Image Data Classification with Multilayer Perceptrons and Convolutional Neural Networks

Brandon Park

Sean Li

Yanis Mouazar

Editor: Brandon Park, Sean Li, Yanis Mouazar

Abstract

The main objective from acquiring the CIFAR-10 dataset was to implement our concepts of multilayer perceptron, to use it to perform and classify images. The challenge lies in honing our collective understanding of basic neural networks and their algorithms under one common foundation, and to find the most effective way in evaluating images and classifying them into the right class. After acquiring our relevant data, we configured our multilayer perceptron to have a wide range of parameters, to truly evaluate the performance of our algorithm when configured to different implementations. Given the ability to express our method of experiments, we found that using backpropagation and mini-batch gradient descent was crucial to the findings of our results and properties of our model. We discovered that the accuracy of these models would be more precise with an increase of network depth, while the activation that worked the best were ReLu and Tanh. Fine-tuning hyperparameters such as learning rate and batch size were crucial to effectively help gain exposure to how much the network could be adjusted in order to give the best accuracy.

1. Introduction

By implementing a multilayer perceptron from scratch, we developed a holistic approach to the finding and tuning of parameters to find the most optimal way to classify image data. The dataset consisted of 60000 images in total, split into 50000 for testing, 10000 for training. From acquiring the datasets, our collective focus was to classify image data through multilayer perceptrons, but our development of such algorithms extended far beyond the basics alone. We acknowledge that the scope of our investigation to follow was not enough to cover the extent of all possible adjustments (Vassiliskrikonis, 2018), such as, the increase/decrease of the number of units in each layer or the convolutional's neural networks and its tuned hyperparameters.

We felt it necessary to justify which learning rates and batch size would have the best accuracy on the different model implementations of our neural networks (Figure 1). We discovered that by controlling the batch sizes and learning rates, 64 and 0.1 respectively, were our model's optimal values.

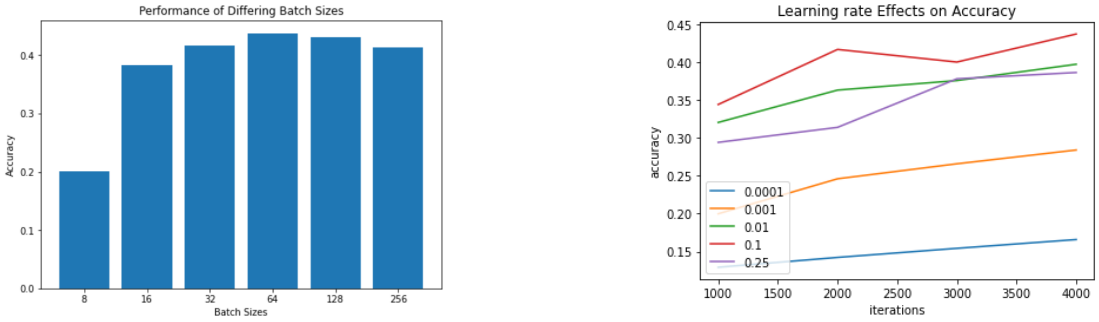


Figure 1: Accuracy from Differing Batch Sizes Learning Rates

By conducting experiments where our independent variables were batch sizes and learning rates, we strived to obtain the most optimal hyperparameters. However, we agreed that experiments that were crucial to our hypothesis and testing included but not limited to the number of hidden layers (Uzair and Kamil, 2021), different types of activations, and comparisons with our model implemented from scratch to a convolutional neural network. In the first case, we estimated that the number of hidden layers would increase the accuracy of the model as a larger time complexity would allow the network to better understand the image classification, albeit without any overfitting/underfitting situations (Chang, 2022).

CNNs are particularly suited to address these challenges, as they are able to learn hierarchical representations of image features, starting from low-level features such as edges and textures, and gradually building up to high-level features such as shapes and objects. We will explore different CNN architectures and techniques for training and optimising them on the CIFAR-10 dataset. We will compare their performance and analyse the impact of different hyperparameters and regularisation techniques on the accuracy and generalisation of the models. Our goal is to achieve results on this challenging dataset, and to gain insights into the strengths and limitations of CNNs for image classification tasks.

2. Dataset

The dataset consists of a total of 60000 images as mentioned previously. There are 10 different classes that these images can fall under, with 6000 images per class. When extracting the data from its raw format, it is divided into another 6 different batches. 5 of them being training batches and the lone other being the testing batch. Each batch is divided evenly again with 10000 images in each, with no uniform distribution of image classes in each batch. However, the training batches combined have exactly 5000 images from each class. In order to properly process this into our implementation, we first needed to vectorize, normalise and one-hot encode our data. Expressed in a matrix, each image can be represented by 3072 individual neurons (pixels). 32 x 32 x 3, 32 being the dimension of each image and 3 being the RGB values of each pixel. For each batch, a 10000 by 3072 matrix can be used to represent each image.

3. Results

Our first experiment we conducted evaluated the performance of our multilayer perceptron model based on the number of hidden layers. To control this independent variable,

we benefited from maintaining all other parameters unchanged, using ReLu activations with a softmax layer at the end to achieve our image classification. To prevent underfitting/overfitting from these hidden layers, the 256 neurons were used in each hidden layer. The time complexity of our model required a larger time span with the increase in the number of layers, while it achieved high accuracy when this time span increased (Figure 2).

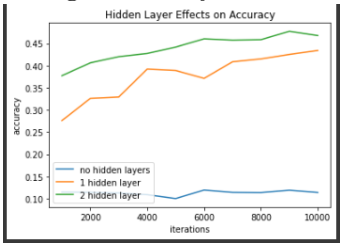


Figure 2: Number of Hidden Layers and Affect on Accuracy

A reasonable explanation would be that the larger time complexity helps the model understand as the network gets more complicated. To truly test the positive correlation and its accuracy, we added another hidden layer to 3 which gave us an accuracy of around 0.45. Since this was not a significant increase in performance at the expense of time, having 2 hidden layers would be the most efficient.

We followed up this experiment by controlling which activation functions would optimise our accuracy between ReLU, tanh, and Leaky-Relu (Figure 3).

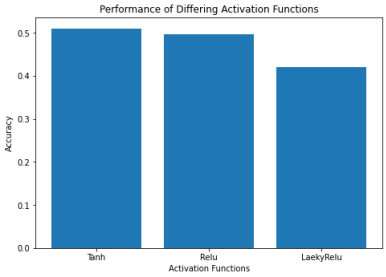


Figure 3: Accuracy on Activation Functions

Our hypothesis that ReLU would perform the best was not proven correct. Tanh outperformed the other activation functions, while beating ReLu with a slight margin. The time complexity from using the activation was also less due to its nature and its simplicity of collecting the maximum between 0 and value, hence converging faster. While we noticed the performance of Leaky-Relu was not on par with it, it allowed us to work with our normalised vectors configured between -1 and 1. It can be noted that this is why ReLu is the most used activation function and competing in our experiments to be on level as the best, as seen forthcoming in our convolutional neural network.

Implementing regularizations, whether that would be L1 or L2, meant we would be able to prevent overfitting problems that could occur if the network became too complex for the training data. While our network did encounter problems with overfitting since training accuracy was higher than the testing accuracy of around 10 percent, we deemed it to be healthy practice to implement regularisation. This helped decrease the overfitting and brought down the difference between the training and testing accuracy to less than

3 percent. The RIDGE regression performed better than the LASSO regression with an accuracy score of 0.48 to 0.4 respectively. So while it reduced overfitting, it also reduced accuracy.

Without normalisation, the accuracy of the model completely decayed. There is also some evidence that batch normalisation can contribute significantly to addressing the vanishing gradient problem common with deep learning models. The accuracy reduced to 0.1 without normalisation, as accurate as a random guess, which may be explained by overflow from exploding weights, a common issue with datasets that are not normalised (Chadha, 2021).

Finding inconsistencies between different forms of the same dataset raised an intriguing question: what form of the dataset would enhance the accuracy of our model? One that was missing from our evaluation was the standardised dataset. When compared to the normalised score of 0.3658, the standardised testing set gave us roughly the same value. We could conclude that images that were highly irregular to their classification will prefer data to be normalised or standardised.

The main reason for this is that CNNs are specifically designed to handle spatial information in images, by using convolutional layers that scan the input image with a set of learnable filters to extract local features, followed by pooling layers that downsample the feature maps while preserving the most salient features. This hierarchical feature extraction approach is well-suited for capturing complex image patterns and variations, and has been shown to be highly effective for a wide range of image classification tasks. In contrast, MLPs are designed to handle tabular data, and treat each input feature as independent and equally important. This approach is not well-suited for image classification, as it does not take into account the spatial relationships between the pixels in the image. In our experiment 5, it can be demonstrated that our model shows higher accuracy after only 25 epochs along with it not overfitting our data (Figure 4).

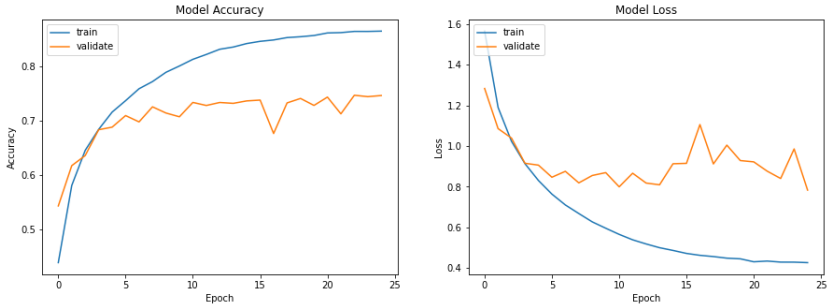
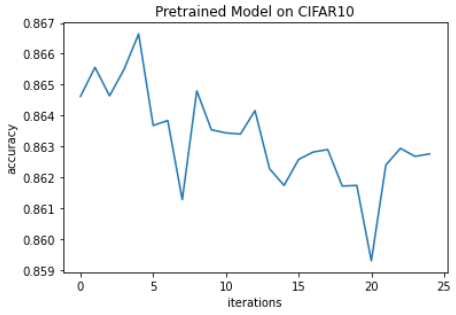


Figure 4: Model Accuracy and Loss as Epoch increases

A pre-trained model, especially one that has been trained on a large dataset, can often achieve higher accuracy than a MLP or regular CNN trained from scratch on a smaller dataset like CIFAR-10. This is because the pre-trained model has already learned a set of generalised image features that are transferable to the target dataset, allowing it to achieve better performance with less training data. In terms of accuracy, the pre-trained model is likely to outperform the MLP and regular CNN trained from scratch on CIFAR-10. However, the exact difference in accuracy would depend on the specific models and their

hyperparameters. In comparison to our Q5 model, we can see that the pre-trained model performs 10 percent better (Figure 5).



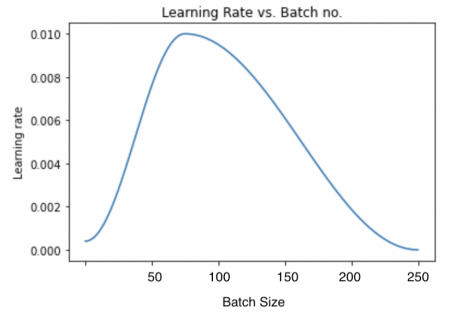


Figure 5: Pre-trained models on CIFAR-10 Figure 6: Learning Rate vs Batch Size

Lastly, it is worth noting that a batch size of 64 and a learning rate of 0.01 was chosen to run the experiments, this decision was made based on the data where we can see how the batch size and learning rate interact with each other (Figure 6).

4. Discussion and Conclusion

The analysis of our multilayer perceptron implementation not only gave us a solid foundation on which hyperparameters would best optimise our model, but further enriched our understanding when compared to different networks like CNN. The ability to find the best image classification model stemmed from having the freedom to configure our network to the best result on a step-by-step basis. Although classification using our model and its configuration could differ with various datasets, we found through crucial experiments that led us to believe some results could stay consistent. For instance, the number of hidden layers that increases the depth of the network would increase the time complexity but increase the accuracy as the network's ability to train itself will improve. It raised an important question whether we would sacrifice time complexity in order to achieve marginally better results. Furthermore, normalising our dataset to handle our activation functions was crucial in building the right model. To develop our neural network to classify images further beyond the 10 categories, accommodating numerous datasets while still performing at optimal accuracy will be a future task to handle. From advancements in understanding the techniques of Artificial Intelligence to tackling real-world issues like predicting the right amount of anaesthesia in a healthcare operation, the possibilities are truly open-ended.

5. Statement of Contributions

Each group member shared the components of the project equally. Each member was designated with a task to work on, and continued this approach when conducting the tests. The write-up was written with our shared expertise on our analysis of the multilayer perceptron implementation.

6. References

Chadha, A. What do normalization and standardization mean?. Becoming Human: Artificial Intelligence Magazine. 2021, July 19. https://becominghuman.ai/what-does-feature-scaling-mean-when-to-normalize-data-and-when-to-standardize-data-c3de654405ed

Chang, Z. Using normalization layers to improve deep learning models. Machine Learning Mastery. 2022, June 19. https://machinelearningmastery.com/using-normalization-layers-to-improve-deep-learning-models/

Uzair, M. Jamil, N. Effects of hidden layers on the efficiency of neural networks. IEEE Xplore. 2021, January 20. https://ieeexplore.ieee.org/document/9318195

Vassiliskrikonis. CIFAR-10 analysis with a neural network. Kaggle. 2018, November 18. https://www.kaggle.com/code/vassiliskrikonis/cifar-10-analysis-with-a-neural-network