Student Name: Nhat Pham (NUID: 002958106)

CIFAR-10 Image Classification Using Support Vector Machine and Convolutional Neural Network

I/ Introduction

The CIFAR-10 dataset contains 60000 tiny color images with the size of 32 by 32 pixels. The dataset consists of 10 different classes (i.e. airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck), in which each of those classes consists of 6000 images. There are in total 50000 train images and 10000 test images (1).

In this project, I will classify images from the CIFAR-10 dataset using two machine learning models we have discussed in class, Support Vector Machine (SVM) and Deep Convolutional Neural Network (CNN).

II/ Support Vector Machine

1. Model Overview:

Support Vector Machine is usually used for classification and regression analysis. Its core idea is the principle of structural risk minimisation, which in practice is to find a hyperplane that represents the maximum margin of separation between classes (2). SVM performs well in high-dimension space and data sets which have many attributes. Therefore, SVM can model complex image classification problems. While using non-linear classification in high-dimension space, there will be many support vectors, so the training speed is much lower than linear classifier. Therefore, I choose linear SVM to train CIFAR-10.

2. Model Architecture:

First, I split data into a training set (49,000), a validation set (1,000, a subset of training set), and a testing set (10,000). I also created a development set of size 500 which is a subset of training data and was used for development so my codes run faster. Then I preprocessed the data, including reshaping the image data into rows, normalizing the images by subtracting the mean image from training and testing data to have inputs that are (on average) centered around zero.

I created a function to calculate the multiclass SVM loss, implementing a vectorized version of the structured SVM loss. Next, I trained a linear classifier using stochastic gradient descent, which takes in training data, training labels, learning rate, regularization strength, number of iterations, batch size as input and output a list containing value of loss function at each training iteration.Then I ran a linear SVM model on training data and test its performance on the validation set. Since the accuracy was low (0.09), I used the validation set to tune hyperparameters (regularization strength and learning rate), experimenting with different ranges for the learning rates and regularization strength. For each combination of hyperparameters, I trained a linear SVM on the training set, computed its accuracy on the training and validation sets, and stored these numbers in the results dictionary. Finally, I used the model that results in the best validation accuracy and the LinearSVM object that achieved this accuracy to evaluate on the test set.

3. Results

When plotting the loss as a function of iteration number, I see a nicely decreasing loss curve. The trained model works well on the test datasets, which has 33.54% accuracy. Then I visualized the learned weights for each class (Fig 2).

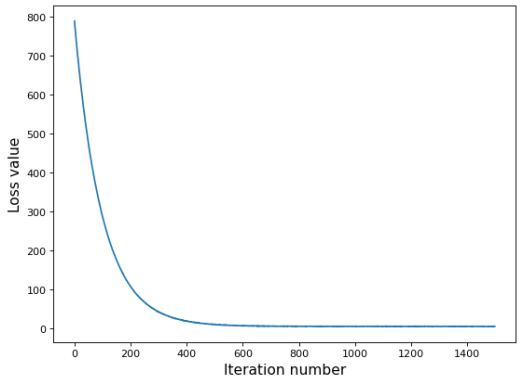

Fig.1: Loss vs iteration

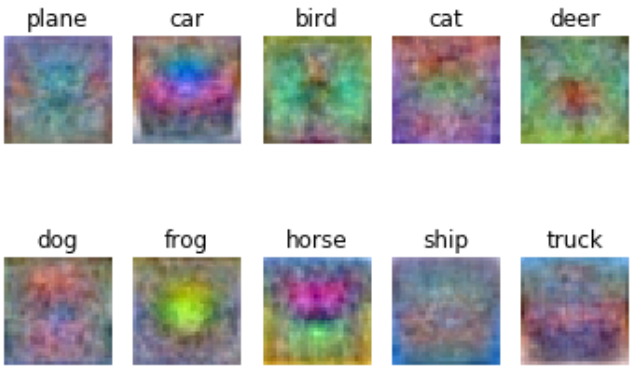

Fig.2: Visualized learned weights for 10 classes

The SVM weights look like the blurred outlines of a typical object belonging to the corresponding class. For example, the visualized weights for the car shows an outline of the front of a reddish car. As the score is the inner product between the sample and the corresponding weight, the weights should be more parallel to the represented sample, in order to get correctly labeled.

III/ Convolutional Neural Network

1. Definition:

Intuitively, a convolutional neural network is specialized to learn useful local correlations and combine features extracted in low level layers to support higher order learning. Besides using fully connected (FC) layers like a general feed-forward neural network, CNN also relies on several convolutional and pooling layers before FC layers. In short, a CNN learns spatial features in convolutional layers, down samples in pooling layers, and predict classes in the fully connected layer.

Because of its local connectivity and ability to use local structure, convolutional networks are usually favored over fully connected neural networks. Consider the CIFAR-10 dataset, where each has 32x32x3 = 3072 features, a common fully connected layer with 512 neurons would have 1,572,864 parameters. Using a convolutional layer consisting of, say, 512 filters of size 3x3, there will be only 4,608 weights to learn. To conclude, a convolutional network successfully utilizes information that exists in neighboring features, leading to reduction in parameter space and therefore is considered as an ideal classifier for image dataset (3).

2. Model set up:

Since there are 10 classes, we will now perform one hot encoding for the class element of each sample, transforming the integer into a 10 element binary vector. Since the pixel values for each image range from 0 (no color) to 255 (full color), some scaling will be required for modeling. I normalized the pixel values, rescaling them to the range [0,1]. This involves first converting the data type from unsigned integers to floats, then dividing the pixel values by the maximum value.

Next, I defined a neural network model. I applied the general architectural principles of the VGG models which achieved high performance in the ILSVRC 2014 competition (4). The architecture involves stacking convolutional layers with small 3×3 filters followed by a max pooling layer. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network such as 32, 64, 128, 256 for the first four blocks of the model. Padding is used on the convolutional layers to ensure the height and width of the output feature maps matches the inputs. Each layer will use the ReLU activation function and the He weight initialization. I will apply this architecture with 3 blocks. The feature maps output from the feature extraction part of the model were then flattened. I could then interpret them with one or more fully connected layers, and then output a prediction. The output layer had 10 nodes for the 10 classes and used the softmax activation function.

The model was optimized using stochastic gradient descent. I used a learning rate of 0.001 and a large momentum of 0.9, both of which are good general starting points. The model optimized the categorical cross entropy loss function required for multi-class classification and monitored classification accuracy on the test dataset.

3. Results:

The model had an accuracy of 73.71% with training time of 572 seconds. The model was able to learn the training dataset, showing an improvement on the training dataset that at least continued to 40 epochs. Reviewing the figures showing the learning curves, however, we see dramatic overfitting within the first 20 training epochs (Fig.3). This result suggests that the model is in need of regularization to address the rapid overfitting of the test dataset. More generally, the results suggest that it may be useful to investigate techniques that slow down the convergence (rate of learning) of the model.

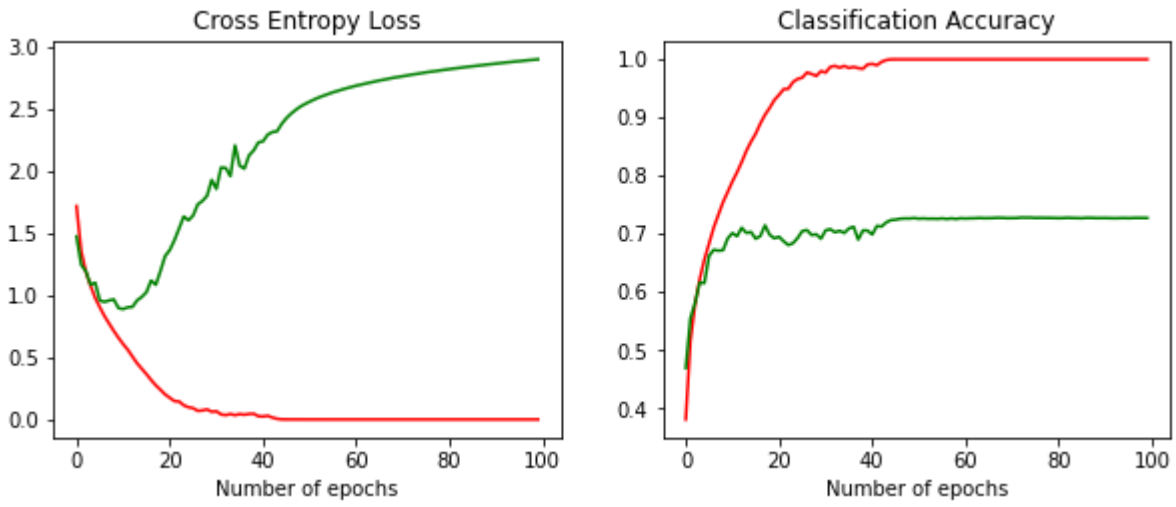

Fig.3: Cross Entropy Loss and Model Accuracy (Train (red) vs Test (green) Data)

IV/ Discussion

In conclusion we can see that the choice of classifier and pre-processing technique will differ significantly. By looking at the convergence speed, training the Linear SVM classifier took only 7.2

seconds, while training the CNN model took 572 seconds. However, the accuracy score was much higher for the CNN model (0.737) than for the SVM model (0.37). Hence, if an application requires extremely detailed results, and speed is not an issue then an algorithm such as the CNN will be best suited due to the much higher overall accuracy.

In the future, I would like to make some improvements to both models and compare the results again. Specifically, I would like to apply dimension reduction methods such as principal component analysis to the SVM model. For example, comparing model accuracy and training time when using different numbers of n components of PCA. For the CNN model, a lot of improvements can be implemented based on the baseline model described above. I have not tuned the hyperparameters of the learning algorithm, such as the learning rate, which is perhaps the most important hyperparameter. The results suggest that the model needs regularization to address the rapid overfitting of the test dataset. More generally, the results suggest that it may be useful to investigate techniques that slow down the convergence (rate of learning) of the model. This may include techniques such as data augmentation as well as learning rate schedules, changes to the batch size.

References:

1. CIFAR-10 and CIFAR-100 datasets. (n.d.). Retrieved May 6, 2022, from https://www.cs.toronto.edu/~kriz/cifar.html
2. *Support-Vector Machine*. Wikipedia. Retrieved May 6, 2022, from https://en.wikipedia.org/wiki/Support-vector_machine
3. Maladkar, K. (2020, November 21). *Overview of convolutional neural network in image classification*. Analytics India Magazine. Retrieved May 6, 2022, from https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/
4. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.