ENPM673 - Perception for Autonomous Robots

Project 6

Convolution Neural Network

Authors:
Trevian Jenkins (116781381)
Karan Sutradhar (117037272)
Markose Jacob (117000269)

Date: May 16th, 2020

Objective:

The basic objective of this project is to classify the images of dogs and cats in the given dataset using the implementation of a Convolution Neural Network(CNN). While human beings naturally develop the capability to easily tell cats, as in figure 1, from dogs, imposing this ability upon a computer is a much more difficult task.

Fig. 1: A sample subset of cats among the images to be classified as “cat” or “dog”

Convolution Neural Networks:

Convolution Neural Network (CNN) also known as comp net is an artificial neural network popularly used for analyzing images. Image analysis has been the widespread use of CNN’s and they can also be used for other data analysis or classification problems. Most generally CNN is an artificial neural network that has some type of specialization for being able to pick out or detect patterns and make sense of them. This pattern detection is what makes CNN very useful for image analysis. A CNN has hidden layers called convolutional layers and also has other non-convolutional layers. But the basis of CNN is the convolutional layers. A convolutional layer basically receives inputs and transforms the input in some way and then outputs the transformed input to the next layer, with the convolutional layer this transformation is a convolution operation.

Convolutional neural networks are able to detect patterns in images more precisely the convolutional layers are able to detect patterns. With each convolutional layer the number of specific filters need to be specified that the layers should have and these filters are actually

what detects the patterns. Patterns in a single image are basically multiple edges, shapes, textures, objects etc. So, one type of pattern that the filter can detect is edges and then the filter will be called edge detector. Some filters detect corners, squares and circles and these are geometric filters in CNN. The deeper the CNN is the more complex these filters become. So, in a larger scenario and in a deeper layer the filter may be able to detect larger objects like dogs and cats in our problem.

Convolution layers:

Convolutional layers are the layers where filters are applied to the original image or other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.
The use of convolution layers is to get the features of the image to be extracted. The first convolution layer is used to get the low level features of the image such as color, orientation, edges, etc. With more and more convolution layers added to the network it is much easier to get more high level features of the image.

Pooling Layers:

The pooling layers used to reduce the spatial dimensions, but not depth, on a CNN model, basically this is the gain:

- By having less spatial information you gain computation performance.
- Less spatial information also means fewer parameters, so less chance to over-fit.
- You get some translation invariance.

Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

Accepts a volume of size $W_1 \times H_1 \times D_1$
Requires two hyperparameters:

- their spatial extent F,
- the stride S,

Produces a volume of size $W_2 \times H_2 \times D_2$ where:

$$W_2 = \frac{(W_1 - F)}{S+1}$$

$$H_2 = \frac{(H_1 - F)}{S+1}$$

$$D_2 = D_1$$

Optimizer:

Optimizers update the weight parameters to minimize the loss function. Loss function acts as guides to the terrain telling optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum.

Stochastic Gradient Descent:

Gradient descent is basically an optimization algorithm which helps in reducing the cost function This will help models to make accurate predictions. Gradient indicates the direction of increase. As we want to find the minimum point in the valley we need to go in the opposite direction of the gradient. We update parameters in the negative gradient direction to minimize the loss.

$$\theta = \theta - \eta \nabla J(\theta;\ x;\ y)$$

θ is the weight parameter, η is the learning rate and $\nabla J(\theta;x,y)$ is the gradient of weight parameter θ.

Activation Function:

Rectified Linear Unit (ReLU) is a type of activation function. Mathematically, it is defined as y = max(0, x). The activation function is the nonlinear transformation that we do over the input signal. This transformed output is then sent to the next layer of neurons as input.

TensorFlow:

Tensorflow is a library used for the computation of the numerical using graph data flow. For example if the first convolutional layer has 5 filters with a kernel size of 5x5 with the same padding. The same padding means both the output tensor and input tensor should have the same height and width. Tensorflow will add zeros to the rows and columns to ensure the same size. In a graph, the nodes represent the mathematical functions, while the ends of the graph represent the multidimensional data arrays, which is also called a tensor. The major advantage of this architecture is that it can order multiple calculations to one or more GPU or CPU without writing code multiple times.

Workflow:

We implemented our CNN using tensorflow's Deep Neural Network model, which utilizes the DNN class in tflearn. We are first resizing our image to 50 by 50 and converting it to grayscale before we pass it through the Neural Network. We are using 4 neural network layers in our network. 3 convolutional layers and 1 fully connected layer. 1 layer is for linear problems and 3 layers for nonlinear problems. Our workflow uses 5 epochs, with a learning rate of 0.001. For training data, we used a total of 18000 images of cats and dogs.

Results:

Our results were accurate to a reasonable degree. With successive epochs, we were able to achieve up to 88% accuracy. ***This figure was achieved after executing the program approximately 3 times***. Figure 2 shows a subset of our results from our program.

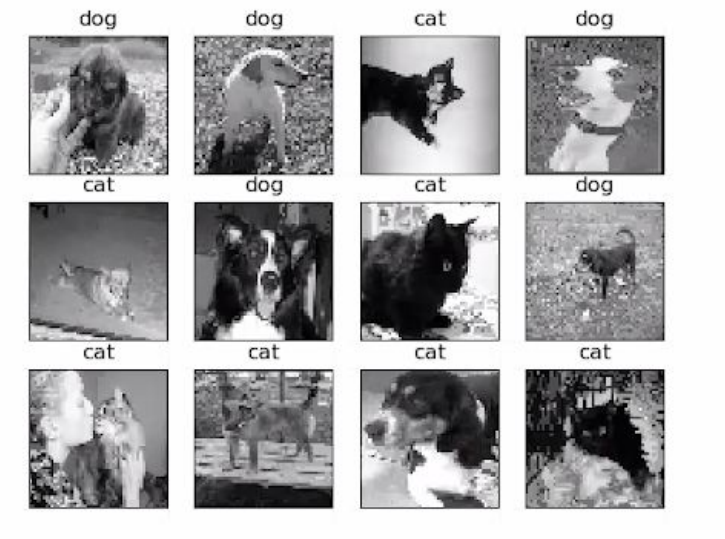
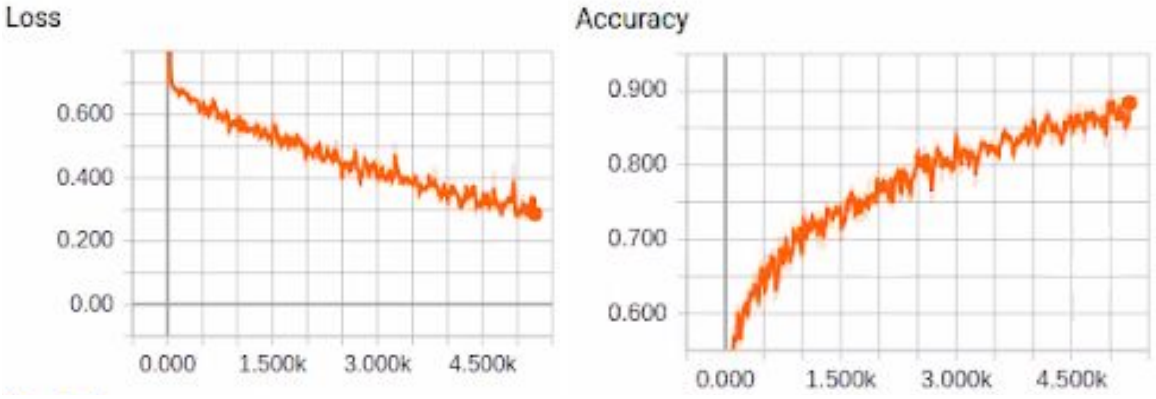


Fig. 2: Montage of sample results from our test data, showing 9 out of 12 correct estimations

Using Tensorboard, we visualized the accuracy of our CNN as the model was trained with more images. Figure 3a shows the graph of our accuracy as we tested images over the course of the last epoch, while figure 3b shows the loss over the same epoch.

Using the DNN model we created from the test set, we tested our model on all of the images in the training set. We compiled two CSV files of our test results, with one CSV file containing the image indices that belong to the cat category, and the other CSV file containing the images



(a) (b)

Fig. 3: (a) Accuracy and (b) loss over the course of last epoch

belonging to the dog category. The probability score *p* for each image ranges from 0.0 to 1.0, with values closer to 0.0 indicating a higher likelihood of the animal being a cat, and values closer to 1.0 more indicative of the image containing a dog. Adjacent to each index in our CSV files is a value indicating the likelihood that the animal is either a cat or a dog, as described in the equation below. We converted the probability score *p* into a new likelihood score x using the following equation, saving *x* into the CSV file:

$$x = 0.5 + |p - 0.5|$$

A value closer to 1.0 indicates a higher likelihood of the given animal being the assumed animal. The total loss *L* of our test data (indicated as 979 in figure 4) was calculated from the cross-entropy function:

$$L = -\sum_{i}^{m}\sum_{j}^{n} y_{ij} log(p_{ij})$$

In the above equation, the category j (out of n = 2 categories) is used to indicate that y_{ij} has a value of 1 if the image belongs to category j, and 0 otherwise. The value p_{ij} is the probability of the image belonging to category j. Figure 4 contains a table of metrics for our results of testing the training data. The loss and accuracy for the training set are included in figure 4.

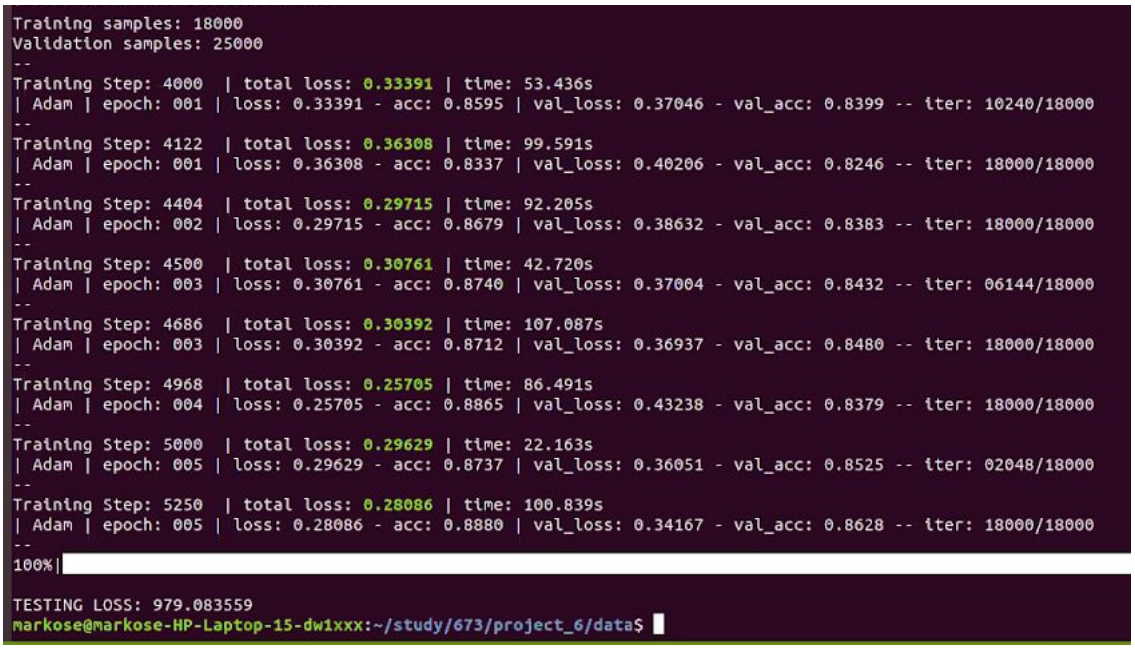


Fig. 4: Metrics from testing of the training data

Conclusion:

Our results gradually improved through successive iterations of epochs, for a maximum animal classification accuracy of 88%. After training the model, the results of testing our training data are displayed in figures 3 and 4, while the results of our testing data are shown in the montage sample in figure 2, as well as the CSV files for the full result.

References:

https://cs231n.github.io/convolutional-networks/#pool
https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3
https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17
https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/