Application of Convolutional Neural Networks and Deep Learning in Multi-Class Image Classification

Taylor Han

Department of Cognitive Science
University of California, San Diego

Nathan Orenstein

Department of Cognitive Science
University of California, San Diego

ABSTRACT

Neural networks and deep learning algorithms have been acknowledged for demonstrating their strengths in predicting multi-class outputs given original multi-dimensional inputs. This paper attempts to create the optimal convolutional neural network for the CIFAR-10 dataset using the neural networks library Keras. A varying selection of filters, activation functions, pooling functions, and optimizers were tried for the learning algorithm. The functionality of each learning algorithm was determined using their corresponding test accuracies, and the algorithm that produced the highest test accuracy is considered to be the optimal convolutional neural network.

I. INTRODUCTION

Image classification is a vital task that is used in many different applications. To perform image classification efficiently, deep machine learning has been adopted to run this process. Despite its wide use and many implementations, there is no definitively "best" way to create a machine learning algorithm for the purpose of image classification.

The most commonly used framework for image classification is the convolutional neural network, but there are many ways to implement one. The different factors that were analyzed in this paper were the filter size, the activation and pooling functions used, and the type of optimizer used. Altogether, there were 90 different combinations that were tested to find the optimal convolutional neural network with the least testing error.

II. METHODS

A. Dataset

The dataset used was the CIFAR-10 dataset. It consists of 60,000 32x32 colored images that fall into ten distinct classes: airplanes, birds, automobiles, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images from each of the ten classes, resulting in the total of 60,000 image instances. The dataset has a 50,000/10,000 training/test split.

1) 32x32 Image Samples with Ground-Truth Labels

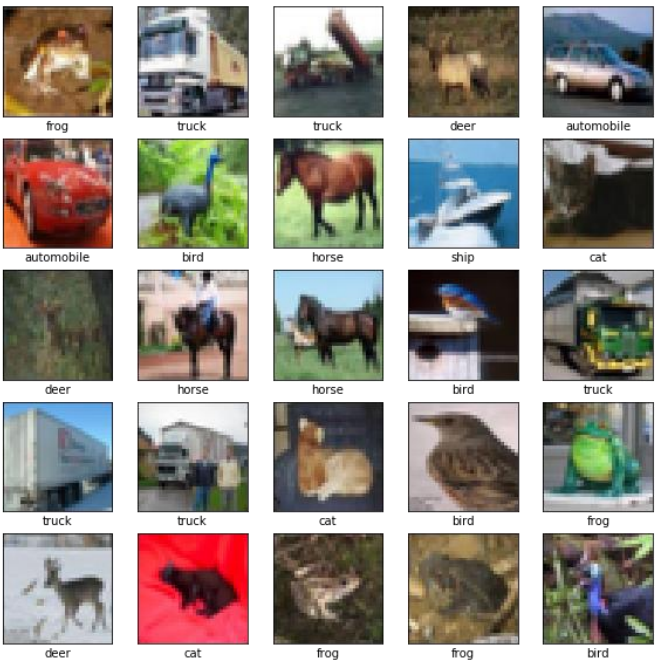


Table A. CIFAR-10 Dataset Overview

Pixel Size	32x32
# of Classes	10
# of Instances	60,000
Training Size	50,000
Testing Size	10,000

B. Learning Algorithms

The main task of this paper was to tune the hyper-parameters to find the ones that lead to the optimal convolutional neural network. The specific hyper-parameters that were adjusted were the filter size, type of activation function used, type of pooling function used, and type of optimizer used.

1) Filter Size

The size of the filter refers to how much larger the "net" is when doing convolution. The bigger the filter, the more spatial reduction happens at each level of convolution. The filters employed in this paper are 64, 128, and 256.

2) Activation Functions

The activation function is a non-linear transformation that is applied to every item in the input. It does not reduce the dimensionality of the input at all. The different activation functions that were tested are ReLU, Sigmoid, and Softplus. ReLU (or rectified linear unit) returns max(0, x), where x is the value of the input. Softplus returns $\ln(1 + e^{x})$. While ReLU is a piecewise function, Softplus smooths it out into a continuous function and is often referred to as SmoothReLU. Sigmoid returns $1 / (1 + e^{-x})$.

3) Pooling Functions

The pooling function serves to reduce the dimensionality of the input by summarizing the values that the function is applied to. The pooling functions that were applied in this paper were in the size of 2x2, meaning that the pooling reduced every four items into one. The two pooling functions that were used were AveragePooling and MaxPooling. AveragePooling takes the average of the four inputs and then outputs that average. MaxPooling takes the maximum values of the four inputs and then outputs that value.

4) Optimization Methods

Optimization methods are what is used to update the model parameters and to minimize the loss of the algorithm. The optimization methods that were used in this paper are Stochastic Gradient Descent, AdaGrad, RMSProp, Adam, and Adamax. Stochastic Gradient Descent works by evaluating the gradient at a randomly selected sample of the input before updating, rather than taking the gradient of the entire input like in classical gradient descent [1] [2]. It has a constant learning rate across the entire process. Adaptive Gradient Algorithm (AdaGrad) adapts the learning rate to the frequency of the parameters, meaning the learning rate is larger for parameters with infrequent features and smaller for parameters with frequently occurring features. Root Mean Square Propagation (RMSProp) similarly has different learning rates for different weights, but it divides the learning rate for each weight by a running average of the magnitudes of previous gradients for that weight. Adam, derived from adaptive moment estimation, also adapts the parameter learning rates in a similar way to RMSProp, but it also uses the average of the second moment of the gradients [3]. Finally, AdaMax is a special case of Adam where its second-order moment is replaced by an infinite-order moment [4].

III. Network Structure

In this section, a graphical overview of the network structure variant of filter size of 128, ReLU, and AveragePooling is shown. Every structure deployed epoch value of 10 and used the Softmax activation function in the final layer as Softmax function distributes the probability for each class. Additionally, a Dropout layer value of 0.2 was augmented to prevent overfitting during the training process.

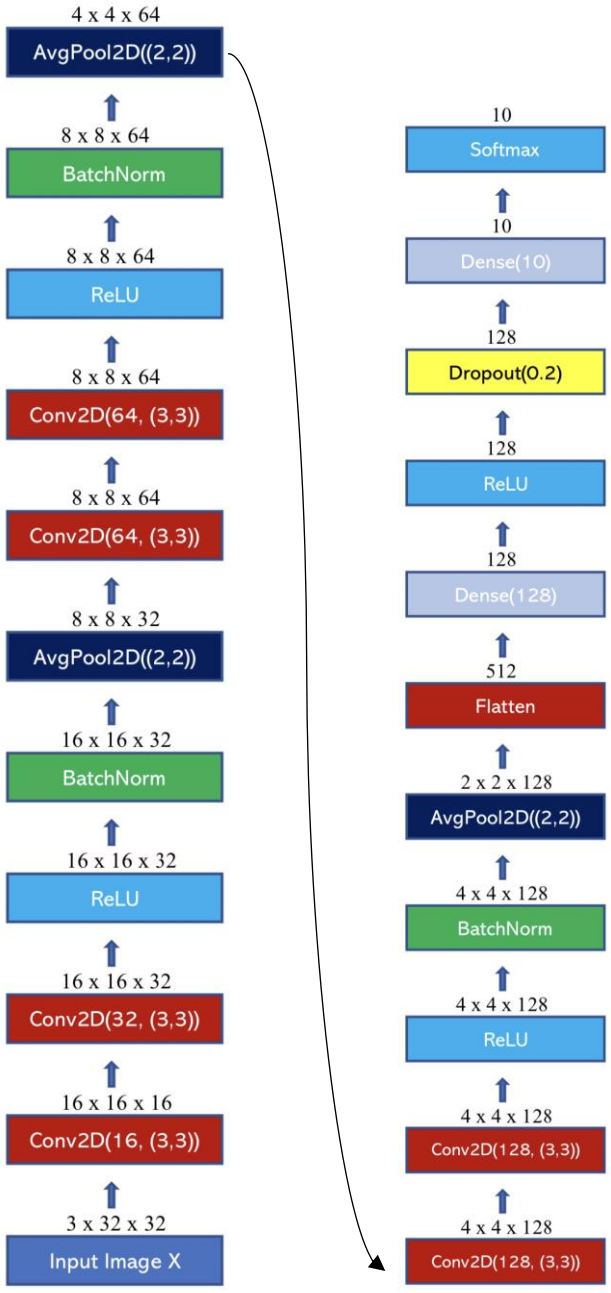


Table B. Layer Statistics

Filter Size	# of Layers	# of Trainable Parameters
64	8	126,954
128	11	349,354
256	14	1,236,522

IV. Experiments & Results

A total of 18 Jupyter Notebooks were used to provide one choice of parameters from the 3 filter options, 3 activation

function options, and 2 pooling function options. Within each of these Notebooks, the data was first preprocessed. Prior to using the images in the various tests, they were always prepared by first normalizing them, and the classifications were one-hot encoded. Five trials were conducted within each of the notebooks: one for each optimization method that were to be tested. Across these trials within each individual notebook, the convolutional neural network model was held constant with the designated parameters, aside from the varying optimization functions. For each trial, the categorical cross-entropy loss and classification accuracy were plotted for the training and testing sets, and the cross-entropy loss and classification accuracy were then recorded for the testing data for observation in the provided tables later in this paper.

There were definitely some noticeable patterns that arose from the data, though nothing was absolute. One observation is that when AveragePooling was the pooling function, the activation function that provided the best testing accuracy was almost always ReLU. However, when MaxPooling was the pooling function, the activation that provided the best testing accuracy was almost always Softplus. A possible explanation for this is that ReLU uses a piecewise maximum function as its base. Since average pooling summarizes the input by using the mean, it more accurately represents the data that it was trying to summarize. Putting this input to a strictly piecewise function like ReLU will produce strong results. On the other hand, max pooling simply selects the largest value from the four entries (2x2) that it spans. This does not summarize the data as accurately as average pooling does. However, the Softplus activation function is more or less a smoothed-out version of ReLU. By smoothing ReLU out, it becomes more forgiving for the data that is not summarized as well. This could be a possible explanation as to why the ReLU model is more accurate for AveragePooling and the Softplus model is more accurate for MaxPooling.

Furthermore, the Sigmoid activation function was almost never the best choice of the three activation functions. This could be since all values for the output of the Sigmoid activation function fall between zero and one, exclusive. This is much different from ReLU and Softplus, where the minimum is zero and the maximum has no limit. By limiting the range of outputs to be between zero and one, larger outputs get limited and will therefore have less impact on the overall algorithm. This is one theory for why the Sigmoid activation function consistently produced results that were worse than the Softplus and ReLU results.

In terms of the filter size, there does not seem to be any discernible pattern across the various trials. This suggests that the filter size might not be a strong factor in determining the accuracy of the convolutional neural network. Convolution is used to reduce the spatial volume of the data, and it seems that the amount that you reduce it by does not have any visible effect on the overall success of the algorithm. Of course, only three different filter sizes were tested in this paper, so an experiment that specifically examined the effect of filter size might be more likely to produce statistically significant results. The exception to this is that the trials with Sigmoid consistently produced worse results as the filter size increased, aside from the trials that used Stochastic Gradient Descent.

Regarding the choice of optimization function, the Adamax and RMSProp functions both produced strong results consistently. Adamax and RMSProp both keep different learning rates that are tailored for each individual parameter, so it makes sense that these two performed well overall. They also have arguably the most complex optimization techniques of the functions tested, which likely contributed to its overall success as well. Stochastic Gradient Descent was on the lower end of the spectrum in terms of testing accuracy produced from the trials. Stochastic Gradient Descent is the foundation for the other optimization methods tested, and each of them have expanded on the methods used by Stochastic Gradient Descent. Stochastic Gradient Descent has the most basic process of the five tests, so it is no surprise that it produced the weakest results overall.

The combination of parameters that yielded the most effective convolutional neural network had a filter size of 128, AveragePooling, ReLU activation functions, and RMSProp for its optimization method. The overall testing accuracy for this convolution neural network was 0.766. The combination of AveragePooling and ReLU is a strong one, as stated earlier, but it should be noted that the same combination of parameters, except with MaxPooling and Softplus activation, produced strong results as well. RMSProp as the optimization function also makes sense following the conclusion that was previously made. The filter size being 128, as stated previously, does not seem to have any correlation with the rest of the parameters, and it should be noted that 256 was the filter size when RMSProp was used with MaxPooling and Softplus.

Overall, each and every one of convolutional networks demonstrated its ability to minimize the loss function, hence proving that the model was learning every epoch. This was shown with the decreasing trend of categorical cross-entropy loss value and increasing classification accuracy value.

V. Reported Data Summary

TABLE 1A: SGD, AVERAGEPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.695	0.561	0.611
128	0.719	0.582	0.640
256	0.721	0.627	0.660

TABLE 1B: SGD, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.875	1.220	1.075
128	0.817	1.170	1.004
256	0.834	1.072	0.963

TABLE 2A: SGD, MAXPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.667	0.656	0.700
128	0.692	0.669	0.695
256	0.656	0.667	0.690

TABLE 2B: SGD, MAXPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.947	0.991	0.877
128	0.932	0.953	0.923
256	1.149	1.007	1.066

TABLE 3A: ADAM, AVERAGEPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.739	0.652	0.711
128	0.763	0.691	0.724
256	0.763	0.618	0.734

TABLE 3B: ADAM, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.784	0.993	0.878
128	0.758	0.883	0.919
256	0.726	1.090	0.842

TABLE 4A: ADAM, MAXPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.727	0.715	0.734
128	0.746	0.664	0.739
256	0.737	0.400	0.748

TABLE 4B: ADAM, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.830	0.847	0.867
128	0.821	0.971	0.855
256	0.824	1.802	0.824

TABLE 5A: ADAGRAD, AVERAGEPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.711	0.631	0.637
128	0.739	0.656	0.686
256	0.731	0.631	0.685

TABLE 5B: ADAGRAD, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.831	1.048	1.020
128	0.772	0.972	0.902
256	0.846	1.065	0.941

TABLE 6A: ADAGRAD, MAXPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.677	0.697	0.708
128	0.697	0.697	0.712
256	0.695	0.665	0.700

TABLE 6B: ADAGRAD, MAXPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.941	0.873	0.855
128	0.949	0.899	0.964
256	1.120	1.012	1.116

TABLE 7A: ADAMAX, AVERAGEPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.749	0.661	0.693
128	0.758	0.674	0.722
256	0.747	0.657	0.708

TABLE 7B: ADAMAX, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.749	0.965	0.898
128	0.771	0.974	0.907
256	0.812	1.035	0.979

TABLE 8A: ADAMAX, MAXPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.720	0.708	0.724
128	0.735	0.715	0.737
256	0.734	0.680	0.730

TABLE 8B: ADAMAX, MAXPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.832	0.862	0.847
128	0.902	0.856	0.877
256	0.892	0.940	0.937

TABLE 9A: RMSPROP, AVERAGEPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.734	0.652	0.690
128	0.766	0.653	0.703
256	0.737	0.608	0.713

TABLE 9B: RMSPROP, AVERAGEPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.861	1.070	0.971
128	0.808	1.011	0.975
256	0.911	1.131	0.923

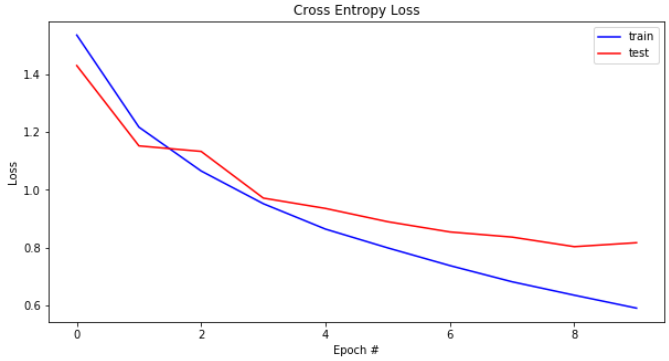
TABLE 10A: RMSPROP, MAXPOOLING ACCURACY

Filter Size	ReLU	Sigmoid	Softplus
64	0.710	0.680	0.730
128	0.749	0.642	0.735
256	0.739	0.497	0.752

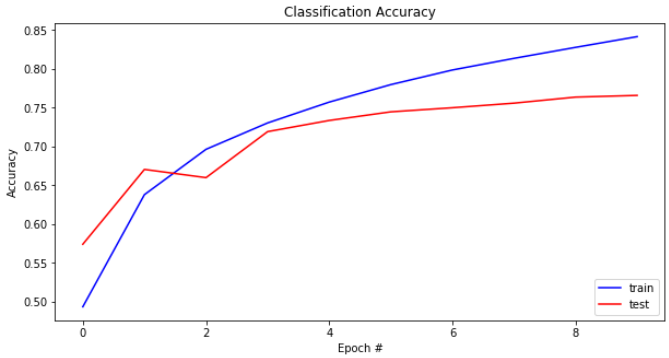
TABLE 10B: RMSPROP, MAXPOOLING LOSS

Filter Size	ReLU	Sigmoid	Softplus
64	0.895	0.976	1.036
128	0.809	1.036	0.905
256	0.978	1.417	0.888

A. *Cross Entropy Loss of filter size of 128, ReLU activation function, AveragePooling method, and RMSProp optimization*



B. *Classification Accuracy of filter size of 128, ReLU activation function, AveragePooling method, and RMSProp optimization*



VI. DISCUSSION

Image classification is a difficult task, and even across ninety different trials, the optimal testing accuracy that was arrived at was 0.766. There is clearly room for improvement, but some important observations were made. The pairing of ReLU with AveragePooling and Softplus with MaxPooling is certainly strong, as is evident from the results, and RMSProp and Adamax have also displayed promise as optimization methods. No conclusive evidence was available to make any claims regarding the filter size, but there is likely a way to optimize it, given more research. Despite the previous remark that the filter size of 128, paired with AveragePooling, ReLU, and RMSProp provided the strongest testing accuracy, this was not by a considerable margin, and there were other combinations that provided accuracies that were fairly close. Even across ninety trials, the perfect convolutional neural network remains elusive, but the results obtained from this paper are still desirable and strong.

REFERENCES

[1] Kathuria, A. (2019, December 02). Intro to optimization in deep learning: Gradient Descent. Retrieved June 11, 2020, from https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[2] Brownlee, J. (2019, November 13). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Retrieved June 11, 2020, from https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[3] Hinto, G., Srivastava, N., & Swersky, K. (n.d.). Lecture 6a Overview of mini-batch gradient descent. Retrieved June 10, 2020, from https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[4] Ruder, S. (2020, March 20). An overview of gradient descent optimization algorithms. Retrieved June 11, 2020, from https://ruder.io/optimizing-gradient-descent/index.html

ACKNOWLEDGEMENT

The authors of this paper would like to share a feeling of appreciation to Dr. Zhuowen Tu and his teaching team for introducing the concepts of neural networks, deep learning, and convolutional layers.