Project 3 - Group 52

Mamoun Benchekroum, Yasmeen Hitti, Ines Moreno, Julien Mounthanyvong

April 2020

1 Abstract

In this mini-project we explored image classification methods and compared the performances of two different machine learning models: Multilayer perceptron and Convolutional Neural Network. We evaluated these models using the CIFAR-10 data set and compared their accuracies in order to compare their performances. Additionally, we compared the effect of tuning different parameters on each model in order to determine the ones that lead to the best performance. Overall, we found that the CNN achieved a higher accuracy than the MP.

2 Introduction

In this project, we were given the CIFAR-10 image dataset and we tested the performance of two different machine learning models through various experiments. The models tested were Multilayer Perceptron (MP) and Convolutional Neural Network (CNN).After implementing our models, we created a set of functions to tune some parameters on the models and pick the ones optimizing its performance. For the CNN we tested both network structure hyperparameters (number of filters, activation function) and network training hyperparameters (learning rate, number of epochs, optimizer). We found that generally the CNN classified the data better than the MP.

3 Related Work

The concept of CNNs, as their name implies is based on biological phenomena. Early works in the 1950s showed that visual cortexes in brains of monkeys and cats responded to specific patterns [5]. Work by Fukushima [3] introduced the concept of neocognitron and convolutional, downsampling layers used in CNNs. Waibel then worked on the time delay neural network using shift invariance being effectively the first convolutional network [7]. Image recognition systems then started to appear [2] but LeCun was the first to use back-propagation for learning, thus making it fully automatic with much better performance for different image types [6]. More recent work on the CIFAR-10 database by Aszemi et al showed which hyper parameter values influenced the accuracy the most [1].

4 Data set and setup

The CIFAR-10 dataset counts with 60,000 32×32 pixel colour images that are organized in 10 different classes with 6000 images per class.The classes in the dataset are: plane, car, bird, cat, deer, dog, frog, horse, ship and truck, which are completely mutually exclusive. The dataset is divided into a train set and a test set. The trainset counts with five batches of 10000 images and the testset contains one batch that contains 100 randomly-selected images from each class. The input to our models is as such: $3 \times 32 \times 32 = 3072$, the 3 represents the 3-channel color of red blue green (RGB). As the data is loaded using torchvision, the representation of each image is a tensor that is normalized in range [-1, 1].

5 Methodology

After loading the data, we implemented the two following models and ran a series of experiments on both, tuning their hyperparameters to obtain an optimal performance.

As a preprocessing step, we decided to create some new images in order to increase the size of the training set. The operations we used to that end are as follows : get the symmetric of the image ; get the transpose of the image ; get a copy of the image with added noise.

- **Multilayer Perceptron(MP):** The multilayer perceptron designed for this classification task has 3 hidden layers with 1024 hidden nodes for the first layer, 32 for the second and 32 for the third. Different activation functions were explored for the activation of the hidden layers. The activation functions included: sigmoid, ReLu, Leaky ReLu and tanh. The network includes the feedforward pass. Training is done by using mini-batch stochastic gradient descent (SGD) to optimize each parameter, with backpropagation to compute the actual gradient, following the loss is created, in our case our criterion was cross-entropy. The weight are initialised randomly in hope of getting more significant gradients in the learning.

- **Convolutional Neural Network(CNN):** The CNN used for this classification task has two convolutional layers, two max pooling layers as well as 3 fully-connected layers. Different structures for the convolutional layers were explored by varying the number of filters in each and the activation functions. Our criterion was cross entropy and we tested different optimizers such as Adam and mini-batch gradient descent.

6 Results

6.1 Model Comparison

On average, CNN classified the images with a higher accuracy than the MP (0.55 vs 0.25 on average). This makes perfect sense since CNN is the algorithm used for analyzing image data par excellence. On the other hand, for both models, some classes were better classified than others. For instance, the CNN had a harder time distinguishing dogs and cats and trucks and cars. This is due to the similarity of these things in real life. (Figure 1).

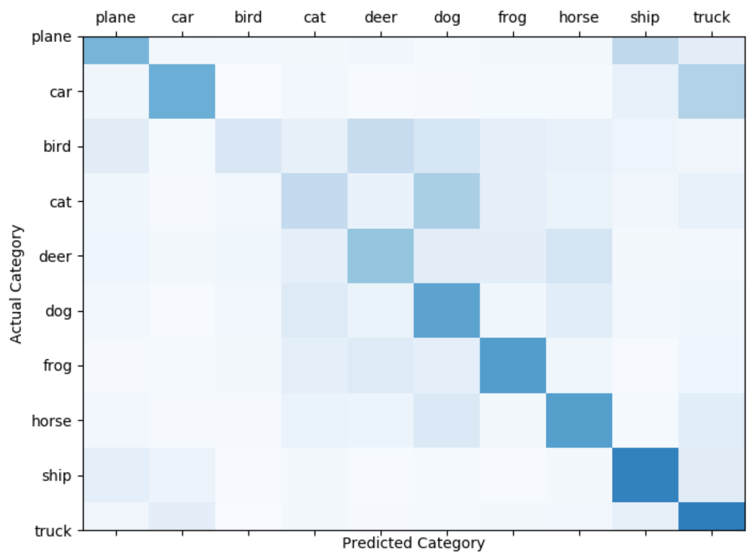


Figure 1: CNN correlation matrix

6.2 Experiments on the Multilayer Perceptron

The very long running time of our code made it hard to properly run the different tests that we had planned. More precisely, we tested different hyperparameters for our model : different activation function such as sigmoid, ReLU and tanh ; different learning rates ranging from 0.1 to 0.001 ; different ending criterions (ϵ used from 0.01 to 0.2 as well as the maximum number of iteration) ; different size for the network.

We could observe differences in the gradients obtained depending on the activation functions used. For instance sigmoid created an oscillating gradient, equally as number of layers increases this activation function is no longer of interest as it may cause vanishing gradients. The vanishing gradient problem is something that also could occur with tanh as the number of layers increase. As for the Relu activation function, we obtained our best results and was computationally more efficient.

As of now, the best results we got where when we used ReLU activation in a network with 3 hidden layers of 1024, 32 and 32 neurons respectively, a learning rate of 0.005 and $\epsilon = 0.1$. This gave us an accuracy of 28%. It is important to note if we would have increased the amount of epochs, our loss would have decreased. We believe as the learning rate was low the weight updates became in some ways unstable and this led to our model not being able to learn from the picture images.

To avoid the exploding of the gradient, different initializations could have been explored such as the method proposed by He et al., which is specific to ReLU activation [4].

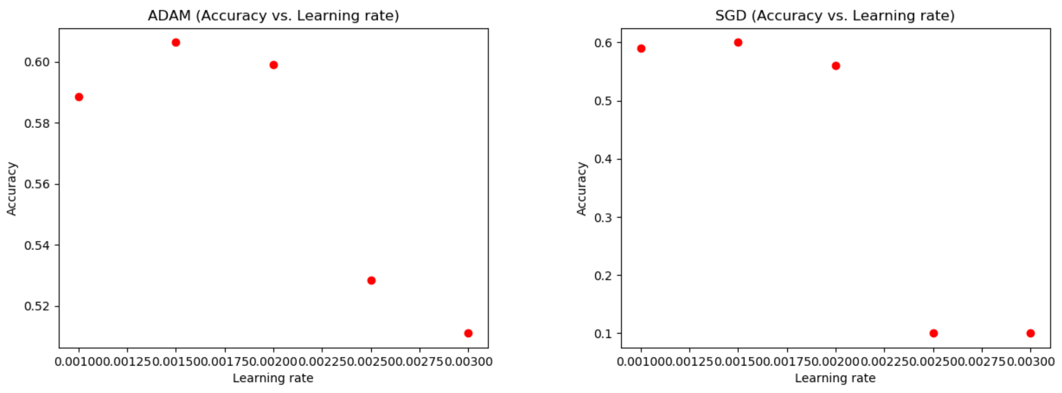
Epochs	2	3	4
Accuracy	0.27	0.28	0.28

6.3 Experiments on the Convolutional Neural Network

We first started by by increasing the number of epochs given in the tutorial and found that it improved the performance of our model. However, it also yielded to a considerably longer running time, for not such a big performance enhancement, hence we decided to run the following experiments with 5 epochs only.

Epochs	2	3	4	5	6	7	8	9	10
Accuracy	0.54	0.58	0.58	0.58	0.60	0.60	0.61	0.62	0.62

We wanted to follow the experiments described Aszemi et al. [1] to optimize the hyperparameters of the CNN but replicating their study was too computationally expensive so we decided to test just a subset of them. We started by trying different optimizers: Stochastic Gradient Descent and Adam and found their corresponding optimal learning rates (Figure 3). We found that, the Adam optimizer yielded to a higher accuracy and that its optimal learning rate was 0.0015.



(a) (b)

Figure 2: Accuracy vs. Optimizer

We then performed experiments on network structure hyperparameters such as the number of filters used in each convolutional layer and their activation functions (up to this point we had kept the structure given in the tutorial). In terms of activation function, we found that ReLu yielded to a slightly better performance (0.58 vs. 0.54 on average). Finally, we found that the accuracy was better when we augmented the size of the filters in each convolutional layer but this was true to a certain extent since then our model started overfitting after a certain point.

Layer 1	Layer 2	Accuracy
8	16	0.51
16	32	0.57
32	64	0.55

7 Conclusion

Throughout this project we investigated and compared the performance of two different models for an image multi-class classification task. We compared the accuracy of our models and explored which were the optimal parameters to use in each case to obtain the best performance. For its best performing hyperparameter CNN was the best performing model. For future work, we could hope to run tests more computationally expensive to find a better optimisation of each model. For example, using grid search to find the best combination of hyperparameters. Furthermore, we could still investigate additional parameters such as the number of convolutional or hidden layers in the CNN and the MP respectively.

8 Statement of contribution

Mamoun Benchekroum & Ines Moreno: CNN + write-up
Yasmeen Hitti & Julien Mounthanyvong: Multilayer perceptron + write-up

References

[1] Nurshazlyn Mohd Aszemi and P. Dhanapal Durai Dominic. "Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms". In: *International Journal of Advanced Computer Science and Applications* 10 (2019).

[2] John S. Denker et al. "Neural Network Recognizer for Hand-Written Zip Code Digits". In: *Advances in Neural Information Processing Systems 1*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1989, pp. 323–331. URL: `http://papers.nips.cc/paper/107-neural-network-recognizer-for-hand-written-zip-code-digits.pdf`.

[3] K. Fukushima. "Neocognitron". In: *Scholarpedia* 2.1 (2007). revision #91558, p. 1717. DOI: `10.4249/scholarpedia.1717`.

[4] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[5] D. H. Hubel and T. N. Wiesel. "Receptive fields of single neurones in the cats striate cortex". In: *The Journal of Physiology* 148.3 (Jan. 1959), 574591. DOI: `10.1113/jphysiol.1959.sp006308`.

[6] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Comput.* 1.4 (Dec. 1989), 541551. ISSN: 0899-7667. DOI: `10.1162/neco.1989.1.4.541`. URL: `https://doi.org/10.1162/neco.1989.1.4.541`.

[7] A. Waibel et al. "Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339.