FOLLOW ME

Project 4

Youssef Khaky
youssefkhaky@hotmail.com

Abstract

Us humans are smart; our brains are capable of complex tasks that we take for granted. We get to appreciate these amazing abilities by studying how to implement them on a computer. The task of segmenting a scene is very easy to us humans, we can see a person sitting on a chair and be able to realize that these are two different things, or a car in front of a building and realize that the car and the building are two different items. However, it is hard for the computer to deduce that from a 2D image and some algorithms need to be implemented correctly for a computer to be able to classify each pixel in the scene. That is what we explore in this project. We want to make the computer segment an image to the best of it's ability using machine learning.

Introduction

We would like to build a simple follow me program that can be installed on a drone to be able to follow a specific person closely. But before a drone can follow a person, it must first be able to detect that person and understand it's environment. In this project we train a neural network to be able to recognize a hero to be able to follow them.

We discuss the architecture of the neural network selected, the role of different layers in the network, the data collected, the hyper parameters used and finally the accuracy of the trained model.

Network model used

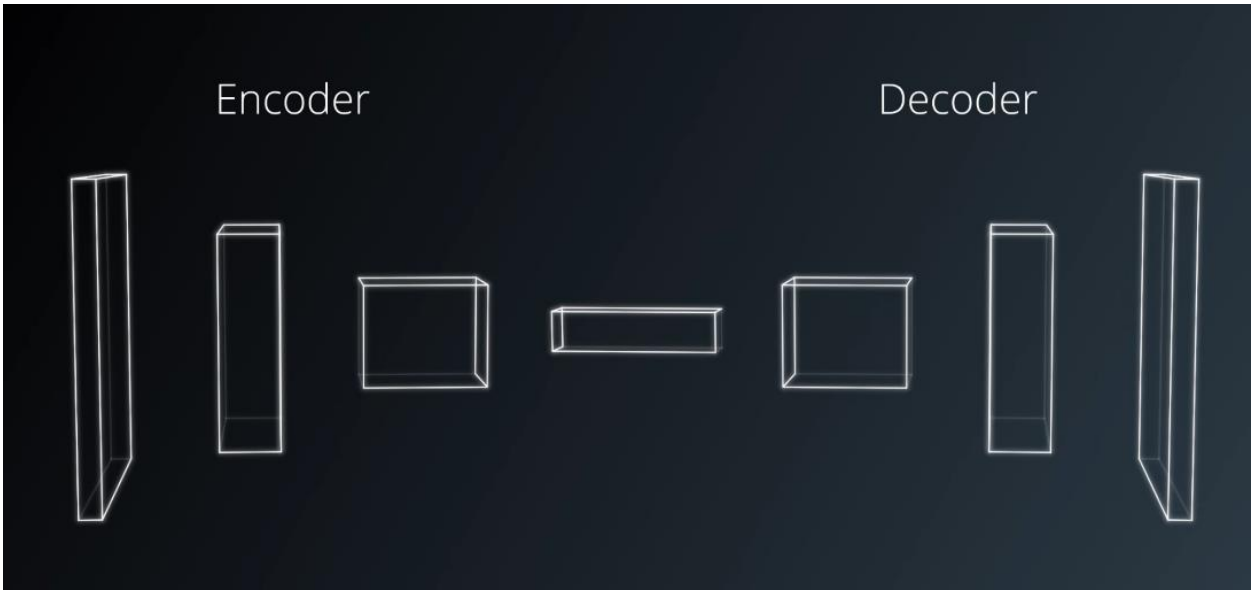


Figure 1 Architecture of FCN (udacity)

In this project we use a specific kind of network called a Fully Convolutional Network. The architecture is like that of a regular convolutional neural network, however after the fully connected layer there is a decoding process which is not in the convolutional network. More on decoding later in the report.

The FCN is divided into two parts encoding and decoding. A regular CNN is just the encoding part of an FCN, so the FCN has this additional decoding part to it. The reason we use an FCN rather than a CNN is be because FCN answers a different question that a CNN. CNN answers the question of “is this an image of a cat?” but an FCN answers “what is the classification of every object in this image”. FCN are able to preserve spatial information, however this information is lost in a CNN.

That last layer in a CNN just before softmaxing is typically a fully connected layer. In FCNs this layer is replaced by a 1x1 convolution layer, which can preserve spatial information.

Encoding

series of convolutional layers such as VGG and ResNet. Goal of encoder is to extract features from the image.

Decoding

The decoder upscales the output of the encoder such that it is the same size as the original image resulting in a prediction for each pixel in the original image. Decoder is comprised of transposed convolutions; transposed convolution is a reverse convolution at which the forward and backward passes are swapped. Transposed convolutions help up sample the previous layer to a desired size.

FCN vs CNN, in FCN we;

1. Replace FCL with 1x1 conv layers
2. Up sampling using transposed convolutional layers
3. Use skip connections, allowing the network to use information from multiple resolution scales, resulting in a more precise segmentation.

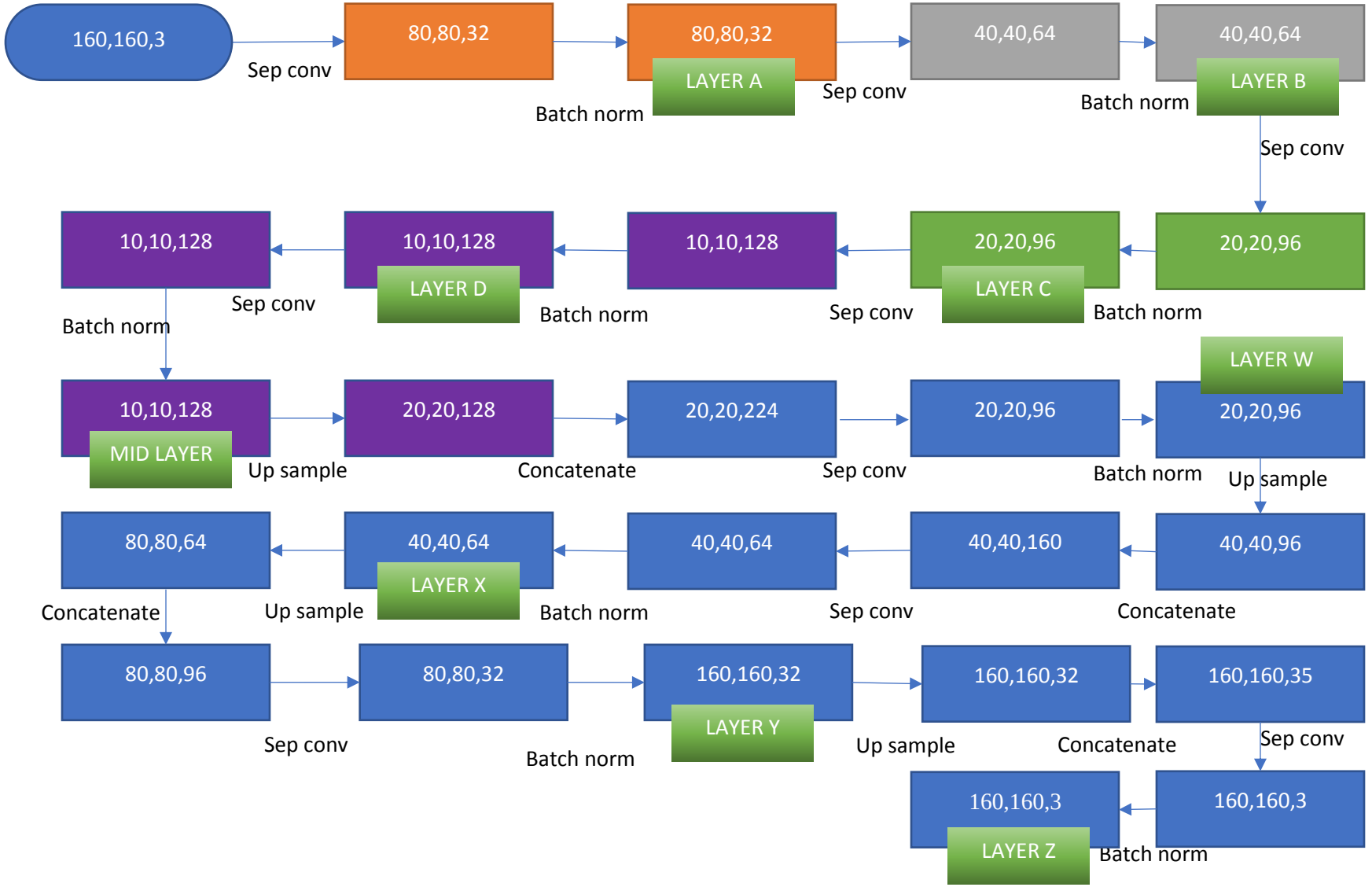
1x1 conv

1x1 convolutions are normal convolutions with the kernel looking at one pixel only. Following the layers of a NN with 1x1 conv is a cheap way of making the model deeper with more parameter. In the context of FCN it has another use. To convert an CNN to an FCN we replace the FCL with a 1x1 conv to preserve the spatial information. In FCN, fully connected layers are replaced by 1x1 convolutions. That will keep the output tensor at a dimensionality of 4 rather than 2. Thus, the spatial information is preserved. Since the objective is not just classification but also pixel wise classification, spatial information is important.

Fully connected layer

Perceptrons in a FCL have connections to all preceptrons in the previous layer. In DNN or CNN these layers come at the end. The FCL is passed through a softmax which determines the scoring for the different classes in the classifier. There is no FCL in an FCN.

Flowchart of the architecture of FCN used in project



Explanation of the different layers and the operations used from one layer to the next

Separable Convolutions (used in encoding)

In regular convolutions we have a kernel scanning the image across all of it's dimensions.

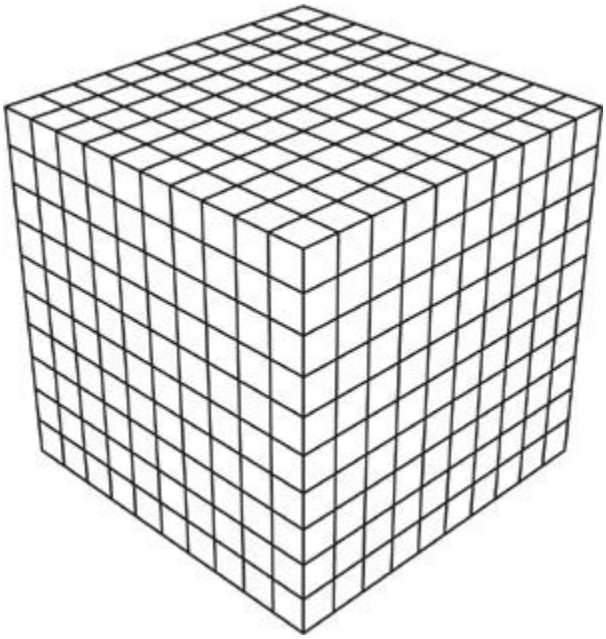


Figure 2 example of an input image [1]

Imagine having a smaller cube scanning this the above cube, it has to be scanned in all the directions X,Y,Z. That results in a huge number of scans. An important question presents itself, is it worth convolving in the Z direction? These layers in Z direction are usually colour channels. So, this is not some kind of point cloud data were the depth channels are physically linked. Therefore, one can argue that in most cases it is not important to look at the colour channel in the context of another colour channel, and these channels can be scanned separately as three different unrelated layers. This is equivalent to convolving over 3 images that have only one channel each, which is acceptable because finding a feature across multiple colour channels may not be meaningful most of the time. The spatial information can be obtained from one layer from one of the channels because the edges of the objects are still distinguishable in each channel.
However, if the edges of the objects in an image appear in one channel but not the other, then this approach might not work so well. That is the point of a separable convolutions, to reduce parameters by convolving over each colour layer independently thus reducing the number of parameters significantly.

After each layer of colour is convolved with a 2D kernel, the separate layers that are produced are combined into one layer using a 1x1 convolution.

Batch normalization (used in encoding)

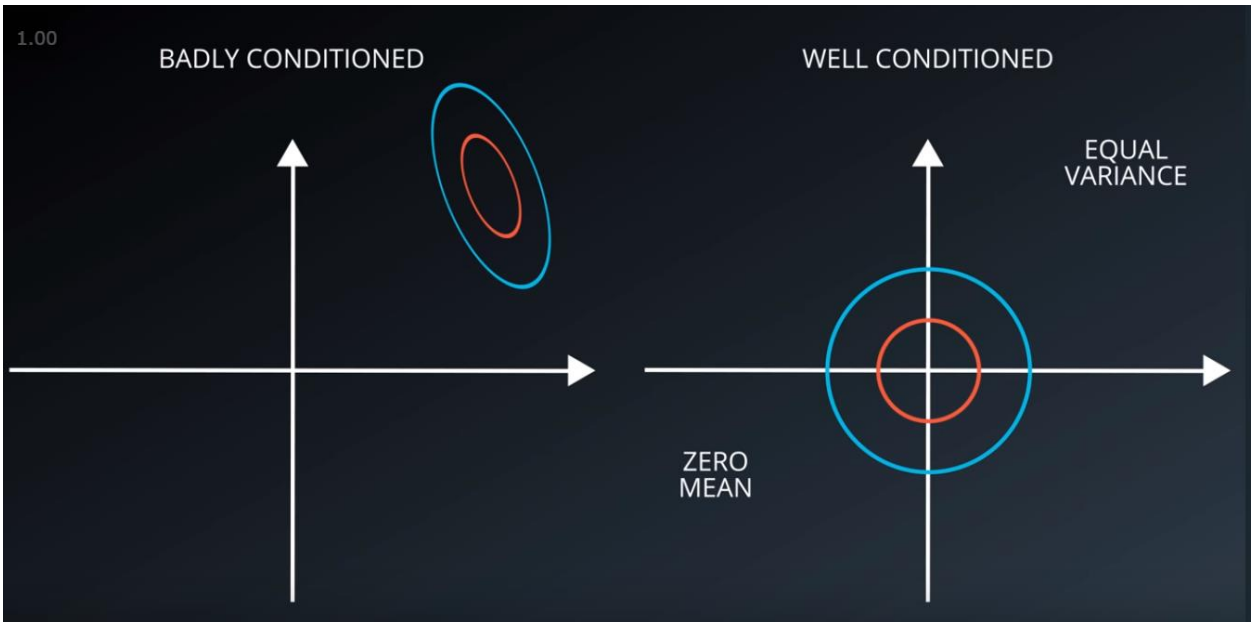


Figure 3 visualising a well conditioned data (ucadiy)

It is important for our data to be conditioned properly to help our network with it's learning. Batch normalization is an extension to the idea of normalization. Every layer can be thought of as an input layer to a network for the layers that are to come next, so we can normalize before every layer. The variance and the mean are derived from the current batch used in training, hence the name, batch normalization.

Bilinear up sampling (used in decoding)

Bilinear up sampling is a part of the decoding step. It is a more intuitive replacement to the transposed convolutions. In bilinear up sampling, pixels that are close to each other are separated by some distance and the gaps in the middle are filled up using bilinear interpolation. The value of a new pixel in the middle is calculated using the distance from the original pixel values from the previous layer.

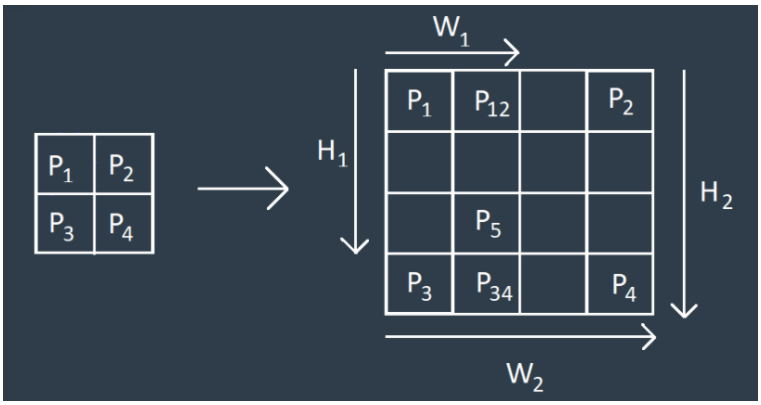


Figure 4 showing bilinear up sampling (Udacity)

Layer Concatenation

Layer concatenation is a substitute to element wise addition operation used when we want to impalement skip connections. We concatenate two layers to retain details from previous layer as we decode the layer to original image size.

It does what the name suggests, it concatenate layers along a certain dimension. It gives more freedom than addition because in addition the layers must be exactly same size, were as concatenation the axis of concatenation can vary (depth channel is usually the axis of concatenation).

Data Collected

The images provided for training were not enough to train the model to reach 40% final score with my network architecture, so I had to collect more data. I have reached a total of 10753 images. I obtained different kinds of data.

- Data with the hero far away
- Data with the hero close
- no hero
- hero with many pedestrians
- hero with low pedestrians, etc.

10753 images were used for training

Parameters used

Here are the parameters used in the model. All of them were used using trial and error. The number of epochs is not as is presented here. I did not train the model in “one go”, I did few epochs at a time, so I am able to check the final score in between. Overall I used around 20 to 30 epochs.

learning_rate = 0.01, it was a smooth curve when I chose this number.

batch_size = 32, not too high so the training is done fast

num_epochs = 5, this is not the true number, it is just the last time I ran it. Actual number was between 20 and 30

steps_per_epoch = 200, this number seems to have yielded good results.

validation_steps = 50

workers = 2

Project Results

The network was assessed using 3 criterions

1. IOU when the target was near
2. IOU when the target was far
3. IOU when target was not visible

Criteria	Number of samples	Avg IOU background	Avg IOU people	Avg IOU Hero	True –	True +	False +	False -
Hero near	542	99.5%	35.1%	87.9%	3	538	0	1
No Hero	270	98.78%	76.5%	43.4%	264	0	6	0
Hero far	322	99.6%	0%	19.2%	21	110	0	191

The score of the network was calculated in the following manner:

true_pos = true_pos1 + true_pos2 + true_pos3

false_pos = false_pos1 + false_pos2 + false_pos3

false_neg = false_neg1 + false_neg2 + false_neg3

weight = true_pos/(true_pos+false_neg+false_pos)

final_IoU = (iou1 + iou3)/2

final_score = final_IoU * weight

The final score calculated was 41.03%

One significant reason why the network is getting a low score is due to the high false negative with the hero is far, if the model can learn how to detect a hero that is far, a better score would be obtained. More on this in the improvement section*

Potential improvements

As seen from the Project Results section. When the hero is near, the network gives true positive (True+ + False+ / total images = 538/542) 99.26% of the time. Which is good. So, when the drone is locked and is in follow me more, everything is good and there is not much to improve here.

When the hero is not in the image we get accuracy of (True+ + False+ / total images = 6/270) 2.22%, which is very bad, the network is almost always reporting that there is a hero even though there is no hero in the picture. The amount of false negative is very high.

When the hero is far we also get a bad result, the accuracy is 34.16% which is also bad, but not as bad the no hero. The network is reporting many false negatives where there is a hero in the picture, but it is still not able to detect it.

So from the above we see that most of the errors is coming from two locations, True negative when there is no hero, and false negative when the hero is far. If these two negative reports can be reduced, a significant improvement in the score will be achieved.

Now that we know where is the issue, what comes next is how we can solve it.

Reducing true negative for no hero

The network needs to be trained much more on differentiating between the people and the hero. More pictures with no hero and pictures of the environment needs to be used in training.

Reducing false negative for far hero

Reduce the aggressiveness by which the image is down sampled. When the hero is far, there are only going to be few pixels that are resembling the hero, so care should be taken so when down sampling these pixels don't get lost right away. These pixels should propagate and have their effect down the network and not be removed right away from the first down sampling.

Also, as usual, more pictures where the hero is far would help. I have many images and I think I may have reached the point of saturation, but I am not sure, perhaps more training samples would help.

Generalizability of model

This FCN model can work on other objects. If we have a training dataset for a cat for example, the network can be trained to pixel wise classify for cats. Some things may differ such as the number of skip connections. In the case of cats, since cats are smaller than humans, perhaps more skip connections are needed since the pixels containing the cat can be retained. Also, the aggressiveness at which the image is down sampled would also be reduced to account for the cat size.

Appendix

[1] https://www.physicsforums.com/threads/space-time-distortion-grid-representations.486222/

Several of the explanations such as the explanation of a 1x1 conv were obtained from Udacity videos.