AI Capstone: Project 1 Report

110550088 李杰穎

March 10, 2023

1 Public Image Datasets: CIFAR-10

1.1 Datasets Description

CIFAR-10 is a popular image classification datasets that is frequently used as a benchmark for computer vision and deep learning tasks. It contains 60,000 32x32 color images of 10 different objects, with 6,000 images per class. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The datasets is divided into 50,000 training images and 10,000 testing images. CIFAR-10 is challenging due to the small image size and the high variability of the objects within each class. It is often used for developing and testing new machine learning algorithms and architectures for image classification.

Figure 1: Ten randomly select images from 10 classes.

In this project, I will directly use the datasets provided in the `torchvision` library, which is part of the PyTorch deep learning framework. Torchvision is a library consists of lots of datasets (including MNIST, CIFAR-10, ImageNet, ...) and pretrained model (including ResNet, VGGNet, BERT, ...).

As for the test set, I will use the original train/test split of the CIFAR-10 datasets. The datasets is divided into 50,000 training images and 10,000 testing images, with an equal number of images from each class in both sets. This train/test split is commonly used in the literature and provides a fair evaluation of the performance of machine learning models on the CIFAR-10 datasets.

It's also worth noting that cross validation can be also used for the CIFAR-10 datasets. However, the number of images in the datasets is large and most of the research paper use the original train/test split. Therefore, it's reasonable for this project to use the original train/test split.

Noted that I also use torchvision to download and process the datasets.

1.2 Algorithms

For the CIFAR-10 datasets, I use two classifier, ResNet-18 and Multilayer Perceptron (MLP).

1.2.1 ResNet-18

ResNet-18 is a popular deep convolutional neural network architecture used for image classification tasks. It was introduced by Microsoft Research in 2015 and is part of a family of ResNet models, which are designed to tackle the problem of vanishing gradients in very deep neural networks. ResNet has different variance, from ResNet-18 all the way to ResNet-152. The number after ResNet indicates the number of layers. Therefore, ResNet-18 is a relatively shallow model, with 18 layers, but it still achieves state-of-the-art performance on a range of image classification benchmarks, including ImageNet, CIFAR-10, and CIFAR-100.

For the sake of convenience, I directly use the ResNet-18 models which provides in torchvision library. Torchvision library also provides the pretrained weights. However, those weights are trained on ImageNet Datasets, not CIFAR-10. Therefore, I also compare the performance with or without using pretrained weights. There are more details in appendix.

It's worth noting that because the image size of CIFAR-10 is only 32 x 32. And ResNet is original designed for ImageNet, which image size is 224 x 224. Therefore, the first convolution layer with kernel size equals to 7 is too large for CIFAR-10 datasets, this will make lots of features lost from the first convolution layer. Moreover, the max pooling layer may also make features lost. Therefore, I change the kernel size from 7 to 3 and also remove the max pooling layer. In the analysis section, I will discuss the performance difference after changing these two layers.

1.2.2 MLP

A multilayer perceptron (MLP) is a type of artificial neural network (ANN) that consists of multiple layers of interconnected nodes (also called neurons) in a feedforward network architecture. It is one of the most commonly used neural network models for supervised learning tasks such as classification, regression, and pattern recognition.

An MLP typically consists of an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer is determined by the number of features in the input data, while the number of neurons in the output layer corresponds to the number of output classes or the number of output variables in a regression task. The number of neurons in the hidden layers is a hyperparameter that can be tuned during the training process.

We know that MLP has one input layer, one output layer and several hidden layers. In this project, I will only use an MLP that consists of two hidden layers with same number of neurons. The number of neurons is also a hyperparameters that will compare in the analysis section. As for the input layer, I just flatten the images into an one dimensional vector, with $3 \times 32 \times 32 = 3072$ elements as input. For the output layer, MLP will output a one dimensional vector with 10 elements, corresponding to 10 classes.

1.2.3 Epochs, Loss Function, Optimizer and Learning Rate Scheduler

In the following experiments, I will use the below configuration for both MLP and ResNet-18.

- Epochs: 200
- Learning Rate: 0.1
- Loss Function: `CrossEntropyLoss`
- Optimizer: `SGD`
- Learning Rate Scheduler: `ReduceLROnPlateau` (monitor on `val_acc,patient=10,factor=0.1`)
- Batch Size: 128

Learning rate scheduler is a tool to change learning rate while training. And the LR scheduler I use will monitor on `val_acc`. When `val_acc` doesn't increase for 10 epochs, the learning rate will be reduced by 90%.

1.3 Analysis

First of all, because CIFAR-10 already has provided a train/test split, I will not use cross-validation for this datasets.

1.3.1 Compare Different Architecture of ResNet-18

As I mentioned earlier, because the kernel size of first convolution layer is too large for images in CIFAR-10 datasets, I change the kernel size and also remove the first max pooling layer. This version is called "Modified ResNet-18", and the unmodified one is called "Original ResNet-18". As we can see in the chart from Figure 3 to Figure 6, the modified one is significant better than the unmodified one. The highest test accuracy for the modified one is 0.9227, in the contrast, the unmodified one is 0.8766.

In conclusion, changing the architecture of ResNet can indeed significantly improve the model's accuracy.

1.3.2 Comparing whether there is a pre-trained weight for ResNet-18

PyTorch provides a pretrained weight for ResNet-18. However, the pretrained weight is trained on ImageNet datasets. Therefore, it might perform well on CIFAR-10 datasets. Because of this I make an experiment to test how the existence of pretrained weight affect the performance of model.

As we can see in Figure 7 and Figure 8, the ResNet-18 with pretrained weight doesn't perform very well. The highest test accuracy of ResNet-18 without pretrained weight is 0.9227, as for the model with pretrained weight, this number is 0.9177. I will discuss more in 1.4.1.

1.3.3 Compare Different Architecture of MLP

For the MLP, I fixed the number of hidden layer to two. And by setting the number of neurons in hidden layers to 100 and 2500, I trained two MLP model.

As we can see in the chart (Figure 11 to Figure 14), the MLP with 2500 hidden neurons is significant better than the 100 ones. The highest testing accuracy for the 2500 one is 0.5886, as for the 100 ones is 0.5324.

1.3.4 Compare Different Amount of Training Data

In this section, I will first analysis ResNet-18 case, and then analysis MLP case.

Noted that though the number of training data is different, the testing data for both model is exactly the same.

First, for ResNet-18, we can look at Figure 15 and Figure 16. As we can observe, the "half" one train much more

slower than the ResNet-18 that trained with full training CIFAR-10 datasets, which contains 50000 images. And the final test accuracy, the "half" one's accuracy is 0.8782. Which is less than the full-trained model by around 5%.

Second, let's look at the MLP. Also we can observe that in Figure 17 and Figure 18, the training accuracy of the MLP with full training datasets is slightly better, and for the testing accuracy, it is nearly the same. The "Half MLP" is 0.5816, and the "Full MLP" is 0.5886.

1.3.5 Compare with SOTA Model

After researching, the state-of-the-art model on CIFAR-10 is Vision Transformer (ViT). In particularly, the ViT-H/14 model can achieve 99.5% test accuracy, which is astonishing results. However, this model require 632M parameters, in the contrast, ResNet-18 only requires 11M parameter.

In conclusion, ViT is 57 times larger than ResNet-18, but it only increase 7% of accuracy.

1.3.6 Overall Comparison

From the above comparing, we know that the best performance for ResNet-18 is the modified ResNet-18 without pretrained weights. As for MLP, the best model is with 2500 hidden neurons. And the SOTA model for CIFAR-10 is ViT-H/18.

For the overall comparison table, please refer to **??**.

1.4 Discussion

1.4.1 Assessing Expected Results and Behaviors in Experiments

1. **Using pretrained weights or not**

   In 1.3.2, I compare the performance of the ResNet-18 with or without pretrained weights. And the result is the model without pretrained weights has better performance.

   One can guess the model with pretrained weights might be better, but it's not. I think one main reason is that I already changed the model architecture, therefore the idea of "transfer learning" didn't work well here. And also, the distribution of CIFAR-10 and ImageNet is also difference. Therefore, trained the model from scratch might be a better idea.

   In conclusion, I think these are the two main reasons why the model without pretrained weights is better.

1.4.2 Factors Affecting Performance

1. Model architecture
2. Different hyperparameters
3. Number of epochs count
4. Number of training data
5. LR scheduler

   Using LR scheduler is actually really important for machine learning, especially when SGD is used. Because larger LR in the later stage of training might make the model can't converge to optimal point. Therefore it's necessary to have LR scheduler.

1.4.3 Future Experiments

1. **Using adam optimizer instead of SGD.**

   The adam optimizer can change the learning rate of each weights, which is might be a better method compared with LR decay. The SGD along with LR decay method may take longer time to converge.
2. **Apply features extractions before sending into MLP.**

   Apply features extractions on images can extract useful features, and using these extracted features might be able to improve the MLP performance.

1.4.4 Findings and Open Questions from Experiments

1. **MLP doesn't perform very well**

   Before conducting the experiment, I thought it may achieve an accuracy around 70%. But it turns out only get 60%.
2. **Changing the architecture of the CNN is useful**

   I improved the model's accuracy by 5% by only changing the kernel size of one layer and removing one max pooling layer. This is a new inspiration for me that sometimes we need to compare the difference of the images we need to classify now with the designated image, and based on the input images to design the proper architecture.

2 Public Non-Image Datasets: 20 Newgroups

2.1 Datasets Description

The 20 Newsgroups datasets is a popular datasets used in natural language processing and machine learning research. It consists of a collection of approximately 20,000 documents, partitioned into 20 different newsgroups,

each representing a different topic. The datasets was first collected by Ken Lang and others at the University of California, Irvine, and has since been widely used in research and experimentation.

Each document in the 20 Newsgroups datasets is a posting to one of the 20 newsgroups, and is represented as a single text file. The datasets includes a variety of topics, including politics, religion, sports, science, and technology, among others.

In this project, I will use vectorize version of this datasets. This is done by using the `CountVectorize` in scikit-learn library. In this way, the text data is transformed to real-valued vector, which can be processed by `RandomForestClassifier`, `GradientBoostingClassifier`, `AdaBoostClassifier`, etc.

2.2 Algorithms

In this datasets, I use four kinds of ensemble learning algorithms, Random Forest Classifier, Gradient Boosting Classifier, Histogram-based Gradient Boosting Classifier and AdaBoost Classifier. One SVM algorithm, linear SVM. And a MLP Classifier, which is same MLP architecture in the first datasets. Therefore, I will only talk about the four ensemble learning algorithms and the SVM algorithm.

Random Forest Classifier, Gradient Boosting Classifier, Histogram-based Gradient Boosting Classifier and AdaBoost Classifier are all powerful machine learning algorithms commonly used for classification tasks. These algorithms are known for their ability to handle complex datasets and provide accurate predictions.

And SVM (Support Vector Machines) is also a machine learning algorithm used for classification and regression analysis. It was developed by Vapnik and his colleagues in the 1990s. SVM is a supervised learning algorithm that works by finding the optimal hyperplane that separates the data into different classes. The hyperplane is chosen in a way that maximizes the margin between the closest data points from each class. These closest data points are known as support vectors.

These six classifiers can be found in the `scikit-learn` library. In the following experiments, I directly apply the default value of `scikit-learn` provides. I state the option clearly in different classifier.

2.2.1 Random Forest Classifier

Random Forest Classifier is an ensemble learning algorithm that creates multiple decision trees and combines their outputs to make a final prediction. Each tree is trained on a random subset of the features and samples of the dataset, making the algorithm robust to overfitting. The final prediction is made by taking the mode of the predictions of all trees in the forest.

- The number of tree: 1000 or 500 (compare the difference)
- Criterion: Gini Impurity

2.2.2 Gradient Boosting Classifier

Gradient Boosting Classifier is another ensemble learning algorithm that creates multiple weak learners and combines their outputs to make a final prediction. Unlike Random Forest Classifier, it creates trees sequentially, with each new tree correcting the errors of the previous tree. This iterative process continues until a stopping criterion is met, resulting in a strong learner that makes accurate predictions.

Gradient Boosting algorithm is widely used in many machine learning competitions in Kaggle. This is reason why I choose this algorithms in this project.

- The Number of Tree: 100 or 50 (compare the difference)
- Loss Function: `log_loss`
- Criterion: `friedman_mse`
- Learning Rate: 0.1
- Max Depth: 3

2.2.3 Histogram-based Gradient Boosting Classifier

The Histogram-based Gradient Boosting Classifier is a variation of the standard gradient boosting classifier in scikit-learn. The primary difference between the two lies in the way they handle large datasets.

The traditional gradient boosting classifier typically uses the entire dataset to build the decision tree at each boosting iteration. This can be computationally expensive and slow for large datasets. In contrast, the Histogram-based Gradient Boosting Classifier uses histogram-based gradient boosting, which aggregates and precomputes information about the dataset in bins or buckets, making it more efficient for large datasets.

Furthermore, the Histogram-based Gradient Boosting Classifier uses a technique called gradient-based early stopping, which allows it to automatically stop training once the model starts to overfit the training data. This is different from the traditional gradient boosting classifier which requires the user to manually specify the number of boosting iterations.

Overall, while the standard gradient boosting classifier is a powerful algorithm that can produce accurate predictions, the Histogram-based Gradient Boosting Classifier is a more efficient and scalable algorithm that is particularly useful for large datasets with high dimensionality.

The `scikit-learn` implementation is inspired by LightGBM.

- Epoch: 100 or 50 (compare the difference)
- Loss Function: `log_loss`
- Learning Rate: 0.1
- Maximum Leaf Nodes: 31
- Maximum Number of Bins: 255
- Tolerance: 10^{-7}
- Patience: 10

2.2.4 AdaBoost Classifier

AdaBoost Classifier, short for Adaptive Boosting Classifier, is also an ensemble learning algorithm that creates multiple weak learners and combines their outputs to make a final prediction. It is similar to Gradient Boosting Classifier in that it creates trees sequentially, but it assigns weights to each sample in the datasets based on how difficult it is to classify correctly. Samples that are misclassified by a weak learner are given higher weights, making them more likely to be correctly classified by the next weak learner.

- Estimator: Decision tree with maximum depth 1
- Number of Estimators: 50 or 500 (compare the difference)
- Algorithm: SAMME.R

2.2.5 Linear SVM Classifier

Linear SVM is a specific type of SVM that assumes the data is linearly separable, meaning that a straight line can be drawn to separate the data points into different classes. Linear SVM is used when the data is linearly separable, which means that the data can be classified into two or more classes by a straight line. In this case, the SVM algorithm finds the optimal hyperplane in a way that maximizes the margin between the closest data points from each class while keeping the classification error rate as low as possible.

Linear SVM is a simple and effective algorithm that works well on a wide range of problems. However, it may not work well when the data is not linearly separable. In this case, other types of SVMs or non-linear methods may be more appropriate.

- Penalty: L2
- Loss: squared hinge
- dual: True or False (compare the difference)

2.2.6 MLP Classifier

Noted that this MLP Classifier is the built-in version in scikit-learn.

- Epochs: 200
- Optimizer: Adam
- Learning Rate: 0.001
- Momentum: 0.9
- Loss Function: `CrossEntropyLoss`
- Tolerance: 0.0001
- Patient: 10
- Number of hidden layer: 1
- Number of hidden neurons: 100

2.3 Analysis

2.3.1 Compare Different Random Forest Classifier

2.3.2 Compare Different Gradient Boosting Classifier

2.3.3 Compare Different Histogram-based Gradient Boosting Classifier

2.3.4 Compare Different AdaBoost Classifier

2.3.5 Compare Different MLP Classifier

2.3.6 Compare Different Amount of Training Data

2.3.7 Compare with SOTA Model

Currently, to my best knowledge, the SOTA of 20 newsgroups datasets is LinearSVM+TFIDF. TFIDF is a kind of vectorizer and LinearSVM is just a simple SVM algorithm.

2.3.8 Overall Comparison

2.4 Discussion

2.4.1 Assessing Expected Results and Behaviors in Experiments

Before conducting the experiment, I guess the Adaboost or histogram-based gradient boosting algorithm may perform pretty well. Because I heard that GB algorithms perform really well on lots of Kaggle competition. However, after the experiments, I found that the best model is simple linear SVM classifier, and it only took around 2 minutes to finished training. Compared to histogram-based gradient boosting, it takes nearly 40 minutes to binning the data, and 4.6 hours to train the model. Which is really time-consuming, and the result is not really good. This really surprised me. I will have more research on this behavior after this project.

2.4.2 Factors Affecting Performance

1. Different algorithms
2. The latent distribution of data
3. Different vectorizer and preprocessing method

2.4.3 Future Experiments

1. **Try to use different vectorizer**
2. **Try more text preprocessing**
   Including removing stop words, stemming, etc.
3. **Try to use NLP Model like BERT**

2.4.4 Findings and Open Questions from Experiments

1. Linear SVM classifier is a great model for some cases
2. Histogram-based gradient boosting is time consuming
3. For simple classification task, the traditional machine learning can already perform really well

3 Self-made Datasets: Satellite Images of 5 Regions

3.1 Datasets Description

In this datasets, I collect the satellite images of mountain area from 5 regions over the world, including Taiwan, Canada, Himalaya, Hengduan and Argentina. The task is to classify the satellite images to these five regions. The satellite images are from MapTiler, a tile map service. I first calculate which tiles should be download, then combine those tiles into a PNG file. Each image is an 256×256 RGB images.

figure/terrain_regions.png

Figure 2: This figure shows 5 satellite images from 5 regions

3.2 Algorithms

Because this datasets is also a image datasets, I just use the two classifiers (ResNet-18 and MLP) which used in the CIFAR-10 datasets with the exact same hyperparameters and configurations.

For MLP, because directly use $3 \times 256 \times 256$ as input layer is too large, I down-sampling the image from 256 x 256 to 128 x 128.

3.3 Analysis

3.3.1 Compare Different Architecture of ResNet-18

3.3.2 Compare Different Architecture of MLP

3.3.3 Overall Comparison

3.3.4 Compare Different Amount of Training Data

3.3.5 Compare with SOTA Model

3.4 Discussion

3.4.1 Assessing Expected Results and Behaviors in Experiments

3.4.2 Factors Affecting Performance

3.4.3 Future Experiments

3.4.4 Findings and Open Questions from Experiments

Appendix A Figures, Charts and Tables

A.1 Public Image Datasets: CIFAR-10

Figure 3: Training accuracy of two ResNet-18 models on CIFAR-10 datasets

Figure 4: Training loss of two ResNet-18 models on CIFAR-10 datasets

Figure 5: Testing accuracy of two ResNet-18 models on CIFAR-10 datasets

Figure 6: Testing loss of two ResNet-18 models on CIFAR-10 datasets

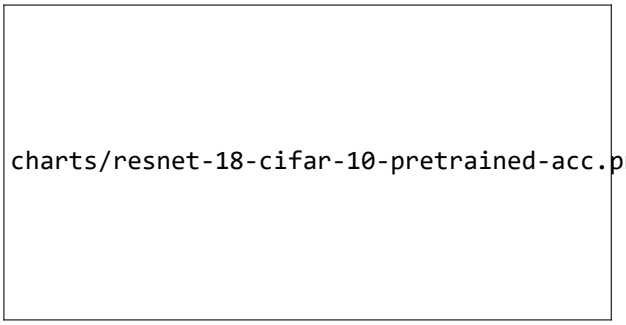

Figure 7: Training and testing accuracy of two ResNet-18 model with and without ImageNet pretrained weight

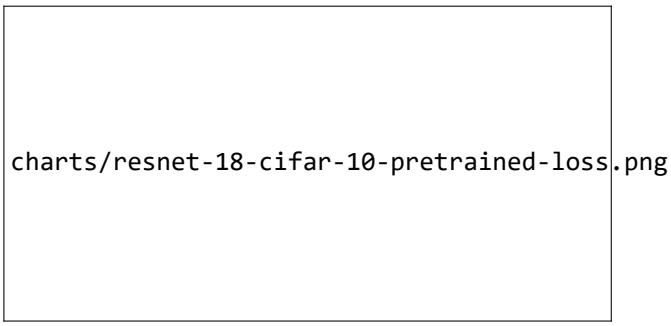

Figure 8: Training and testing loss of two model with and without ImageNet pretrained weight

Figure 9: Accuracy table of pretrained ResNet-18

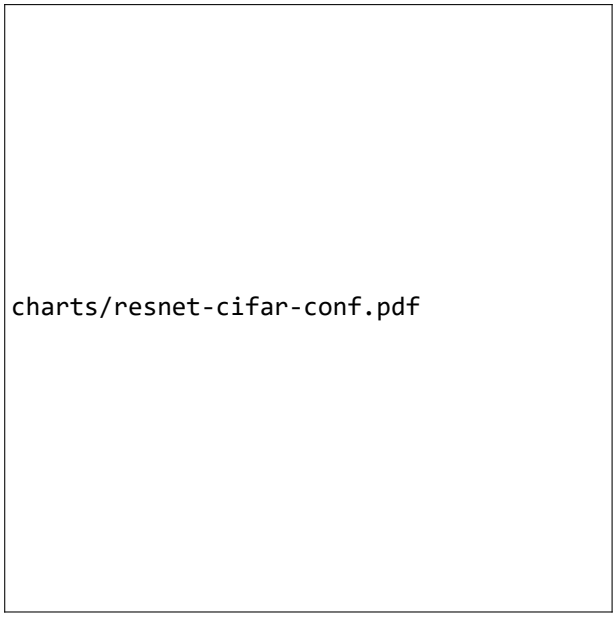

Figure 10: Confusion matrix of pretrained ResNet-18

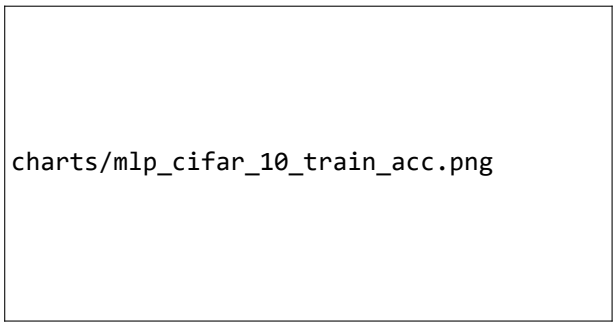

Figure 11: Training accuracy of two MLP models on CIFAR-10 datasets

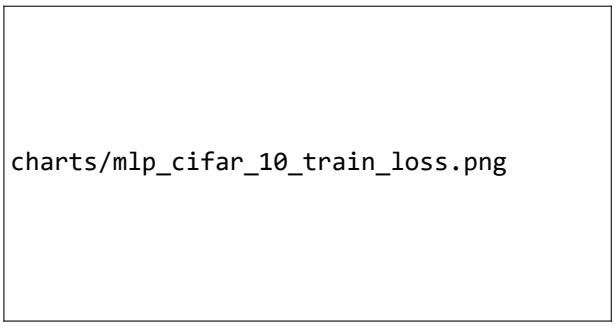

Figure 12: Training loss of two MLP models on CIFAR-10 datasets

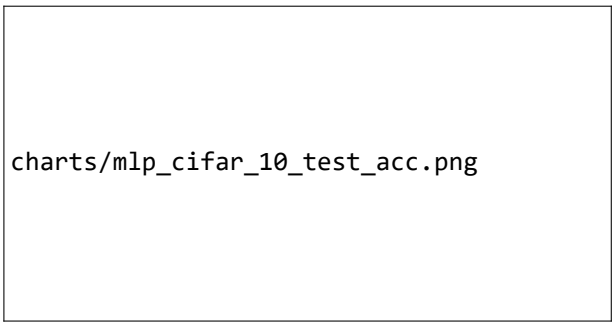

Figure 13: Testing accuracy of two MLP models on CIFAR-10 datasets

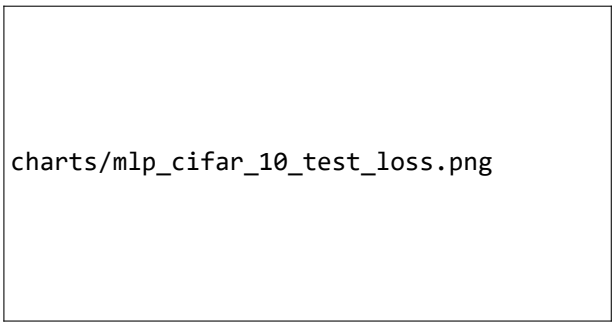

Figure 14: Testing loss of two MLP models on CIFAR-10 datasets

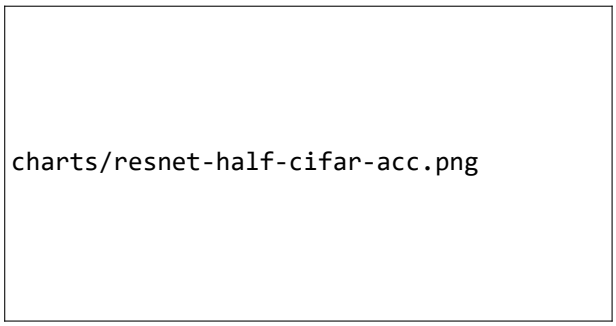

Figure 15: Training and testing accuracy of ResNet-18 with different training data amount. The "Half-ResNet-18" is trained with only 25000 training images.

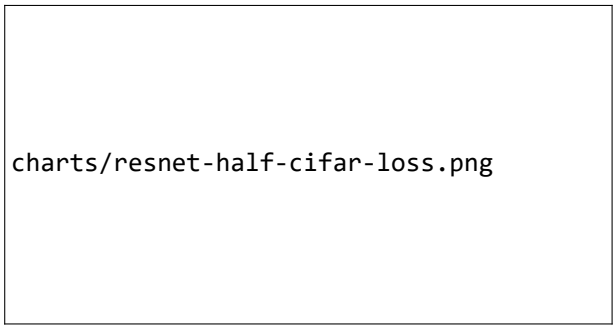

Figure 16: Training and testing loss of ResNet-18 with different training data amount. The "Half-ResNet-18" is trained with only 25000 training images.

Figure 17: Training and testing accuracy of MLP with different training data amount. The "Half MLP" is trained with only 25000 training images.

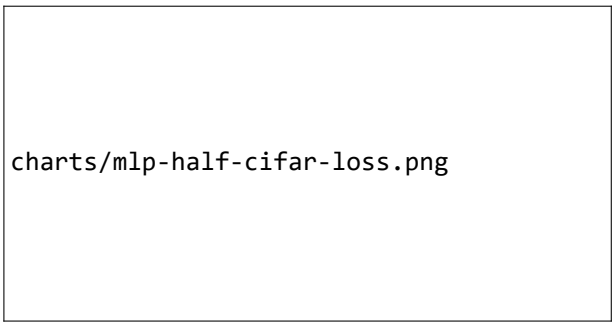

Figure 18: Training and testing accuracy of MLP with different training data amount. The "Half MLP" is trained with only 25000 training images.