Audience Choice Award for Best Project Presentation

Object Detection

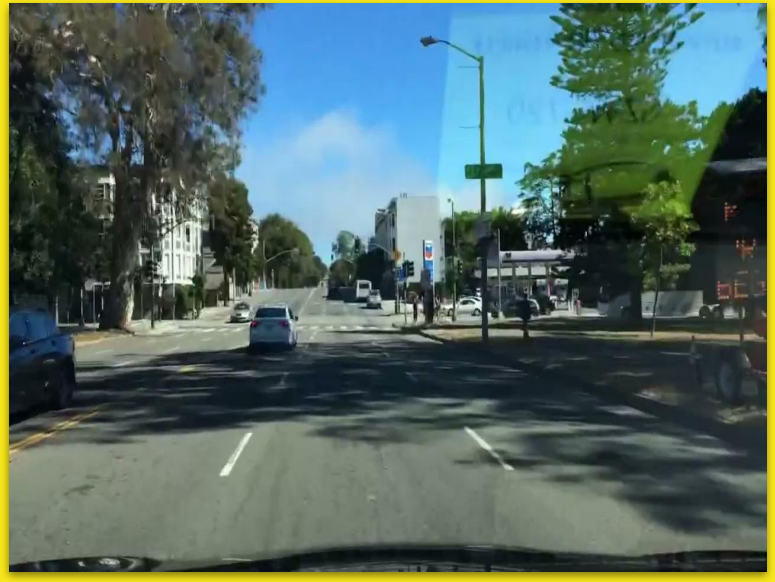
Chinmay, Phoebe, Lakshit, Tyler, Ian, Nishanth
Instructor: Eric

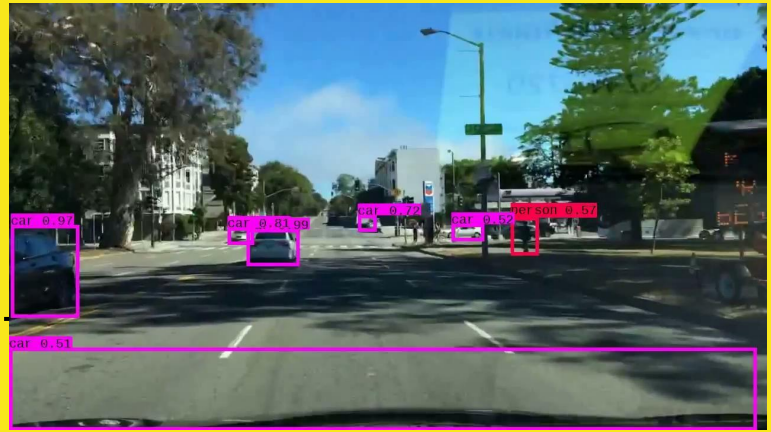
Project Overview

Object Detection for Driverless Vehicles

- **I**nput: Computer Vision
- **P**rocess: Neutral Networks/YOLO
- **O**utput: Labels with Bounding Boxes

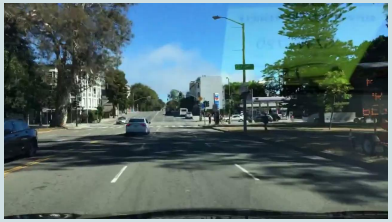
*Enable self-driving cars to **"see" and identify objects** such as traffic lights, cars, and people.*





Detection in a Video

Video



Video ⇒ Images

Some other frames

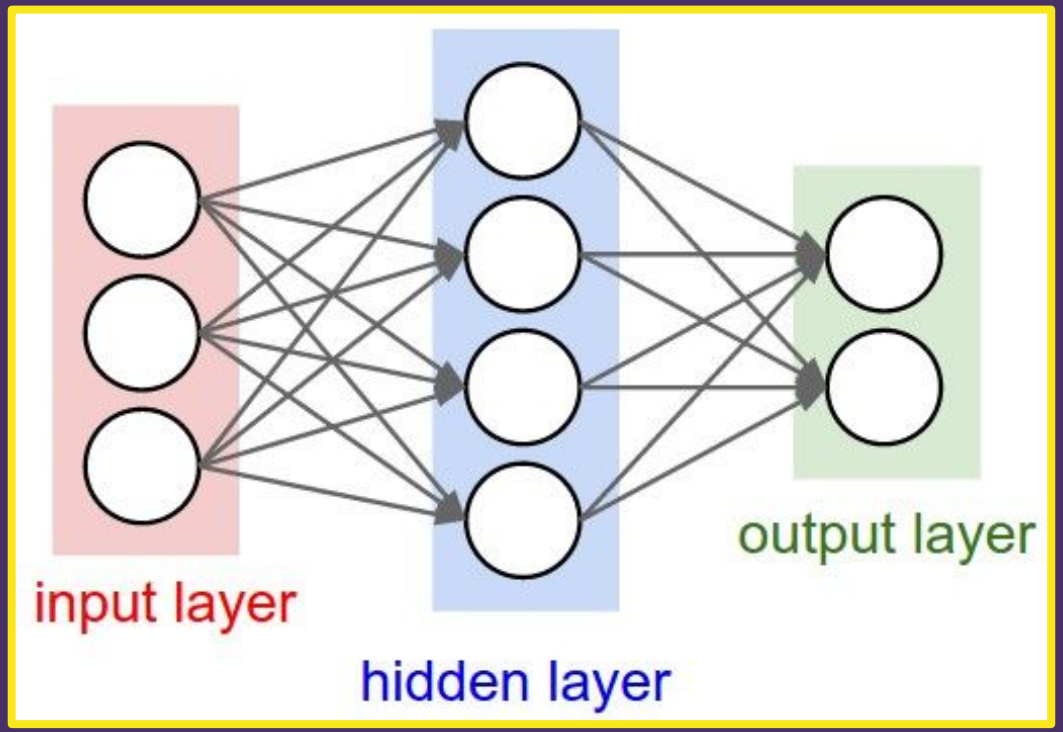
Some 100+ frames

500+ other frames

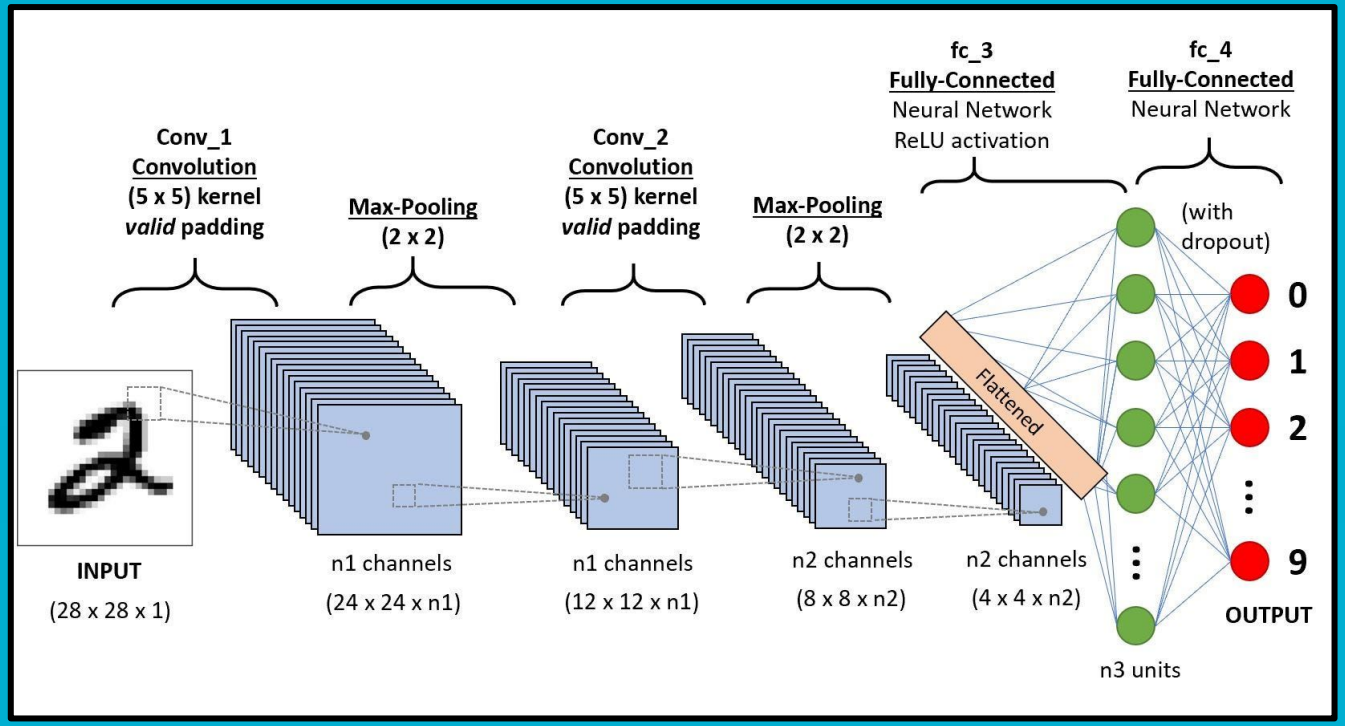
e.g. Frame 32

e.g. Frame 169

Neural Networks



Convolutional Neural Networks



Neural Network (Perceptron) Prediction Using Sliding Windows

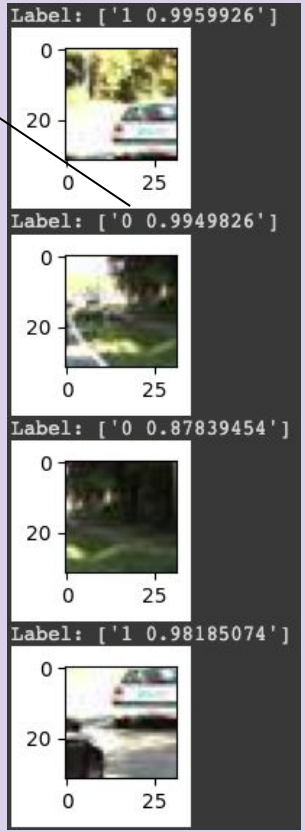
- Creating 32x32 rectangles in the image
- Building models and layers
- Preprocessing + Training
- Only plotting each rectangle with confidence level higher than the specified threshold
- Accuracy of only ≈77.8% on testing data
- Overfitting

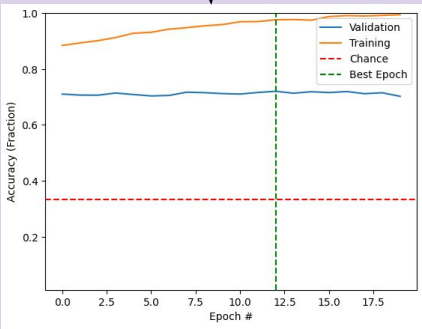
```
for y in range(0,100,16):
  for x in range(0,160,16):
    cropped = new_image[y:y+window_h, x:x+window_w]
    if cropped.shape == (window_w, window_h, 3):
      windows.append(cropped)
```

Label: ['0 0.9949826']

Category:
0=Background
1=Car
2=Truck

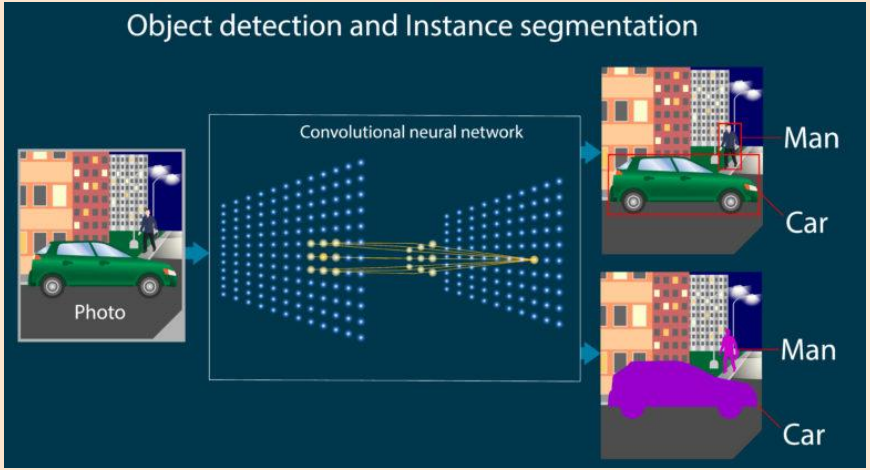
Confidence %





Building a **Convolutional Neural Network**

- Adding different layers
  - Pooling, flatten, dense, convolutional, etc
- Compiled CNN
  - Specified optimizer, loss, and accuracy as the metric
- Then, trained the model for **20 epochs** on training data
- Got an accuracy of ≈85.85%




```
perceptron = Sequential()
perceptron.add(Flatten(input_shape = (32, 32, 3)))
perceptron.add(Dense(units = 128, activation = 'relu'))
perceptron.add(Dense(units = 3, activation = 'softmax'))

perceptron.compile(loss='categorical_crossentropy',
              optimizer=optimizers.SGD(learning_rate=1e-3, momentum=0.9),
              metrics=['accuracy'])
```

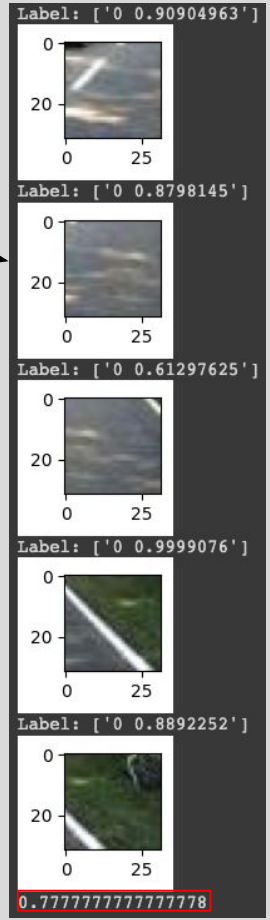
Neural Network

```
cnn = Sequential()
cnn.add(Conv2D(64, (3, 3), input_shape=(32, 32, 3)))
cnn.add(Activation('relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Flatten())
cnn.add(Dense(units = 128, activation = 'relu'))
cnn.add(Dense(units = 3, activation = 'softmax'))
# compile the network
cnn.compile(optimizer = optimizers.SGD(learning_rate = 1e-3, momentum=0.95), loss = "categorical_crossentropy", metrics = ['accuracy'])
```

Convolutional Neural Network

Testing **CNN** on Sliding Windows

```
sliding_predictions(cnn, windows, threshold=0.6, labels = labels)
```

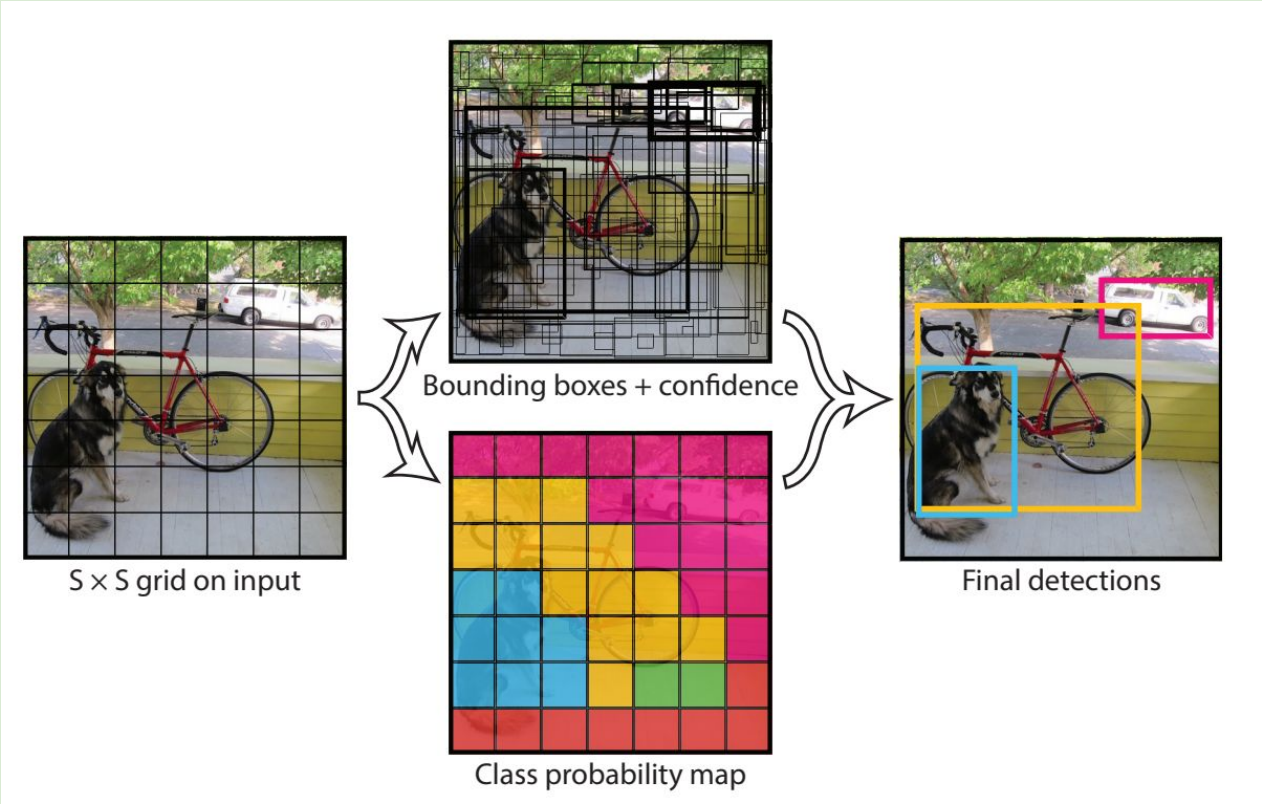


- Just changed model on **from NN (perceptron) to our CNN**
- Once again, only results with confidences above the threshold were plotted
- Accuracy of only 77.8%
- Suggests that our model may have overfitted (conformed to the training data)
    - Due to high training accuracy but not as high testing accuracy


YOLO

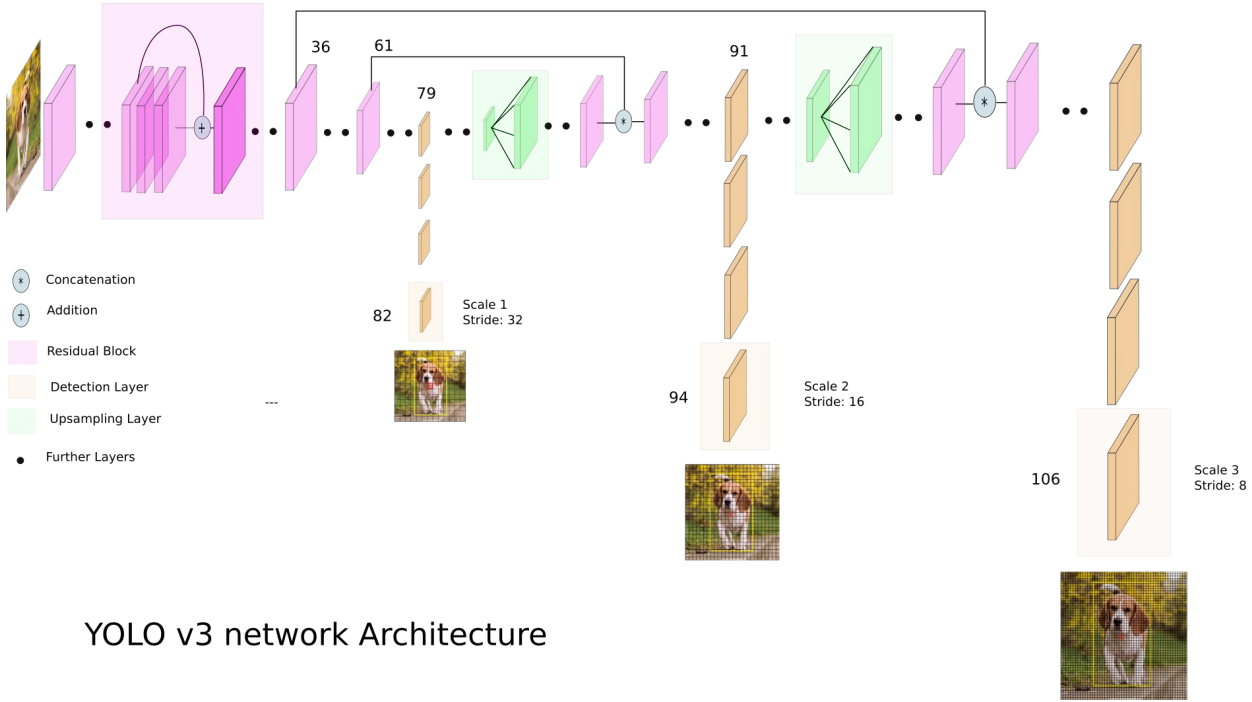
text 0.99

YOLO - Bounding Boxes



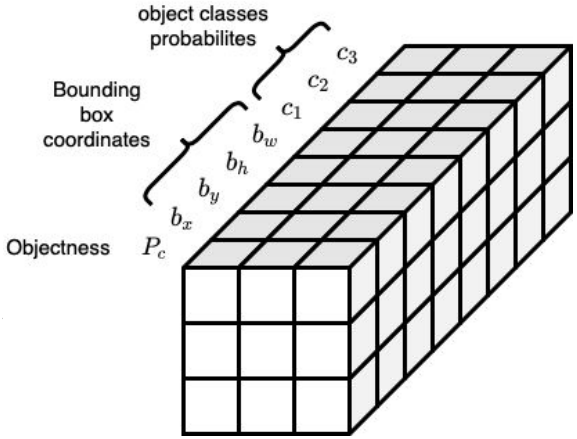
text 0.99

YOLO - Darknet



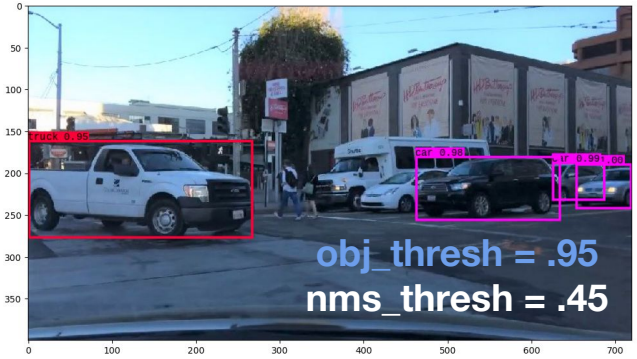
YOLO v3 network Architecture

Output



text 0.99

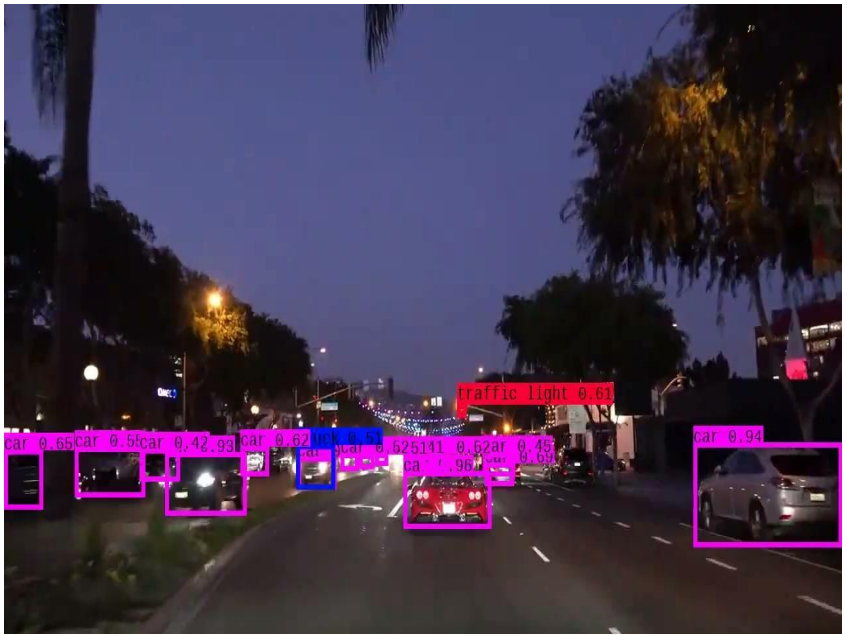
YOLO - Suppression Steps



Best model 👍

Final Result - Additional Video Demo (Night time)

Input



Output

Additional Video Demo (Evening time)

Input

Output

Additional Video Demo (Stop sign + fire hydrant)

Input

Output

Conclusion

Outcome: A computer vision model for object detection in street images precisely

Significance: Enhanced safety, efficiency for self-driving cars

Further Directions:

- Implement object detection fast enough in real time
- Identify the road and non-road areas

👎 *Real Time Detection Demo (laggy)*

Tesla Example 👍

Thank you

~ Object Detection ~