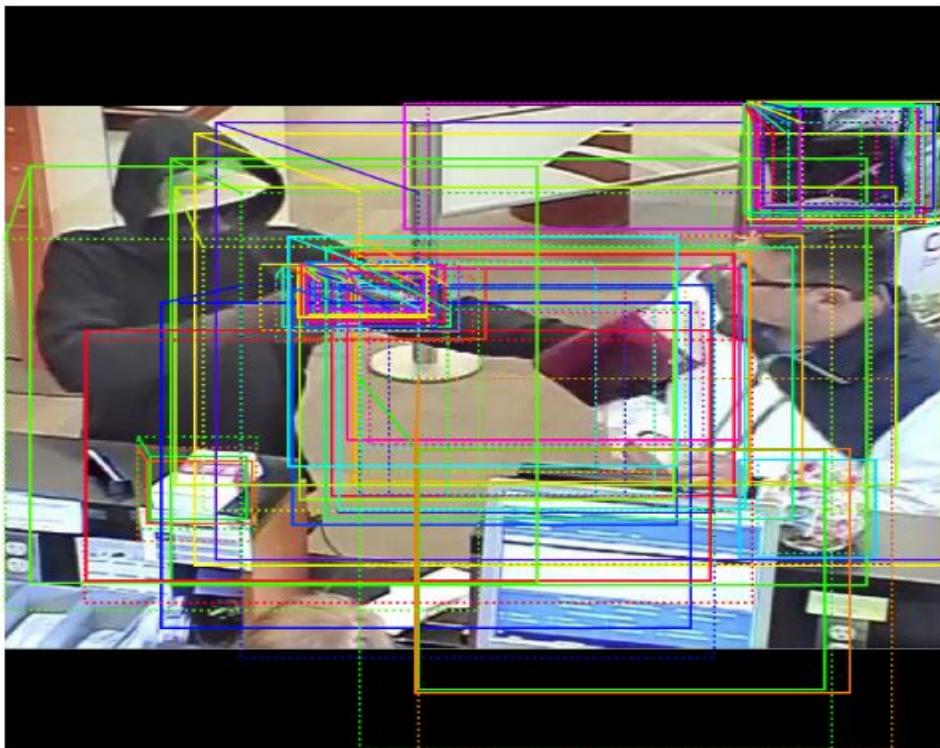


Step 1:

We create randomly 2000 region proposals or anchors for the image.

Only the top 100 have been shown here.

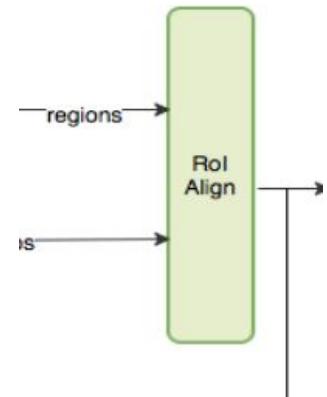
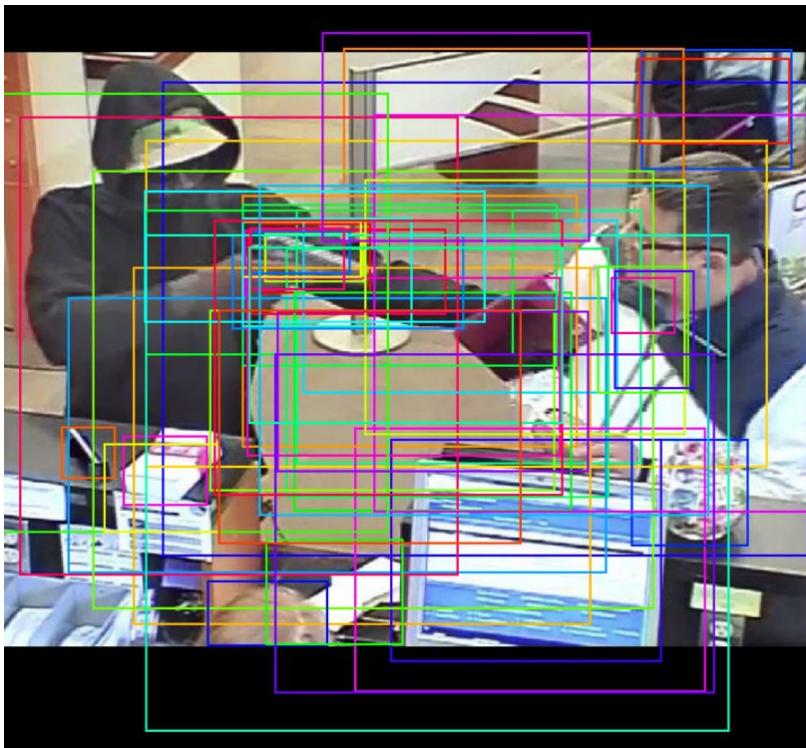
Anchor Box Refinement



Step 2

Anchor Box Refinement: A anchor might not be centered perfectly over the object. So the RPN refines the anchor box to fit the object better.

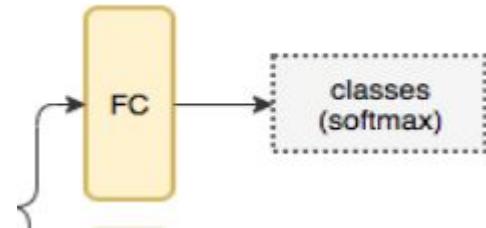
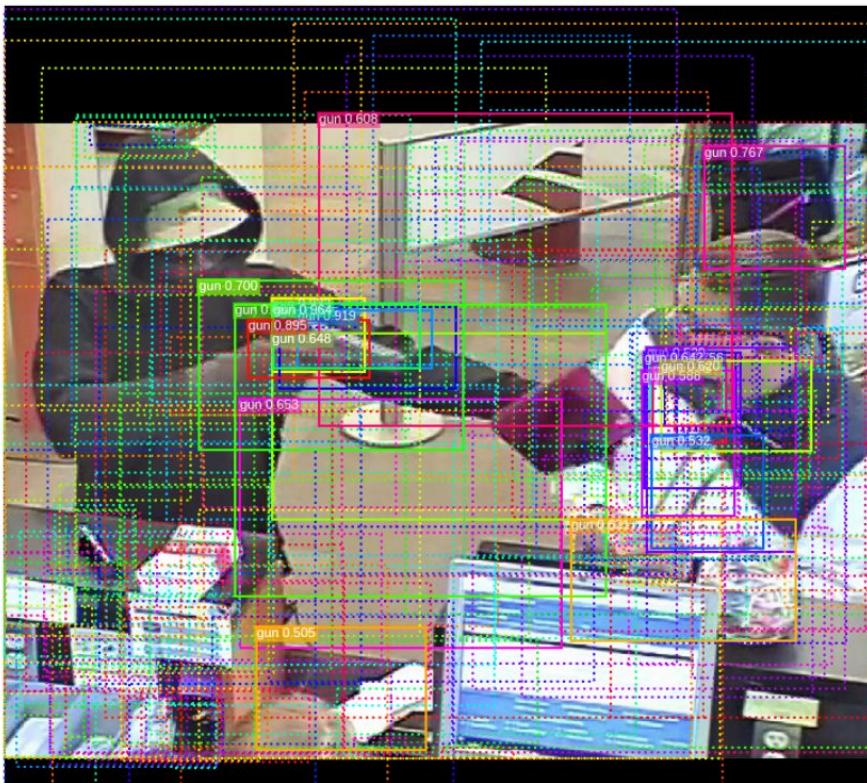
Final ROIs



Step 3:

The final top 50 ROIs after non max suppression are shown here.

Proposal Classification

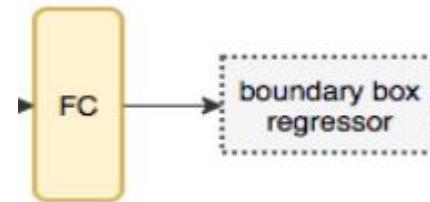
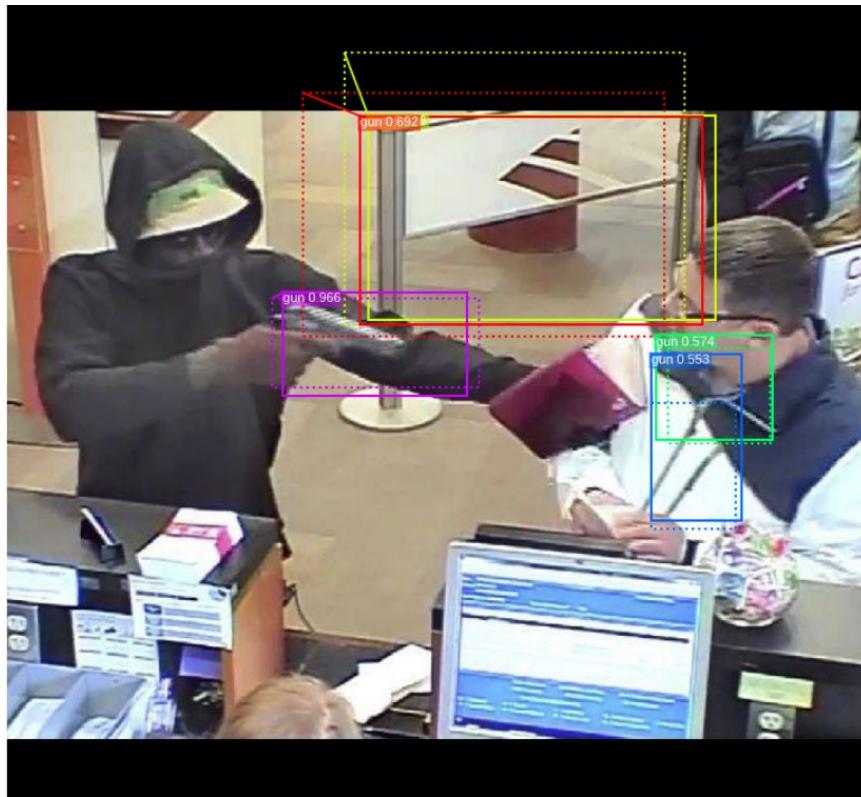


Step 4:

Proposal Classification running
a pretrained ResNet 101 with
COCO weights for classification

The solid lines shows detected
guns with accuracies

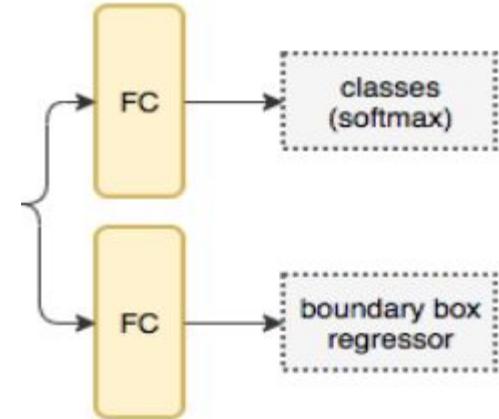
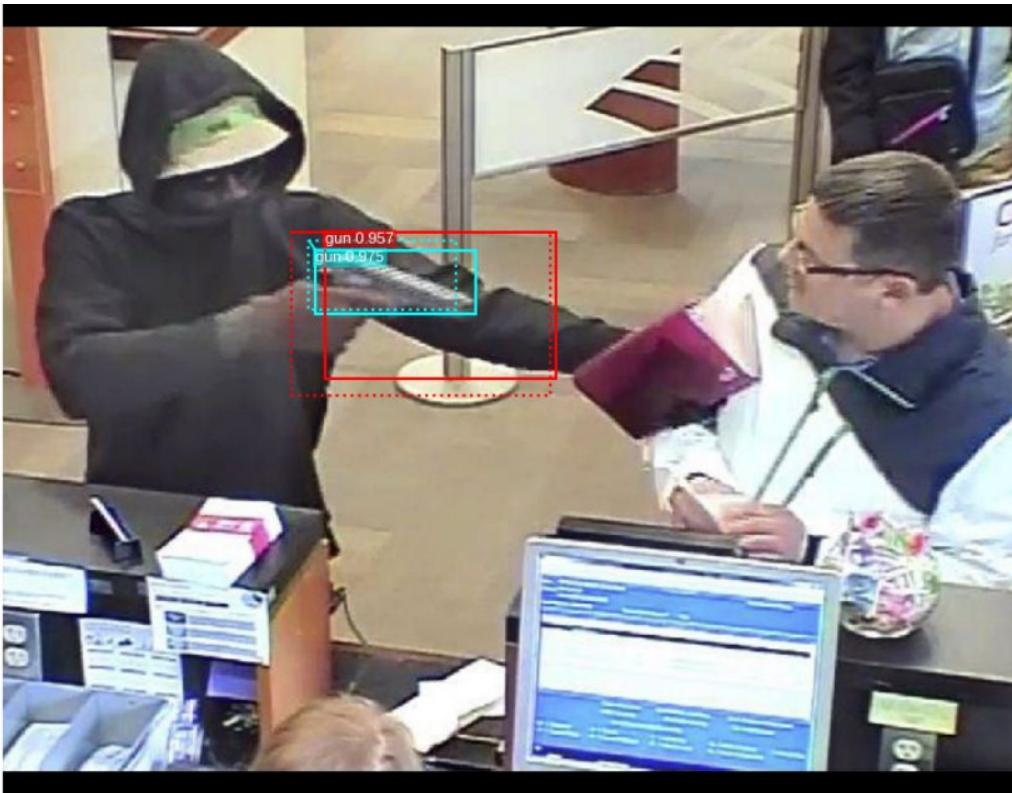
Bounding Box Regressor



Step 5:

Applying Bounding Box
Refinement using bilinear
interpolation

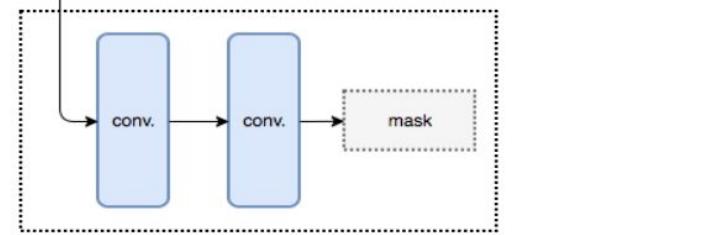
Filtering Low Confidence Detections



Step 6:

We filter out images with accuracies lesser than 93 % percent

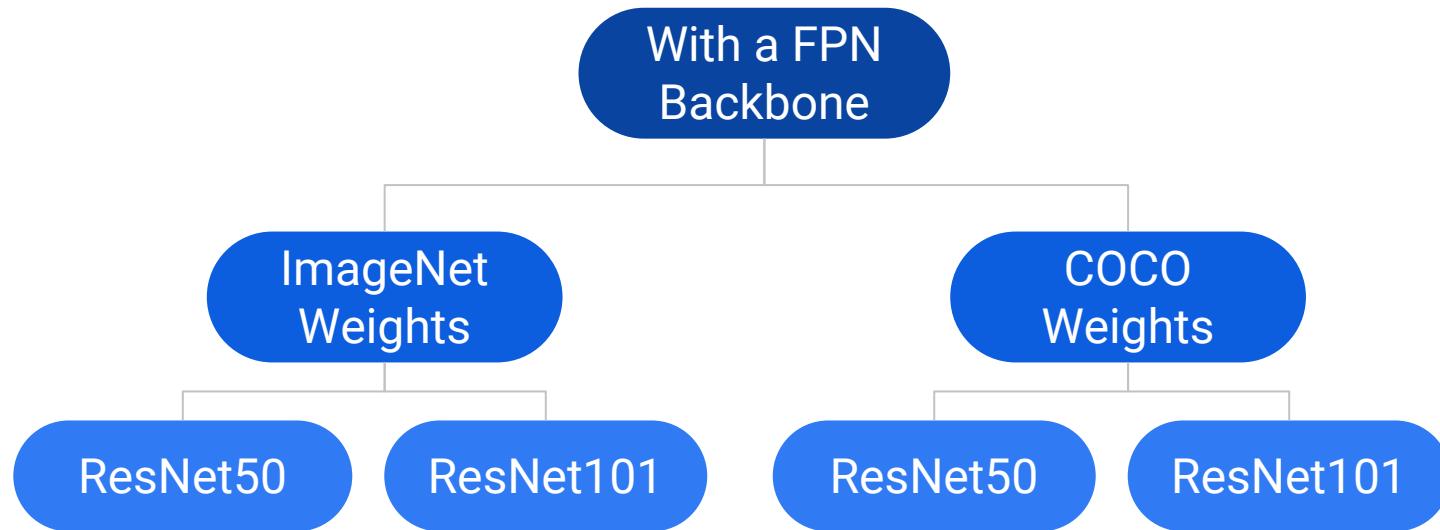
Applying Mask



Step 7:

The masks are generated by connecting two convolution layers back to back and getting the output of the convolution layer as a feature map

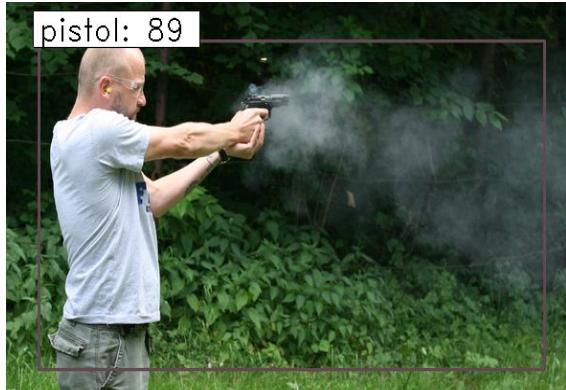
4 Approaches under Mask RCNN



Comparative Analysis under Mask RCNN

Approaches under MASK RCNN	Validation mask_loss	Validation class_loss
ResNet50 (ImageNet)	0.5927	0.0597
ResNet101 (ImageNet)	0.5885	0.0403
ResNet50 (COCO)	0.2532	0.0536
ResNet101 (COCO)	0.2294	0.0527

Comparative Analysis: Faster RCNN v/s Mask RCNN



But we did have some False Positives!



Misclassification errors due to -
Overfitting & Inability of training
more blurry images

Fine Tuning the Network

Transfer learning to reduce FPs

Classification Task:

3405 (800 guns/ 2605 not guns) images trained on ResNet101 (ImageNet Weights) improved the scores of ResNet101 ImageNet Mask RCNN Model.

Using Transfer learning we reduced false positives with metrics -

Validation Mask_loss = 0.475 Validation Class_loss = 0.0383



Another Challenge with Exploding Gradients

Review Weight Stats

```
[11]: # Show stats of all trainable weights  
visualize.display_weight_stats(model)
```

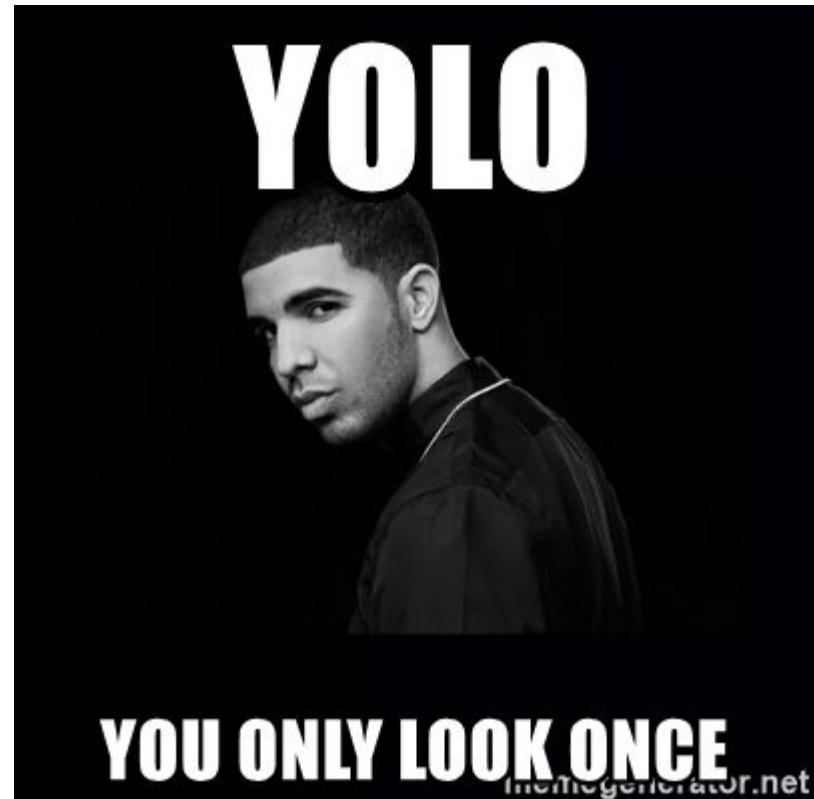
WEIGHT NAME	SHAPE	MIN	MAX	STD
conv1/kernel:0	(7, 7, 3, 64)	-0.8616	+0.8451	+0.1315
conv1/bias:0	(64,)	-0.0002	+0.0004	+0.0001
bn_conv1/gamma:0	(64,)	+0.0835	+2.6411	+0.5091
bn_conv1/beta:0	(64,)	-2.3931	+5.3610	+1.9781
bn_conv1/moving_mean:0	(64,)	-173.0470	+116.3013	+44.5654
bn_conv1/moving_variance:0*** Overflow?	(64,)	+0.0000	+146335.3594	+21847.9668
res2a_branch2a/kernel:0	(1, 1, 64, 64)	-0.6574	+0.3179	+0.0764
res2a_branch2a/bias:0	(64,)	-0.0022	+0.0082	+0.0018

EXCEPTIONAL RESULTS with MASK RCNN



Wait..What about Speed?

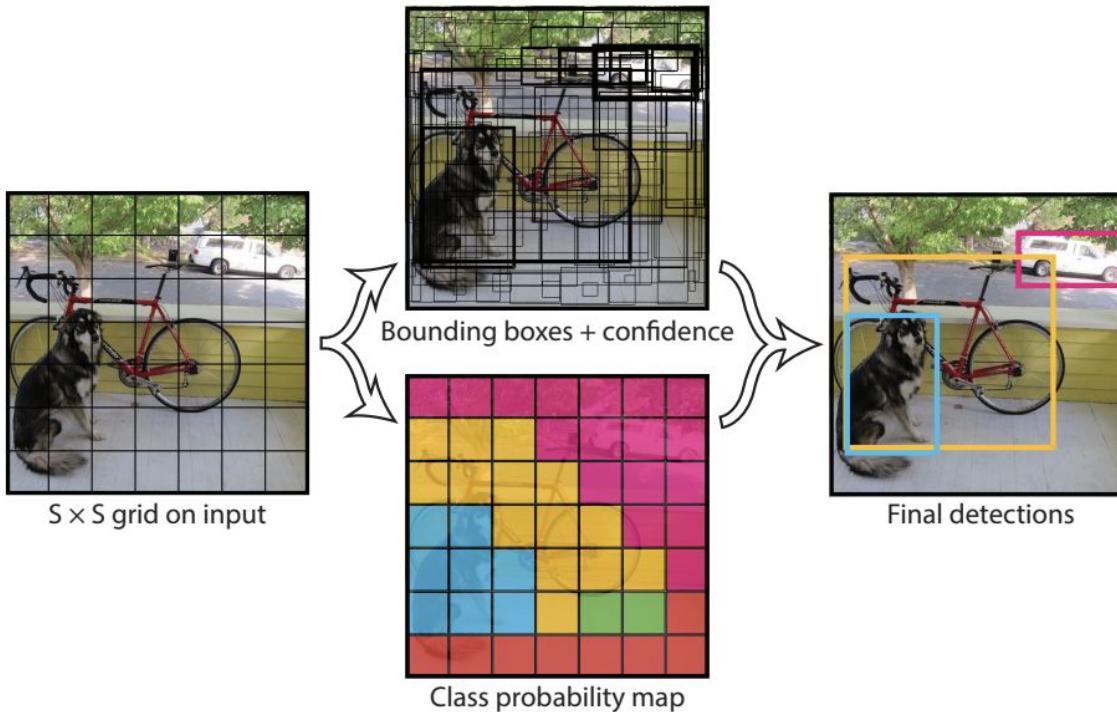
- You only look once (YOLO) is a state-of-the-art, real-time object detection system.
- Mobile App Demo.(Reference:
<https://github.com/gauravsawant/Weapon-Detector>)
- **How?** The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object.



~ YOLO is out of scope for this implementation.



Grid to Bounding Boxes to Detections with Classification



Speed Vs Accuracy ?

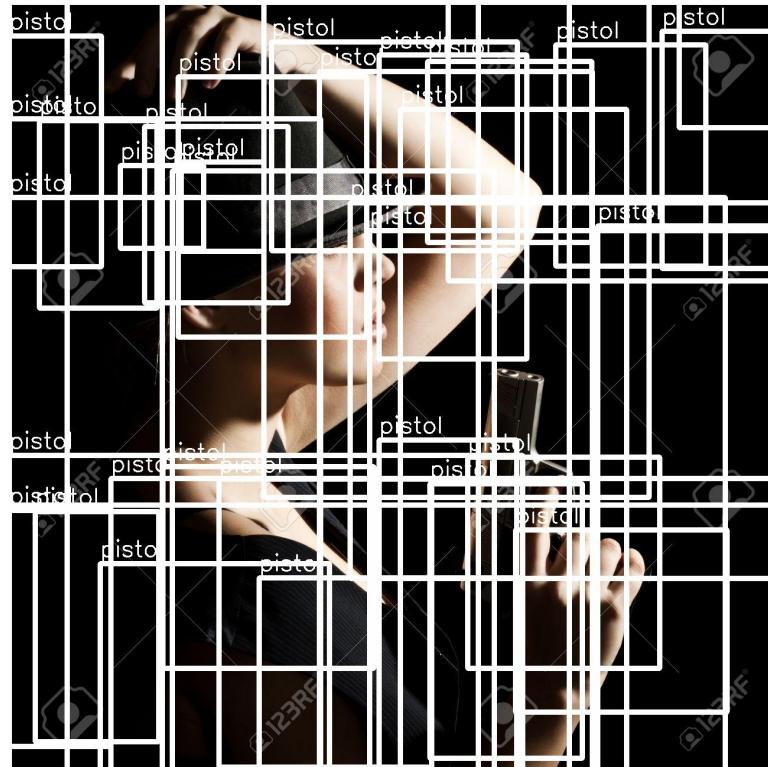
YOLO makes predictions with a single network evaluation due to which it is much faster than other object detection algorithm.

Limitation - YOLO algorithm struggles a lot with small objects within the image.

Model	Speed of Detection(On a single frame)
Faster RCNN	300 ms
Masked RCNN	350 ms
YOLO	10 ms

Note: Speed will vary with different CPU architectures. These figures are from GPU used on Google Colab

Glimpse of our bad predictions using YOLO



Overall Challenges we faced...

1. Data and time constraints for YOLO.
2. Annotating images manually for masked and faster RCNN.
3. Shortage of Hardware Requirements for video and real time analysis.
4. Couldn't try experimentations with training due to GPU limitation on Google colab and CC Labs.
5. Steep Learning curve while studying for the project.

Future Scope >>>

1. Real-Time implementation with YOLO.
2. Unified loss metric(like Mean Average Precision) for all the three approaches
3. Detecting categories of weapons and identifying fake from real ones
4. Detecting handheld guns using Pose and Human Key point Estimation.
5. Ensemble techniques for detection combining Instance segmentation with Human Keypoint estimation
6. Creating a model API and pushing it to realtime deployment in a drone or CCTV using a Raspberry PI
7. Trying out more faster algorithms like MXNet or MobileNet for gun detection.
8. Trying out RNN based approach for creating automated annotations(DEXTR and PolygonRNN++)

THANK YOU!