Image Classification on CIFAR10

Final Project

Phillip Stefan Niehaus
Tandon School of Engineering
New York University
New York, USA
stefan.niehaus@nyu.edu

Muhammad Osama Khan
Tandon School of Engineering
New York University
New York, USA
mok232@nyu.edu

***Abstract*—The CIFAR 10 dataset (with 60,000 32 x 32 RGB images) was used in this research project. The dataset was separated into 50,000 training images and 10,000 test images. Three machine learning algorithms, namely Support Vector Machine (SVM), fully connected Neural Network (NN) and Convolutional Neural Network (CNN) were trained on the data. Linear, radial basis function (RBF) and polynomial kernels were used when training the data with SVMs, with the penalty parameter, $\mathcal{C}$, of the error term varying over a range 0.0001 - 100. The linear kernel (with $\mathcal{C} = 0.1$) was found to produce the best results with a classification accuracy of 30.6%. Mean normalized and un-normalized data produced an accuracy of 30.6%, whereas scaling the pixel values between -1 and 1 produced a slightly poorer accuracy of 29.3%. On the other hand, training the data with a fully connected NN produced an accuracy of 53% on the 15th epoch with 4 layers, and 100 units in each hidden layer. A further increase in the number of hidden units or number of layers increased the number of parameters drastically, with very little increase in accuracy. A CNN architecture was then used to train the data, producing an accuracy of 71% on the 15th epoch. Finally, a hybrid architecture with a chained full Convolutional Network and fully connected NN was trained on the data. However, this hybrid model did not improve the accuracy in these experiments. This was probably a result of poor random initialization of the parameters, which resulted in convergence to poor local optimums.**

***Index Terms*—Neural Networks, Convolution, SVM**

I. Introduction

The CIFAR10 dataset was chosen for this research project, which consists of 60 thousand 32x32 color images in 10 different classes. The 10 classes are: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. These common examples of day-to-day objects provides a case of point to create and test machine learning algorithms for classification.

A particularly appealing feature of the CIFAR10 dataset is the small size of the images (i.e., 32x32 pixels). As a result, one can rapidly train various models over this large dataset, combining the benefits of a large amount of data and the ability to prototype various models and fully test those models within a short time-frame. Moreover, application of training classifiers on small images are finding growing research interest in medicine, as the sections of the images that are of interest will often occupy only a small space (i.e., tumor cells, inter alia).

The thee machine learning algorithms that were chosen are the Support Vector Machine (SVM), a fully connected Neural Network (NN), and a fully Convolutional Neural Network (CNN). The algorithms were chosen, in part, because of their wide-spread applications and established success across various domains. The models would be tested on the dataset and compared according to their accuracy. Since each class has a similar number of training and test examples, there would be no need to extensively consider other metrics (i.e., precision, recall, i.a.).

The dataset was separated into 50 thousand training images and 10 thousand test images.

II. Classification Experiments

For the first set of classification experiments, the SVM model was tested on various kernels with different regularization constants. The CIFAR-10 training dataset was imported from the keras library and had a shape of (49000, 32, 32, 3).The 32 x 32 x 3 color images were then flattened into a vector, reshaping the training dataset to have the shape (49000, 3072). Hence, it is important to note that only 49 thousand images were used when working on the SVM. However, since the training times were unfeasible on the entire set, the problem was more centered on the model itself than the availability of data.

A. SVM Kernels and Regularization

Three types of kernels were used to train the data, namely the linear kernel, the radial basis function (RBF) kernel and the polynomial kernel. During the initial prototyping phase, a relatively small training set of 3000 images was used to train the data, and the penalty parameter, $\mathcal{C}$ determined using a separate validation set. In this way, one could relatively quickly determine the highest performing hyperparameter and respective kernel. The penalty parameter, $\mathcal{C}$, of the error term was varied over the range 0.0001 - 100. The results are summarized in Table I.

TABLE I
TRAINING RESULTS FOR SVM MODELS

Kernel	$\mathcal{C}$	Training Accuracy (%)	Test Accuracy (%)
Linear	0.0001	35.5	9.8
	0.001	48.5	18.8
	0.01	70.9	25.3
	0.1	98.9	28.6
	1	1	27.9
	10	1	27.9
	100	1	27.9
RBF	0.0001	10.7	7.9
	0.001	10.7	7.9
	0.01	10.7	7.9
	0.1	30.7	11.9
	1	48.7	11.9
	10	84.8	10.5
	100	1	11.9
Polynomial	0.0001	10.7	8.7
	0.001	10.7	8.7
	0.01	10.7	8.7
	0.1	10.7	8.7
	1	13.9	8.7
	10	33.6	15.1
	100	80.4	26.7
	200	89.1	26.3
	500	95.3	26.3
	1000	97.8	25.9

In Table I, the best test accuracy of 28.6% was obtained for $\mathcal{C} = 0.1$. Following closely behind, the Polynomial Kernel demonstrated peak performance at an accuracy of 26.7%, with a penalty parameter $\mathcal{C} = 100$.

The plot of accuracy against $\mathcal{C}$ for the linear kernel is shown in Figure 1. The performance of the SVM on both the test set and training set improve substantially until $\mathcal{C} \approx 0.1$, after which the performance diminishes.

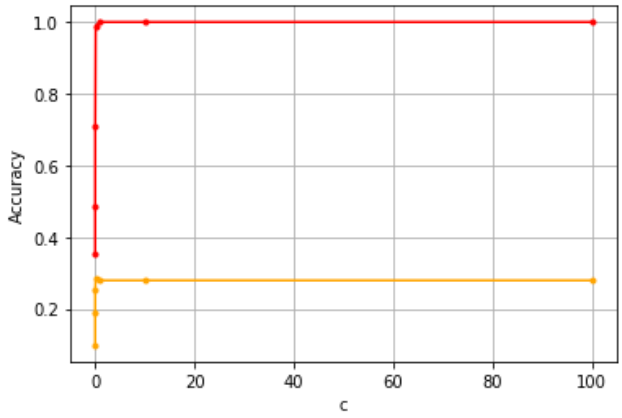


Fig. 1. Accuracy against $\mathcal{C}$ for SVM Linear Kernel

The RBF kernel was tested next, with the penalty parameter, $\mathcal{C}$, ranging from 0.0001 to 100. In this case, the best test accuracy of 11.9% was obtained for $\mathcal{C} = 0.1$. Note that after $\mathcal{C} = 0.1$, although the training accuracy keeps increasing, the test accuracy starts to decrease, illustrating the effects of over-fitting. The results are plotted in Figure 2.

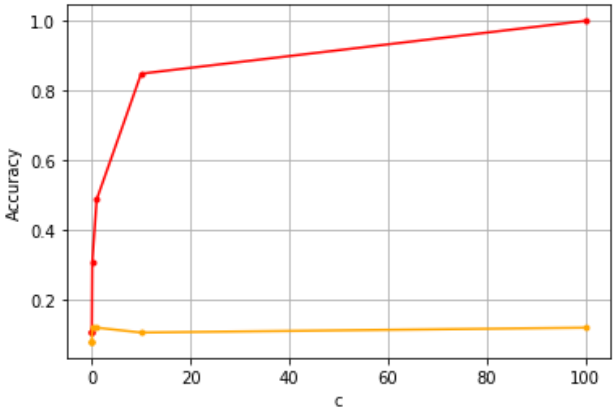


Fig. 2. Accuracy against $\mathcal{C}$ for SVM RBF Kernel

Thereafter, the Polynomial kernel was tested, with the penalty parameter, $\mathcal{C}$, ranging from 0.0001 to 1000. In this case, the best test accuracy of 26.7% was obtained for $\mathcal{C} = 100$. The results are plotted in Figure 3.

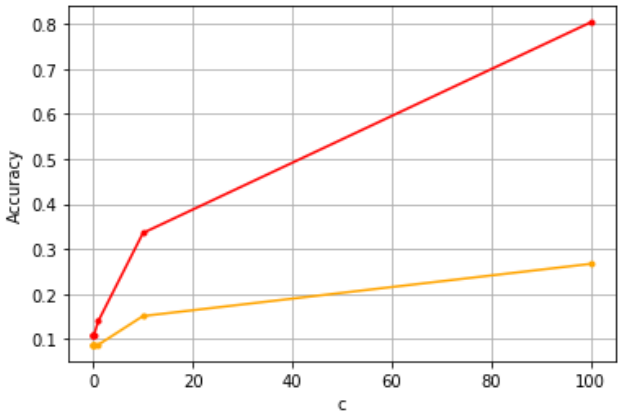


Fig. 3. Accuracy against $\mathcal{C}$ for SVM Polynomial Kernel

B. Selected SVM Model

The best results were obtained with a linear kernel SVM for $\mathcal{C} = 0.1$. Therefore, 10 thousand training images were trained with the linear kernel ($\mathcal{C} = 0.1$). This resulted in a test accuracy of 29.3%.

A small set of misclassified images were taken from training and shown in Figure 4 with the respective Truth labels and predictions.

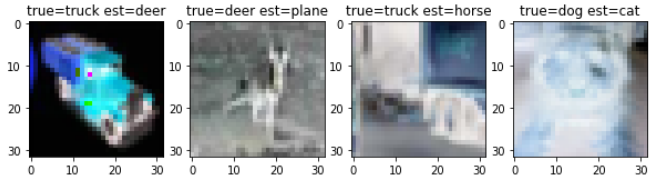


Fig. 4. Misclassified Images with Truth Labels

a) Effect of Normalization on Test Accuracy: In order to improve the classification accuracy, three different normalization schemes were applied on the training and test images. More specifically, the optimal model, the linear kernel ($\mathcal{C} = 0.1$), was trained on 10 thousand training images under three different normalization schemes: (i) un-normalized data (pixel values from 0-255), (ii) normalized data #1 by subtracting the mean of all the input data from each image in the training set and (iii) normalized data #2 by scaling the pixel values to be between -1 and 1. The results of the three different normalization schemes are summarized in Table II.

TABLE II
SVM CLASSIFICATION ACCURACY FOR NORMALIZATION SCHEMES

Normalization	Test Accuracy (%)
Un-normalized data	30.6
Normalized data #1	30.6
Normalized data #2	29.3

The next test series focused on more modern image classification techniques, including fully connected networks and convolutional networks.

C. Loss Criterion and Optimizer

The authors will use cross entropy loss to extend the negative log-likelihood function to the multi-class setting.

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{10} \mathbb{1}\{y = j\} \log \hat{y}_j \tag{1}$$

The optimizer that will be used in this series will be the Stochastic Gradient Descent optimizer provided by the Pytorch `optim` package since small batches of 4 pictures will be tested at a time. Moreover, momentum was used at a default value of 0.9, which is a standard base-line value for the SGD optimizer. The momentum value of 0.9 will, essentially, provide a moving average value that will be used in updating the network parameters in order to reduce 'noise' in the update that may take the network, on average, away from the optimal.

D. Learning Rate

For the neural networks and convolutional neural networks, the learning rate was decreased when the loss function, during training, would show that performance was no longer improving. Therefore, the learning rate, which will be denoted as lr, was incrementally adjusted in order to optimize the performance on the training set.

Indeed, this adjustment was shown to significantly decrease the final loss that could be achieved during training.

- $lr = 0.001$ for $0 \leq epoch \leq 5$
- $lr = 0.0003$ for $5 \leq epoch \leq 10$
- $lr = 0.0001$ for $10 \leq epoch \leq 15$

As shown in Figure 5 and Figure 6, at both 5 and 10 epochs, there is a significant decrease in the overall loss. However, after 15 epochs, the model was not showing any improvement.

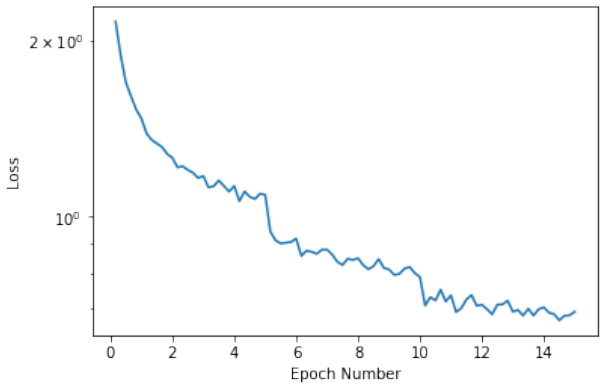


Fig. 5. Loss for Convolutional Neural Network

The reasoning for adjusting the learning rate during training is due to the non-linear architecture of the NN and CNN models, as well as the high-dimensionality of the inputs (i.e., color images). In other words, the backpropogation algorithm will not converge to a local optimal - not to mention the global optimum - if the learning rate is too large. If one were to imagine the possible loss positions as a three dimensional plane, it is easy to 'step-over' potential weights that would allow the loss function to converge. Therefore, the loss curves were observed the learning rate dynamically adjusted according to the aforementioned schedule.

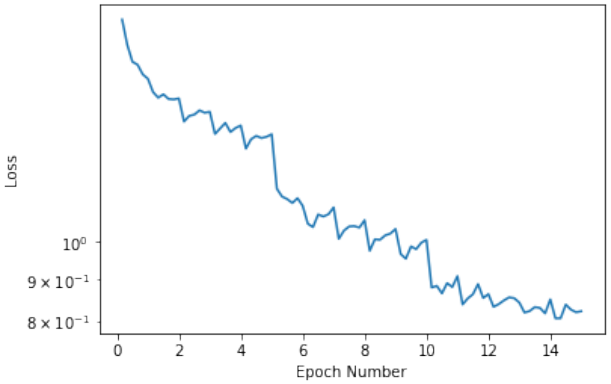


Fig. 6. Loss for Fully Connected Neural Network

During training, the loss function was observed and the learning rate was decreased once the loss was no longer showing improvement over several iterations. In order to achieve this, the network weights were saved and reloaded with a new defined learning rate.

Interestingly, a cyclic learning rate adjustment [1], as proposed by Leslie N. Smith, could yield a higher performing NN and CNN. The cyclic learning rate adapts the previous approach by both decreasing the learning rate more systematically as well as periodically resetting the learning rate back to some maximum value, and then decreasing it once more. The pattern is as shown in Figure 7.

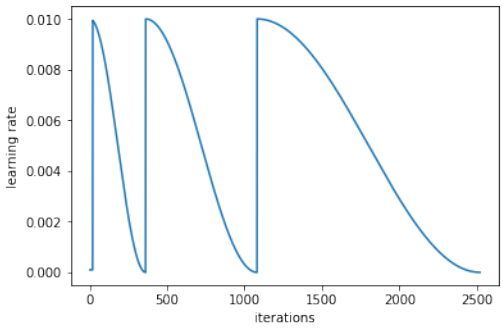


Fig. 7. Loss for Fully Connected Neural Network

In this way, if one were to consider an example 3D loss 'plane' of the non-linear architecture of the NN and CNN models, as shown in Figure 8, one could avoid - to some degree - falling into local minimum values that would not generalize well, or that could be easily improved.

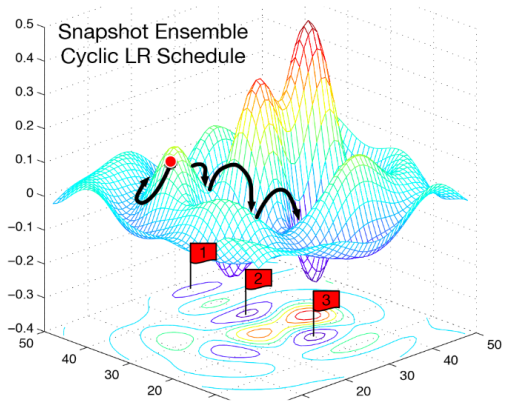


Fig. 8. Cyclic Learning Rate Schedule

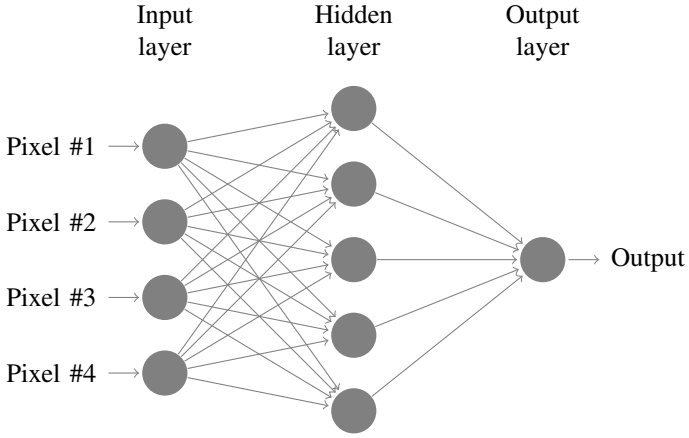
E. Model Complexity

Prior to implementing the cyclic learning rate schedule, the authors followed the more simplistic manual adjustment of the learning rate as shown in Figures 5 and 6. The results for three different fully connected NN architectures are shown in Table III. The input layer takes 3 * 32 * 32 = 3072 inputs, since we have to consider the RGB layers collectively, the hidden layers all had the same number of units, while the output had 10 units, one for each class.

TABLE III
RESULTS FOR FULLY CONNECTED NEURAL NETWORK

Model	Layers	Hidden Units	Epochs	Accuracy (%)
1 DNN	4	50	5	44
			10	47
			15	50
2 DNN	4	100	5	48
			10	52
			15	53
3 DNN	6	100	5	49
			10	53
			15	54

In order to visualize the neural network, a sample case with 4 inputs, one hidden layer with 5 units, and 1 output is shown below.



Note that when the number of hidden units was increased from 50 to 100 (i.e., Models 1 and 2, respectively), the accuracy on the 15th epoch increased by 3%. However, increasing the number of hidden units further showed only a marginal improvement, while the number of weights that needed to be learned increased drastically - Model 1 had 153600 parameters, while model 2 had 307200 parameters, and so on. Although this did not result in a significant increase in the training time, the diminishing returns and major increase in memory requirements for training are problematic. Moreover, the cyclic learning rate schedule seemed to have nominal effects on the final accuracy, improving results by only 1% on average.

Therefore, in order to increase the accuracy of the classification, a Fully Convolutional Neural Network was used in order to extract relationships for a *collection* of pixels [2]. Since the pictures were only 32x32 pixels, the chosen kernel was 3x3 and the activation function used in this study was the ReLU, which has been widely adopted in other studies. The exact architecture used in this case is shown in Figure 9.

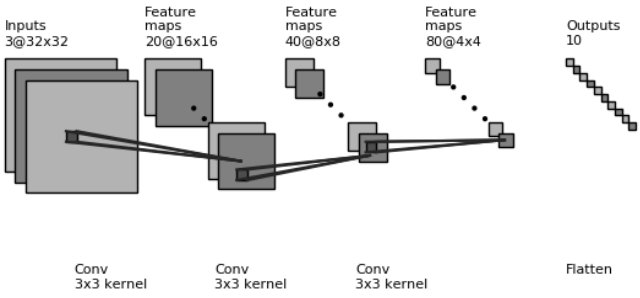


Fig. 9. CNN Architecture

In our experiment, a total of 20 filters were used for the mapping of the first 3 RGB layers to 20 hidden convolutional layers each 16x16. In this way, hidden fully convolutional layers were formed, each reducing the number of pixels to a quarter of the respective input. In this way, it was surmised that the CNN could learn complex patterns and create

combinations of these patterns in order to more accurately classify the images. In the last layer, a 4x4 max-pool was used in order to flatten out the 80 feature maps in the last convolutional layer (i.e., each having a 4x4 matrix of values), to 80 individual values. Thereafter, these 80 features were used to create 10 outputs with a simple softmax regression.

The average results are shown in Table IV

TABLE IV
RESULTS FOR FULL CONVOLUTIONAL NEURAL NETWORK

Model	Epochs	Accuracy (%)
4 CNN	5	61
	10	66
	15	68

Note that in this case, the total number of learned parameters was only 37490, which is 4 times less than the number of weights used for DNN Model 1, which achieved only 50% accuracy, and 8 times less than DNN Model 2, which achieved 53% accuracy.

A potential improvement to this model would be to increase the number of layers and include max-pool layers. This will create 'summaries' of patterns, which would then be combined. Moreover, a fully connected neural network could be used to more accurately determine the 10 final outputs from the 80 input features from the last convolutional layer.

Increasing the number of convolutional layers, such that the final layer is only 2x2 yields the results shown in Table V.

TABLE V
RESULTS FOR FULL CONVOLUTIONAL NEURAL NETWORK

Model	Epochs	Accuracy (%)
5 CNN	5	67
	10	71
	15	71

Instead of the softmax regression, the authors also used a NN that takes the input of the last convolutional layer. The experiment was run with Model 4 (i.e., the CNN with 3 hidden convolutional layers). Unfortunately, this hybrid model did not improve the accuracy for the test runs. This may be in part because of poor random initialization of the parameters, which resulted in convergence to poor local optimums. A summary of the results are shown in Table VI. The design of the hybrid architecture is shown in Figure 10.

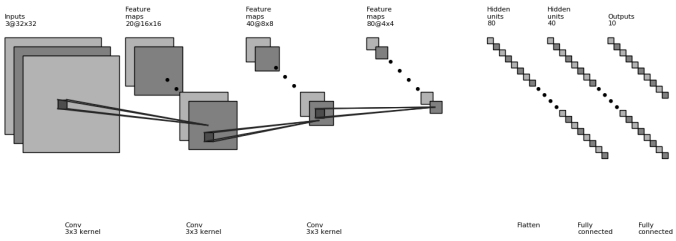


Fig. 10. CNN Architecture

III. FINAL RESULTS

For brevity, the results for the SVM models will be excluded in the summary table since the final accuracies were substantially lower than those of the NN and CNN models.

The accuracy results for all DNN and CNN models are shown in Table VI.

TABLE VI
SUMMARY OF MODEL ACCURACIES PER CLASS

	Accuracy (%)			
	DNN - 3	CNN - 4	CNN - 5	Hybrid - 6
Plane	63.2	74.4	82.8	69.7
Car	63.1	82.0	85.6	79.4
Bird	45.5	50.5	64.9	58.9
Cat	38.8	54.0	48.9	55.5
Deer	45.5	62.1	66.7	55.8
Dog	41.4	63.9	64.5	56.8
Frog	60.9	74.4	73.4	70.1
Horse	60.3	76.7	75.1	69
Ship	68.8	68.2	77	77.7
Truck	57.4	69.0	78.9	75.4
Overall	**54**	**68**	**71**	**66**

IV. CONCLUSION

Out of the three machine learning algorithms used to train the CIFAR 10 dataset, the best results produced for each of the three algorithms were as follows: (i) SVM - 30.6%, (ii) fully connected NN - 53% and (iii) and fully CNN - 71%. Several hyperparameters were varied when training the dataset with each of the three learning algorithms. For the SVMs, a linear kernel with $\mathcal{C} = 0.1$ was observed to produce the best results (accuracy = 30.6%). On the other hand, a fully connected NN with number of layers = 4 and number of hidden units = 100 was observed to produce an accuracy of 53%, with any further increase in number of hidden units or number of layers resulting in a drastic increase in the number of parameters, with very little increase in accuracy. Finally, the CNN architecture with 4 fully convolutional layers was observed to produce the best results, with an accuracy of 71%.

REFERENCES

[1] L. N. Smith, "Cyclical learning rates for training neural networks," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 464–472, IEEE, 2017.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.