McGill

COMP 551 - Applied Machine Learning

Applied Machine Learning

Assignment 3

Group members
Dan Ning Yang
Richard Gao
Charles Huang

Group number:

May 30, 2020 (Winter 2020)

1 Abstract

We approached labelling the CIFAR image dataset using Multi-layer perceptrons and Convolutional Neural Networks via Pytorch. We found that for the LeNet convnet, increasing the training epochs did not have a great effect unless tuning the output channels, meanwhile for AlexNet and MLP, the number of epochs had a increasing effect on accuracy.

2 Introduction

The overall project task is to investigate the performance of a custom built MLP, LeNet CNN and AlexNet CNN on the CIFAR 10 dataset. By playing with the hyper parameters, layers and input/output channels we determined AlexNet as the highest performing net.

3 Related Works

[1] Yoonjin Park "Notes on AlexNet", 9 Nov. 2018,
https://www.yjpark.me/blog/jekyll/update/2018/11/09/notes-on-alexnet.html
Mentioned in the reference is an analysis of AlexNet and pytorch's implementation.

4 Data-sets and preprocessing

We use the CIFAR-10 dataset, which contains 50000 training images and 10000 testing images, each image is a 32*32 colour image, meaning each pixel has a R,G,B value. Therefore we have a total of 32*32*3 pixels. The target label is an int label between 0 and 9, indicating the class the image belongs to. For example if Y[4500] = 3, that means the 4501's image is a cat.

4.1 Multilayer Perception

We first need to "flatten" the image data, that means for each 3d array with dimensions 32*32*3, we need to flatten it into a 1d array of 3072 (32*32*3 = 3072). For the training input, we have a 2d matrix of N*D, and for the testing input, we have a 2d matrix of M*D, where N and M means the number of training data, and testing data, in our case 50000 and 10000, and D equal to 3072, number of pixels.
For the training and testing target labels, we create a 2d matrix of N*C and M*C, where C indicates the number of classes, in our case : 10. To fill out theses 10 variables, most of them will be 0, except the predicted/true label, which will be 1. For example, trainY[4500] = [0,0,0,1,0,0,0,0,0,0], since Y[4500] has label 3.

5 Proposed Approach

5.1 Multilayer Perception

Initialization: The model is initialized with a 1d array of size L. H indicates the number of hidden layers we use and the number of nodes in each layer. We use this to initialize the weights L+1 sized list named Ws. W_1 has a size of $D * H_1$, and W_{L+1} has a size of $H_L * C$, (D and C are simply the column numbers of training X and training Y), and the other W_l would be size of $H_{l-1} * H_l$. In our case, H array is [800, 200, 50], so Ws is a list of 4 matrices, with size 3072*800, 800*200, 200*50 and 50*10, we initialize all variables of Ws to be a random number between 0 and 0.1.

Fit: For our MLP model we use stochastic gradient descent and momentum to get the best weights. That means instead of using all datas (50000), we only use ¡ 50000 random data-points at a time (in our case x = 500), to get dWs (Explanation here) to adjust our weights. The disadvantage is that the cost might not always decreasing after each iteration, but the running time of each iteration is slightly faster. During each gradient descent process, we first find out the output of each node in its hidden layer and the predicted label $\hat{Y}$ using the current weights variables:

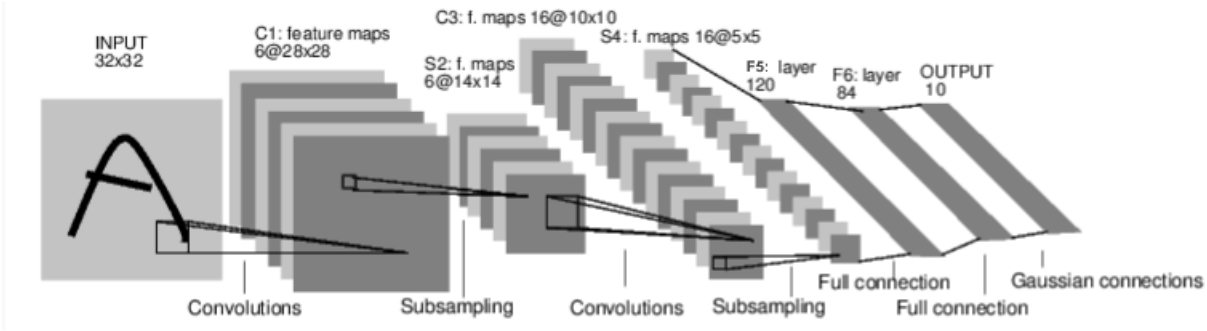
$$Z_l = Relu(W_l * Z_{l-1}) \vee l \in L$$
$$\hat{Y} = softmax(W_{L+1} * Z_L)$$

Then we calculate the derivative of $\hat{Y}$ and Z, where $dY = \hat{Y} - Y$ and $dZ_l = dZ_{l+1} * W_{l+1} \vee l \in L(dZ_{L+1} = dY)$, finally we compute the derivative of each weights where $dW_{L+1} = Z_L * dY$, and $dW_l = Z_{l-1} * dZ_l \vee l \in L$, using those dW, we compute the new weight by substracting dW with it's correspond weights, with a learning rate that decays after each iteration. The iteration stops when dw_{L+1} is smaller than eps.
Predict: We predict the testX with the weights already calculated. We then compare the results with the true testY.

5.2 Convolutional Neural Net

We trained individual models and evaluated the model performance at each epoch. This is according to the template LeNet model provided by the instructions, that is, a 2 layer convnet using max pooling and relu. The model takes in 3x32x32 images. The first layer consists of a 3 input channels and 6 output channels with 5x5 square convolution and 2x2 pooling. The second layer consists of 6 input channels and 16 output channels with 5x5 square convolution. Moreover, We used Stochastic gradient descent as well as Cross Entropy loss as our cost function during training. Furthermore, we played with the number of output channels in the second layers and found when the number of output channels increases, the overall accuracy would also increases.

Figure 1: Convolutional Neural Net

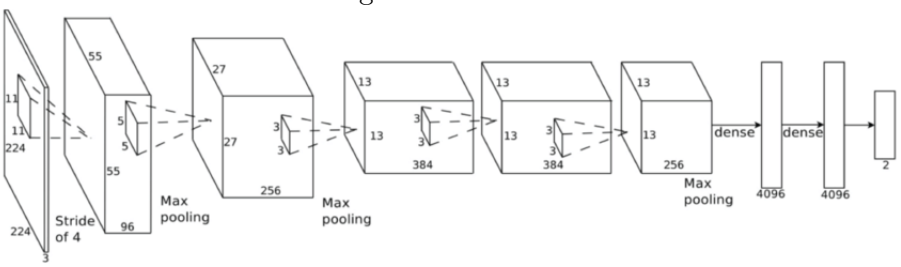


5.2.1 AlexNet

In addition to the two above models we also tested how an AlexNet-like setup performed with the CIFAR Data. The AlexNet model consists of 5 convolutional layers, using Relu as an activation function. Usually, the model takes in 224x224x3 images. The first layer consists of a 96 filters and kernel of size 11x11x3 with stride of 4 and padding of 2 and 2x2 pooling. This would mean that our image would be shrunk to a size of 3x3x96 after the first layer. Therefore

in order to feed our images to the model, we chose to change the first layer to a kernel of size 3 with stride 2 and padding 1 and 96 filters. We also played with the number of filters (64 in the first layer) and found no substantial differences in accuracy and left this alone. Resizing the images was also attempted, however this resulted in poor accuracy and we discontinued this approach.

Figure 2: AlexNet



6 Results

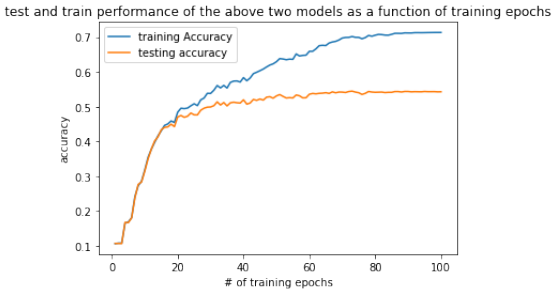
6.1 Multilayer Perception

The graph in **Figure 3** shows that the training accuracy can go up to 72% and the testing accuracy 54%. After 90 training epochs the accuracy flat-lines. We made 3 separate attempts of batch size 200, 500, 1000. We realized a bigger batch size gave a better accuracy, so the best option would be to have a batch size equal to the training set size. However the running time would be too slow. We also realized that more layers won't definitely give better result. By comparing with models with 2 layers, 3 layers and 4 layers, we can see that the best is the one with 3 layers, which is surprising.

Figure 3: Results MLP

Model	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Overall Accuracy
Attempt 1: MLP/SGD+Momentum, lr = 0.1, decay = 0.000001, max iter = 10000, minibatch = 500, momentum = 0.99, 3 layers [800, 200,50]	58.10%	64.40%	32%	31.20%	37.2	38%	61.40%	61.40%	69.20%	57.20%	51.04%
Attempt 2: MLP/SGD+Momentum, lr = 0.1, decay = 0.0000005, max iter = 5000, minibatch = 1000, momentum = 0.99, 3 layers [800, 200,50]	62.2	64.8	40.2	36.3	45.4	43.3	61	62	66.8	60.6	54.28
Attempt 3: MLP/SGD+Momentum, lr = 0.1, decay = 0.0000005, max iter = 5000, minibatch = 1000, momentum = 0.99, 4 layers [1000, 300 ,100,30]	61.8	63.1	41.3	36.1	40.9	40.7	58	57.4	67.6	57.7	52.46
Attempt 4: MLP/SGD+Momentum, lr = 0.1, decay = 0.0000005, max iter = 5000, minibatch = 1000, momentum = 0.99, 2 layers [500, 70]	58.6	64.2	31.7	32.4	42.5	38.4	63.5	60.3	69.2	59.1	51.99
Attempt 5: MLP/SGD+Momentum, lr = 0.1, decay = 0.000001, max iter = 10000, minibatch = 200, momentum = 0.99, 3 layers [800, 200,50]	53.4	59	27.4	24.8	31	38.5	60.9	56.3	63.7	51.7	46.67

Figure 4: Epochs and Training MLP



6.2 LeNet CNN

The graph in **Figure 5** shows that there is only an increase of accuracy during the first two epochs of Convolutional Neural Network. The accuracy almost remains constant after 2 epochs; and the accuracy slight decrease after 8 epochs. This can be explained by the overfitting when the number of epochs increase. Moreover, we also tested the accuracy of the model with a variate number of output channels in the second layer. From **Figure 6**, we can find when the number of output channels increases, the overall accuracy also increases. Each channel in the neural network responds to a different sets of features. Therefore, once the output channels increases, the network would have more information about the input images. This can explain the reason that the accuracy of the model increase along with the dimension of the output channels of the layers.

Figure 5: Epochs and Training CNNs

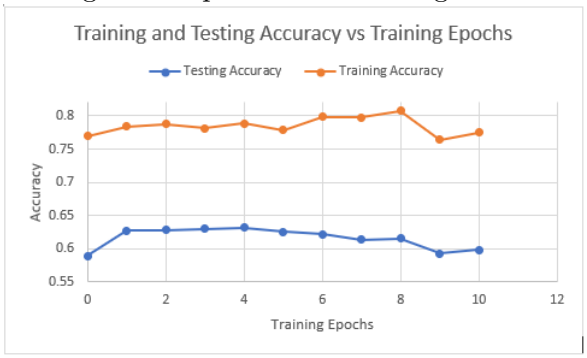


Figure 6: CNN Results

Model	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Overall Accuracy
LeNet Cross Entropy/SGD + Momentum:lr=0.001, momentum=0.9m, 8 output channel	39	66	32	52	34	27	46	72	75	63	51
LeNet Cross Entropy/SGD + Momentum:lr=0.001, momentum=0.9m, 16 output channel	67	57	29	35	42	42	76	60	53	78	54
LeNet Cross Entropy/SGD + Momentum:lr=0.001, momentum=0.9, 32 output channel	57	75	50	40	55	55	78	73	78	69	63

6.2.1 AlexNet CNN

The graph in **Figure 7** shows substantial growth in accuracy during the first five epochs of AlexNet. Over time, the accuracy almost rivals that of the pre-trained AlexNet model provided by Pytorch. At around 5 epochs the training accuracy increases, but the testing accuracy remains the same and flatlines. Compared to a 2 epoch AlexNet model, 10 epoch AlexNet categorizes birds and planes better as a whole, showing that these two images were very difficult to categorize and required more training for the model to distinguish them.

Figure 7: Epochs and Training AlexNet-CNN

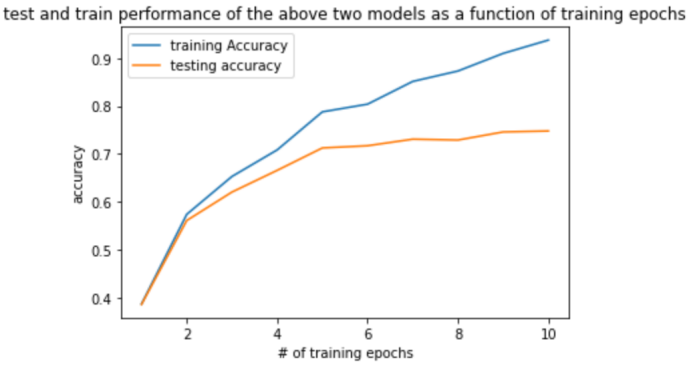


Figure 8: AlexNet Results

Model	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Overall Accuracy
Modified AlexNet / SGD + Momentum: lr=0.001, momentum=0.9 + 5 Layers (2 Epochs)	58	65	39	23	47	63	61	62	84	63	56
Modified AlexNet / SGD + Momeentum: lr = 0.001, momentum=0.9 + 5 Layers (10 Epochs) Pretrained	79	82	59	53	79	72	77	78	89	83	75
64-Modified AlexNet / SGD + Momeentum: lr = 0.001, momentum=0.9 + 5 Layers (10 Epochs)	82	86	65	52	72	66	85	76	84	75	74
96-Modified AlexNet / SGD + Momeentum: lr = 0.001, momentum=0.9 + 5 Layers (10 Epochs)	79	85	69	55	65	62	63	83	74	87	74

7 Discussion and Conclusion

The LeNet CNN's training accuracy did not increase much with the number of epochs (unlike the other two models) unless output channels were altered, the best of which was a 32 output channel model. This likely was due to the fact that the more channels we have, the more optimized our model is able to become. Changing the number of filters/output input channels for the first layer of AlexNet did not affect accuracy, however it did change accuracy of certain categories. This may be an area of study in the future.

8 Statement of Contributions

Richard Gao: AlexNet.
Charles Huang: Convolutional Neural Network.
Dan Ning Yang : MLP.