Programming Assignment 1 Part 2

Katarina Vuckovic
CAP5415 Computer Vision
Oct 24, 2021
Instructor Dr. Yogesh Singh Rawat

I. Convolutional Neural Network (CNN) Classification

A. Introduction

The purpose of this part of the assignment is to train and test the performance of different CNN architectures for image classification. A total of five architectures (models) are considered, starting with the simplest model and increasing the complexity for each subsequent model.

B. CNN Architecture

The architecture for each model is summarized as follows:

- **Model 1**: One fully connected (FC) layer with 100 hidden layers followed by a sigmoid activation function. Trained using stochastic gradient descent (SGC) with learning rate (α) of 0.1 for a total of 60 epochs, a mini-batch of 10, and no regularization.
- **Model 2**: Two convolutional (ConV) layers followed by a FC layer from model 1. Each convolution layer is followed by sigmoid activation function and a maximum pooling (MaxPool). The parameters for the ConV layers are 40 kernels, stride of 1x1, and kernel size of 5x5. The MaxPool is over a region 2x2.
- **Model 3**: In model 2, the sigmoid function is replaced with ReLU function and with new α of 0.03.
- **Model 4**: In model 3, a second FC layer is added with 100 hidden layers.
- **Model 5**: In model 3, the number of hidden layers is increased to 1000. Regularization using Dropout with rate of 0.5 is added between the FC layers. The total epochs is reduced to 40.

All five models have a final FC layer that reduces the output from the number of hidden layers (100 or 1000) to the number of classes (10). For the training and testing *CrossEntropyLoss* Pytorch function is applied. The function combines the LogSoftMax and the negative log likelihood loss (NLLLos). Softmax is required for multi-class classification and NNLOS is the loss function that measures the training and testing accuracy.

C. Dataset

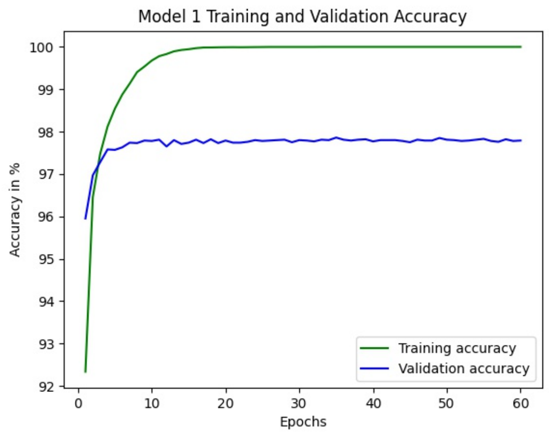
The project uses the NMIST digit dataset for evaluation. The dataset consists of images with digits from 0 to 9 (i.e. 10 classes). The size of the images are 28x28 and the total number of samples in the dataset is 6000 training samples and 1000 testing samples. The NMIST dataset is avaialbe from the *torchvision* library.

D. Results and Discussion

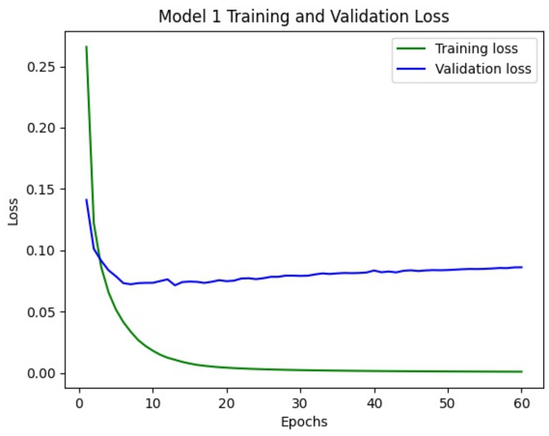
The results for the five models are summarized in Table I. The table shows the accuracy and the number of epochs at which the model converges. As may be seen from the table the accuracy increases with each subsequent model as the complexity of the model increase. Comparing model 2 to model 3, we observe that replacing the sigmoid activation function with the ReLU function improves the performances. Next, adding a FC layer in model 4 also increase the performance and finally, increasing number of hidden layers and adding dropout also improves the accuracy. However, since the accuracy of model 2 is already at 99 %, the improvements between model 2 and 5 is only a fraction of a percent. Depending on the application of the object detection, this improvement may be negligible. However, if the application requires nearly perfect accuracy, then these improvement may be significant. Furthermore, NMIST is a relatively simple dataset; hence, it does not require a complex CNN to achieve high accuracy. However, if a more complex dataset was trained using the same architectures, a more significant improvement could be observed between the models. Regarding the third columns in Table I, the number of epochs required for the network to converge is significantly less than the number of epochs that the network is trained on. The number of epochs over which the network is trained on has to be large enough to ensure conference. However, selecting an arbitrarily large number increases the computation time. Fig. 1-5 show the training and testing accuracy in a), while b) shows the training and testing loss over all the epochs.

TABLE I: Accuracy of Models

Model	Accuracy (%)	No. of epochs to converge
1	97.86 %	35
2	99.23 %	37
3	99.33 %	13
4	99.38 %	31
5	99.47 %	32

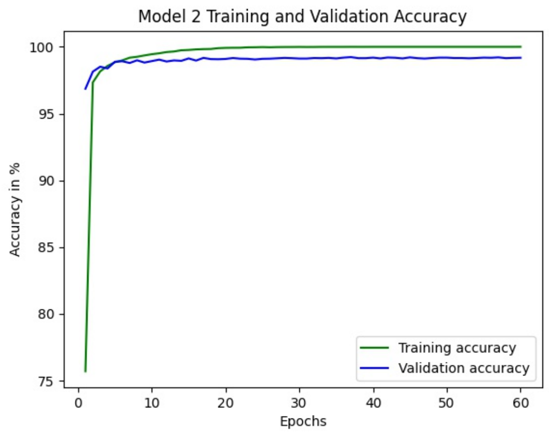


(a) Training vs. validation accuracy at every epoch.

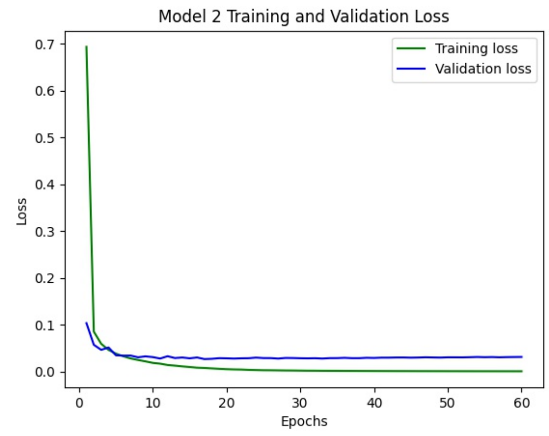


(b) Training vs. validation accuracy at every epoch

Fig. 1: Results for Model 1

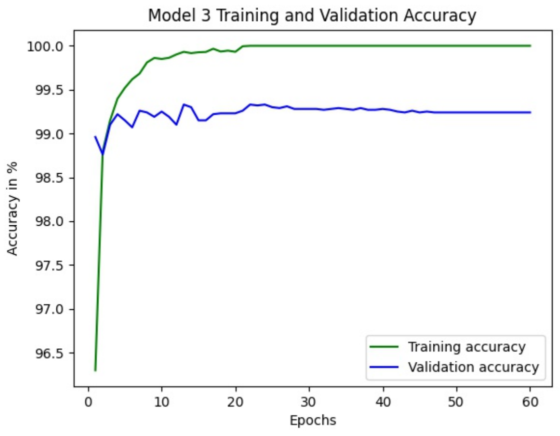


(a) Training vs. validation accuracy at every epoch.

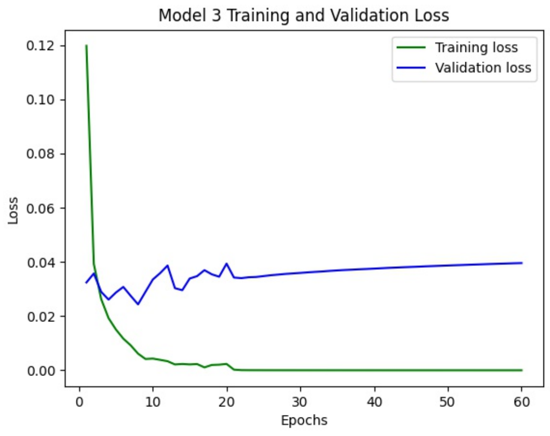


(b) Training vs. validation accuracy at every epoch

Fig. 2: Results for Model 2

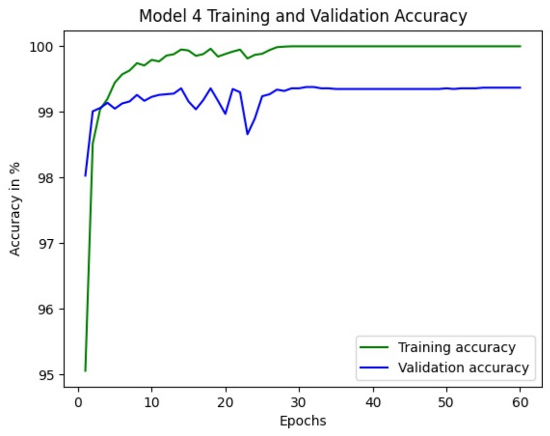


(a) Training vs. validation accuracy at every epoch.

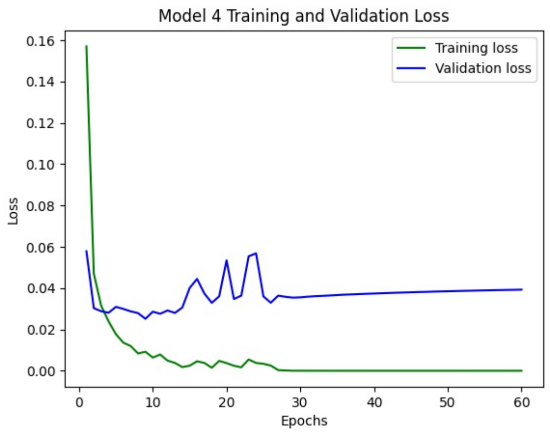


(b) Training vs. validation accuracy at every epoch

Fig. 3: Results for Model 3

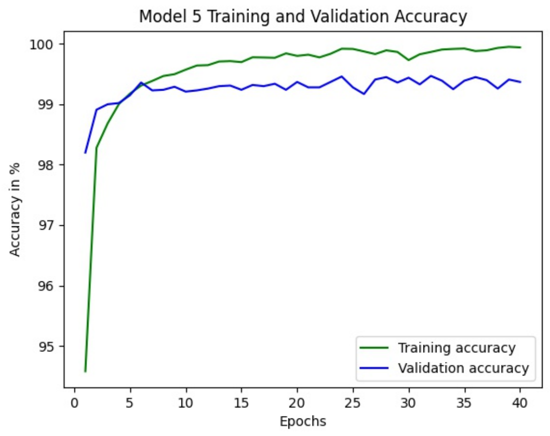


(a) Training vs. validation accuracy at every epoch.

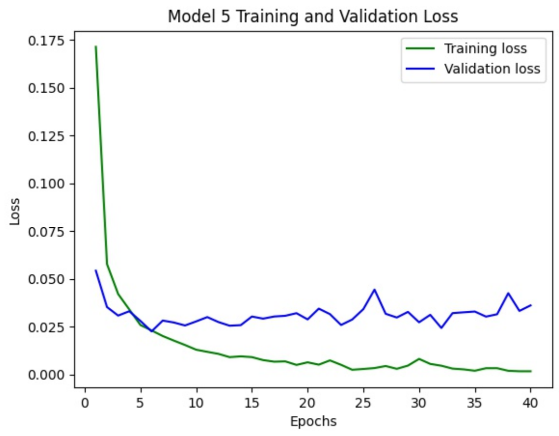


(b) Training vs. validation accuracy at every epoch

Fig. 4: Results for Model 4



(a) Training vs. validation accuracy at every epoch.



(b) Training vs. validation accuracy at every epoch

Fig. 5: Results for Model 5