Deep Learning Lab: Convolutional Neural Networks

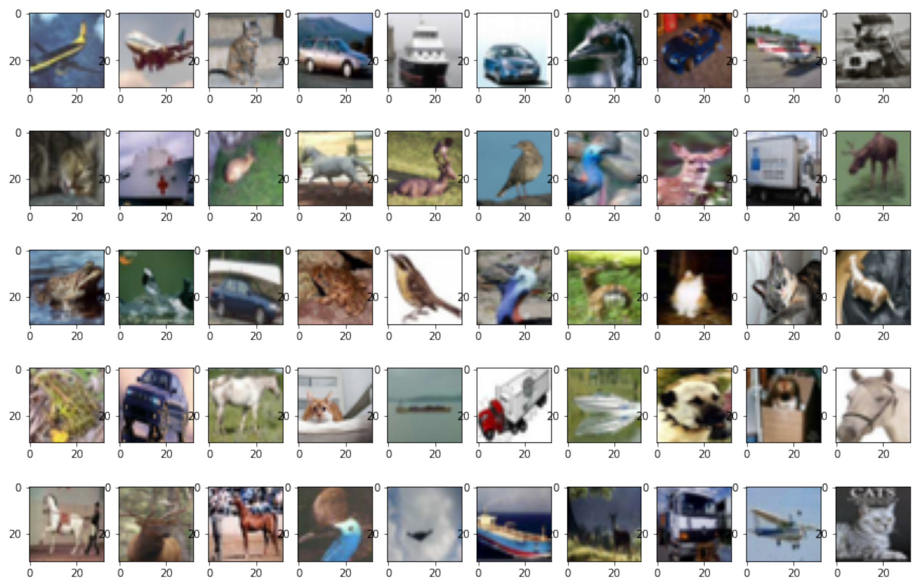
Anthony Bugatto

13 November 2022

1 Loading the CIFAR-10 Dataset

1.1 Loading the Dataset

The CIFAR-10 dataset is a set of 60000 photos in grouped by ten different classes of objects. In order to load the data, $torchvision.datasets.CIFAR10()$ function is used. Because the data is needed in batches during training pytorch has a DataLoader class that allows loading of the data in batches from file. Each time a dataloader is called the data is shuffled to prevent bias. A sample of the data takes the form of:



1.2 Normalizing the Dataset

In order to make sure the model learns the features of any image in an unbiased manner we must normalize the dataset, meaning that we want every image to be of a normal distribution where $mean = 0$ and $std = 1$. Pytorch has a transform function that allows normalization via the equation:

$$z = \frac{x - \mu}{\sigma}$$

and using the CIFAR-10 distribution:

$$\mu = [0.49139968, 0.48215841, 0.44653091]$$

$$\sigma = [0.24703223, 0.24348513, 0.26158784]$$

The transform can be input into the data loading function so the input data can be normalized. Note: this means that the network only works well on normalized data.

1.3 Creating a Valdation Set

Creating a validation set is relatively easy by using the $torch.utils.data.SubsetRandomSampler$ function. We use a validation set size of 1000 and training set of 49000.

2 Model

The image classification model takes the form of a convolutional network with four convolutional layers, two max pooling layers, a fully coneccted layer, and a softmax layer. The layers are designed as:

- Convolutional layer with 3x3 filter size, 1x1 stride, and 32 filters
- Convolutional layer with 3x3 filter size, 1x1 stride, and 32 filters
- Max Pooling layer with 2x2 filter and 1x1 stride
- Convolutional layer with 3x3 filter size, 1x1 stride, and 64 filters
- Convolutional layer with 3x3 filter size, 1x1 stride, and 64 filters
- Max Pooling layer with 2x2 filter and 1x1 stride
- Fully Connected Layer with 512 hidden units
- Softmax layer with 10 outputs corresponding to CIFAR-10 classes

3 Training

3.1 Training Pipeline

The training pipeline uses the model and trains it on CIFAR-10 images using the cross entropy loss:

$$Loss = \sum_{i=1}^{outputsize} y_i^* log(y_i)$$

and stochastic gradient descent with momentum:

$$w = w - [momentum\nabla_{t-1} + rate\nabla_w]L(W, X, y)$$

3.2 Training Results

The training starts by rapidly reaching 60%-70% validation accuracy but then slows down as the accuracy starts to plateau. This signifies that the model is starting to overfit, evidenced by the loss continuing to decrease at the same rate despite the decrease in accuracy.

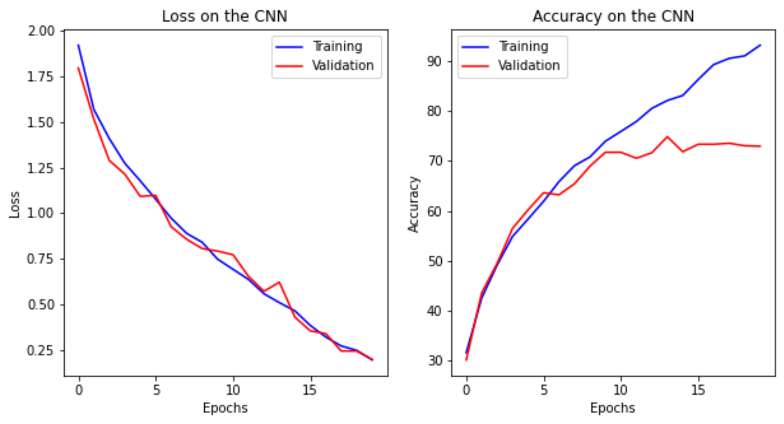


Figure 1: Dropout Model with p = .2 and 69.18% test accuracy

3.3 Dropout

Dropout is a layer that randomly drops activations with the probability p in each iteration. This increases generalizability by allowing the model to effectively be an ensemble of the different dropout models, acting as a form of regularization. The layers are now designed as:

- Convolutional layer with 3x3 filter size, 1x1 stride, and 32 filters

- Convolutional layer with 3x3 filter size, 1x1 stride, and 32 filters
- Max Pooling layer with 2x2 filter and 1x1 stride
- Dropout layer with probability of 35%
- Convolutional layer with 3x3 filter size, 1x1 stride, and 64 filters
- Convolutional layer with 3x3 filter size, 1x1 stride, and 64 filters
- Max Pooling layer with 2x2 filter and 1x1 stride
- Dropout layer with probability of 35%
- Fully Connected Layer with 512 hidden units
- Dropout layer with probability of 35%
- Softmax layer with 10 outputs corresponding to CIFAR-10 classes

After using dropout the loss converges slower but the model tends to generalize on the validation set better. Trying the p values of .2, .5, and .8 we see that dropout increases the final accuracy noticeably, three percent higher with p = .2. Additionally we see that the overfitting with p = .2 is significantly decreased but as p goes towards .5 and more so at .8 we see the model underfitting. This is evidenced by the loss decrease, the drop off in training accuracy, and the validation accuracy being greater than the training acuracy. This is likely due to the model over-generalizing due to the difference between the ensemble models in the same way that an over-regularized model would perform worse.

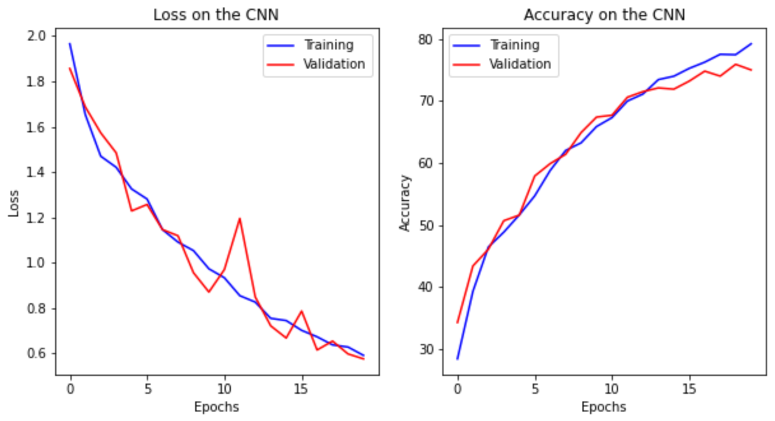


Figure 2: Dropout Model with p = .2 and 73.11% validation accuracy

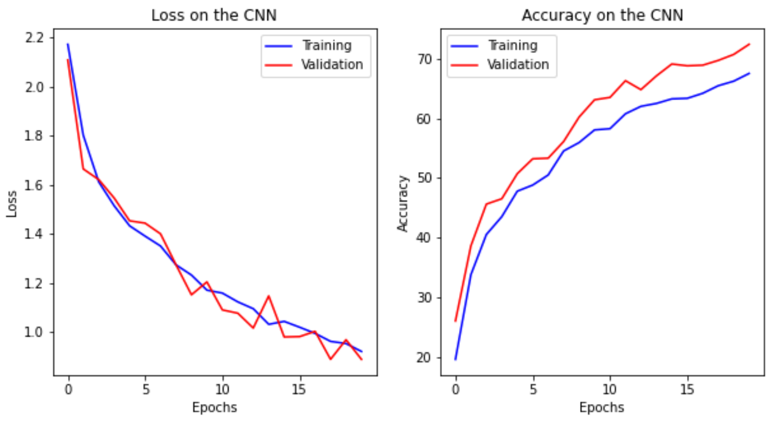


Figure 3: Dropout Model with p = .5 and 71.84% validation accuracy

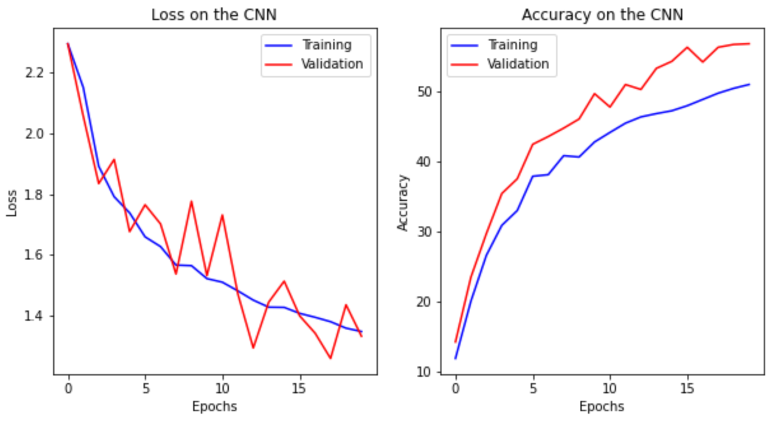


Figure 4: Dropout Model with p = .8 and 56.09% validation accuracy

3.4 Best Dropout Model Results

With the p = .2 model we display some test images, which show that our model does indeed classify the images with approximately 73% accuracy.

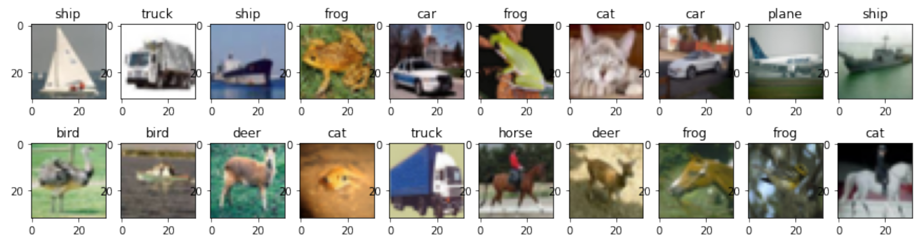


Figure 5: Test image classifications for the p = .2 dropout model

3.5 Final Model Results

The first optimization I did was train models until I found the best learning rate and momentum. This turned out to be:

$$rate = .0012$$

$$momentum = .085$$

and allowed the model to gain a few percent in accuracy before plateauing. Furthermore it was found via [2] that setting different dropout values for each part of the network increases the accuracy significantly. This was experimentally verified in the final model, which achieved 76.5% accuracy by epoch 20 and 80.4% accuracy by epoch 40. We can see that the model doesn't start to plateau until around 78%, while peaking over the 80% threshold. Generalizing well due to the new dropout scheme, the test accuracy was 78.8%.

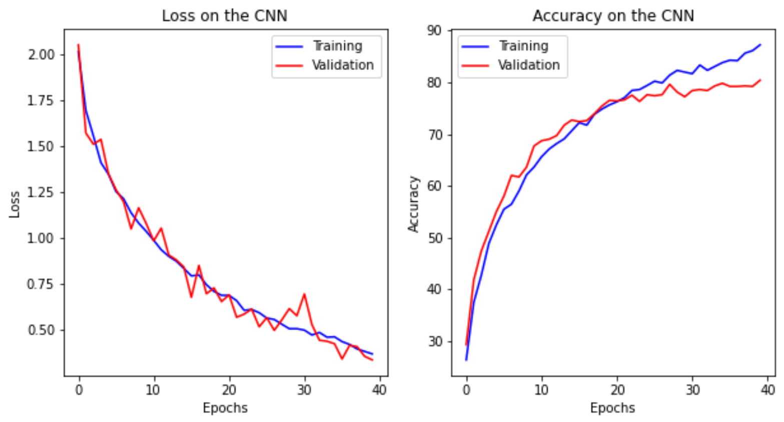


Figure 6: Final Model with 80.4% validation accuracy

4 Questions

4.1 When (for what purpose) do you use softmax activation?

The softmax layer is used as the last layer of the network in order to turn the values into a probability distribution.

4.2 Here we used a non-zero momentum in the SGD optimizer. In this case, the optimizer accumulates a certain quantity with a decay factor specified by the momentum value. Name this quantity. Hint: read the PyTorch documentation of torch.optim.SGD.

The momentum SGD optimizer accumulates velocity with the momentum parameter.

4.3 Here we used 2D convolution for images. Name one example type of data for which you may want to use 1D convolution

1D convolution could be useful for time series data because a CNN could potentially be used to find a signal processing filter.

4.4 Test time behavior of dropout is different from its behavior during training. Describe both behaviors (max two sentences). Hint: you can read the PyTorch documentation of nn.Dropout.

At training time dropout will act as a regularization technique via ensemble while at test time the removed nodes will decrease the accuracy of the predictions.

5 Citations

1. My report from last year, from which some of the explanations are based on.

2. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. R. (2012, July 3). Improving neural networks by preventing co-adaptation of feature detectors. arXiv.org. Retrieved November 13, 2022, from https://arxiv.org/abs/1207.0580