AlexNet for Deep Convolutional Neural Networks

Sanket Waghmare
Department of Software Engineering
Rochester Institute of Technology
Rochester, NY-14623, USA
sw1725@rit.edu

Advisor - Rui Li
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY-14623, USA
rxlics@rit.edu

December 6, 2021

1 Introduction

Machine learning and its various applications have been very resourceful in the current climate of object recognition. More and more detailed insights can be generated and more and more information can be used to have a better understanding of the data that we are provided with by just having a supplemental stream of data. Better detailing and segmenting of data can prove to be revolutionary in terms of learning from the original dataset in order to make predictions more precise and accurate. Learning capacity needs to also be analysed to have optimal conditions wherein learning takes place. Convolutional neural networks are very flexible in terms of their manoeuvrability which makes them pivotal to the model. They can be operated and strategized with to come up with an optimal solution. So, bringing into play these convolutional neural networks coupled with a set of neat strategy is what makes this model get ideal results.

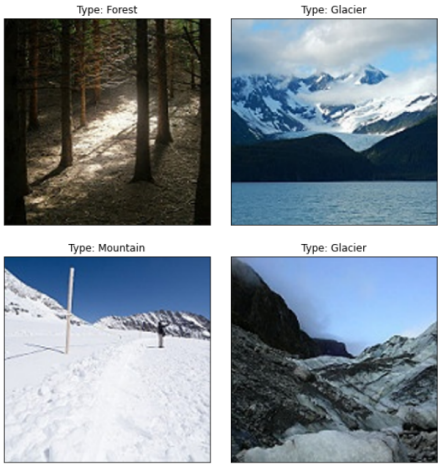

Figure 1: Training Images

2 Related Work

The Convolutional Layer model that is used here was first published in 'ImageNet Classification with Deep Convolutional Neural Networks' [1]. This paper details the use of the model with extensive background research on the base research that was done for the development of the model. This paper was a part of Adit Deshpande's 9 most important deep learning papers list, where it was mentioned to be a pioneering paper in this field. Similar ideas have been developed earlier, by Yan LeCunn [2], which applied Gradient Based Learning. This model has two convolutional layers which, like the first paper by Alex Krizhevsky, are followed by max pooling layers. After that, there are three dense layers which forms the basis of the model. This methodology can be traced even further back to Kunihiko Fukushima [3], whose model was called Neocognitron. Originally, Neocognitron was used for visual pattern recognition, which had an ability to recognize stimulus patterns based on the geometrical similaritybut without having an affect by their positions. Image Classification has also been widely researched. Dan Ciresan's Multi

Column Deep Neural Networks for Image Classification [4] used a biologically plausible deep artificial neural network architecture for this research, which had the best error rate on the MNIST digit recognition task. AlexNet proved that using supervised learning for modeling a deep convolutional neural net gives notable results on large datasets.

3 Methodology

Starting with the dataset, we see that there are separate folders for training images, testing images are images separated just for predictions. So, the dataset contains 14000 images that are used for training. Other than that, there are 3000 images that are used for testing and another set of 7300 images that are used for prediction purposes. First step is to input all the training images along with their labels. The labels are then categorised so that they can then be of use for attaching to the original training images. For testing images as well, images are all brought in and their labels are stored to check on with the test images and we get the predicted labels. The size of the training and testing images are set to a predefined value so that they match the values with the input layer of the model and there is no discrepancy. For the input layer of the model and the first convolutional layer, the filters are set to 96 and the kernel size is set to 11x11x3 with the strides being 4. After this, a max pooling layer of pool size 3 and strides of 2 is set. Batch normalisation is also added which is then accepted as input for the second convolutional layer.

Second convolutional layer uses a filter of 256 with a kernel size of 5 and strides of 1. This convolutional layer is also followed with a max pooling layer with a pool size of 3 and strides of 2 and a batch normalisation layer. Once the first two convolutional layers are set with a max pooling and batch normalisation layer following them, the next and final 3 convolutional layers are coupled so as they are one after the other. The third convolutional layer has a filter of 384 with a kernel size of 3 and strides of 1. Fourth convolutional layer is similar to the third one in the sense that it also has filters of 384 with kernel size of 3 and strides of 1. The fifth and final convolutional differs to its two predecessor layers in the sense that this layer has filters of 256 but similar to the two previous layers, it has a kernel size of 3 and strides of 1. After the final three convolutional layers are done, it again follows the pattern set by the first two convolutional layers by having a max pooling layer of pool size 3 and strides of 2 with a batch normalisation layer as well. All 5 convolutional layers use relu as their respective activation.

The dense layer performs matrix-vector multiplications where parameters are the matrix values that are trained. These values are updated using backpropagation. The dense layer produces an output dimensional vector therefore can be used to alter dimensions of the given vector as well. The dropout layer is used with the aim to prevent overfitting. It sets, at random, some input units to 0 with a frequency specified in the argument which indicates rate at each step during training. So, a hierarchy of layers combining dense and dropout are added following the 5 convolutional layers. First set of dense and dropout layers have a shape of 4096 with an input shape the same as the first convolutional layer with an activation of relu. The dropout at this stage is 0.4 and these are followed by a batch normalisation layer. The second set of dense and dropout layers also follow the same set of features, with the shape being 4096 and activation being relu with the dropout being 0.4. No input shape is provided here as it is just pivotal in the first set. The final set of dense and dropout layers have a shape of 1000 with relu activation and a dropout of 0.4 which is then followed with a batch normalisation layer. Now, in the context, the output prediction is Multi class rather than two class, so the output layer has an activation of softmax with the shape of the dense layer being 20.

4 Evaluation

After the model is set up, it is compiled with Adam as the optimizer and sparse categorical entropy as the loss. Metrics on which we evaluate the progress is mainly the accuracy of the model. Once the model has been defined and compiled, the next step is to fit it with the training data as well as the labels for the training data. A number of epochs are tested and then finally set to 25 so that we can get a good accuracy as well as wait for the training accuracy to stabilize, which happens around the end of the model fit in this model. For the first epoch, we get a training accuracy of 0.5043 at a loss of 1.5002. On average, it takes around 570 seconds to execute each epoch. At the end of the model fit, we get a final training accuracy of 0.9296 at a loss of 0.2046. The next step is to evaluate the

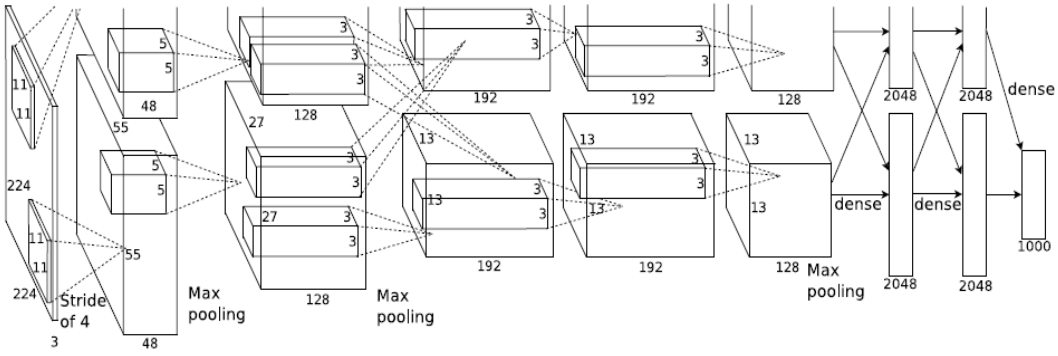

Figure 2: Alexnet Architecture as portrayed in [1]

Figure 3: Testing Images

model with the testing data and the labels used for the testing data. Here, the test accuracy is 0.7863 at a loss of 0.8114. Based on this score, the test images are shown and predicted with their original test labels as well as the predicted labels shown on top of each predicted image. After that, the predicted images folder is selected and is prepared in a similar way as the training and testing images, with the exception of labels as the prediction images do not have labels. After they are set, the model is used to predict the label on the prediction images and they are shown with the predicted labels above them.

5 Research Questions

1. Importance and need for convolutional layers.

Convolutional layers are very important to help with the workability of the model. They are also pivotal in defining a structure and providing an initial base for the model to start doing its work. In this model, there are 5 convolutional layers that are uniquely defined and play a role in the big picture. The first two convolutional layers are followed by the max polling and the batch normalization layers, but the last three layers are concurrent to each other.

So, to validate the utilization capabilities of these layers, we run the model with and without these layers being placed right after each other. We set the epochs to a predefined value, so that they remain consistent across all the three runs. First, we run the model with just the first three convolutional layers and validate their final accuracies and loss. Here, we see that after just one epoch, the training accuracy already surpasses 0.56 at a loss of 1.3602. After the end of all the predefined epochs, we see that it achieves a training accuracy of 0.7325 at a loss of 0.7440. For the test data, this achieves an accuracy of 0.6770 at a loss of 0.9170.

Next, we add the fourth convolutional layer and run the model. Here we see that after the first epoch, it achieves an accuracy of 0.5247 at a loss of 1.4581. After the end of all the predefined epochs, we see that it achieves a training accuracy of 0.7242 at a loss of 0.7800. For the test data, this achieves an accuracy of 0.5233 at a loss of 1.8660. Next, we add the fifth convolutional layer. Here we see that after the first epoch, it achieves an accuracy of 0.5043 at a loss of 1.5002. After the end of all the predefined epochs, we see that it achieves a training accuracy of 0.6958 at a loss of 0.8336. For the test data, this achieves an accuracy of 0.3730 at a loss of 2.2425. Hence, we see that when three convolutional layers are used, is when we get the best results for a predefined set of epochs, as well as the best test accuracy coupled with the lowest loss. This may be due to the fact that the model which uses five convolutional layers performs way better when

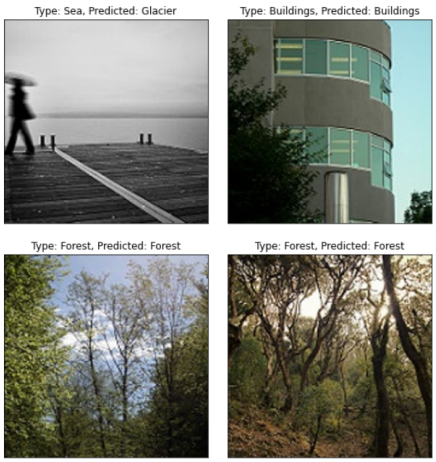

Figure 4: Predicted Images

it has an abundance of training data, for example the ImageNet dataset which has over a million sets of data. But, when using the data used in this project, usage of only three convolutional layers is sufficient to extract a paramount amount of knowledge from the dataset. This research question proved very useful in getting this result and provided various insights into the convolutional layers and their optimal utilization.

2. Analysis of the Importance of Dropout.

We saw earlier that the dropout layer is used with the aim to prevent overfitting. Inputs that have not yet been set to zero are scaled to make sure that the overall sum of inputs remain unchanged. So, to validate, we run the two models with a predefined set of epochs, where one model has dropout in the context that is defined by the model, and the other has no dropout layer. When we run the first model, we see that after the first epoch, it achieves an accuracy of 0.5043 at a loss of 1.5002. After the end of all the predefined epochs, we see that it achieves a training accuracy of 0.6958 at a loss of 0.8336. For the test data, this achieves an accuracy of 0.3730 at a loss of 2.2425. Next, we run the second model, where we see that after the first epoch, it achieves an accuracy of 0.5078 at a loss of 1.4411. After the end of all the predefined epochs, we see that it achieves a training accuracy of 0.6779 at a loss of 0.8677. For the test data, this achieves an accuracy of 0.6153 at a loss of 1.1393. So, when we compare the results of the two models, we see that they both achieve similar train accuracy at a similar loss as well across all the epochs. But, the test accuracy of the second model is way better than the first model, as well as nearly half the loss of the first model. This can also be contributed to the fact of the dataset being the main difference and causing this result. It also shows us the importance of dropout, as well as how and when to use it so that it can get the optimal result.

3. Analysis of the number of Epochs.

The dataset passing through the network once is what constitutes an epoch. It is crucial to set a number to have an optimal result and not underfit or overfit. Final number of epochs depend on the size of the dataset and the relationship between the various individual features. Validation loss also needs to be looked at to see if it stays in its natural rhythm. If validation loss increases, there may be a case of overfitting. The model is run on different set of epochs, to explore the training accuracy across epochs and to also have an eye on the loss for the training accuracy. The first epoch has a training accuracy of 0.5043 at a loss of 1.5002. This loss halves by the third epoch to 0.8336 with a training accuracy of 0.6958, which is nearly 20 percent increase in a small set of epochs. The training accuracy increases steadily across the next few epochs, with the loss also going down at a steady rate. The fifteenth epoch gets a training accuracy of 0.8408 at a loss of 0.4459. After that, the rate of increase of the accuracy and decrease of loss slows down, as it normally should. The final accuracy is 0.9296 at a loss of 0.2046, which is a 43 percent increase and an 89 percent decrease of loss since the first epoch. This goes to show the learning rate of the model across epochs as well as the insights gained from the varied series of loss and accuracy values that were calculated.

6 Conclusion

Deep Convolutional Neural Network model by Alex Krizhevsky is a very robust and powerful model in terms of its learning as well as training. It used five convolutional layers, two of which are followed by max pooling as well as batch normalization layers. The five convolutional layers are then followed by a combination of three dense and dropout layers. The build of the model used on the dataset in this project got a final training accuracy of 0.9296 at a loss of 0.2046, and a testing accuracy of 0.7863 at a loss of 0.8114. Accuracy of 92 percent is very high, but as seen in the research questions, we see specific deficiencies due to the nature of the dataset coupled with the large neural net. So, this paper shows the working and detailed analysis of the model and the research questions dive a bit deeper into

the inner workings to uncover fascinating information, which is definitely useful for any future references as well as advancement of the model in consideration.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[4] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.