Summary

This project is about the implementation of image recognition using convolution neural networks. We make use of TensorFlow, Theano and keras deep learning libraries to build the convolution neural network and implement image recognition.

We make use of Convolution, Flattening, Max pooling and Full connection to create a CNN. In convolution, the image is first converted into pixels. Then it is passed to feature detector and a feature map is generated. We then make use of the ReLu Layer where the ReLU function f(x)=max(0,x) is applied element-wise to the output. Flattening is used to pooling layer into an input layer of a future ANN. After which, max pooling is used which selects the max value in matrix i.e. Selecting and preserving the features. And finally, the full connection is used to add a whole artificial neural network to a convolution neural network.

The objective of the project is the distinguish cats from dogs by using the features extracted from the images in the training dataset. We then test the accuracy of our convolution network model by using it on our test data. The model learns if it has performed an incorrect classification and then make changes in the weights or the feature detectors and runs this process at least 1000 times which helps it in learning and improving the accuracy.

Keywords: [Neural network, convolution, ANN, TensorFlow, Max pooling, image recognition, ReLU, Flattening, model, keras, cats and dogs]

[Title Here, up to 12 Words, on One to Two Lines]

We are implementing image recognition using convolution neural network. The computer utilizes machine vision technology with artificial intelligence to achieve it. It is easy for humans to recognize objects but same task if difficult for computers. If we look at an object such as a pencil or a broken pencil, we don’t have to study it consciously to tell what it is. For a computer identifying an object is a difficult problem. Image recognition can be achieved using machine learning where it is designed to resemble the same way a human brain thinks.

There has been considerable growth in deep learning due to the rapid improvements in fast information storage capacity, increase in computation power and graphical processors. Deep learning techniques are inspired by the information processing and communication pattern similar to that of the human brain.

We decided to learn more about deep learning and use it to implement a real-world problem. We will make out model train itself by looking at images of cats and dogs. Based on the features distinctions it will learn to distinguish a dog from a cat. It will get a reward for every correct guess and it will keep on optimizing the accuracy as we keep on providing it with a larger input.

**Methodology and Analysis**

We have the data of X observation we divide the data into X test data and X train data. The first step is convolution. A convolution is a mask or a small matrix we use it to blur, sharpen, edge detection. This is possible using convolution between a convolution matrix and an image.

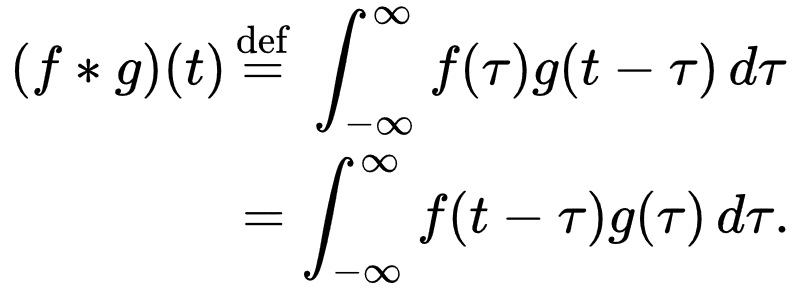
Example:

A close up of a clock

Description generated with very high confidence

It is a process of adding each element of an image to the local neighbors which are weighted by the kernel.

Mathematically it can be represented as,



We take the input image from train data; each image is represented in pixels. It is then passed to the feature detector and a feature map is generated.

A close up of a keyboard

Description generated with high confidence

The multiplication is direct multiplication and not as we typically multiply matrices. The input image is multiplied with the feature detector which gives us the feature map. We can also see that we have reduced the size of the image. This is a very important feature of feature detector of the convolution step which is to make the size of image smaller, so it is easier to process. We are losing some information as we can see from the resulting matrix. But the purpose of feature detection is to detect some features and ignore the other irrelevant information. And this is how we see in real life we don’t look at every pixel to speak what we see in real life. We look at the features of an object and recognize it. It allows to bring forward and get rid of the unnecessary things.

A picture containing object

Description generated with high confidence

We create multiple feature maps because we use different layers. And that is how we preserve lots of information so we don’t have just one feature map.

Implementation in python:

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

A picture containing object, antenna

Description generated with high confidence

We have our convolutional layer which we apply to rectifier function. The reason we apply rectifier is to increase non-linearity in our image. Images are highly non-linear especially if we are recognizing objects next to each other and the image is going to have a lot of nonlinear elements and the transition between pixels would be nonlinear. When we apply convolution, we risk of creating something linear therefore there is a need to break it up.

We then make use of pooling, which combines outputs of neuron clusters at one layer into a single neuron in the next layer. We could use max pooling where we make use of the maximum value from each cluster of neurons at a prior layer another example is average pooling where we use the average value.

A screenshot of a cell phone

Description generated with very high confidence

As we can see here that we take the 2x2 matrix in the feature and consider the maximum point in the pooled feature map.

A rough example from our analysis could be that 4 in the feature map represent the pointy cat ears and even if the image is rotated and is distorted we would be able to look at it from the pooled feature map. Therefore, through max pooling we are preserving the features and also reducing the size significantly.

We then perform flattening step in which we pull the feature maps and flatten them, so we put it into one long column sequentially.

A screenshot of a cell phone

Description generated with very high confidence

This would be our input for our artificial neural network.

After that we apply the last step of full connection, we add a whole artificial network to our convolution neural network. So, we have the input layer and the hidden layers all put together which will get us the fully connected layer. As compared to an artificial neural network hidden layers are not fully connected whereas in the convolutional neural network we are using fully connected layer. So, basically whole column of vector of output that we have after flatting we are passing it into the input layer. The main purpose of an artificial neural network is to combine features into more attribute that predict the classes even better.

A close up of a map

Description generated with high confidence

Here we got a better looking artificial network where we have five attributes on the input and we have six neurons in the first hidden layer. We have eight neurons in the second hidden layer. We have two outputs one is Dog and the other is Cat.

Backpropagation is a technique which is used to minimize error. If we have a prediction of 80% that it is a dog, but the actual result is a cat, then the error calculated. We will use the cost function and we will use cross entropy and the mean squared error. But for now, we will say it as loss function which tells us how well our network is performing. Each time an error is calculated and then it backpropagates through the network and again and the weight is adjusted in the network. So, this is a way in which our network trains by repetitive learning. A close up of a map

Description generated with high confidence

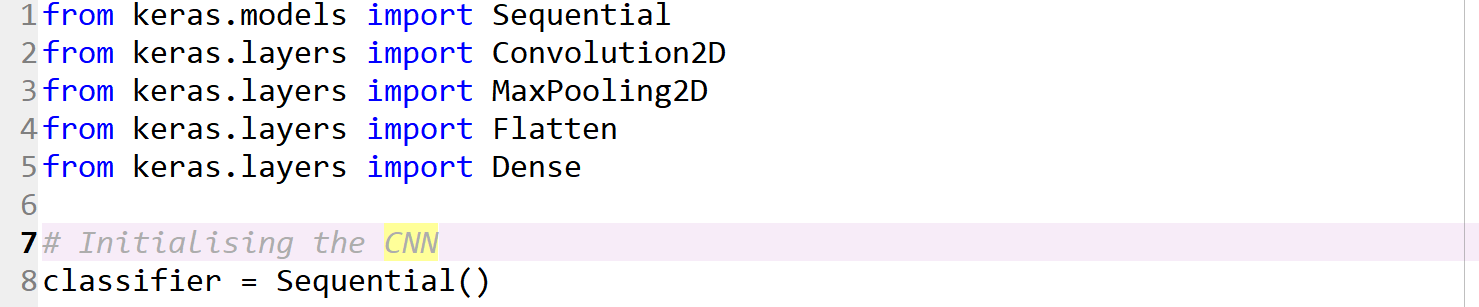
How does these two classes work, or how does this classification or images play out? To understand let us start with the top neuron going to the dog. The main purpose is to assign the weights to each synopsis that connects the dog so that we know which previous neurons are important for the dog. Below is the following hypothetical example in which we got weights between 0 and 1. 1 means that neuron is confident that it found perfect match for this feature and 0 means it didn’t find any feature.

A picture containing text

Description generated with very high confidence

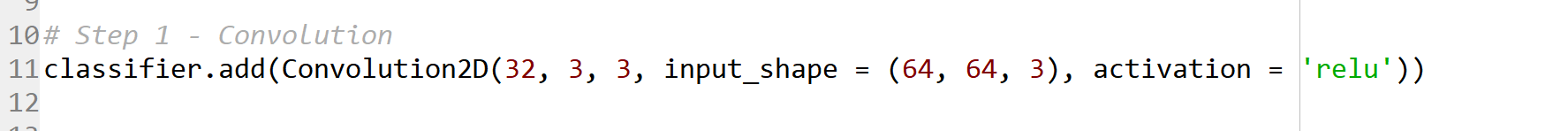
Let us pass an image of the dog and the values are propagated through the network and we get certain values and so this time the dog and the cat neurons don’t know they don’t have the image of the dog here they don’t know whether it’s a dog or a cat, but they have learned to listen to what weights are being assigned to the synopsis. Based on the high values the class output is decided.

**Data analysis**



Sequential library is used to initialize our neural network. Convolution2D is used to perform the first step of CNN in which we add the convolution layer so we are working on images and since the image is in 2D we use a convolution2D to deal with images.

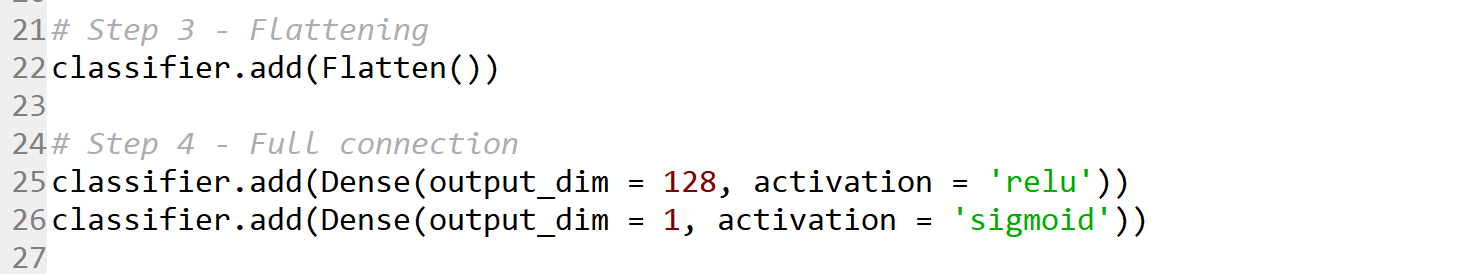
The next package is Maxpooling2D and this package is used to extract maximus values (features). The next package is Flatten which is used to convert the pooled layer into a large feature vector which becomes the input of the fully connected layer. Dense package is used to add the fully connected layer and a classical neural network so basically each package corresponds to one step of the construction at the CNN. Next step is initializing the sequential object.



In this step it will convert the image into a matrix of pixel containing 0 and 1. “This layer creates a convolution kernel that is convolved with the input layer to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well”.



We are creating in this step and max pooling matrix size is 2\*2.

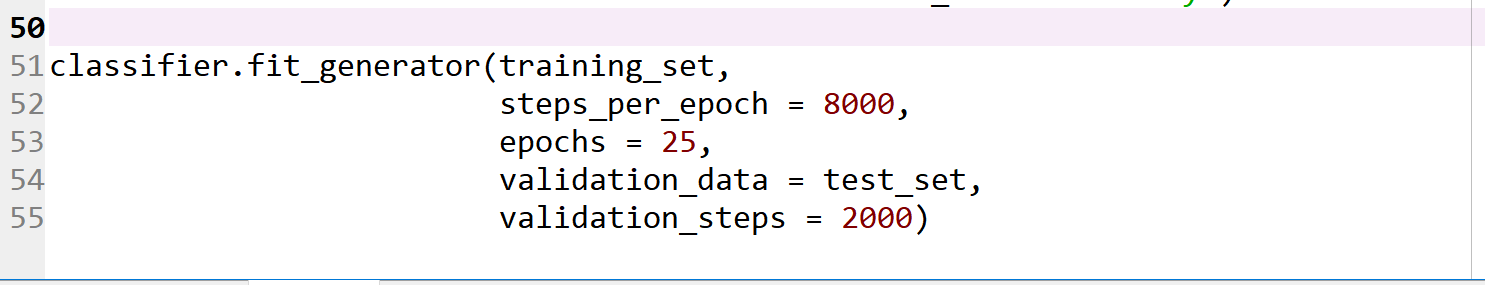


All our pooled feature maps have been put into a single vector. The vector contains all the different cells of all the different map.

A screenshot of a social media post

Description generated with very high confidence

For compiling our CNN, we are using stochastic gradient decent algorithm as an optimizer and binary cross entropy for the cost function. We are creating test dataset and train dataset. ImageDataGenerator function is used for image augmentation.



This step is all about fitting the model with epochs value as 25 and we validate it with the test data.A screenshot of a cell phone

Description generated with high confidence

In this, we can observe the train and test dataset accuracy. In starting(Epoch) training set accuracy is 60% whereas test dataset accuracy is 67%. While if we consider epoch 6th we can observe the training set to be 75% and test dataset to be 76% and we can observe on 10th epoch the accuracy of trainset is 80 % and test set is 78.5%.

**Conclusion**

Thus, we can see the implementation of image recognition using convolution neural networks. We were able to achieve an accuracy of 80% after using our model on the test set.

* After the implementation of CNN for image recognition we can see that the accuracy is dependent on the size of our data. If we keep on feeding our model with more data, the accuracy keeps on increasing.
* We took the target size of 64 X 64 and if we increase our target size we could further improve our accuracy but increasing the target size would increase the computational time required.
* Also, the accuracy can be improved by increasing the number of epochs. In this example we took 10 epochs and the computational time required was around 20 minutes. We could have increased the number of epochs, but it would have considerably increased the computational further.

References

1) N., & A., M. (1970, January 01). Neural Networks and Deep Learning. Retrieved from http://neuralnetworksanddeeplearning.com/chap1.html

2) Nicholson, C. V., & Gibson, A. (n.d.). Introduction to Deep Neural Networks (Deep Learning). Retrieved from <https://deeplearning4j.org/neuralnet-overview>

3) Kernel (image processing). (2018, March 21). Retrieved from <https://en.wikipedia.org/wiki/Kernel_(image_processing)>

4) KDnuggets. (n.d.). Retrieved from <https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html>

5) Deep learning. (2018, May 18). Retrieved from https://en.wikipedia.org/wiki/Deep\_learning