Machine Learning Project 2

Aidan Licoppe, Kyle Vamvakas & Yanying Zhang

March 9th, 2023

Abstract

The goal of this report is to implement a multilayer perceptron (MLP) from scratch for image classification of the CIFAR-10 dataset. The MLP is trained on a varying number of hidden layers ranging from 0 to 2 and with different activation functions including ReLU, leaky ReLU, and hyperbolic tangent (tanh). Furthermore, the MLP model is also trained with unnormalized images, trained with L1 and L2 regularization individually, and trained with momentum. Also, hyperparameter tuning using sklearn is performed. In the second portion of the report, a convolutional neural network (CNN) was trained with 2 convolutional and 2 fully connected layers for the same classification task using existing libraries such as Pytorch and Tensorflow. Finally, a pre-trained CNN model was loaded and tested on CIFAR-10, and its results were compared to the previous models' results.

Our key findings begin with our hyperparameter tuning, which found a batch size of 16, a learning rate of 0.001, and L1 and L2 hyperparameters of 0.0001 and 0.01 to be optimal for the MLP model. We also found that the non-linearity activation and depth of neural networks improve the test accuracy of the MLP model from 0.4 to 0.53, and that ReLU and leaky ReLU activation outperformed tanh activation. Additionally, training the MLP model on unnormalized data made the model overfit much more easily as it was susceptible to biases, as well as generalize to unseen data poorly. Also, both L1 and L2 regularization helped with the MLP model not overfitting to the training data. Finally, momentum was added to the MLP, which helped with faster convergence speed, though similar accuracy. When training a CNN with the CIFAR-10 dataset, the test accuracy increased from 0.53 to 0.77. Alternatively, using ResNet-50 as a pretrained model and adding two fully connected layers to which the CIFAR-10 dataset was trained on resulted in very fast convergence, though a final test accuracy of 0.61.

Introduction

An MLP is a form of feed-forward neural network that works as a universal approximator, in which if the dataset given is a continuous mapping of a certain finite space to another, then the values can be mapped to a certain degree of accuracy given the architecture. In its simplest form, the network architecture typically consists of an input layer which has weight values connecting it to a hidden layer containing some form of activation function, and an output layer typical with a different function, such as SoftMax for classification. The universal approximation theorem states that an MLP can approximate any continuous function with a single layer, given enough bases. In practice, due to computation, space limits, and generalization error, this is not an efficient approach and neural network depth has been shown to have a greater effect than increasing width (Lu et al., 2017). This depth is created through the addition of hidden layers, typically containing the same activation functions, of which include the ReLU function, leaky ReLU, and tanh.

Convolutional neural networks, or CNNs, are another type of feed-forward neural network, which are typically used in the analysis of image data. Unlike MLPs, which treat the input simply as a vector, CNNs assign importance to various aspect of the image, considering pixels in relation to each other through the use of a convolution with a kernel filter. In the convolutional layers, this kernel matrix is moved over the input, performing elementwise multiplication on it, resulting in an output matrix typically of smaller size. These convolutional layers will typically also be connected with fully connected, or MLP layers in practice before computing an output as a means to learn non-linear combinations of the high-level features.

The task of this project is to implement an MLP from scratch, implement a CNN from built-in libraries such as PyTorch or Tensorflow, and use a pre-trained CNN model. All three of these broad tasks are for image classification of the CIFAR-10 dataset, and to see how differing hyperparameters and models affect prediction accuracy. The dataset used in this report is CIFAR-10, which is explained in more detail in the following section. Essentially, it is a dataset of images which span 10 classes. Sophisticated neural networks trained on CIFAR-10 have been able to classify unseen images from the dataset with up to 89.91% accuracy, so it will be interesting to see what our simple model can acquire (Thakkar et al., 2018).

Datasets

The CIFAR-10 is an image classification dataset consisting of 60 000 32x32 coloured images. There are 10 classes of images total, consisting of 6 000 images per class. The image classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. CIFAR-10 is split into five training sets of 10 000 images each, and one testing set of 10 000 images. The training sets total 50 000 images with 5 000 images from each class and the testing set consists of 1 000 images from each class for.

To analyze the dataset, we must understand two components: the class distribution and image quality. We know that each class consists of 6 000 images, so CIFAR-10 is a well-balanced dataset. This implies that each class has the same probability of being selected when fitting and testing the MLP model, so the model should not be biased towards any class. As far as image quality goes, CIFAR-10 is not considered to have high quality images with complicated patterns since they are 32x32 images. Therefore, we would expect an MLP model with fewer hidden layers to sufficiently capture the patterns in CIFAR-10 as they are not complicated. If we did have higher quality images portraying complicated patterns, then a deeper MLP model would suffice in capturing the intricacies of the dataset.

Results

It is important to note that our colab document, though it does contain all the code, does not contain all plots and results documented in the report as it took too long to run and coordinate from three different computers. As such, tests were done separately from the colab document.

Hyperparameter tunning

Before discussing our results, it is important to note that we performed hyperparameter tuning for the learning rate, batch size, and L1 and L2 regularization hyperparameters using sklearn. We found that for the MLP model without momentum, the optimal batch size value is 16; the optimal learning rate is 0.001; the optimal L1 regularization hyperparameter is 0.0001; and the optimal L2 regularization hyperparameter is 0.01. These were the hyperparameters used to train the MLP model.

Network depth and non-linearity activation

For complicated data with non-linear relationships, non-linearity activation and network depth are two crucial parameters. Non-linearity enables the MLP model to capture non-linear relationships within the data by recognizing complex patterns and classifying these patterns to their respective image class. By not including non-linearity in an MLP model, the model becomes a linear model like logistic regression. Hence, the patterns would not be learnt and classified by the model. As for the importance of network depth, this allows the MLP model to capture increasingly complicated patterns as the network trains deeper. What is happening here is that each subsequent layer learns an additional level of abstraction of the original input data since it is fed forward through all previous layers.

Figures 1 – 3 summarize the findings from the three different MLP models trained with ReLU activation by varying the number of hidden layers. As expected, the number of hidden layers did affect both the training and test accuracy of the model with test accuracy increasing from 0.4 to 0.5 to 0.53 as the depth of the network increased from 0 to 2. However, the discrepancy in model accuracy was not as large as anticipated. The reason for this could be that 1 and 2 hidden layers are not deep enough to capture the complexity in the CIFAR-10 dataset. Hence, it would be beneficial in the future to tune the number of hidden layers as a hyperparameter.

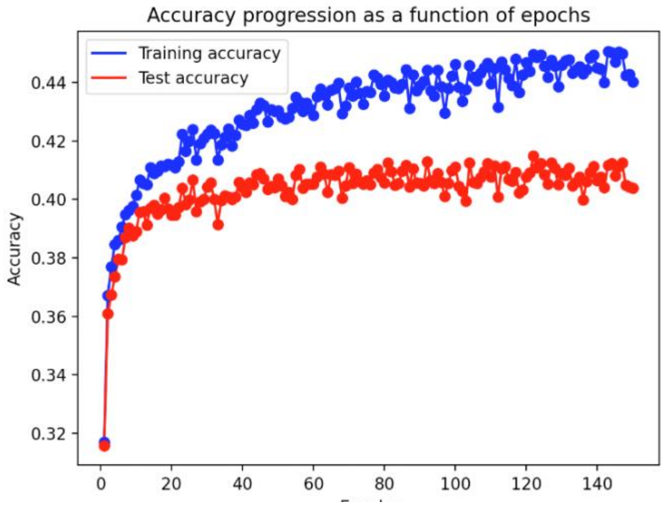

Figure 1 – MLP model accuracy as a function of epochs for 0 hidden layers using ReLU activation. Notice that the model overfits around 30 epochs at 0.4 test accuracy.

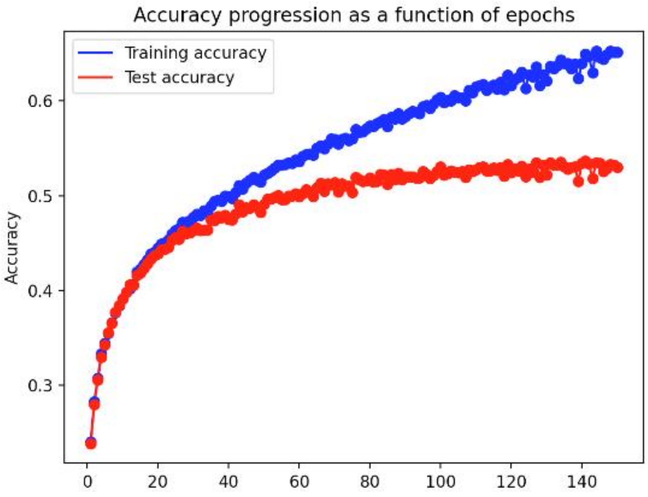

Figure 2 – MLP model accuracy as a function of epochs for 1 hidden layer with 256 units, using ReLU activation. Notice that the model overfits around 70 epochs at 0.5 test accuracy.

Moving on, when working with deep neural networks, ReLU and leaky ReLU are commonly used since they help with the vanishing gradient problem. Leaky ReLU is better at training very deep networks since it allows small gradients to pass through, which significantly helps with the vanishing gradient problem as the network gets deeper. As for the tanh, it too has shown success as an activation function, however, it has two drawbacks which the other two do not have. First, it can give saturation during forward and backward feeds since very small and large values will be near -1 or 1 in the forward feed or near 0 in the backward feed. The second drawback is that the range of the activation is limited to [-1, 1] (Agostinelli et al., 2014).

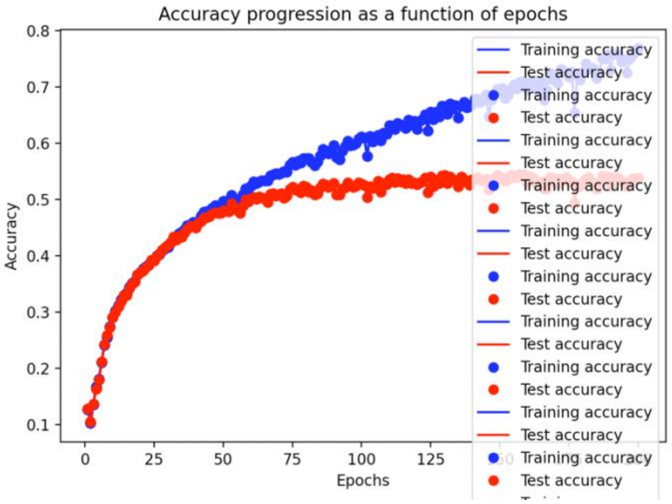

Figure 3 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using ReLU activation. Notice that the model overfits around 75 epochs at 0.53 test accuracy.

Figures 3 – 5 demonstrate how the activation function can influence the MLP model's accuracy. As expected, the model utilizing ReLU (0.53 test accuracy) and leaky ReLU (0.52 test accuracy) activation did perform better than the model using tanh (0.47 test accuracy) activation for the reasons stated above. The ReLU and leaky ReLU models perform similarly because the number of hidden layers is low. Typically, in practice, these two activation functions perform very similarly until the network becomes deep, at which point leaky ReLU performs better as it allows small non-zero gradients to pass through. Hence, it is able to capture higher abstractions of model complexity. However, at the low network depth of 2, both ReLU and leaky ReLU can capture the same complexity within CIFAR-10 (Agostinelli et al., 2014).

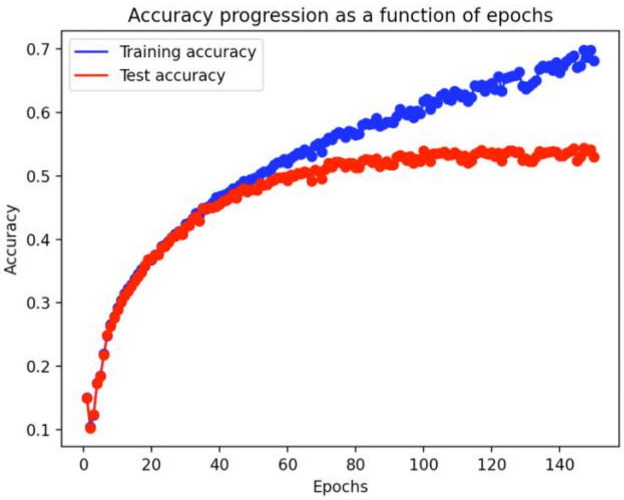

Figure 4 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using leaky ReLU activation. Notice that the model overfits around 80 epochs at 0.52 test accuracy.

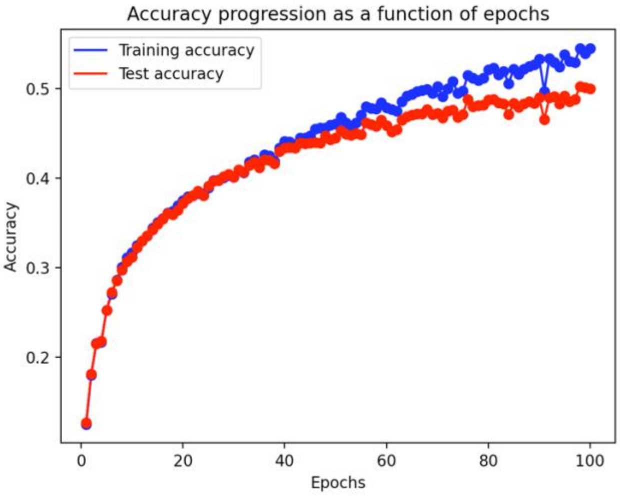

Figure 5 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using tanh activation. Notice that the model overfits around 80 epochs at 0.47 test accuracy.

Regularization and data normalization

Figure 6 summarizes the results of training a 2-hidden-layer MLP model with ReLU activation on unnormalized image data. This model overfits much earlier at around 20 epochs compared to the model trained on normalized image data which started overfitting around 75 epoch (Figure 3). Additionally, the unnormalized model did not generalize as well as its normalized counterpart as its accuracy was 0.47 as opposed to 0.53. These results are exactly what should be expected, emphasizing the reason why we train on normalized data. The unnormalized model is prone to capturing biases and overfitting since the range of its input values are no longer limited to [0, 1], but instead [0, 255]. As a result, certain features like the lighting in the image can dominate the other features, making the model have biases and easily overfit. As a corollary, the model will also not generalize as well to unseen data when testing. In the future, it can be insightful to train a CNN on unnormalized data to see if the discrepancy between the normalized and unnormalized models increases.

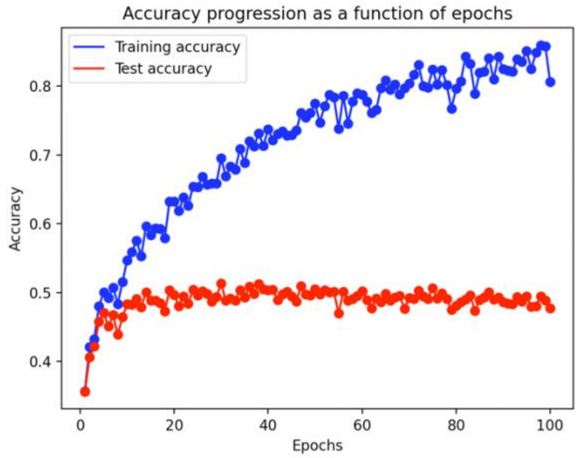

Figure 6 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using ReLU activation and training on unnormalized image data. Notice that the model overfits around 20 epochs at 0.47 test accuracy.

Figures 7 and 8 show the performance of the 2-hidden-layer ReLU MLP model with L1 and L2 regularization. As would be expected when introducing regularization into a model, both L1 and L2 help with the model's overfitting, not reaching a plateau in the testing accuracy until roughly 95 epochs. The overfitting occurs later than the model trained without regularization, which started overfitting around 75 epochs. As for the test accuracy, both the regularized and unregularized models produce similar results. The reason why the regularized models have lower accuracy is because regularization introduces constraints on the model's weights. Thus, regularized models need a longer time to train and learn the complexities of the dataset. In the future, it would be beneficial to let the model run for longer until a clear plateau can be observed. Due a lack of time, this was not possible for this report.

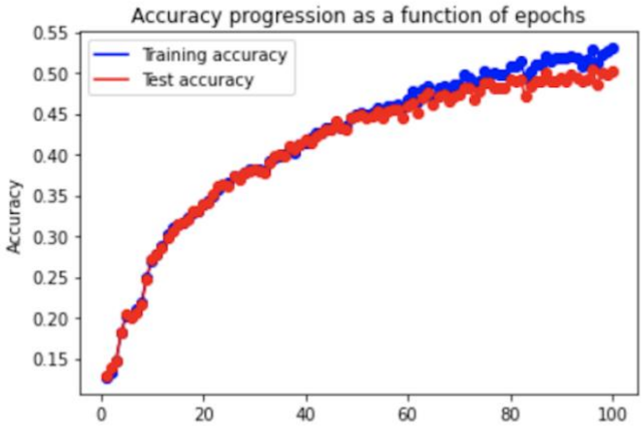

Figure 7 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using ReLU activation and L1 regularization. Notice that the model overfits around 95 epochs at 0.5 test accuracy. The lambda value used was 0.0001.

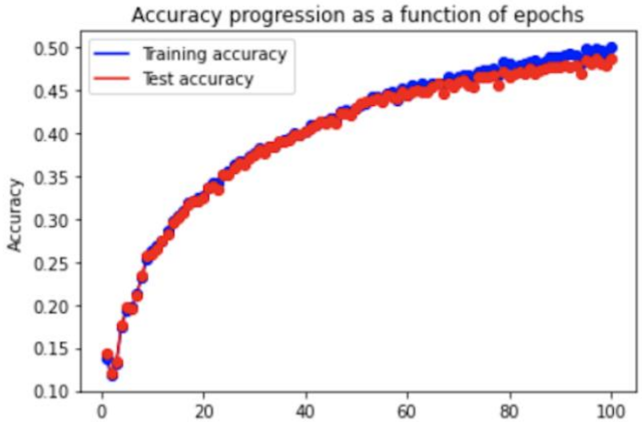

Figure 8 – MLP model accuracy as a function of epochs for 2 hidden layers with 256 units each, using ReLU activation and L2 regularization. Notice that the model overfits around 95 epochs at 0.48 test accuracy. The lambda value used was 0.01.

Adding momentum to the MLP model

As an added parameter to analyze, the use of momentum was added in the backpropagation calculation of gradients, in which the gradient was weighted based on its previous value, and the current value corresponding to a weight factor α for each weight layer. The results of this experiment, using a value of 0.8 for α, and 2 hidden layers of dimensions [256, 256] are shown in Figure 9. A larger learning rate of 0.01 was also used, as each input had less respective weighting, and as a result the accuracy value converged much quicker than the network of similar structure without momentum. Though quicker convergence speed was observed, after about 17 iterations, the maximum accuracy on the test set was similar to the network without momentum.

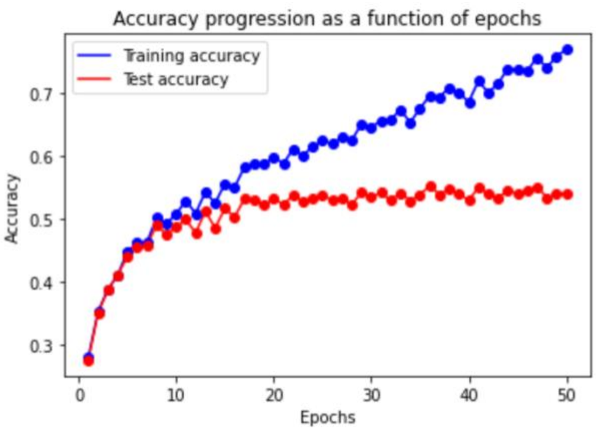

Figure 9 – MLP model using 2 hidden layers with 256 units in each, ReLU activation, and a momentum value of 0.8. The model converges after about 17 iterations, to a test accuracy of ~0.54 from which it begins to overfit the training data.

CNN on CIFAR-10

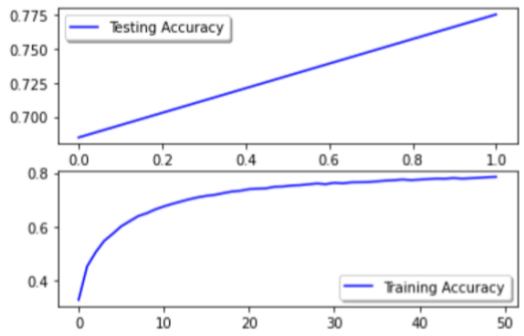

Figure 10 – CNN model using 2 convolutional and 2 fully connected layers with ReLU activation and 256 units per fully connected layer. The model converges around 30 epochs with a testing accuracy of 0.775.

Referencing Figure 10, the CNN yielded a testing accuracy of 0.77, which is higher than the testing accuracy of 0.53 given by the MLP model. This observation is to be expected since MLPs take image input as one-dimensional vectors, whereas CNNs do not. Hence, the spatial structure of the image is taken into account only with CNNs. Also, CNNs share weight parameters, thus reducing the number of parameters in the model which enable the model to generalize well to unseen data and not overfit to training data. Finally, CNNs are better suited for capturing the hierarchical features in images as the network get deeper, as opposed to MLPs. For example, the model learns simple edges in the first layers, and builds off this to learn shapes in subsequent layers much more efficiently than MLPs (Jmour et al., 2018).

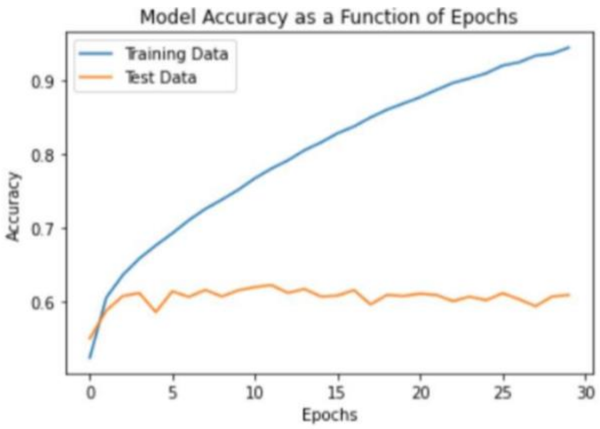

Figure 11 - Pre-trained CNN model using the ResNet-50 network. There were two fully connected layers of size 256, for which ReLU activation was used. Only the fully connected layers were trained on the training data. The model converges to a test accuracy of 0.61

Pre-trained CNN model

The pretrained model used was ResNet-50, a 50-layer deep CNN trained on more than a million images from the ImageNet database. To keep parameters relatively consistent, we added two fully connected layers of size 256 at the end, and trained these layers with our image data from CIFAR-10. As it can be observed in Figure 11, the testing accuracy converged very quickly, after about 2 epochs, where it reached a final value of about 0.61 from which it began to overfit the training data. It should also be noted this was performed on the unnormalized image data, as the normalized image data only reached a maximum value of about 0.35 for test accuracy. This could be due to the fact that the trained weights would have much less of an impact on the overall weighting of the neural network when using smaller normalized image data.

Discussion and conclusions

There are many key takeaways from this report which emphasize the importance of different type of neural networks, their architecture and their hyperparameters. By performing hyperparameter tuning via sklearn, we were able to find optimal hyperparameters to test our most basic version of the MLP model. We determined a batch size of 16, a learning rate of 0.001, and L1 and L2 regularization hyperparameters of 0.0001 and 0.01 to be optimal when training the model without momentum. Moving on, we found that non-linearity activation and network depth helped train the model and improve its accuracy. As we introduced hidden layers (i.e., non-linearity), the accuracy improved from 0.4 to 0.53. As for the depth of network from 1 to 2 hidden layers, the accuracy only improved from 0.5 to 0.53, which is most likely due to the MLP model not being deep enough to capture all the intricacies of the CIFAR-10 dataset. Moving onto how the activation function affected the model accuracy, ReLU and leaky ReLU trained models showed to outperform the tanh model slightly with accuracies of 0.53 and 0.52 compared to 0.47. This was to be expected as ReLU and leaky ReLU help with the vanishing gradient problem and saturation of values that tanh yields.

As for training the MLP with unnormalized image data, this model was overfitting after only 20 epochs compared to 75 epochs for its normalized counterpart. Additionally, it did not generalize to unseen data as well, having an accuracy of 0.47 compared to 0.53. To continue, both L1 and L2 regularization helped decrease overfitting with the MLP model, with similar accuracy to the unregularized model. A momentum parameter of 0.8 was finally added to the experiment, resulting in faster convergence speed, though similar final accuracy of 0.54 to the same model without momentum. After training a CNN with the CIFAR-10 dataset, the testing accuracy increased dramatically to 0.77, emphasizing why CNNs model image data better than MLPs. Comparatively, using ResNet-50 as a pretrained model and training on two additional fully connected layers resulted in very fast convergence, though only to a test accuracy of 0.61 which was less than the CNN. This is likely due to the fact that only the final layers were trained, and a higher accuracy would likely be achievable through training the entire model with the CIFAR-10 dataset.

For better accuracy in the future, we could experiment with different neural network architectures and parameters such as deeper networks with dropout, the addition of max pooling layers, RMSProp, and additional MLP layers added to the CNN. Additionally, letting the regularized models train over more epochs could show to improve the test accuracy of the model even more since regularization helps with reducing overfitting in deeper networks. In a paper published by Doon et al., they were able to achieve almost 88% testing accuracy through combining some of these principles, giving a good starting point for future optimization of the network.

Statement of contributions

Aidan and Kyle were responsible for implementing the MLP model form scratch and for experiments 1, 2, 3, 4, and 6, as well as introducing momentum into the MLP model and hyperparameter tuning of the batch size, learning rate, and L1 and L2 hyperparameters. Yanying was responsible for experiment 5. All members contributed to the report.

References

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). *Learning Activation Functions to Improve Deep Neural Networks*. ArXiv.org. https://arxiv.org/abs/1412.6830

Doon, R., Kumar Rawat, T., & Gautam, S. (2018). Cifar-10 Classification using Deep Convolutional Neural Network. *2018 IEEE Punecon*, 1–5. https://doi.org/10.1109/PUNECON.2018.8745428

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width. *Advances in Neural Information Processing Systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/32cbf687880eb1674a07bf717761dd3a-Abstract.html

Jmour, N., Zayen, S., & Abdelkrim, A. (2018). Convolutional neural networks for image classification. *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. https://doi.org/10.1109/aset.2018.8379889

Thakkar, V., Tewary, S., & Chakraborty, C. (2018, January 1). *Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data*. IEEE Xplore. https://doi.org/10.1109/EAIT.2018.8470438