

APSC 258: Lab 5 Manual

Andre Cox

July 16, 2022

Contents

1	Introduction	2
1.1	Introducing the Convolution Layer	2
1.2	More on Convolution Layers	3
2	Using the Convolution Layer	4
2.1	Creating a Convolution Layer	4
3	Applying the Convolution Layer	5

Chapter 1

Introduction

1.1 Introducing the Convolution Layer

In our last lab you got to work with dense layers. You may have noticed that the neural network was not very effective. In this lab we will introduce the concept of Convolution layers. To explain why we use these layers we must first revisit the concept of a dense layer. You can think of a dense layer as a layer of neurons. The neurons in a dense layer are simply a weighted sum of the inputs. When training the neural network these weights get adjusted until the network is able to predict the correct output. This is a great way to predict simple things such as the average grade a student may achieve based on class attendance and the amount of homework they have done. However they become less effective as the number and complexity of the inputs increase. For instance let's say we have an image of our track that we feed into some dense layers. These dense layers may be able to predict the steering angle of the car however they may get confused depending on the different features of the image. This is because these layers represent the image in 2 dimensions. Let me give you an example.



Figure 1.1: A 1D representation of an image

Can you tell what this above image is? Probably not easily this is essentially what our neural network has to process with dense layers. We need a way to represent the image in a 2 dimensional space.

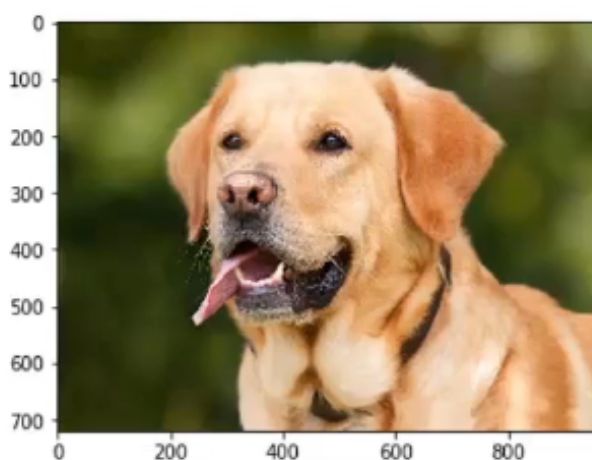


Figure 1.2: A 2D representation of an image

You can see that the image is now represented in a 2 dimensional space and is now much easier to understand. This is why convolution layers are so useful.

1.2 More on Convolution Layers

Convolution layers are useful for feature extraction. For instance if we were running facial recognition we would use multiple layers that would each progressively extract features with more and more detail. An example of this is shown below.

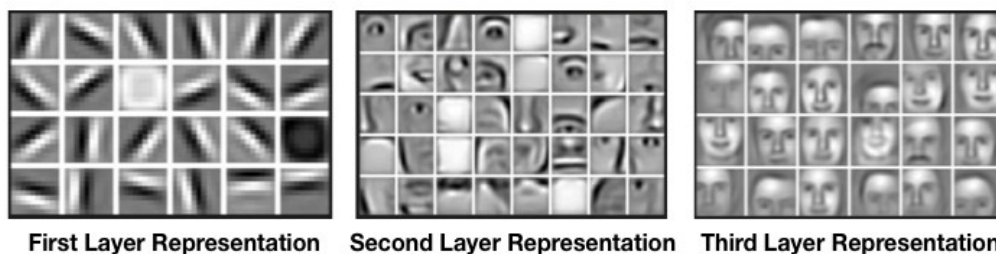


Figure 1.3: Extracting features from an image

Once we have extracted our features we can use a Flatten layer to turn the extracted features back into a 1 dimensional vector. Then this can be used as input to a dense layer. A representation of this is shown below. You can see that there is also a pooling layer all that does is reduce the size of the image. This is useful for increasing the efficiency of the network.

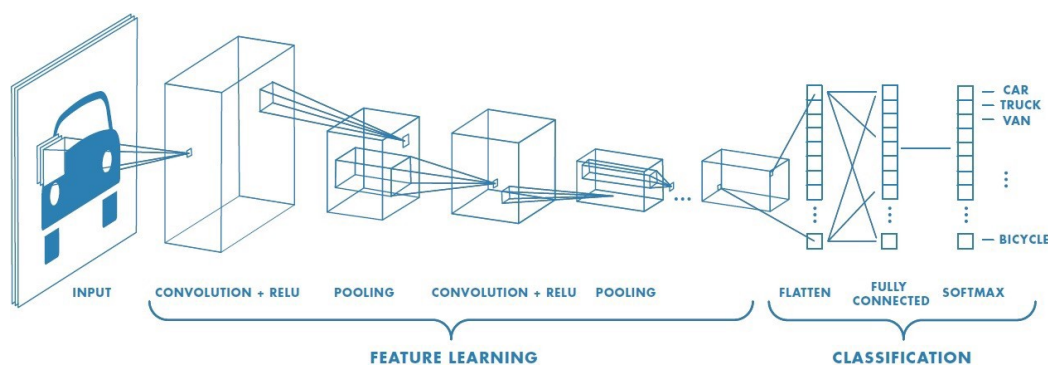


Figure 1.4: Representation of a CNN

Chapter 2

Using the Convolution Layer

Now that we have a better understanding of the convolution layer we can use it in our neural network.

2.1 Creating a Convolution Layer

Below is a simple example of a convolutional neural network. Where all the parameters are explained.

```
1  from keras.layers import Conv2D, MaxPooling2D, Dense, Sequential,
    InputLayer, Flatten
2  # a sequential model is a model that is made up of layers
3  model = Sequential()
4  # the input layer is the first layer in the model
5  model.add(InputLayer(input_shape=(100, 66, 1)))
6
7  # the convolution layer
8  # the different parameters are:
9  # the number of output filters: 32
10 # the kernel size: (3, 3) specifying the height and width of the 2D
    convolution window.
11 # the activation function: ReLU the type of activation function that
    is used in the convolution layer.
12 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
13
14 # the pooling layer
15 # the pooling layer is used to reduce the size of the image.
16 # the pool size: (2, 2) specifying the height and width of the 2D
    pooling window.
17 model.add(MaxPooling2D(pool_size=(2, 2)))
18
19 # the flatten layer
20 # the flatten layer is used to turn the output of the convolution
    layer into a 1 dimensional vector.
21 model.add(Flatten())
22
23 # our dense layers, used to do the classification
24 model.add(Dense(100, activation='relu'))
25 model.add(Dense(50, activation='relu'))
26 model.add(Dense(10, activation='relu'))
27 model.add(Dense(1, activation='relu'))
```

Chapter 3

Applying the Convolution Layer

Now that you have a good grasp of the convolution layer we can apply it to our neural network. Open the model from your previous lab and add try experimenting by adding the convolution layer to the model. See if multiple convolution layers function better than a single one. Once you are done with this and are happy with your model you can save the model and download it. Then you can run the model with the following code. "python runNetwork.py -m yourConvolutionModel.h5"



Roadblock: You may have noticed that while your training model has a very low mean squared error, while your testing model has a very high mean squared error. This is caused by overfitting. Overfitting is when the model is too complex and cannot generalize to new data. This is a problem that we will solve in the next lab.