Applied Machine learning Mini Project 2

Rahma Nouaji, Mohammad Ghavidel, Bita Farokhian

Abstract

In the scope of this project, we investigated the performance of 4 models, namely Multi-layer Perceptron (MLP), Convolution Neural Networks (CNN), and two pre-trained models, Resnet50 and Densenet201, on the CIFAR-10 dataset. MLP with two hidden layers (RelU activation function each), with a learning rate of 0.01, a batch size of 64, as well as L2 regularization, achieves a modest performance of around 51.36%. Moreover, the CNN model with a learning rate of 0.001 converged to its optimal point after 15 epochs reaching a testing accuracy of 68.95 %. Data augmentation techniques and Dropout method were performed for the pre-trained models as well. Resnet 50 revealed a good testing performance reaching around 84%. Finally, the DensNet201 reached the highest accuracy 88% with a learning rate of 0.0001 using Adam optimizer.

1 Introduction

The goal of this project is to investigate the performance of the Multi-layer perceptron model while varying its various hyperparameters on the CIFAR-10 dataset. We also want to highlight the performance of the CNN and pre-trained models ResNet and DenseNet on the same dataset. We found that the pre-trained model DenseNet has significantly outperformed the other models reaching 88% of evaluation accuracy. The CIFAR-10 dataset is commonly used as a benchmark for machine learning algorithms and deep learning models in computer vision research. CIFAR-10 was used in [1] to demonstrate the effectiveness of ResNet in handling the problem of vanishing gradients in very deep neural networks. It was also used in [2] to evaluate the performance of four Convolutional Neural Network (CNN) models for image recognition and classification. In [3], the CIFAR-10 dataset was used to demonstrate that some common regularization techniques (such as weight decay and dropout) may not always improve generalization performance.

2 Dataset

The CIFAR-10 dataset is a well-known computer vision dataset consisting of **60,000 32x32** color images in **10 classes** (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), with **6,000 images per class**. The dataset is divided into **50,000 training** images and **10,000 test** images.
The images are relatively low resolution compared to some other datasets, but the variety of objects and backgrounds, as well as a large number of images, make it a popular choice for image classification tasks.
We used a bar chart to visualize the distribution of the 10 classes, which shows the same number of images (6,000) for each class, indicating a balanced dataset. We also visualized different images for each class in order to get a sense of the different objects of the dataset. The image data must be processed to remove unwanted distortions or enhance certain image features that are important for our model to perform better.

We used a histogram to visualize the distribution of pixel values in CIFAR-10. It reveals a roughly uniform distribution of pixel values between 0 and 255, indicating that the images are not too bright or too dark. Pixels in an image range from 0 to 255, and by dividing each pixel by 255, we can scale them to range from 0 to 1. This makes the convergence of cost faster while training the network.

3 Results

3.1 Multi Layer Perceptron (MLP)

Table 1: Grid search cross validation (Number of folds = 5).

Hyperparameters	Mean validation error	Std of validation error
'lr': 0.001, 'batch_size': 128	1.879673	0.099449
'lr': 0.001, 'batch_size': 64	1.799852	0.099454
'lr': 0.001, 'batch_size': 32	1.732252	0.10832
'lr': 0.01, 'batch_size': 128	1.6327	0.105868
'lr': 0.01, 'batch_size': 64	1.611255	0.102511
'lr': 0.01, 'batch_size': 32	1.703970	0.167403
'lr': 0.1, 'batch_size': 128	1.679505	0.102833
'lr': 0.1, 'batch_size': 64	1.767096	0.136540
'lr': 0.1, 'batch_size': 32	2.053829	0.301430

- **Task 2:** We performed a grid search cross-validation with 5 folds over 3 different sets of learning rates and batch sizes in order to find the best hyperparameter combination for our task, which is defined by the one having the lowest standard deviation and mean validation error. This technique was so time-consuming since it loops over 9 combinations of hyperparameters and runs the model 5 times for 30 epochs. This grid search was performed on a model with 2 hidden layers (Model 3). According to table 1, the lowest standard deviation is for the combination of hyperparameters learning rate 0.01 and batch size 64.

- **Task 3:** We used the Early Stopping technique because we noticed that the model was overfitting the data. For that, we opted for a variable called patience which is a hyperparameter that represents the number of epochs that the model is allowed to continue training without improvement in validation performance before early stopping is triggered.

- **Task 3.1:** We evaluated three different models: the first one is an MLP with no hidden layers, the second is an MLP with a single hidden layer having 256 units and ReLU activations, and finally, the last one is an MLP with 2 hidden layers, each having 256 units with ReLU activations. From Figure 1 and Table 3, we can observe that increasing the non-linearity and network depth leads to an improvement in accuracy. The MLP with no hidden layer is a linear classifier that cannot capture complex non-linear relationships between the inputs and outputs, which explains why it has the lowest accuracy, around 38%; it also has a high training and validation loss which reflects underfitting. On the other hand, the MLP with two hidden layers has slightly outperformed the MLP with a single hidden layer with 50 %, indicating that increasing the depth of the network helps to capture more

complex patterns and improve the accuracy of the model. Notice that the third model is experiencing overfitting; for that, our next step would be regularization.

Table 2: Evaluation accuracy for the three MLP models.

	First model	Second model	Third model
Models evaluation accuracy (30 epochs)	0.3868	0.4896	0.5029

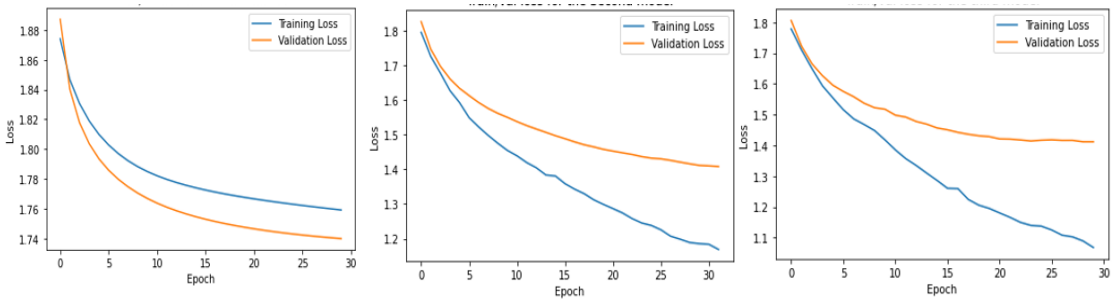

Figure 1: Cross Entropy Loss for three models with batch size 64 and learning rate 0.01 (MLP (no hidden layer), MLP (1 hidden layer), MLP (2 hidden layers).

- **Task 3.2:** In addition to ReLU, we also experimented with other activation functions, such as Leaky ReLU and Tanh in our third model. However, we encountered issues with the Tanh function due to input values that were either too large or too small. This resulted in the vanishing gradient problem and prevented effective training. We tried to set maximum norm constraints on the weights to enforce an absolute upper bound on the magnitude of the weight vector for every neuron and used projected gradient descent to enforce the constraint; however, it did not give any improvement. In contrast, the third model with the Leaky ReLU activation function yielded slightly lower test accuracy results (around 49%) than the model with ReLU activation. We used different sets of alpha, and we found that alpha=0.01 yielded the best results. These results put the emphasis on the RelU activation function as the best one. We expected to have a better performance with leaky Relu. The reason behind this could be the positivity of the input variables since the data was scaled between 0 and 1.

- **Task 3.3:** We added the regularization term L1 and L2 to MLP with 2 hidden layers and Relu activation functions. For this model, we used a learning rate of 0.01, 64 batch size, and 30 epochs based on grid search results. After applying various lambda, we conclude that 0.01 reaches higher accuracy around 51 % which is highlighted in Table 3. Overall, L2 shows better results compared to L1 with respect to lambda.

Table 3: Evaluation accuracy for the regularization MLP models.

	L1 model	L2 model
Models evaluation accuracy Lambda 0.01	0.4925	0.5136
Models evaluation accuracy Lambda 0.05	0.4749	0.5009
Models evaluation accuracy Lambda 0.1	0.3131	0.5036

- **Task 3.4:** When the data is not normalized, it causes large values in the calculations, which can lead to overflow and underflow errors. This can then result in incorrect calculations, such as division by

zero or NaN values, which in turn can cause the training process to fail and produce incorrect results.

3.2 Convolutional Neural Network CNN

- **Task 3.5:** We employed the CNN model with TensorFlow. The model receives inputs that are tensors of shape (image height, image width, color channels). The configuration for this CNN algorithm is to process inputs of shape (32, 32, 3), which is the format of CIFAR images. Layers Conv2D and MaxPooling2D are stacked on top of each other. We set the number of units in the fully connected layers to 256. Figure 2 shows that the model has a slower convergence and lower accuracy with 256 units per layer around 68.95 % compared to 70.98% when we tried using 32 units per layer.

  In order to complete the model, we added one or more Dense layers over the final output tensor from the convolutional base (4, 4, 64). The current output of the dense layer is a 3D tensor, which takes vectors as input (which are 1D). A Dense layer is then added on top of the flattened (or unrolled) 3D output. The Dense final layer has 10 outputs since CIFAR has 10 classes. The CNN model reaches 68.95% accuracy. For the CNN, the use of a ReLU activation function in the output features of a Dense layer with 256 units resulted in better performance compared to using a Tanh activation. As can be inferred from the results, CNN outperformed the MLP models due to the following reasons: 1) Image data has a spatial structure: In MLPs, the input features are flattened into a single vector, while CNN is designed to preserve the spatial structure of the input image. 2) pooling layer in the CNN model is designed to reduce the spatial dimensions of the feature maps. 3) CNNs use convolutional layers that are specifically designed to identify local patterns in the image.

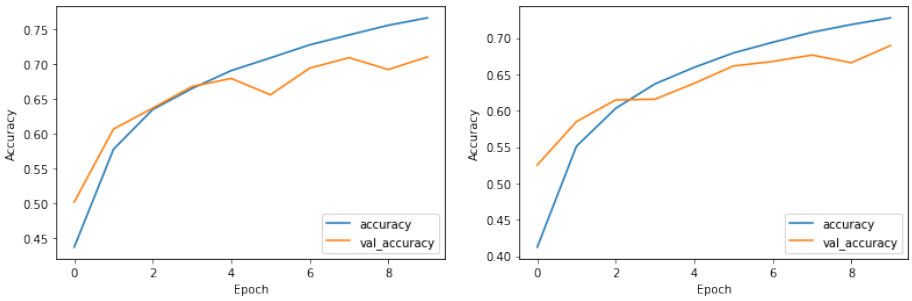

Figure 2: CNN models accuracy (Left 32 unit layer)(Right 256 unit layer).

3.3 Pre-trained Models: Resnet-50 and DenseNet201

- **Task 3.6:** We utilized pre-trained models including ResNet50 and DenseNet201 [4]. First, we implemented the pre-trained Resnet 50 model with different parameters, and the best was the one with the learning rate of 0.01, a batch size of 128, and 5 epochs (we opted for fewer epochs for the pre-trained model for lower computational cost); the accuracy for this model was 84.7 %. DenseNets models have several advantages, such as reducing the number of parameters, solving the vanishing gradient problem, increasing feature propagation, and encouraging feature reuse. In summary, training accuracy of around 90 % and validation accuracy of around 88 % can be obtained by transfer learning from pre-trained CNN algorithms from the Keras database. Figure 3 depicted the result for pre-trained

CNN where DenseNet 201 has outperformed Resnet 50 showing a better validation accuracy (88 %) and faster convergence. These pre-trained models are optimized to have faster convergence with lower epochs (< 10) to reach higher accuracy (above 80).

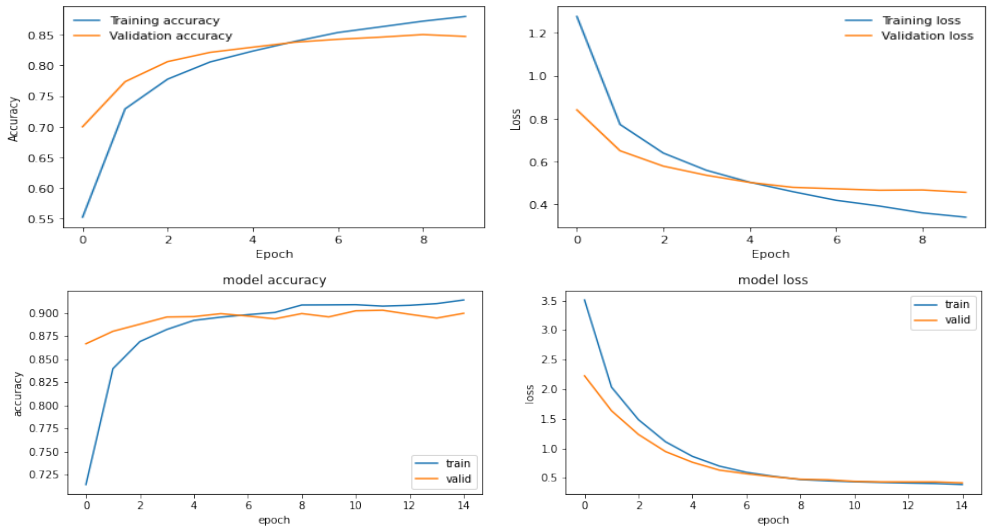

Figure 3: Above: RestNet50 accuracy and Down: DenseNet201 model accuracy.

- **Task 3.7:** We observed after 30 epochs, the MLP model tends to overfit. Thus, we employed early stopping. For more Figures, check the Colab link Ml project group 4.

4 Discussion and Conclusion

We conclude that normalizing the input data before training a neural network ensures numerical stability. Increasing the depth of a model can increase its capacity to learn more complex features and patterns in the data. We conclude that L2 regularization has performed better than L1 since it encourages smaller but non-zero coefficients, leading to a more stable model, unlike L1 regularization which can cause sparsity and make the model less robust. The L2 regularization was used to improve the test accuracy and prevent overfitting along with early stopping for the MLP model and pre-trained models as well. We conclude also that each activation function is specific to a task. Performing data augmentation techniques (cropping, flipping...) and adding dropout to the pre-trained models (0.5 for DenseNet) helped prevent overfitting and improve the test accuracy. Using a pre-trained CNN for CIFAR-10 has a fast convergence thanks to pre-loaded weights. DenseNet 201 has outperformed the Resnet50 and CNN models. We conclude that CNN and pre-trained models are advantageous and more expressive for image processing tasks since they are able to capture more features and patterns in an image. For future investigation, We could tackle an image segmentation problem where we can use these pre-trained models and compare their performance with a well-known model for image segmentation such as UNET-3D [5].

5 Statement of Contributions

The tasks were distributed evenly among the team members.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] R. K. R, G. R. Namita, and R. Kulkarni, "Image recognition, classification and analysis using convolutional neural networks," in *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pp. 1–4, IEEE, 2022.

[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2017.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Springer International Publishing, 2015.