Performance Analysis of Multilayer Perceptron and Convolutional Neural Network on classification of Image Data

Eric Shen, Xingyu Shen and Mingzhou Dong

***Abstract*— Nowadays digit recognition is becoming a heated topic. During this project, we mainly explore two machine learning models: Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN). The goal is to get accuracy of image which would be classified to the correct class in CIFAR-10 dataset.In addition, we also found the effect of hyperparameter adjustment on accuracy, and learned that after the adjustment, the accuracy of MLP and CNN was significantly improved, respectively, by 7% and 18%. The best result is from CNN model with hyper-parameter tuning, which achieves an accuracy of 71.40%.**

I. Introduction

Preliminaries:

MLP and CNN belong to the classification model in machine learning. A multilayer perceptron is a neural network connecting multiple layers in a directed graph, which means that the signal path through the nodes only goes one way. Each node, apart from the input nodes, has a nonlinear activation function. MLP uses backpropagation as a supervised learning technique. MLP is a deep learning technique since there are multiple layers of neurons in it .[1] A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks. Like other kinds of artificial neural networks, a convolutional neural network has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. This simulates some of the actions in the human visual cortex.[1]

We implement MLP by hand, import CNN from torch to classify our image dataset and use keras for hyperparameter tuning. We also try some other machine learning models like KNN and SVM. In this project, we try to predict whether a image is a plane, car, bird, cat, beer, dog, frog, horse, ship or truck.

Project goal:

The goal of this project is to find the best model of predicting image dataset and the relationship between parameters. The whole process can be divided into five parts: first, we preprocessed the datasets by reshape and normolize input data and expected output. Second, we implemented the MLP, CNN, KNN and SVM models on training set. Third, we predict the both train and test set. Fourth, we use hyperparameter search to find the model with highest accuracy on training set. Last, we use the model with highest training accuracy to predict both train and test set then see the improvement of hyperparameter tuning and acheieve the best machine learning algorithm in terms of image prediction.

Important findings:

After applying all models on CIFAR-10 dataset, we compared the accuracy result derived from those models and find out that CNN is the best model with 71.4% accuracy on unseen test data. We achieved this by doing 10-fold cross-validation, 5 convolutional layers of zero padding, 2 of 2X2 pooling layers, relu activation function, 0.01 learning rate, 5 fully connected layers, and Optimization Algorithm of Stochastic gradient descent.

II. Background and Relatedwork

Digit recognition system is the working of a machine to train itself or recognizing the digits from different sources like emails, bank cheque,

papers, images, etc or in different real-world scenarios for online handwriting recognition on computer tablets or system, recognize number plates of vehicles, processing bank cheque amounts and so on...[2].

Multilayer perceptron (MLP) and Convolutional Neural Networks (CNN) have recently gained popularity at image classification tasks since the initial work by Yann LeCun et al(LeNet) involving the classification of handwritten digits and Alex Krizhevsky's AlexNet which obtained a record performance metrics on the CIFAR dataset at ImageNet classification competition in 2012.

MLPs are very useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation. MLPs are also universal function approximators as shown by Cybenko's theorem. They can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical. MLPs make good classifier algorithms. It was a popular machine learning solution, finding applications in diverse fields such as speech recognition, image recognition and machine translation software. However, thereafter faced strong competition from much simpler support vector machines(SVM). Interest in backpropagation networks returned due to the successes of deep learning.[3]

The major highlight of CNNs is their ability to extract higher level representation of image features without feature engineering, a manual and expensive process that uses domain knowledge to create features for training in machine learning algorithms. CNNs comprise several learnable filters that convolve with input images at specified strides. Another major advantage of CNNs are their ability to reduce the numbers of network parameters and consequently the computational burden, while still attaining increased performances.

III. Datasets

In this project, as Fig.1 shown, the provided CIFAR-10 dataset for training consists of 50000 images, each of them contains one of the unique 10 classes [0,1,2,3,4,5,6,7,8,9], represent as 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck' respectively, see Fig.2. Each class has exactly 5000 instances, giving us the perfect balance to train a model. The test set consists of 10,000 images in the same format.

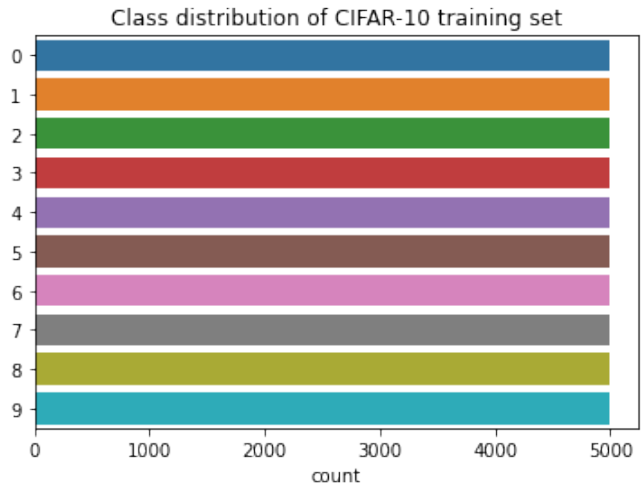

Fig. 1: Proportion of classes in CIFAR-10

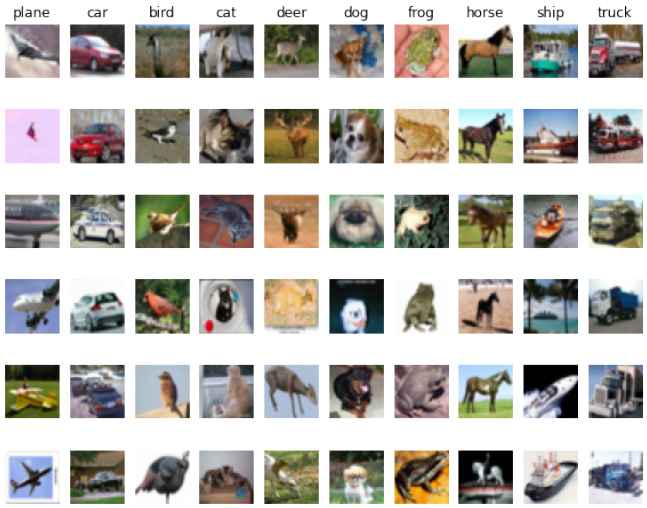

Fig. 2: Sample instance per class

We download data from scratch and use pickle to unpickle the data and merge into 5 training files each contains 10000 images so that the training set contains 5 files and the test set contains 1 file. For training set, we tranform each image into 1x3072 arrays as input data and 1x10 array as expected output by one-hot encoding for MLP and 32x32x3 matrix as input data and 1x10 array as expected output by one-hot encoding for CNN. Then we normalize input data by dividing 255 which helps to remove distortions caused by lights and shadows in an image.

IV. Proposed Approach

A. Model Choice

We have applied different classification algorithms including LinearSVM, KNN, CNN and MLP. In this section, we will discuss them one by one.

1) Multilayer Perceptron:: Multilayer Perceptron (abbreviated MLP) is the configuration of the former artificial neural network , mapping a set of input vectors to a set of output vectors. MLP can be regarded as a directed graph, composed of multiple node layers, each layer is fully connected to the next layer. Except for the input node, each node is a neuron (or processing unit) with a nonlinear activation function. A supervised learning method called back propagation algorithm is often used to train MLP. The MLP is perceptron promotion, can not overcome the perception of linearly inseparable weakness identifying data. Here is a basic explaination of MLP:

$$f(b + \sum_{i=1}^{n} x_i w_i) \tag{1}$$

b donates bias, x is the input to neuron, w is weights, and n is the number of inputs from the incoming layer.

a) Back propagation: The learning process of BP algorithm consists of forward propagation process and back propagation process. In the forward propagation process, the input information passes through the hidden layer through the input layer, is processed layer by layer and passed to the output layer. If the expected output value cannot be obtained in the output layer, the sum of the square of the output and the expected error is taken as the objective function, which is transferred to the back propagation, and the partial derivative of the objective function to the weight of each neuron is obtained layer by layer to form the target The gradient of the function to the weight vector is used as the basis for modifying the weight. The learning of the network is completed during the weight modification process. When the error reaches the expected value, the network learning ends.

b) Mini-batch gradient descent: Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.Implementations may choose to sum the gradient over the mini-batch which further reduces the variance of the gradient. Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.[4]

2) Convolutional Neural Network:: Convolutional Neural Network (CNN) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNN generally is consist of convolutional, pooling, normalizing, fully connected (FC) and accelerating layers. Our CNN is built by hand from scratch, which contains 4 layers with sigmoid activation function at first. The number of layers, the number of units on each layer and the activation function can be modified to achieve the best accuracy.

3) LinearSVM:: Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification analysis. LinearSVM is one of SVM algorithms to solve multi-class classification problems.[5]

4) KNN:: K-nearest neighbors algorithm (KNN) is a nonparametric method used for classification. The only parameter in KNN is the number neighbors(K). We tune different values of K in the code.

B. Process

1) Multilayer Perceptron: When preparing for MLP with 2 hidden layers of 128 nodes and 1 output layer of 10 nodes, we defined the activation function Sigmoid, forward backward layers and mini batch gradient descent for back propagation. The process of neural network starts with initializing layer size, weight and deviation. Forward pass refers to the calculation process, in which variables are propagated forward through the network. It traverses all neurons from the first layer to the last layer. In the backward transfer, the three parameters (layers, units and activation function) remain unchanged as the forward path.

Backward refers to the process of calculating the weight change (actual learning), using the mini batch gradient descent algorithm in the project. In this process, the network is trained: the calculation of the partial derivative loss function is for each property of the chain rule inherited from above by using induction. Then the weights and gradient deviations are updated, the learning rate involved is fixed at 0.01 and the epoch is 30.

2) Convolutional Neural Networks: The CNN model is learned from a PyTorch tutorial. It uses a package called torchvision which has a data loader for CIFAR10, which is used for this modelbuilt section. This model consists of an input layer with two combinations of one 2D Convolutional filter layer with ReLU activation function. The two convolutional layers 'conv1' and 'conv2' both apply a 5*5 kernel with stride=(1, 1) that is the kernel for MaxPool has size of 2 with stride=2, while the kernels for the later two are 3*3. Since additional depth was added, the kernel sizes had to be decreased from 5x5 to 2x2 or 3x3. We used ReLU as an activation function because it helps speed up gradient descent as opposed to a tanh or sigmoid function. Cross Entropy is applied as the loss function for it is popular for classification problems.

Hyperparameter tuning

CNN default model accuracy by class on testset	
Class	Accuracy
plane	66%
car	47%
bird	39%
cat	21%
deer	59%
dog	56%
frog	48%
horse	54%
ship	74%
truck	72%

After we find base models accuracies, we analyze results. As Fig.4 shown, 'cat' class has lowest accuracy 21%. We need to improve lower classes accuracy to overall result. For hyperparameter tuning, we use data from keras dataset, then reshape and normalize the input data and expected output for both training and testing sets as before.

V. Results

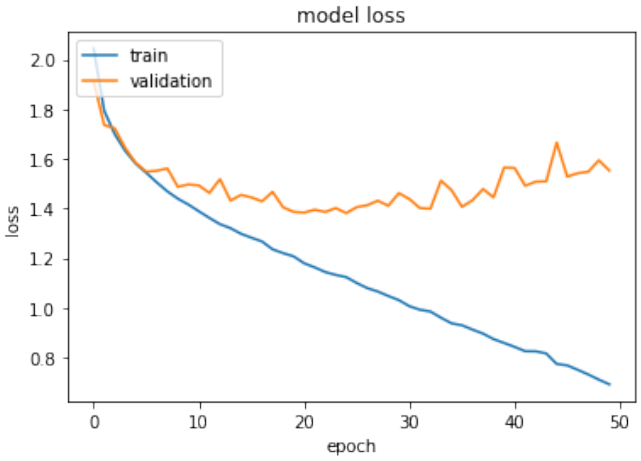

Fig. 3: Model loss per epoch for MLP defaut model

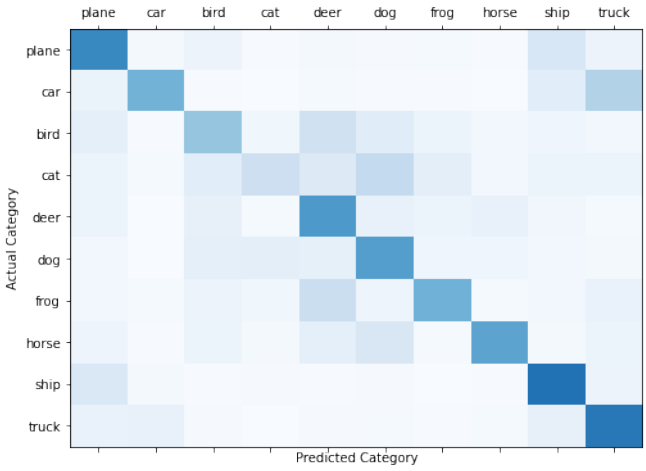

Fig. 4: Visualization of CNN predict accuracy by class on test set

In terms of accuracy on training and test set, the performance of each applied algorithm is shown in Table I.

We start by comparing accuracies of default model, the best result is CNN from PyTorch version whose accuracy is 54% on test set. We also achieve an accuracy of 45% on hand-written MLP with 2 hidden layers of 256 and 256 nodes. Compared with other models, KNN only achieves 10 percent accuracy. Even with hyperparameter tuning, its accuracy is not improved at all. Thus, we can conclude that KNN model fails to predict CIFAR-10 dataset.

Accuracy on CIFAR-10 dataset

	default models (Training Acc / Test Acc)	hyperparameter tuning models (Training Acc / Test Acc)
MLP	0.484 / 0.450	0.650 / 0.530
CNN	0.560 / 0.540	0.650 / 0.530
SVM	1.000 / 0.279	0.289 / 0.306
KNN	0.347 / 0.100	0.362 / 0.100

TABLE I

We also create an epoch vs loss graph for MLP. An Epoch is when an entire dataset is passed forward and backward through the neural network only ONCE. Thus, as more and more epoch(data) pass forward and backward through the neural network, the training loss decreases and training accuracy increases. However, the training loss is not continuing decreasing, as the model becomes overfitting. As Fig.3 shown.

After hyperparameter tuning, for MLP, as the change of number of layers, nodes of each layer and drop out, the accuracy increases by 8% on test set to 53% and by 16.6% in predicting training set to 65%. We notice that the number of layers and nodes per layer have no direct relationship with predicting unseen data. We set hidden layer parameter between 0 to 10, and nodes parameter between 128 to 256, it gives us the best model of 7 hidden layers and 148 nodes per layer. In addition, there is a significant increase in accuracy in CNN as we introduce different pooling layers, fully connecting layers and hyperparameter tuning, it increases the accuracy by 17 percent for test set. It is the highest accuracy for predicting test-set for all of the models we choose. See Table I.

We also notice that there is an over-fit in SVM and CNN, as they achieve or almost achieve 100% accuracy for predicting training-set, but much less for predicting test-set.

VI. Conclusion and Discussion

In this project, We evaluate these models according to their recognition accuracy. Besides, we also experiment on impacts on tuning different parameters.

After doing all experiments, we find CNN model has the best performance on predicting multi-image classe on this CIFAR-10 dataset. It gives us the best accuracy of 71.4%.

However, we have to admit that due to computer performance and other reasons, some of our experiment are not fully developed. For example, we did not split validation set and apply cross validation on it.

For future work, more experiments with regularization and additional hidden layers could be done to solve the problem of overfitting. Furthermore, we want to explore more types of structures of neural networks as we can study more CNN models such as ResNet and other categories of recurrent neural networks like LSTM.

VII. Statement of Contributions

•Xingyu Shen: CNN-pytorch, SVM, KNN and their parameter tuning, Report

•Eric Shen: CNN-pytorch, MLP hand-writing, Report

•Mingzhou Dong: MLP hand-writing, MLP and CNN hyperparameter tuning, Report

References

[1] H. Beniwal, "Handwritten digit recognition using machine learning," 2018.

[2] H. Beniwal, "Multilayer perceptron (mlp)," *https://www.techopedia.com/definition/20879/multilayer-perceptron-mlp*, 2018.

[3] Wikipedia, "Multilayer perceptron," *https://en.wikipedia.org/wiki/Multilayer*$_{perceptron}$.

[4] A. G. I. to Mini-Batch Gradient Descent and H. to Configure Batch Size, "J. brownlee," 2020.

[5] V. V. N. Cortes, Corinna, "Support vector networks," in *Machine Learning*, pp. 273–297, 1995.