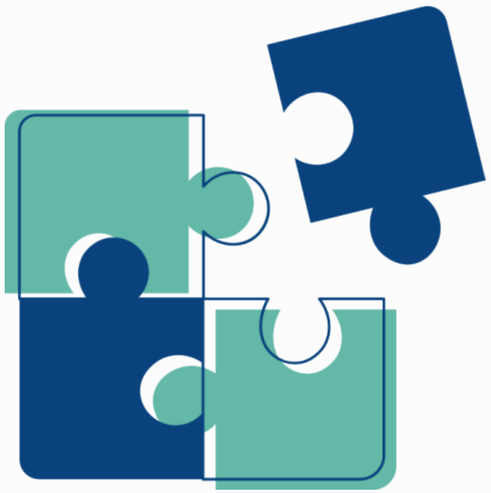
Real-Time Sign Language Detection System

Team 3

Emilio Cabrera, JT Flume, Connor Gilmore, Yashpreet Kaur, Junsu Kim



Content

- Problem Statement and Background
- Dataset Overview
- Object Detection Overview
- The model: Yolov4
- Model training
- Model results
- Conclusion and next steps

Background

- Over **1MM** people communicate using ASL
- **93%** of deaf customers want to communicate in ASL
- **$86B** market
- **Less than 10** not-widely used ASL machine translation services available

Increasing Accessibility

- Leverage computer vision to serve an underserved community
- **98%** of deaf people who do not receive education in sign language
- **72%** of families who do not sign with their deaf children

Why Austin Makes the List as a Top Deaf-Friendly City

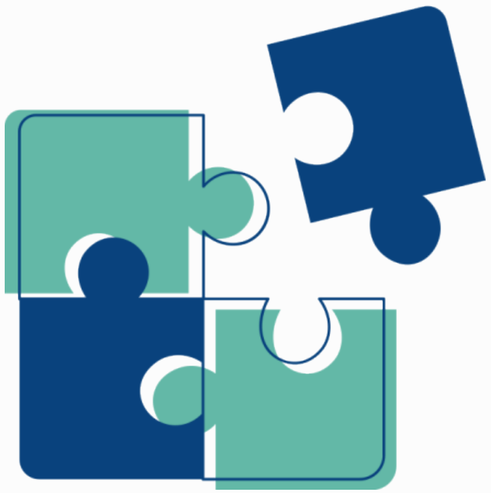
CSD

Problem Statement

Build an American Sign Language (ASL)

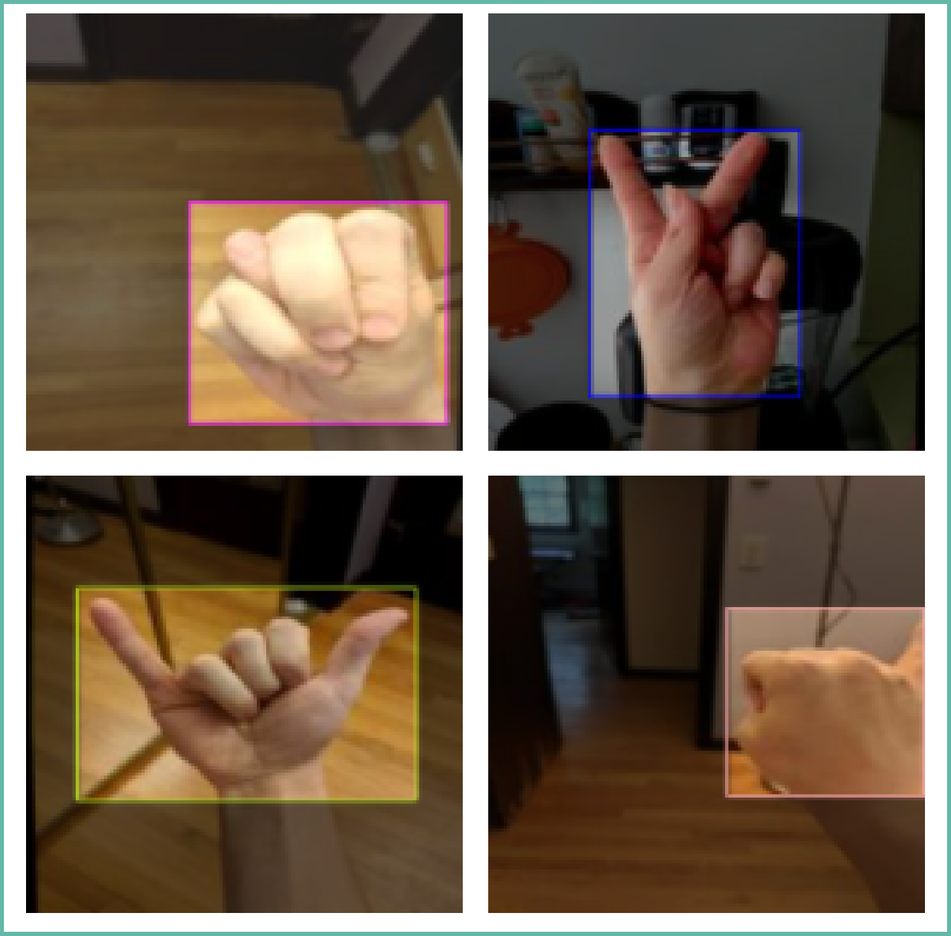
Detection system that works with:

- Images
- Videos
- Real-time sign language inputs



Dataset Overview

ASL Data



- **1,512** jpg files in training dataset
- **26** alphabets (classes)
- Corresponding input image format used like yolo txt/ tfrecord

Object Detection Overview

Classification Vs. Detection

Object Localization

determine where objects are located in a given image

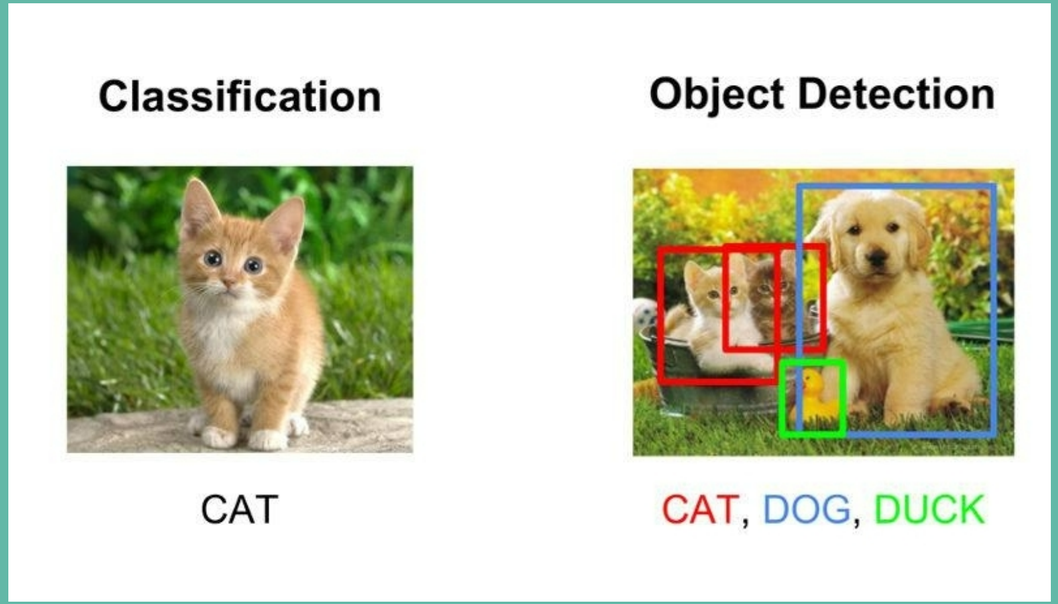
+

Object Classification

determine which category each object belongs to

=

Object Detection



Variables:

- class_name
- bounding_box_top_left_x_coordinate
- bounding_box_top_left_y_coordinate
- bounding_box_width
- bounding_box_height

Object Detection Methods

Two Stage Detectors

- Accuracy: More Accurate
- Computational Speed: Slower
- Examples:
  - R-CNN
  - Fast(er) R-CNN

Single Stage Detectors

- Accuracy: Slightly Less Accurate
- Computational Speed: More Efficient
- Examples:
  - YOLO
  - SSD

The advantage of a one stage detector is the speed it is able to make predictions

quickly allowing a real time use

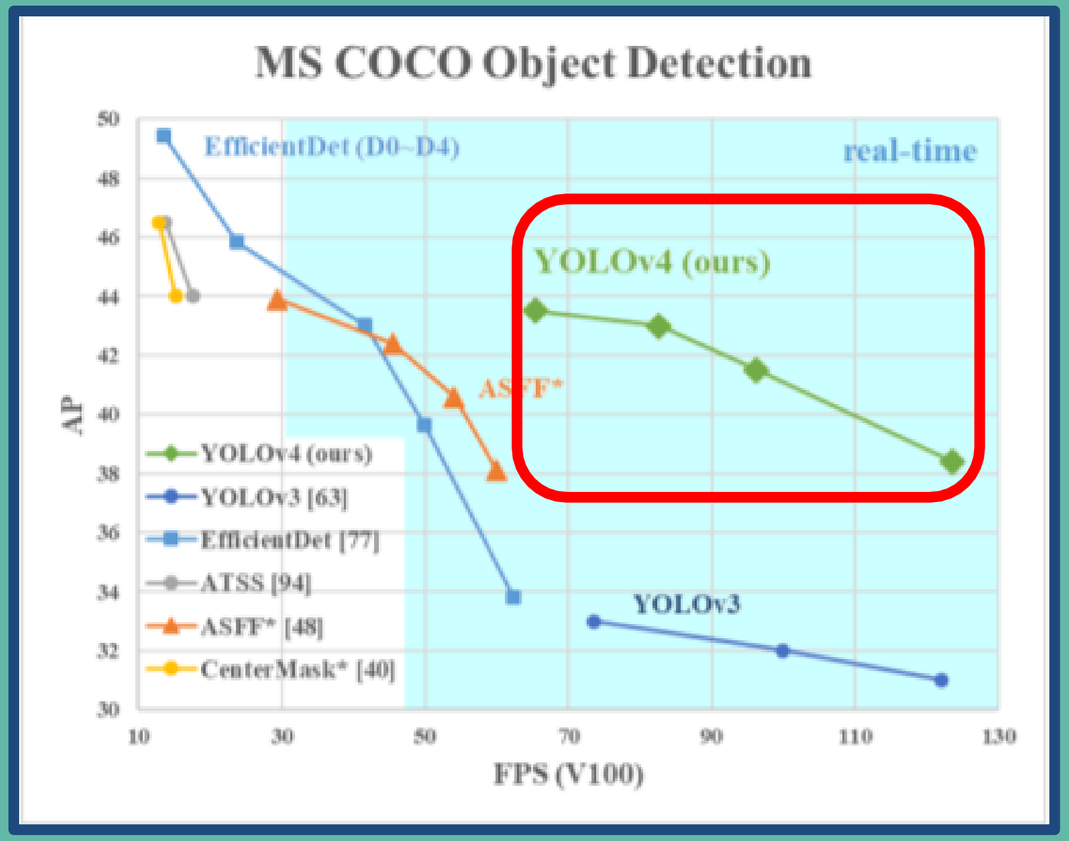
Why YOLO?

2016
https://arxiv.org/abs/1506.02640

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Source: You Only Look Once: Unified, Real-Time Object Detection
Authors: Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

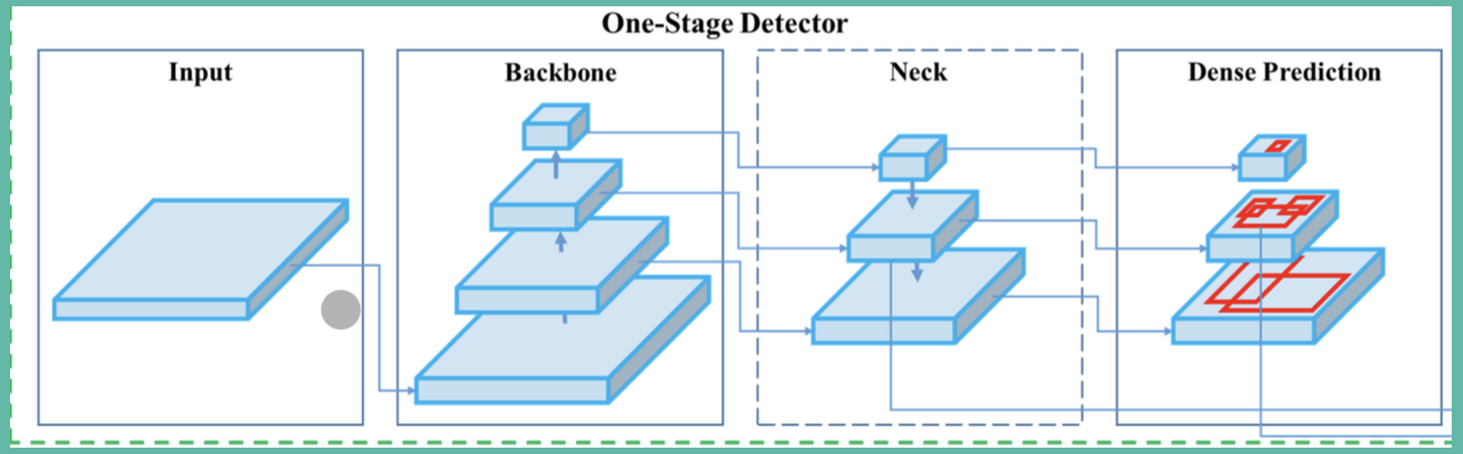
2020
https://arxiv.org/pdf/2004.10934.pdf



Source: YOLOv4: Optimal Speed and Accuracy of Object Detection
Authors: Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao

The model: Yolov4

YOLOv4 architecture



1. **Backbone Network -** Feature Formation: CSPDarknet53
    - Backbone network for an object detector is typically pretrained on ImageNet classification

1. **Neck** - Feature Aggregation - PANet
    - Mix and combine the features formed in the ConvNet backbone to prepare for the detection step

1. **Head** - The Detection Step
    - Perform dense prediction, composed of a vector containing the coordinates of the predicted bounding box (center, height, width), the confidence score of the prediction and the label

YOLOv4: What's new?

- **Bag of Freebies**
  - Improve performance of the network without adding to inference time
    - Data Augmentation
      - Mosaic Data Augmentation
      - Self-Adversarial Training (SAT)

- **Bag of Specials**
  - Add marginal increases to inference time but significantly increase performance
    - Activation functions
      - Mish instead of ReLU
    - DIoU
    - Cross mini-Batch Normalization (CmBN)

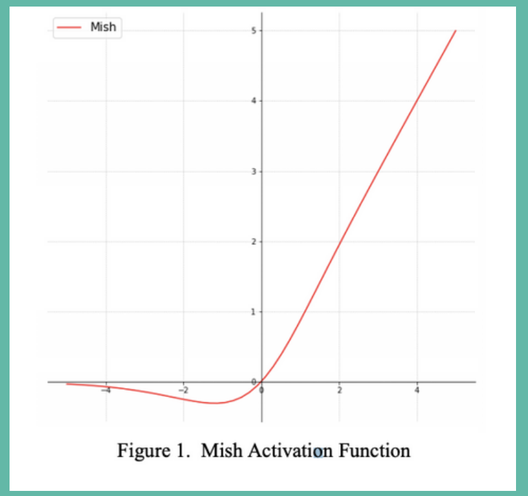
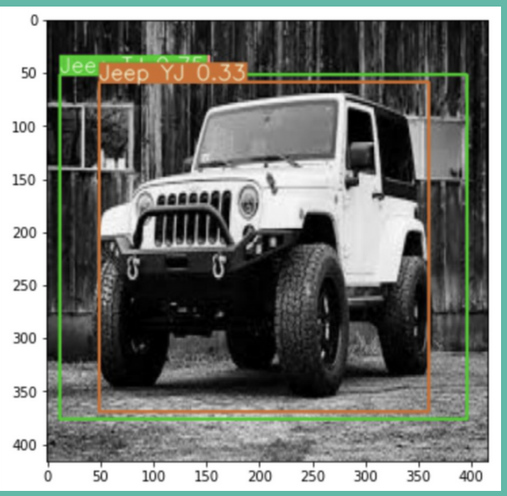


Figure 1. Mish Activation Function



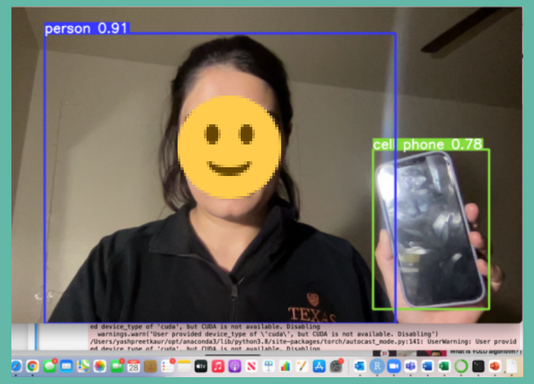
YOLOv4-tiny

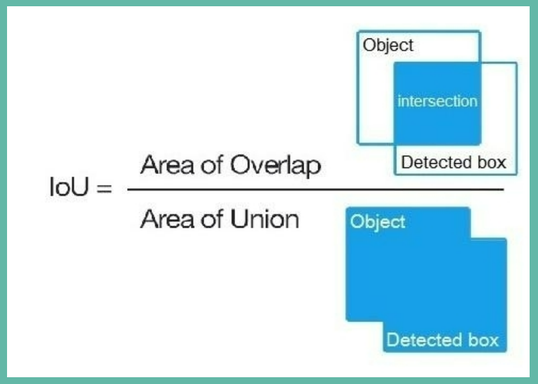
- **Compressed** version of yolov4
- Trained from **29** pre-trained convolutional layers compared to **137** yolov4 layers
- FPS (yolov4-tiny)= **8 times** FPS (yolov4)
- Accuracy (yolov4-tiny)= **2/3** * Accuracy (yolov4)

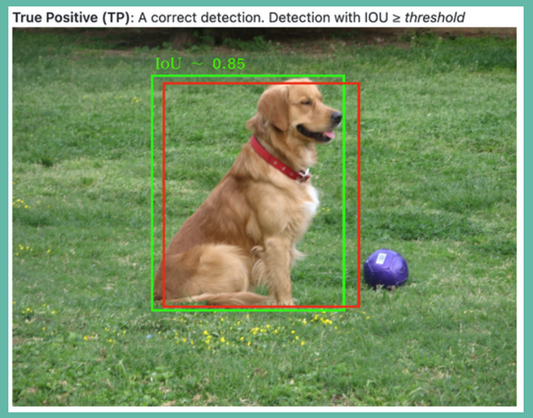
- **Real time object detection**
  - Faster inference time more important than precision or accuracy
  - yolov4-tiny better than yolov4

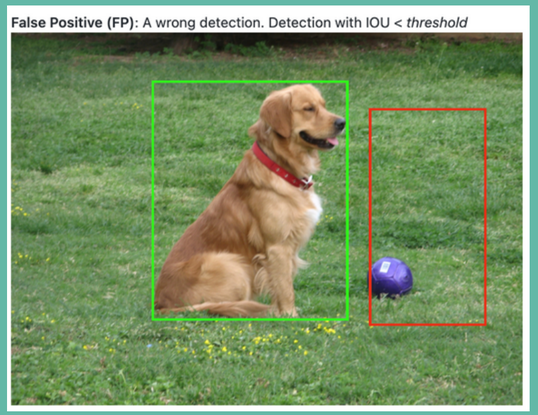
Scoring metrics

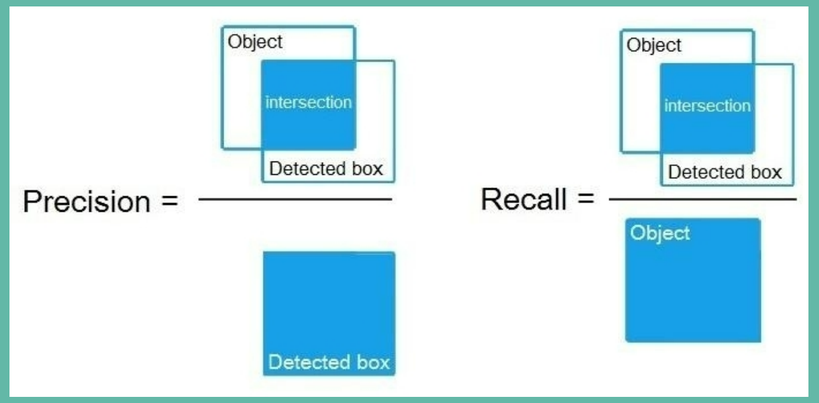
1. Confidence score
2. Intersection over Union (IoU)
3. Confusion Matrix
4. Precision/ Recall
5. Average Precision (AP)/ mean Average Precision (mAP)

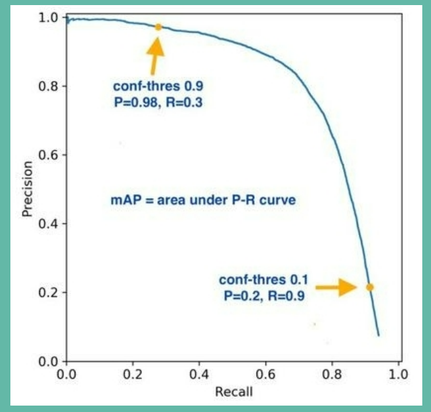


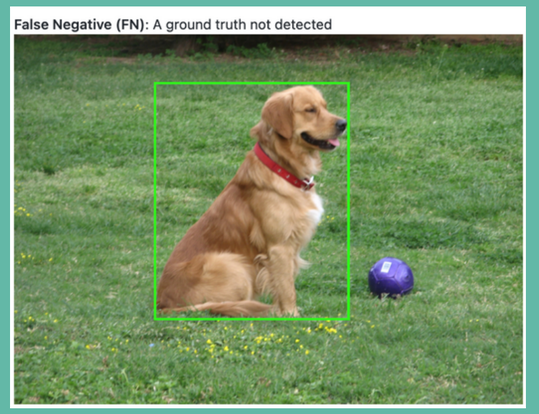












Model training

Model Setting

For each iteration
- 64 images to the network
- 4 samples
- 0.9 momentum

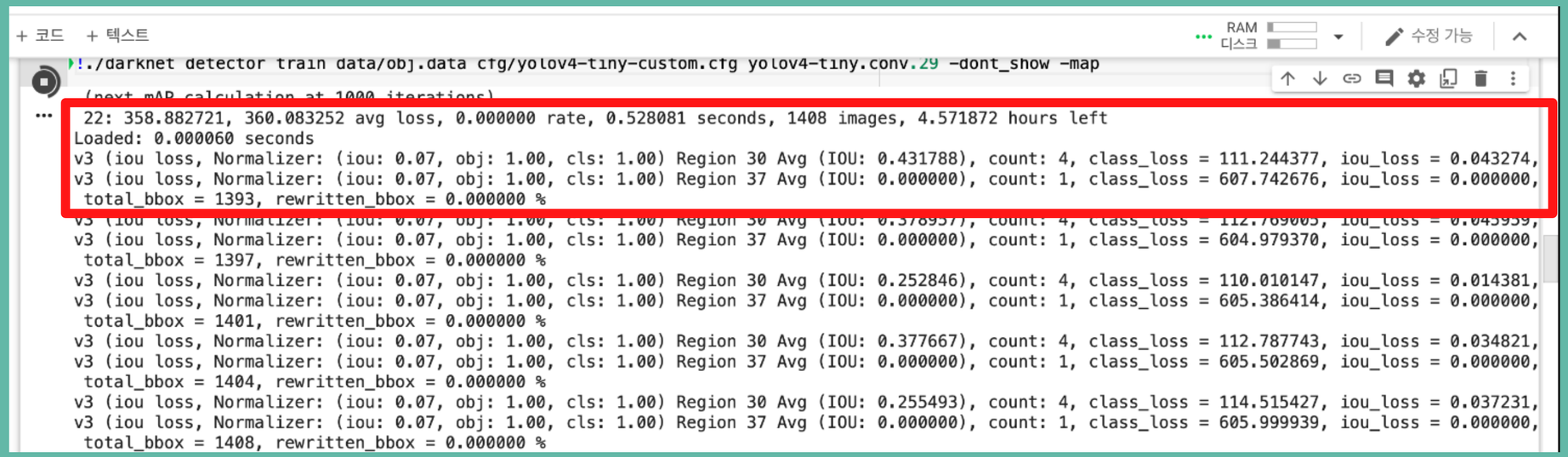
26 classes
> 93 filters = (# of class + 5)*3
>Maximum iteration of
52000 = 2000* # of class

yolov4-tiny-custom.cfg

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=16
width=416
height=416
channels=3
```

```
[convolutional]
size=1
stride=1
pad=1
filters=93
activation=linear


[yolo]
mask = 3,4,5
anchors = 10,14,  23,2
classes=26
```

Overview of Training



```
+ 코드  + 텍스트                                                    RAM  수정 가능
디스크
!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-tiny.conv.29 -dont_show -map

 (next mAP calculation at 1000 iterations)
 22: 358.882721, 360.083252 avg loss, 0.000000 rate, 0.528081 seconds, 1408 images, 4.571872 hours left
Loaded: 0.000060 seconds
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.431788), count: 4, class_loss = 111.244377, iou_loss = 0.043274,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 607.742676, iou_loss = 0.000000,
 total_bbox = 1393, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.378957), count: 4, class_loss = 112.769005, iou_loss = 0.045959,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 604.979370, iou_loss = 0.000000,
 total_bbox = 1397, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.252846), count: 4, class_loss = 110.010147, iou_loss = 0.014381,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 605.386414, iou_loss = 0.000000,
 total_bbox = 1401, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.377667), count: 4, class_loss = 112.787743, iou_loss = 0.034821,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 605.502869, iou_loss = 0.000000,
 total_bbox = 1404, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.255493), count: 4, class_loss = 114.515427, iou_loss = 0.037231,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 605.999939, iou_loss = 0.000000,
 total_bbox = 1408, rewritten_bbox = 0.000000 %
```

```
 (next mAP calculation at 3100 iterations)
 Last accuracy mAP@0.50 = 99.25 %, best = 99.76 %
 3037: 0.141791, 0.152929 avg loss, 0.002610 rate, 0.532965 seconds, 194368 images, 2.542325 hours left
Loaded: 0.000066 seconds
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.747694), count: 4, class_loss = 0.424090, iou_loss = 0.220164, total_loss =
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss =
 total_bbox = 192636, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.681059), count: 4, class_loss = 0.286429, iou_loss = 0.136962, total_loss =
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss =
 total_bbox = 192640, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.762863), count: 4, class_loss = 1.174376, iou_loss = 0.173741, total_loss =
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss =
 total_bbox = 192644, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.707867), count: 4, class_loss = 0.480487, iou_loss = 0.092225, total_loss =
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss =
 total_bbox = 192648, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.714782), count: 4, class_loss = 0.051486, iou_loss = 0.119975, total_loss =
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss =
 total_bbox = 192652, rewritten_bbox = 0.000000 %
```

Training Continued

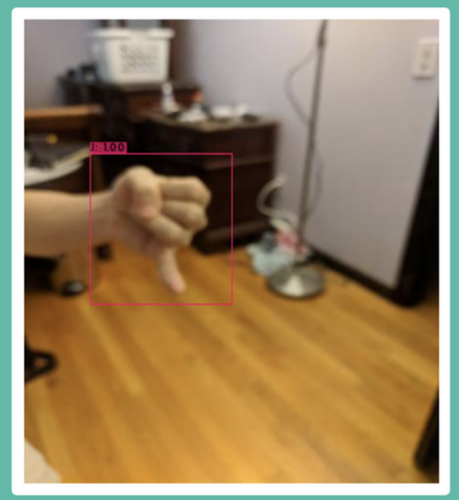
```
2800: 0.101343, 0.147063 avg loss, 0.002610 rate, 0.524939 seconds, 179200 images, 2.552545 hours left

calculation mAP (mean average precision)...
Detection layer: 30 - type = 28
Detection layer: 37 - type = 28

 detections_count = 348, unique_truth_count = 151
class_id = 0, name = A, ap = 100.00%      (TP = 9, FP = 1)
class_id = 1, name = B, ap = 100.00%      (TP = 5, FP = 0)
class_id = 2, name = C, ap = 100.00%      (TP = 2, FP = 0)
class_id = 3, name = D, ap = 100.00%      (TP = 8, FP = 0)
class_id = 4, name = E, ap = 100.00%      (TP = 6, FP = 0)
class_id = 5, name = F, ap = 100.00%      (TP = 4, FP = 0)
class_id = 6, name = G, ap = 100.00%      (TP = 5, FP = 0)
class_id = 7, name = H, ap = 100.00%      (TP = 4, FP = 0)
class_id = 8, name = I, ap = 100.00%      (TP = 12, FP = 0)
class_id = 9, name = J, ap = 100.00%      (TP = 9, FP = 0)
class_id = 10, name = K, ap = 100.00%             (TP = 6, FP = 0)
class_id = 11, name = L, ap = 100.00%             (TP = 4, FP = 0)
class_id = 12, name = M, ap = 100.00%             (TP = 8, FP = 1)
class_id = 13, name = N, ap = 100.00%             (TP = 2, FP = 0)
class_id = 14, name = O, ap = 100.00%             (TP = 1, FP = 0)
class_id = 15, name = P, ap = 100.00%             (TP = 6, FP = 0)
class_id = 16, name = Q, ap = 93.06%      (TP = 6, FP = 1)
class_id = 17, name = R, ap = 100.00%             (TP = 6, FP = 0)
class_id = 18, name = S, ap = 100.00%             (TP = 8, FP = 0)
class_id = 19, name = T, ap = 100.00%             (TP = 4, FP = 0)
class_id = 20, name = U, ap = 100.00%             (TP = 4, FP = 0)
class_id = 21, name = V, ap = 100.00%             (TP = 4, FP = 0)
class_id = 22, name = W, ap = 100.00%             (TP = 5, FP = 0)
class_id = 23, name = X, ap = 100.00%             (TP = 9, FP = 0)
class_id = 24, name = Y, ap = 100.00%             (TP = 4, FP = 0)
class_id = 25, name = Z, ap = 95.24%      (TP = 5, FP = 1)

 for conf_thresh = 0.25, precision = 0.97, recall = 0.97, F1-score = 0.97
 for conf_thresh = 0.25, TP = 146, FP = 4, FN = 5, average IoU = 76.26 %
```

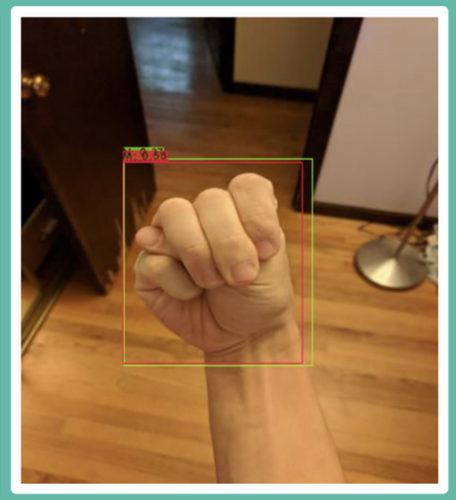
```
!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-tiny.conv.29 -
...
class_id = 13, name = N, ap = 100.00%             (TP = 2, FP = 0)
class_id = 14, name = O, ap = 100.00%             (TP = 1, FP = 0)
class_id = 15, name = P, ap = 100.00%             (TP = 6, FP = 0)
class_id = 16, name = Q, ap = 93.06%      (TP = 6, FP = 1)
class_id = 17, name = R, ap = 100.00%             (TP = 6, FP = 0)
class_id = 18, name = S, ap = 100.00%             (TP = 8, FP = 0)
class_id = 19, name = T, ap = 100.00%             (TP = 4, FP = 0)
class_id = 20, name = U, ap = 100.00%             (TP = 4, FP = 0)
class_id = 21, name = V, ap = 100.00%             (TP = 4, FP = 0)
class_id = 22, name = W, ap = 100.00%             (TP = 5, FP = 0)
class_id = 23, name = X, ap = 100.00%             (TP = 9, FP = 0)
class_id = 24, name = Y, ap = 100.00%             (TP = 4, FP = 0)
class_id = 25, name = Z, ap = 95.24%      (TP = 5, FP = 1)

 for conf_thresh = 0.25, precision = 0.97, recall = 0.97, F1-score = 0.97
 for conf_thresh = 0.25, TP = 146, FP = 4, FN = 5, average IoU = 76.26 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.995498, or 99.55 %
Total Detection Time: 1 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Model Results

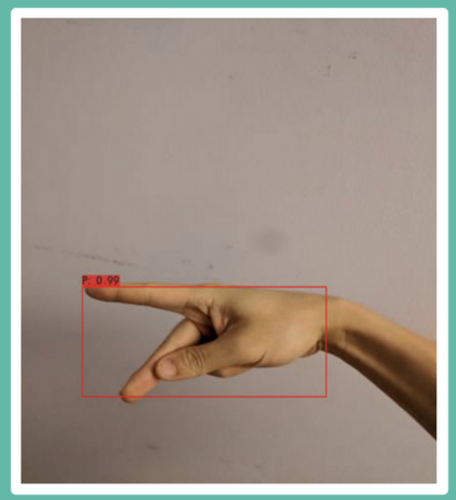
Yolo on Images

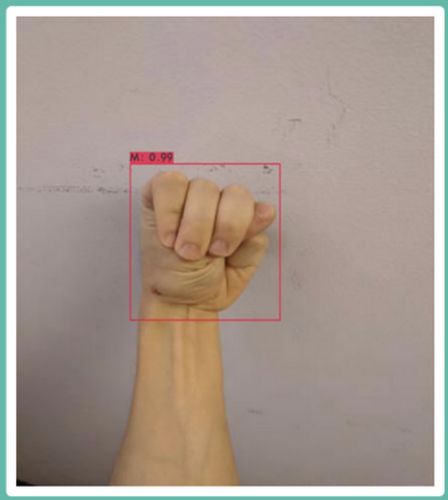
J & M



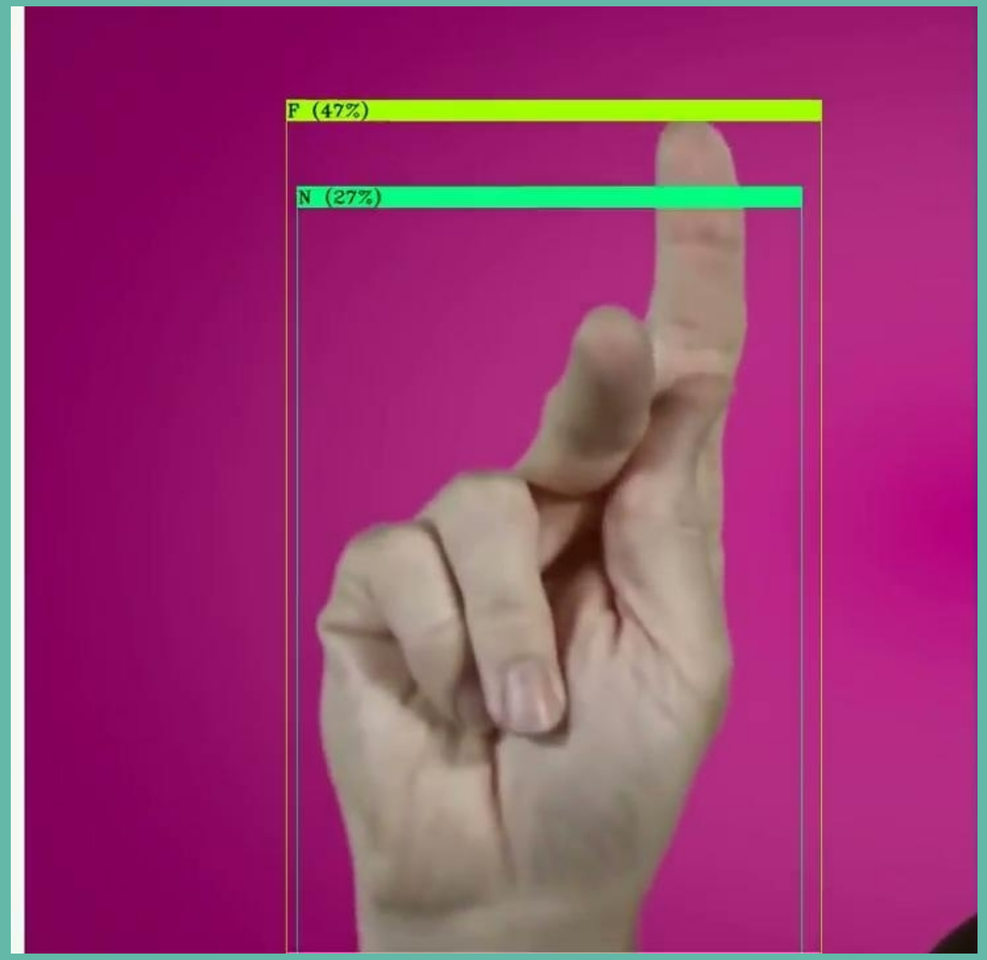


P & M





Yolo on Video



G H I J K L

VS

G H I ? F L

Yolo on Webcam (Real-time)

Conclusion and Next Steps

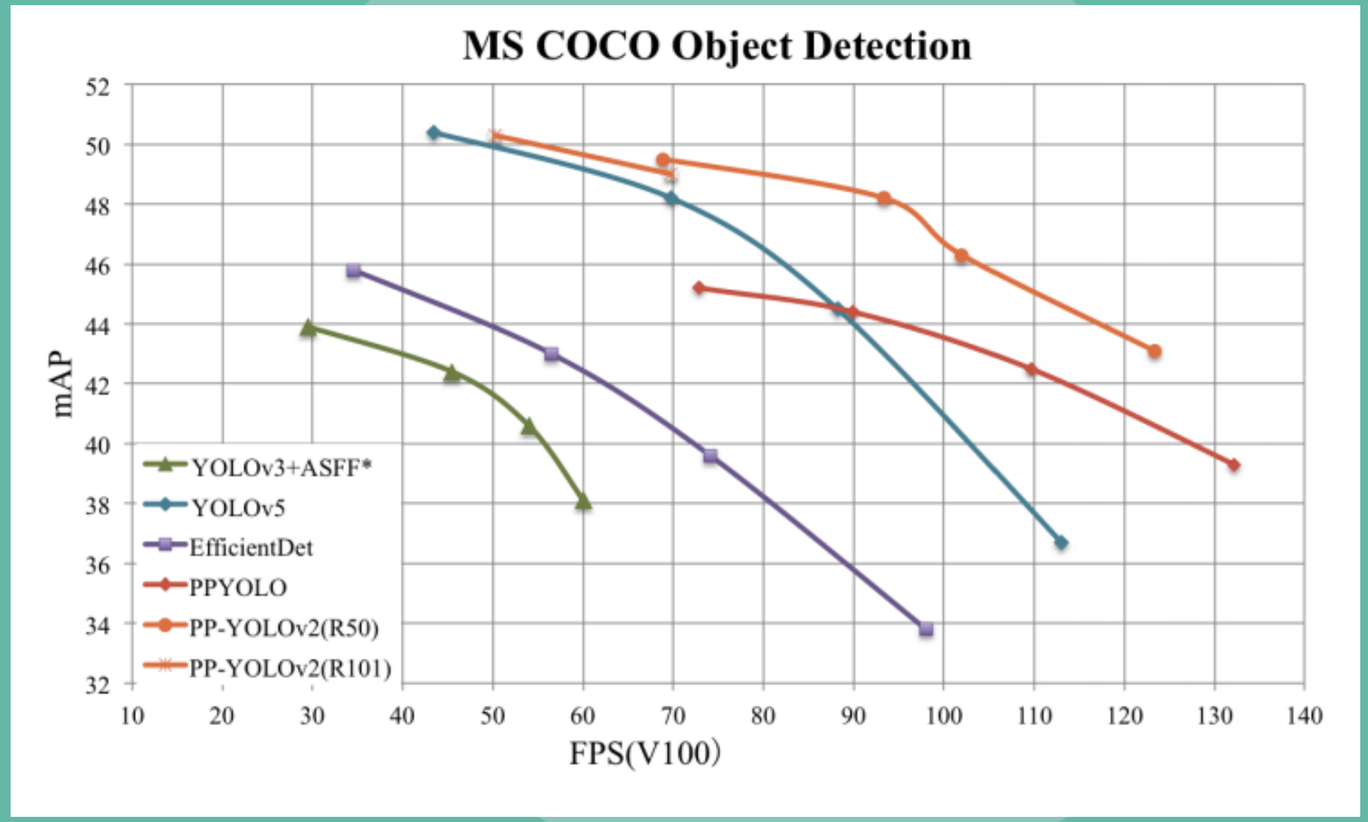
Conclusion

Advantages

- **Speed**
- **High Accuracy**
- **Learning Capabilities**

Areas of Improvment:

- **Newer iterations** and **improved models** like yolov5 and PP-yolo
- **Expanding** amount and variety of input images, i.e., people of different age/race/language etc.
- **Adding more classes** to include sign language phrases

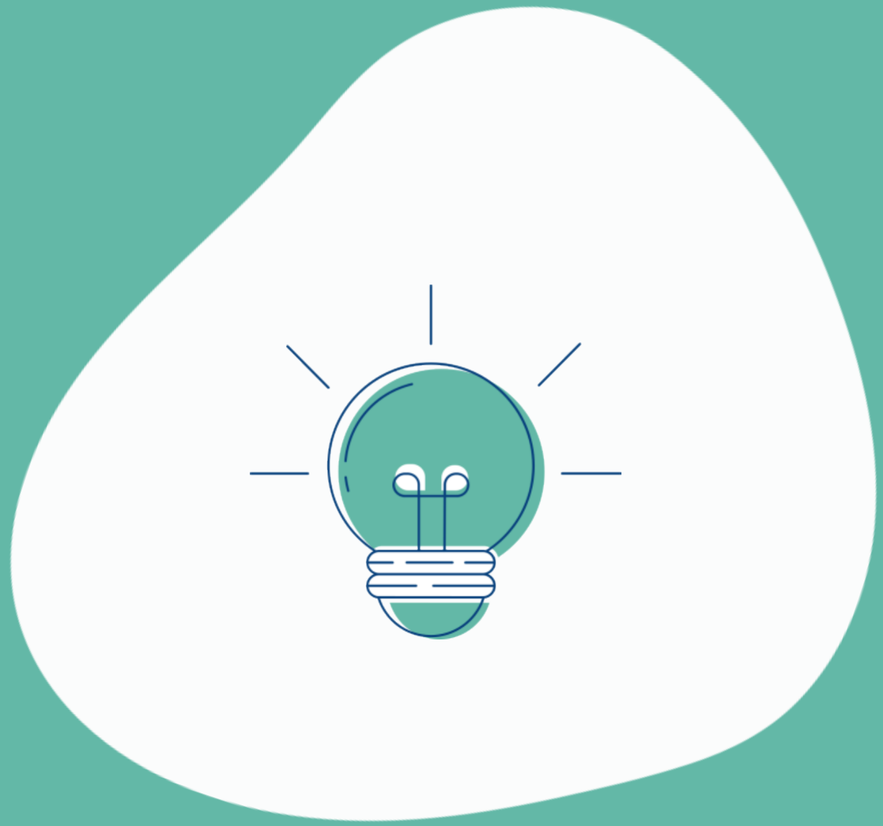


Similar models currently in the business world

- Fingerspelling.xyz
  - Real-time alphabet instruction
- Google Research's video conferencing service
  - 91.5% accuracy, not launched yet
- SignAll
  - Piloting at a school for the deaf

Next Steps and Looking Forward

- Future Applications
  - Adding context with facial recognition
  - Text and audio conversion
  - Educational applications with games
  - Bypassing language barriers
- Additional transfer learning with models
- AI trained to detect lip reading
  - University of Oxford's LipNet
  - Accuracy as high as 93.4%



Thank You!

Appendix

Yolo architecture

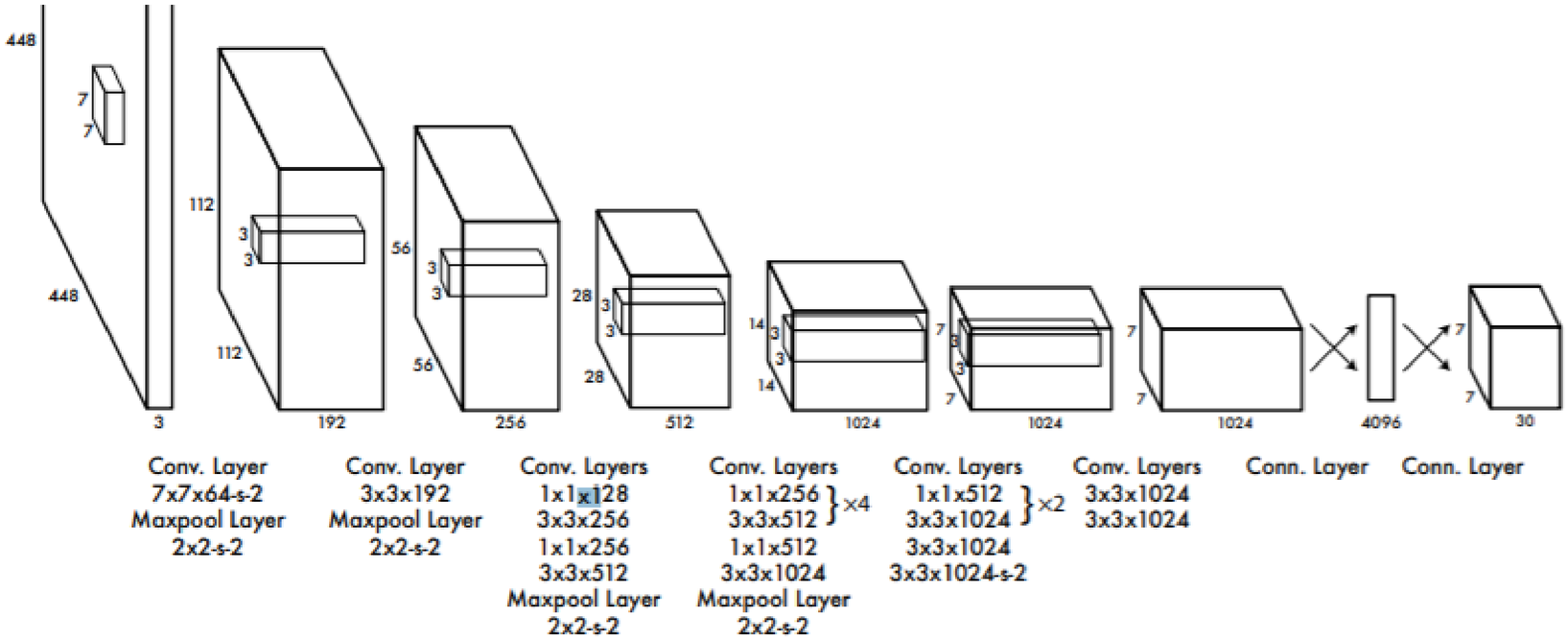


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Yolo loss function

Classification loss

If *an object is detected*, the classification loss at each cell is the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

where

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

Localization loss

The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Confidence loss

If *an object is detected in the box*, the confidence loss (measuring the objectness of the box) is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

If *an object is not detected in the box*, the confidence loss is:

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

Yolo loss function

Loss

The final loss adds localization, confidence and classification losses together.

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

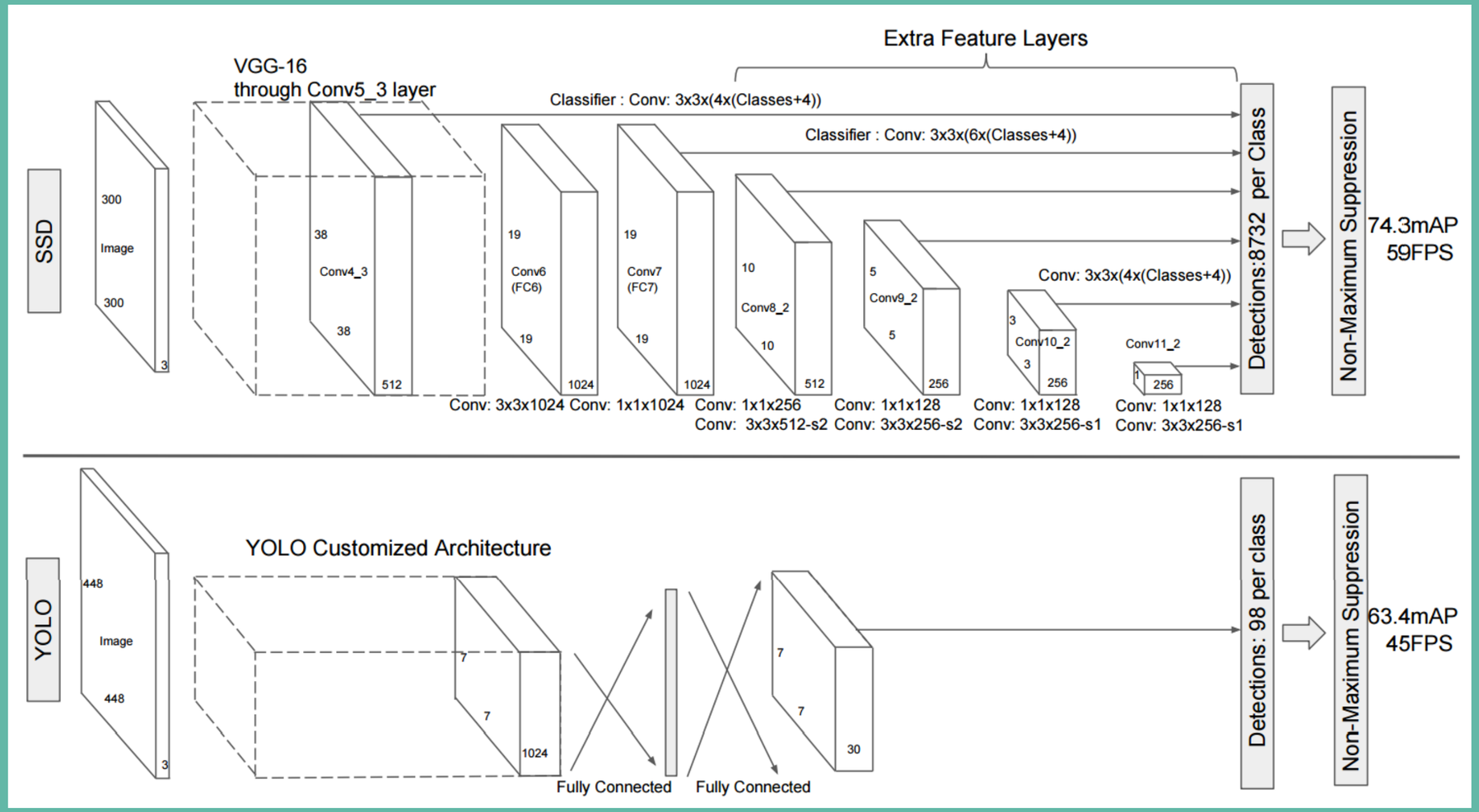
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Single-shot detector vs Yolo



- SSD model trained for image input
- Similar Precision
- Much slower
- Yolo chosen over SSD

CNN architecture

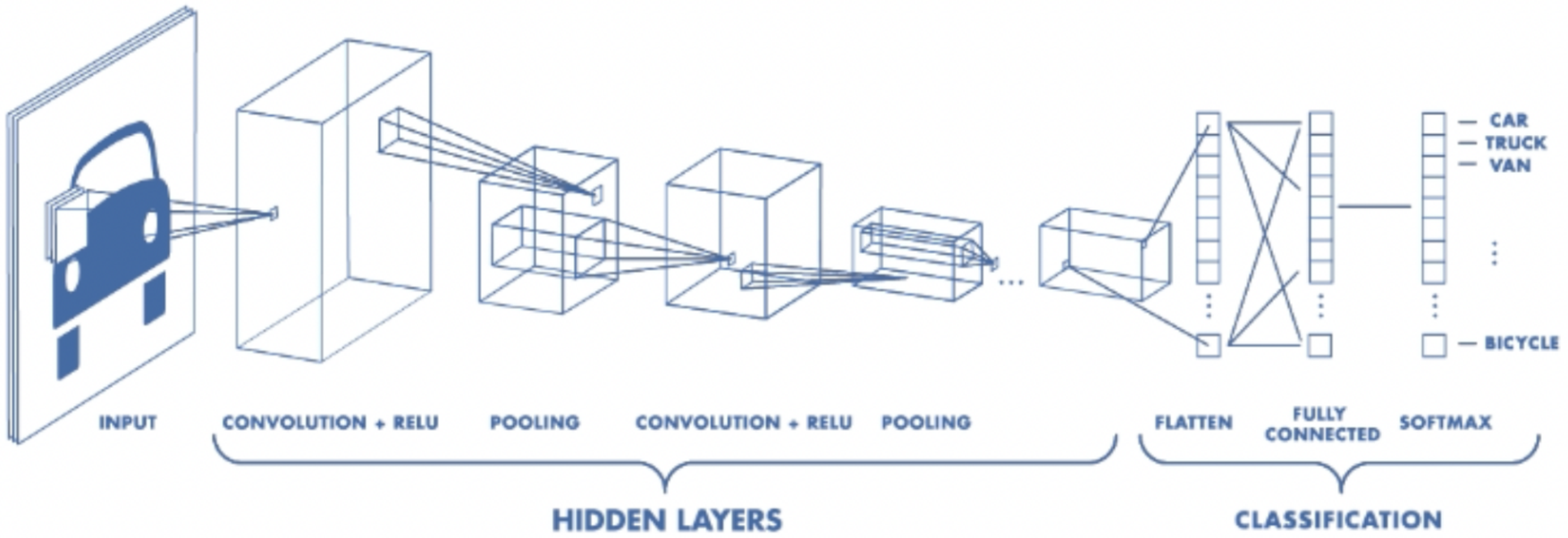


Figure 2: Architecture of a CNN (Source)

CNN steps

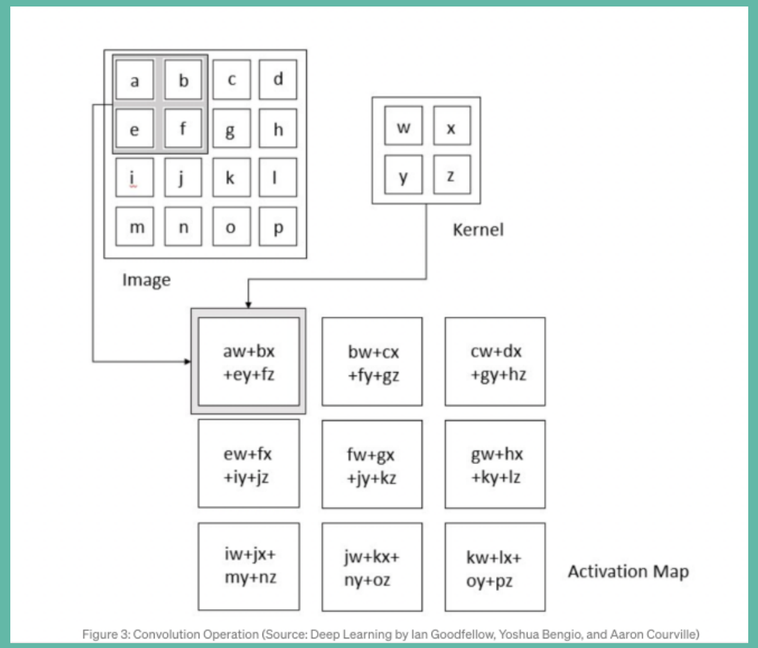


Figure 3: Convolution Operation (Source: Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville)

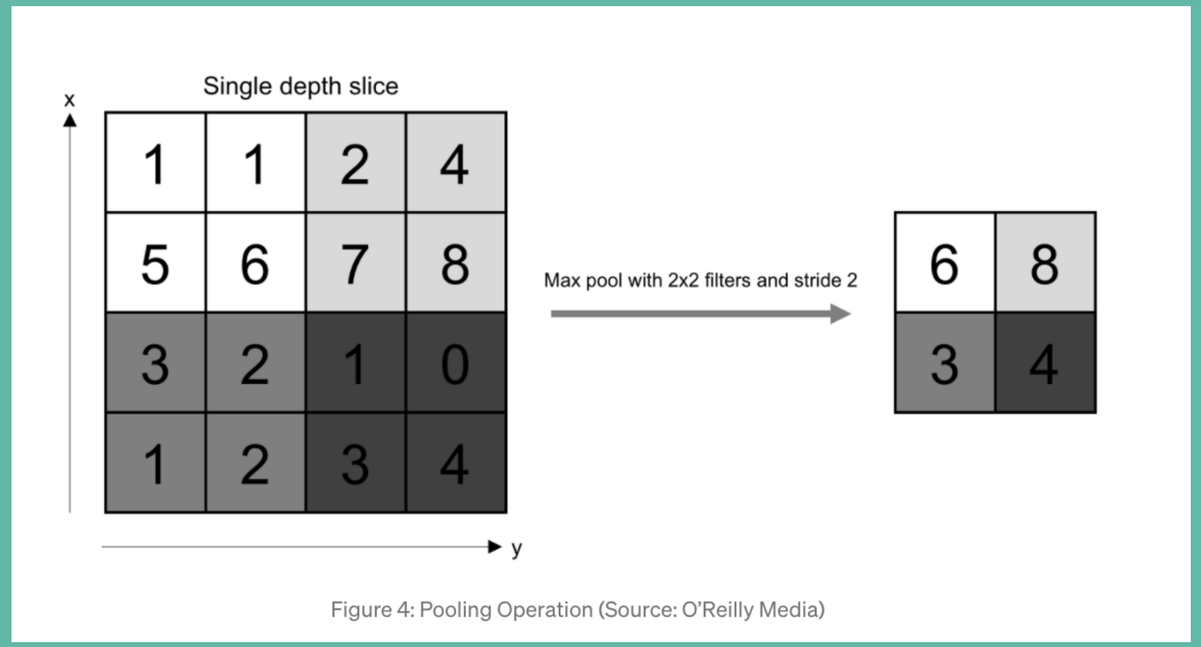
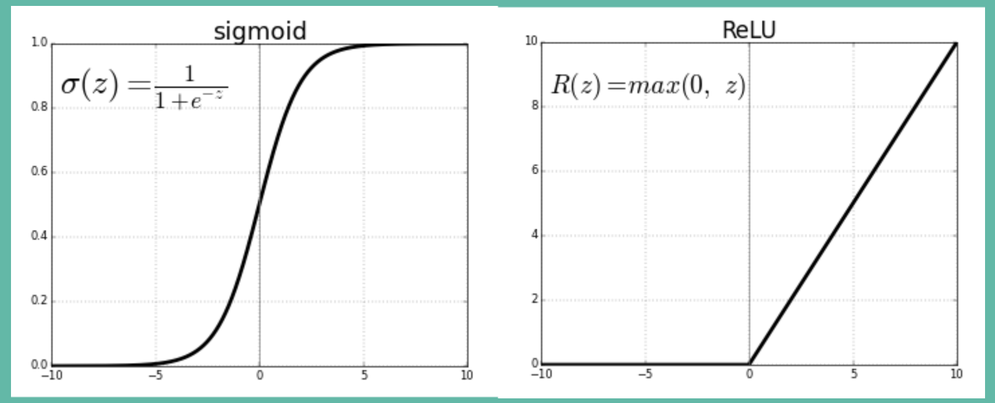
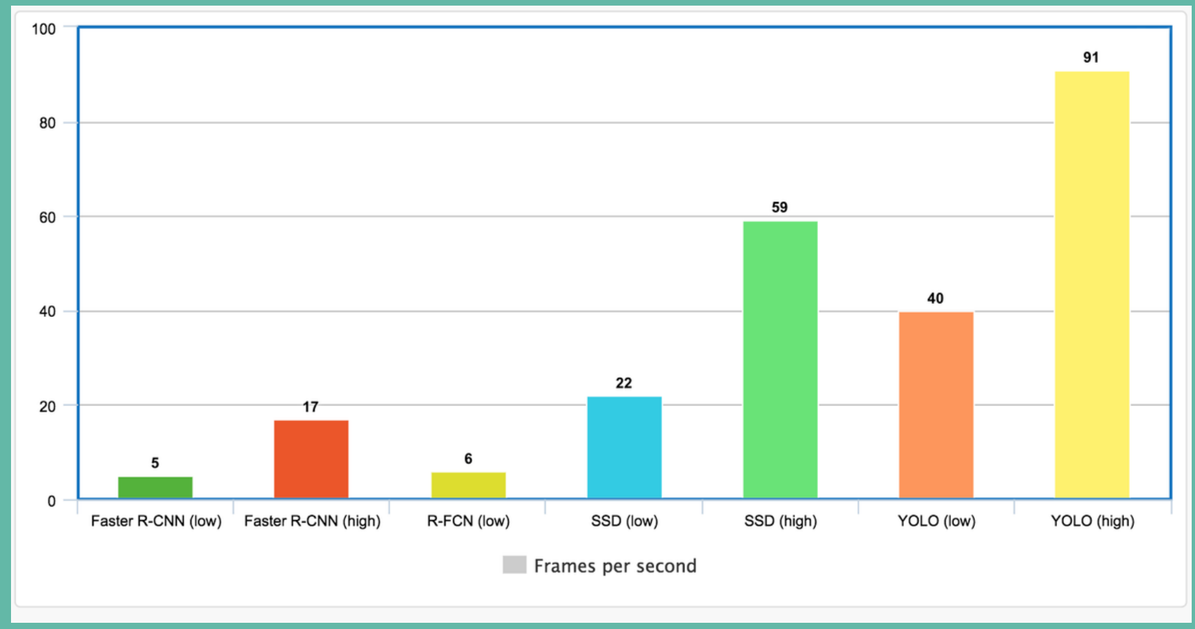


Figure 4: Pooling Operation (Source: O'Reilly Media)

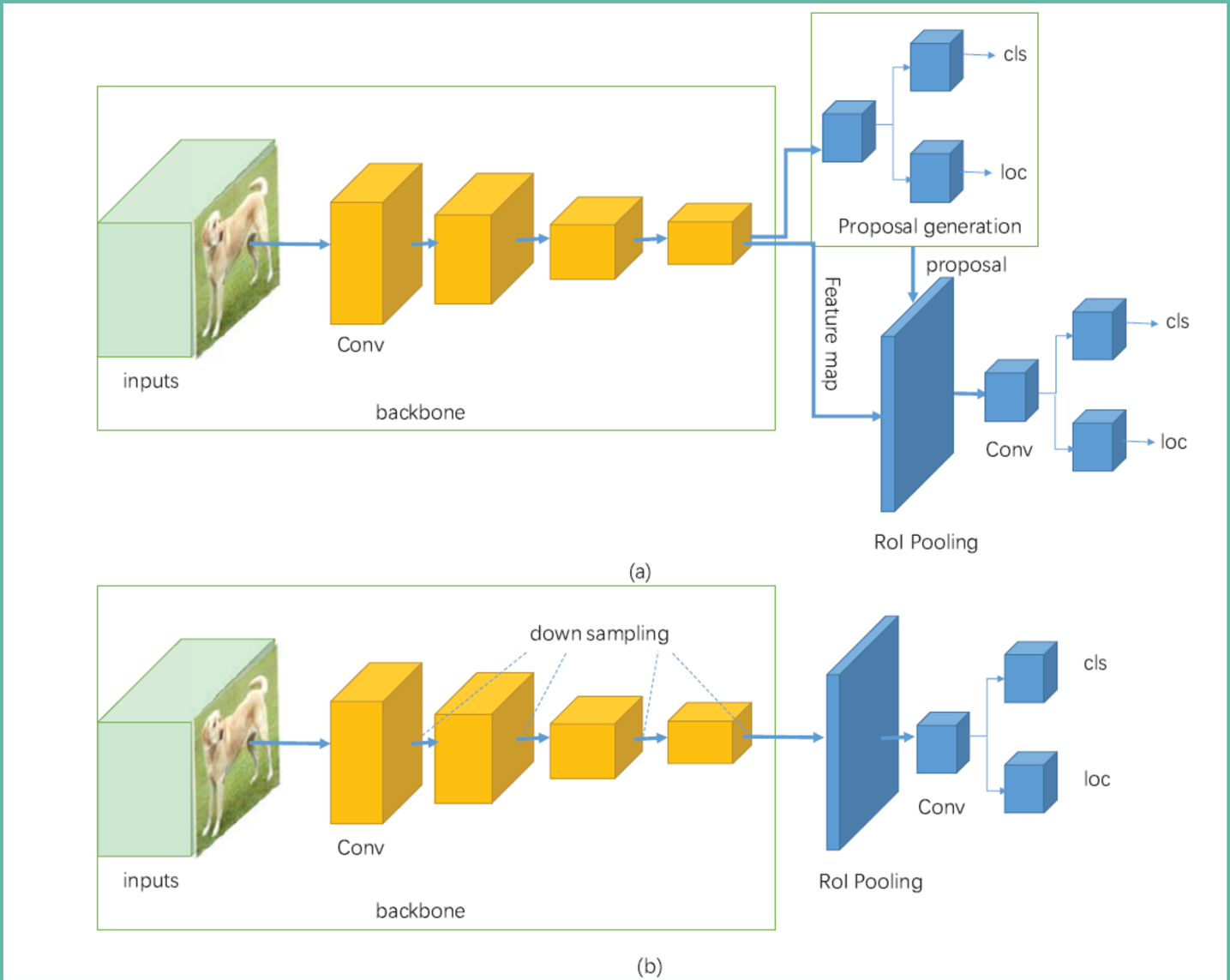


Two stage vs single stage detectors



The advantage of a one stage detector is the speed it is able to make predictions quickly allowing a real time use.

Two-stage vs One-stage detectors



- Regional Proposal Network
  - proposes candidate object bounding boxes
- RoI Pooling

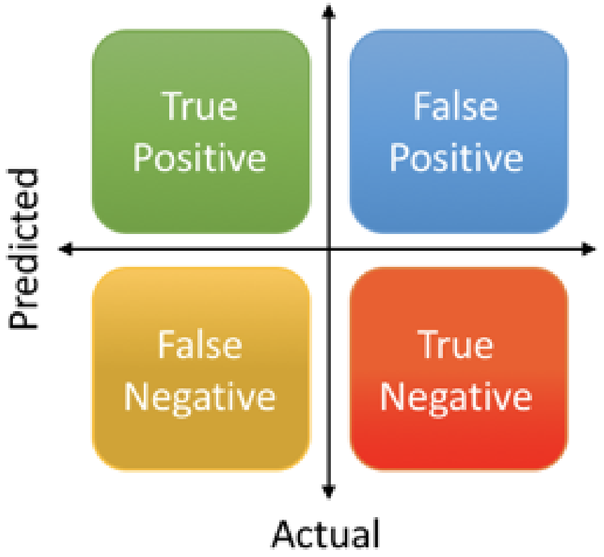
- Propose predicted boxes from input images directly

Precision/ recall

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



After training

Intersection over Union (IoU)
- 85.4%

Precision/Recall
- 100% for both

mean Average Precision (mAP)
- 1.00