Machine Learning Nanodegree
Capstone Proposal

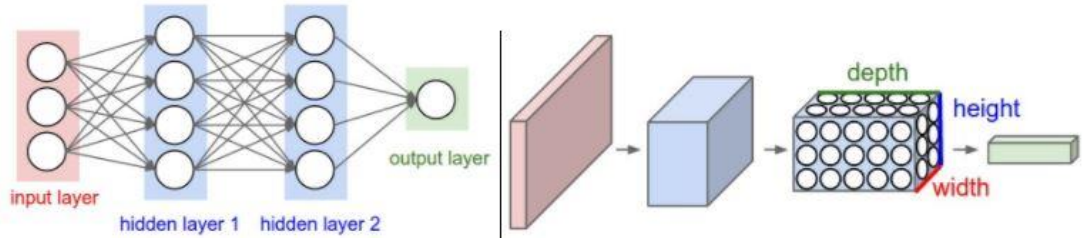
M Harshith
November 18, 2017

1. Domain Background

Object detection is a computer technology related to computer vision and image processing that deals with the detection instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. Well-researches domains of object detection include face-detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Convolution Neural Networks are very similar to ordinary Neural Networks, which are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function; from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what does change? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architectures. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.



2. Motivation

My personal motivation for learning this area in depth is that we have plans for developing web and mobile applications to make people understand the world in a new face which requires a strong Artificial Intelligence. We have participated in hackathons, competitions and also tried exploring, analyzing the datasets in various websites. As we know the world is modernizing and

improving technology at greater rate so we are planning to build better applications to help ourselves live in a better world.

3. Problem Statement

In this project, we will use convolution neural network and object detection to make a model and to detect the car in the given photo or a video.

- **Task:** Detect cars in a given photo or video.
- **Performance:** Accuracy of car predictions in a test dataset.
- **Target method:** using CNN's in keras.

Therefore, I seek to build object detection trained model using deep learning convolutional neural network.

4. Datasets and Inputs

This project will use Vehicle Image Database comprises 3425 images of vehicles rears taken from different points of view, and 3900 images extracted from road sequences not containing vehicles. We will use this dataset to train the convolutional neural network using Keras – a high level neural network API.

An experiment was already done with this dataset using Machine Learning Algorithm SVM which we will discuss and compare with our implementation of solving the problem.

We will use the respective dataset and the input to the convolution neural network would be the 64*64 image where the output would be a binary of detection.

5. Solution

To tackle the problem described in the Section 3, we will use convolution neural network with deep learning to train the model using the above dataset. Unlike the other approaches of detecting car in the whole image we will train the model on individual car images from various angles and then using Opencv we will apply sliding window and identify the car position in the given image or a video.

6. Benchmark Model

Our benchmark model is the first basic model that is shown in the code with total params: 1,059,633, trainable params: 1,059,633

With the accuracy of 46 percentage.
The benchmark given in the project is to detect the car and place it in the rectangle. Thus our benchmark will not have any such root mean squared etc but its just how well we can get the object detected.

7. Evaluation metrics

a. As we are using Keras as our API for developing the neural network we will be using binary_accuracy as our metric to decide the performance of the neural network.

```
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

Binary cross entropy loss is a special case of a multinomial cross-entropy loss for m=2

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(p_i) + (1-y_i)\log(1-p_i)] = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Where i indexes samples/observations and j indexes classes, and y is the sample label (binary for LSH, one-hot vector on the RHS) and $p_{ij} \in (0,1) : \sum_j p_{ij} = 1 \forall i,j$ is the prediction for a

8. Project Design

a. **Programming language and libraries**
   i. **Python 3.**
   ii. **Scikit-learn.** Open source machine learning library for python.
   iii. **Keras.** Open source neural network library written in python. It is capable of running on top of either Tensoflow or Theano.
   iv. **Tensorflow.** Open source software libraries for deep learning.

This project will use the Keras deep learning environment and jupyter notebook to perform the corresponding actions on the dataset.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(64,64,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.summary()
```

We will use the corresponding Sequential network designed in Keras using layers such as Conv2D, MaxPooling, Dense and activations such as relu, sigmoid etc.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
activation_6 (Activation)    (None, 62, 62, 32)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
activation_7 (Activation)    (None, 29, 29, 32)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
activation_8 (Activation)    (None, 12, 12, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                147520
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_10 (Activation)   (None, 1)                 0
=================================================================
Total params: 176,225
Trainable params: 176,225
Non-trainable params: 0
```

This is the detailed summary of the corresponding model.

```
x_train, x_test, y_train, y_test = train_test_split(
    X_train, y, test_size=0.2, random_state=rand_state)
```

Using this we will divide the dataset into two parts.

1. Test dataset - 20 %
2. Train dataset - 80 %

```
for (x, y, window) in sliding_window(imaged, stepSize=10, windowSize=(winW, winH)):
    # if the window does not meet our desired window size, ignore it
    if window.shape[0] != winH or window.shape[1] != winW:
        continue
    clone = imaged.copy()
    clone = cv2.resize(clone[y:y+winH,x:x+winW], (64,64)).astype('float32')/255
    pre = model.predict(np.expand_dims(clone, axis=0))
    if pre[0][0] == 1.0:
            final.append({
                "x" :x,
                "y":y,
                "winH" : winH,
                "winW": winW,
                "pred" : pre[0][0]
            })
```

This code will help us in applying the sliding window concept where we will use a 128*64 rectangle and resize it into 64*64 and then give it as input to the convolution neural network.

```
for i in range(0,len(car_predictions)):
    if round(car_predictions[i][0][0]) == y_test[i]:
        total = total + 1
test_accuracy = (total/len(car_predictions))*100
print('Test accuracy: %.4f%%' % test_accuracy)
```

```
Test accuracy: 97.1331%
```

Giving the final accuracy to be 97 percentage and we will save the model using model check pointer while training provided by Keras API.

References.

SVM : *Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. **20** (3): 273–297. doi:10.1007/BF00994018.*

Neural Networks : McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*. **5** (4): 115–133. doi:10.1007/BF02478259.

Convolution neural network : *LeCun, Yann. "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.*

J. Arróspide, L. Salgado, M. Nieto, "Video analysis based vehicle detection and tracking using an MCMC sampling framework", *EURASIP Journal on Advances in Signal Processing*, vol. 2012, Article ID 2012:2, Jan. 2012 (doi: 10.1186/1687-6180-2012-2)

https://www.pyimagesearch.com -- pyimagesearch

Tatsuyah --- vehicle-detection (github.com)

**Udacity Self-Driving Car Engineer Nanodegree Vehicle Detection