Deep Learning Foundation Nanodegree Program

Part.3 Convolutional Networks

Deep Learning for Cancer Detection with Sebastian Thrun

Mini Project: Dermatologist AI

Takuma Kawahara

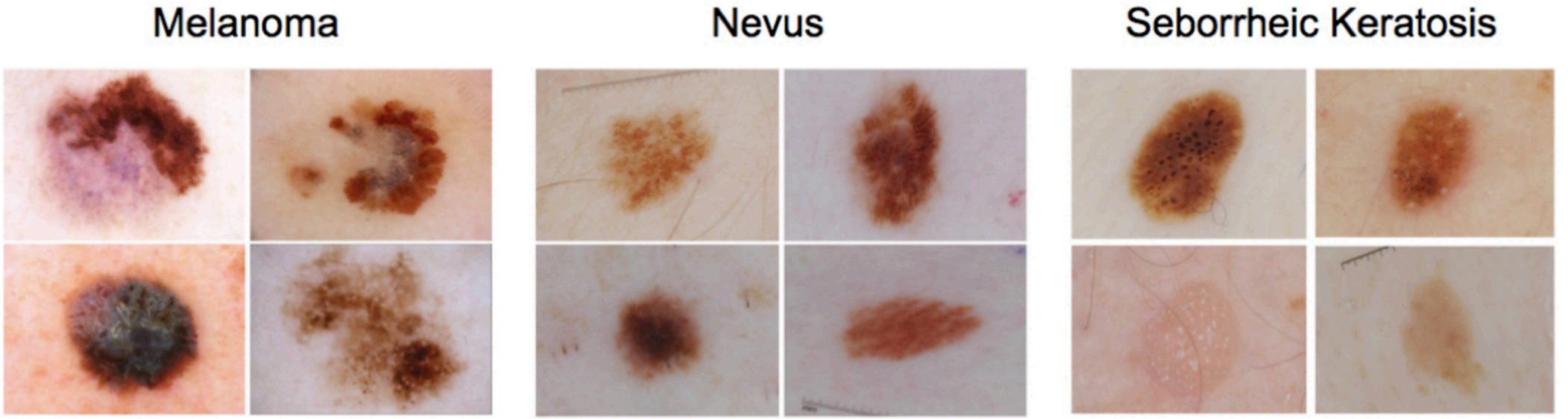
Jan. 20th. 2018

Introduction & Classifier

Introduction[1]

In this mini project, you will design an algorithm that can visually diagnose melanoma, the deadliest form of skin cancer. In particular, your algorithm will distinguish this malignant skin tumor from two types of benign lesions (nevi and seborrheic keratoses).

The data and objective are pulled from the 2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection. As part of the challenge, participants were tasked to design an algorithm to diagnose skin lesion images as one of three different skin diseases (melanoma, nevus, or seborrheic keratosis). In this project, you will create a model to generate your own predictions.

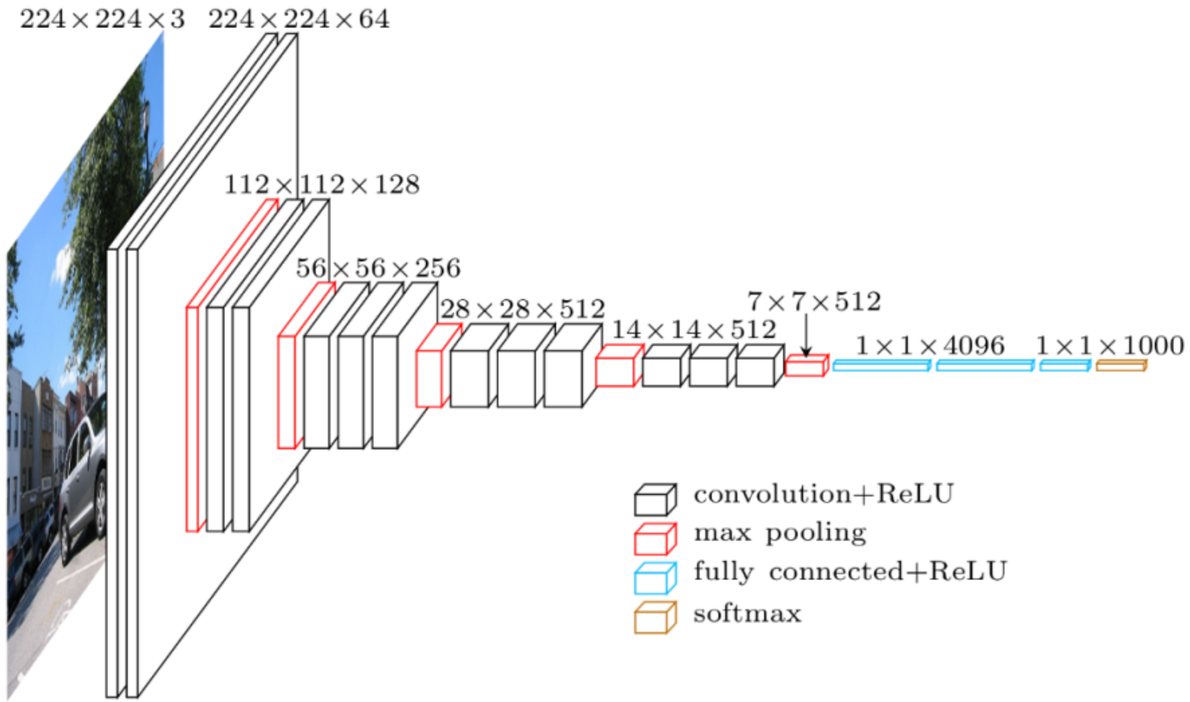


[1] https://github.com/udacity/dermatologist-ai

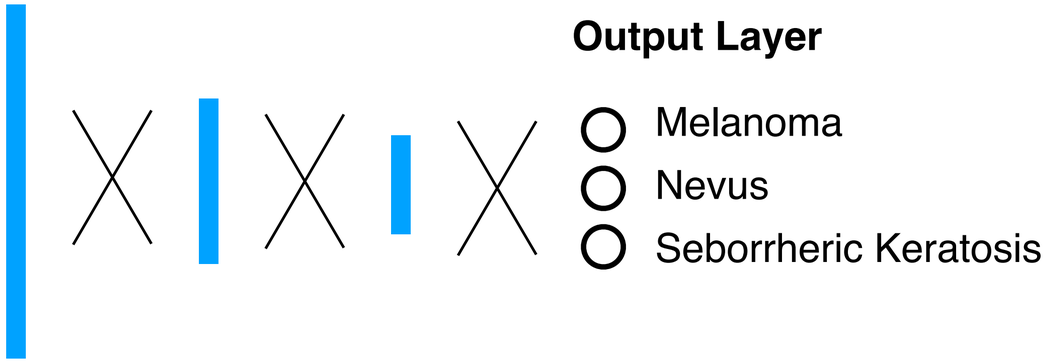
Classifier

I use pre-trained VGG16 and additional some fully connected layers to build the Classifier.

VGG16[1]



+ Fully connected layers



Fully connected layers + ReLU

Cost: Softmax cross entropy with legit
Optimizer: Gradient Descent Optimizer

[1] https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/

Classifier optimization

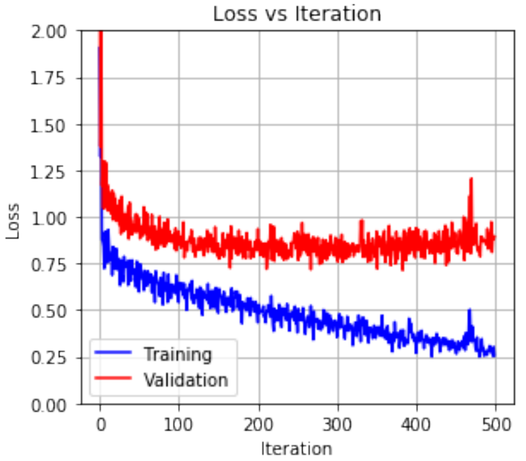
Classifier optimization

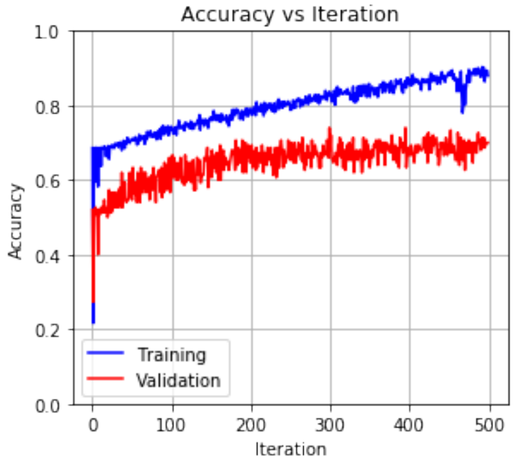
I check Training / Validation Loss & Accuracy with changing some parameters as below.

Parameter

	Parameter	Ref.	Trial
#1	Learning rate	0.01	0.001, 0.005, 0.01, 0.02, 0.05, 0.1
#2	Drop out ratio	0.75	0.25, 0.50, 0.75, 1.00
#3	Initial weight	0.025	0.001, 0.01,0.015, 0.025, 0.05, 0.075, 0.1
#4	Number of fully connected layer	3	2,3, 4
#5	Shuffle the Training data or not	Shuffle	Shuffle or not
#6	Number of trading data set	All data	Use all data (e.g. A:89, B:245, C:124) or Use same number of data (e.g. A:89, B:89, C:89)

Output

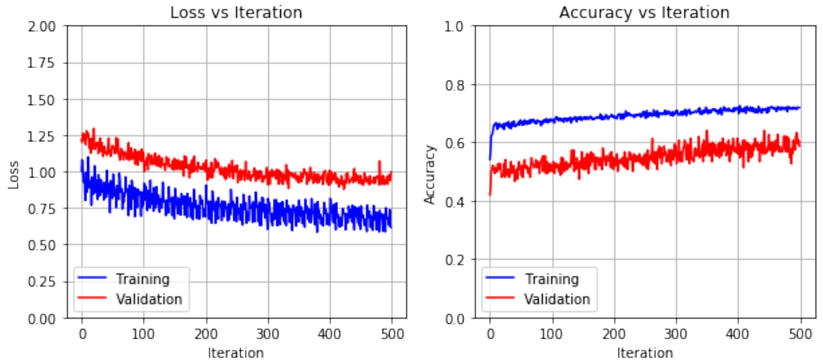




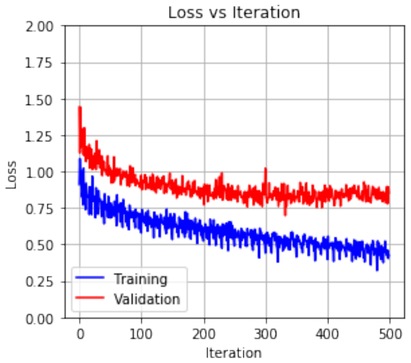
Learning rate dependence

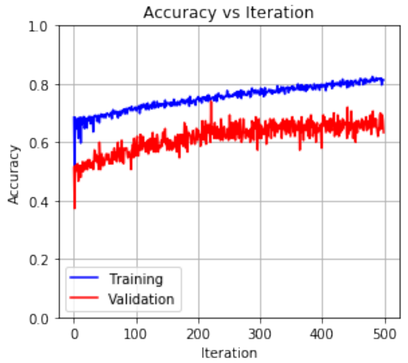
With smaller learning rate, many iteration was needed to learn but noise became smaller.

Learning rate: 0.001

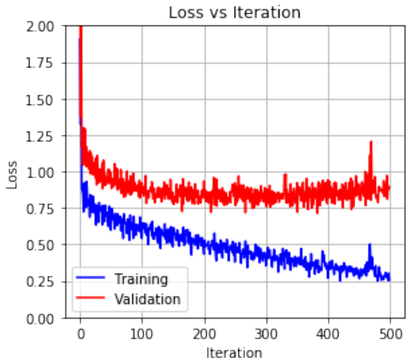


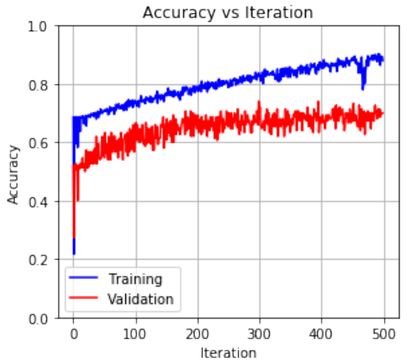
Learning rate: 0.005



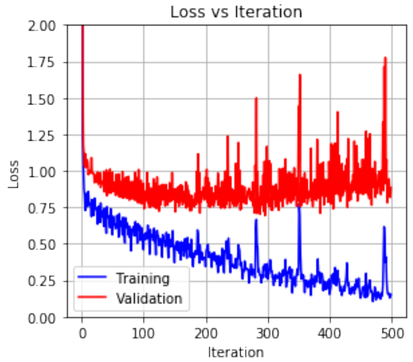


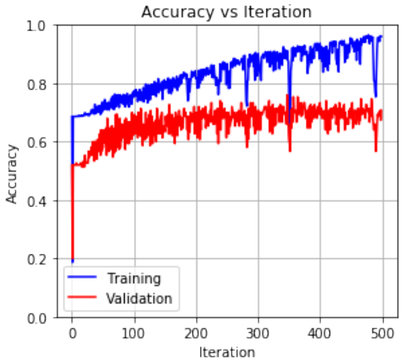
Learning rate: 0.01



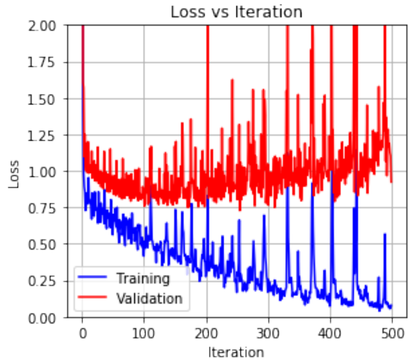


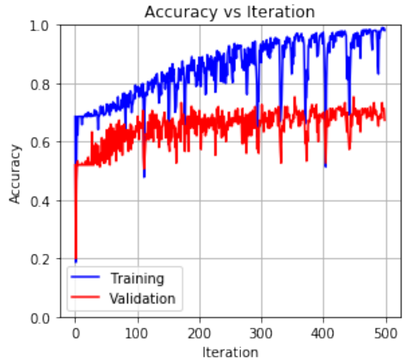
Learning rate: 0.02



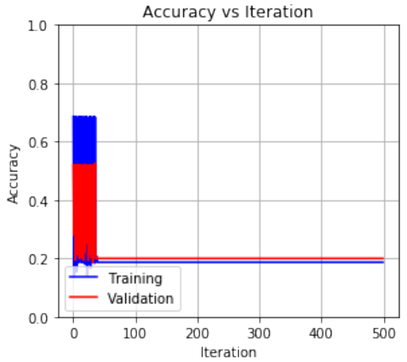


Learning rate: 0.05





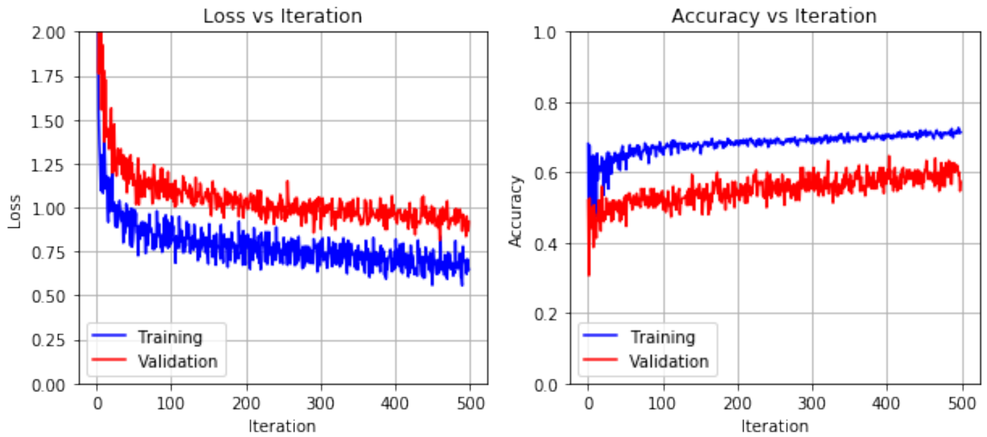
Learning rate: 0.1



Dropout ratio dependence

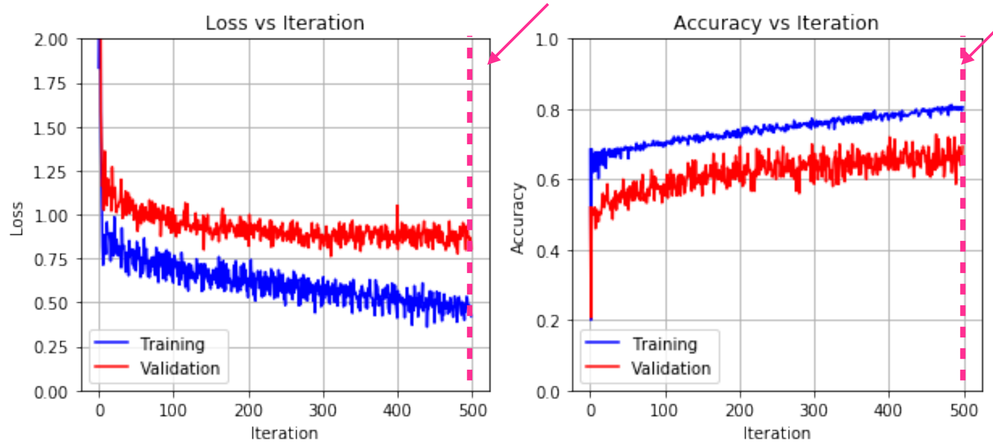
With higher dropout ratio, loss for training set became lower with fewer iteration.
Delta loss (between for the training set and for the test set) did not depend on the dropout ratio.

Dropout ratio: 0.25

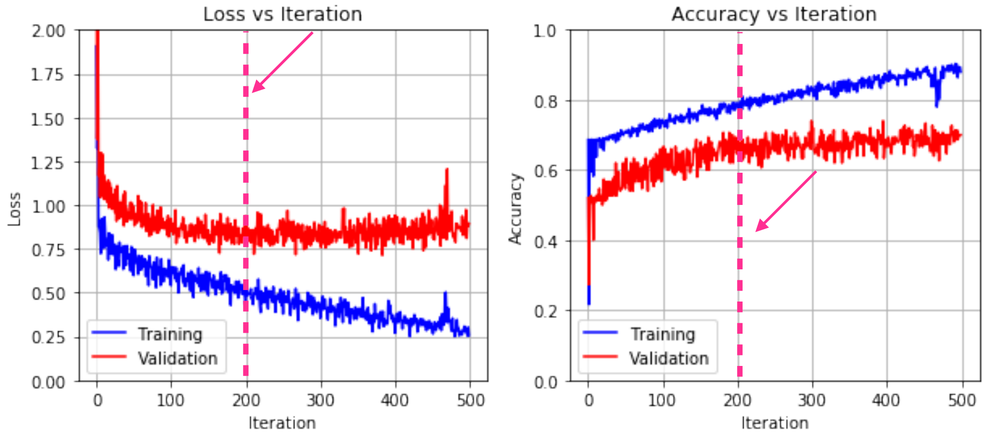


Dropout ratio: 0.50

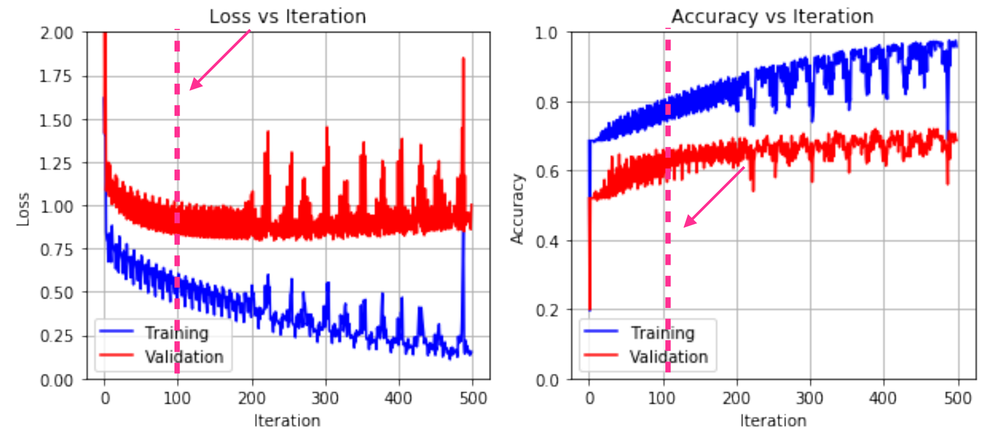
Training loss: ~0.5



Dropout ratio: 0.75



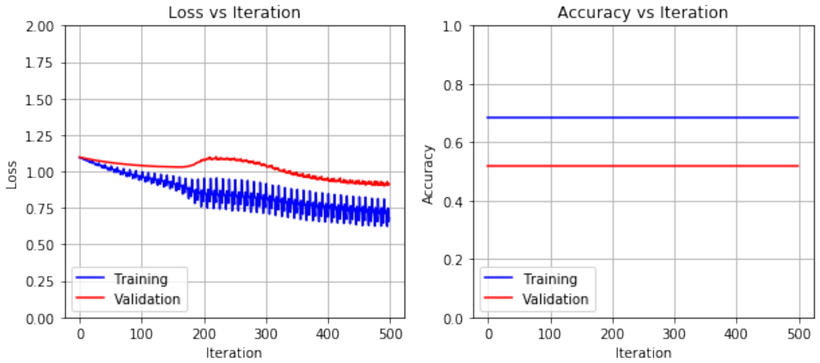
Dropout ratio: 1.00



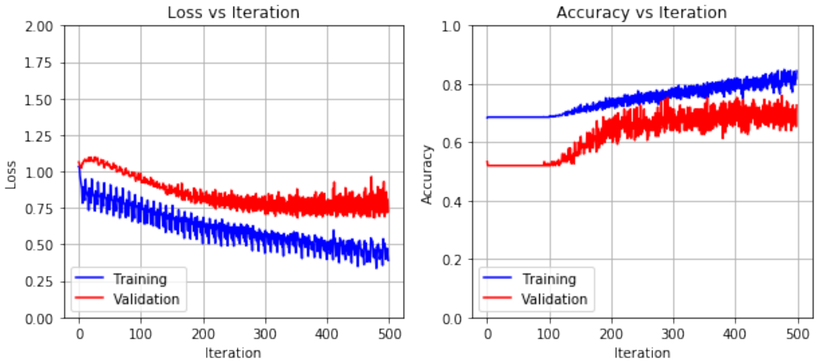
Initial weight dependence

I got smaller Validation loss with initial weight 0.01~0.025.

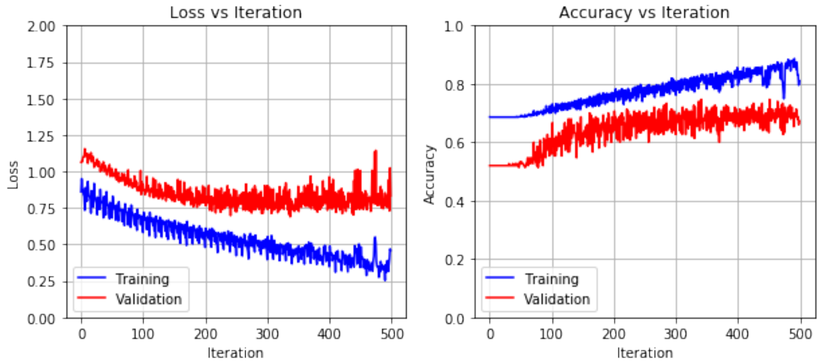
Initial weight: 0.001



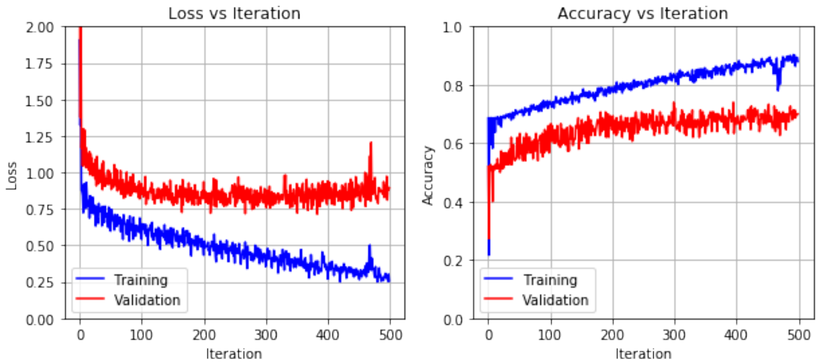
Initial weight: 0.01



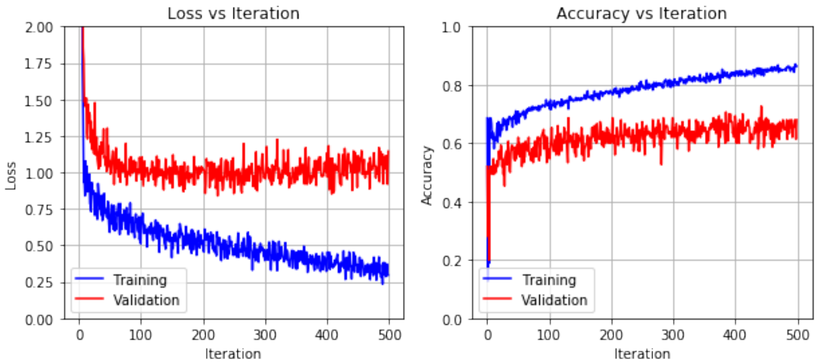
Initial weight: 0.015



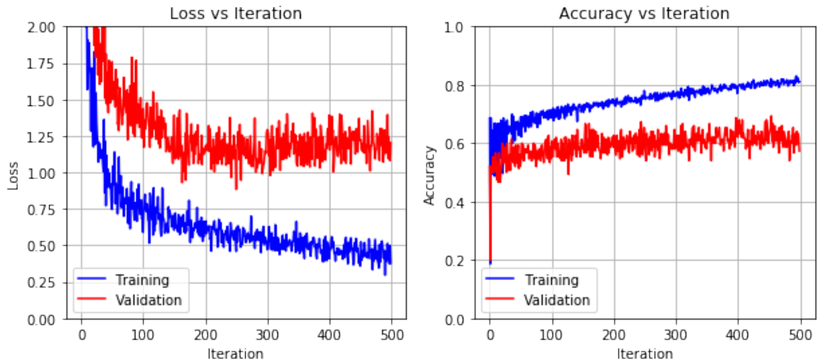
Initial weight: 0.025



Initial weight: 0.05



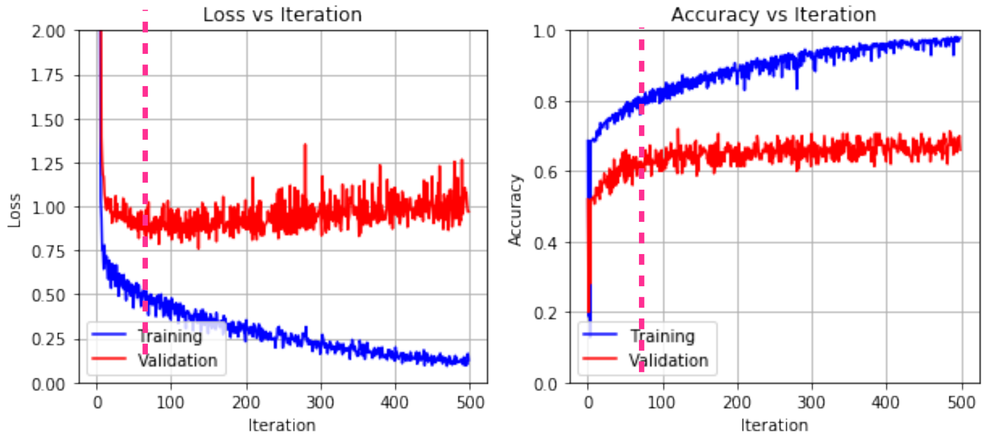
Initial weight: 0.075



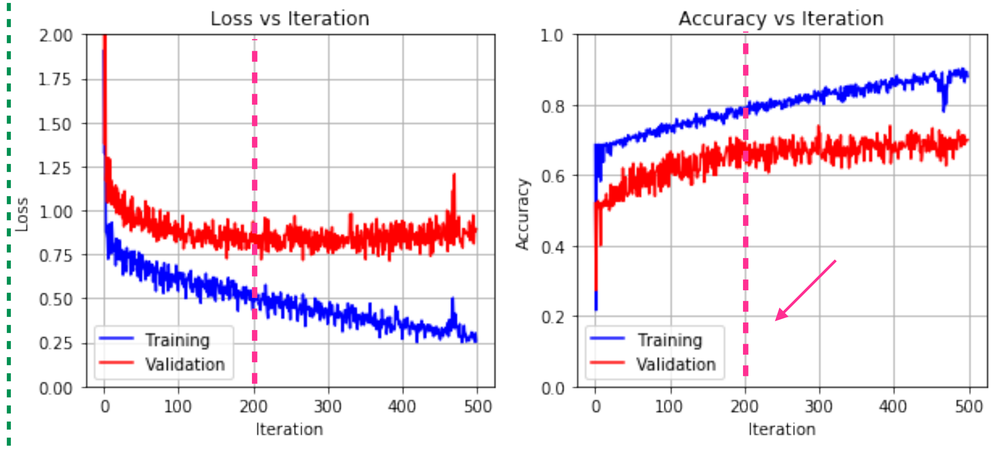
Fully connected layer number dependence

Delta loss (between for the training set and for the test set) with 2 layers were larger than that with 3,4 layers. Many iteration was needed to train model which has more fully connected layers.

Number of fully connected layer: 2

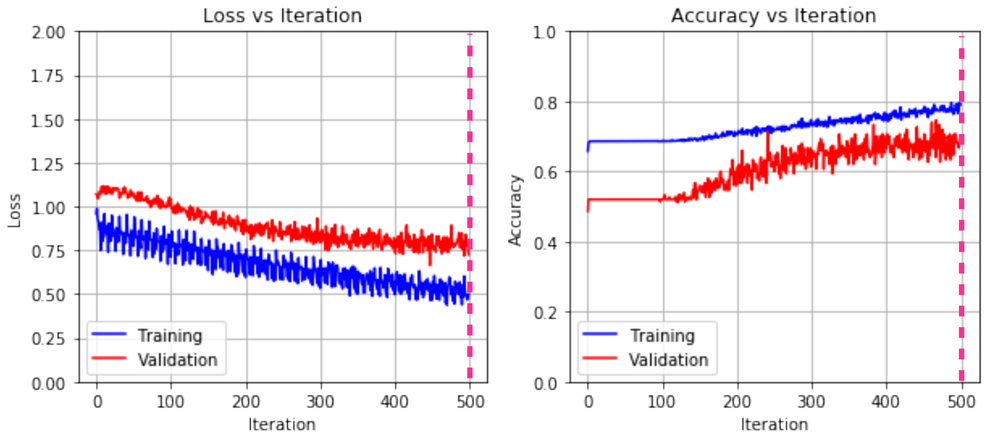


Number of fully connected layer: 3



Training loss: ~0.5

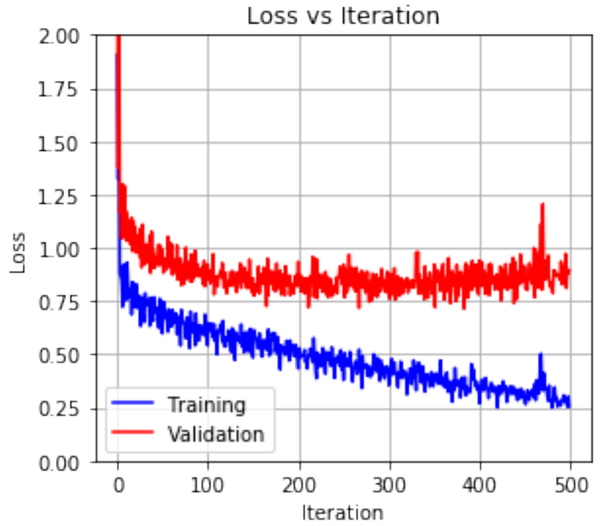
Number of fully connected layer: 4

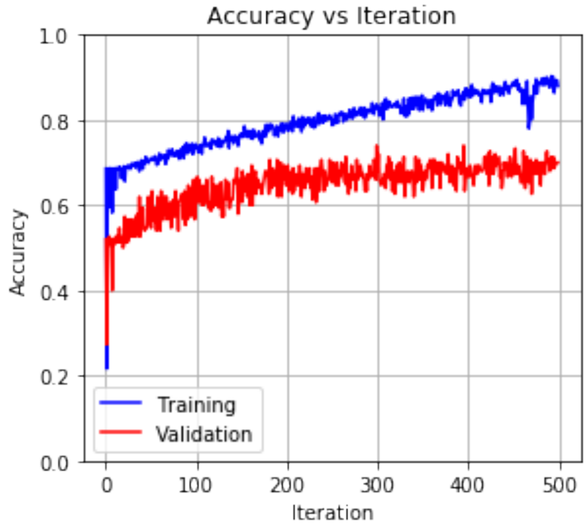


Shuffle the Training data or not

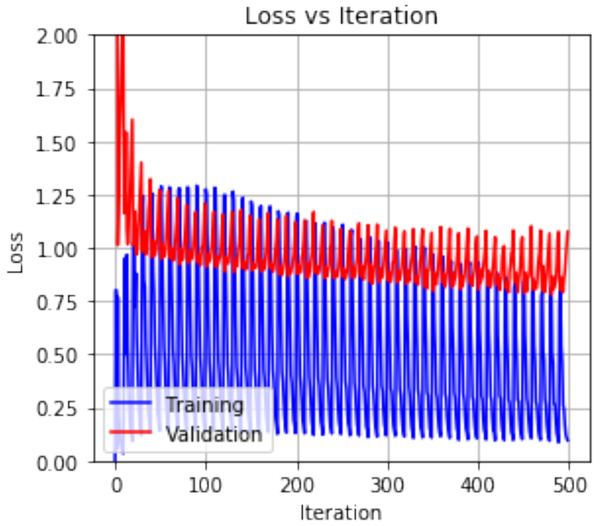
It was difficult to build stable model without shuffling the training data.

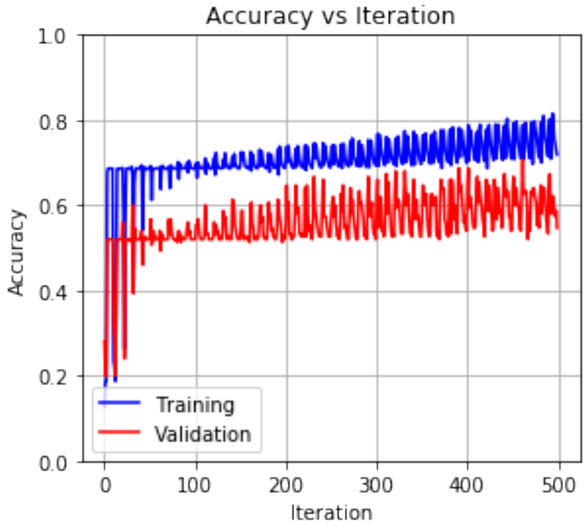
With shuffling





Without shuffling

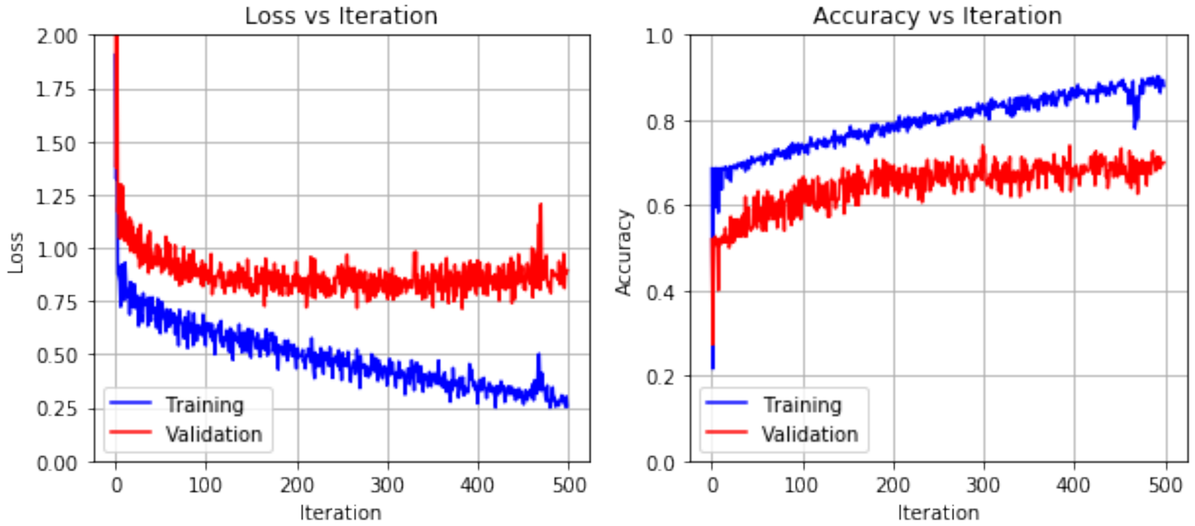




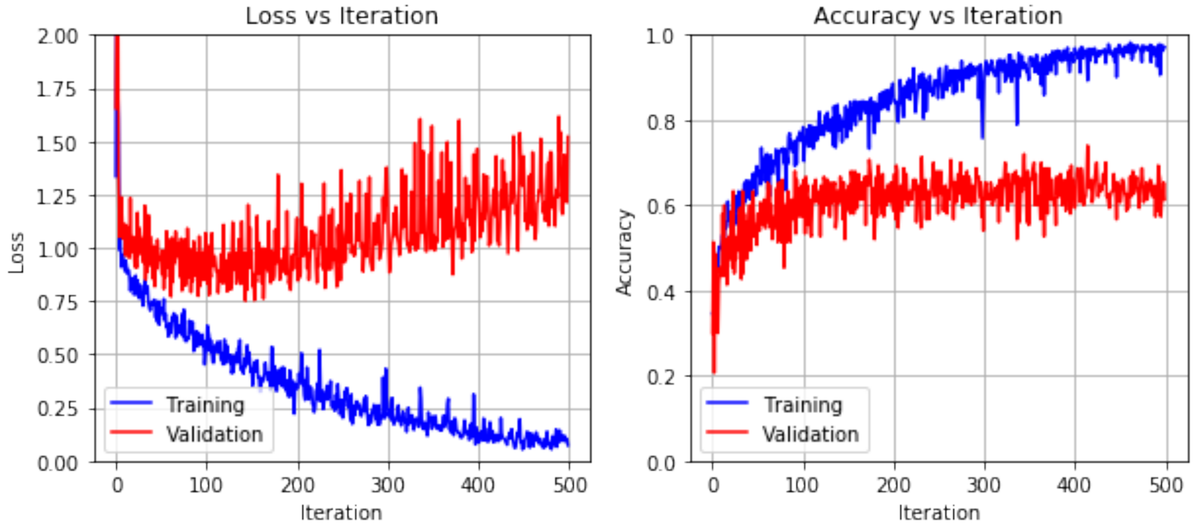
Number of trading data set

With using all data I got better accuracy for the test set and stable model in terms of less noise.

Use all data



Use same number of data (254 each)



Results

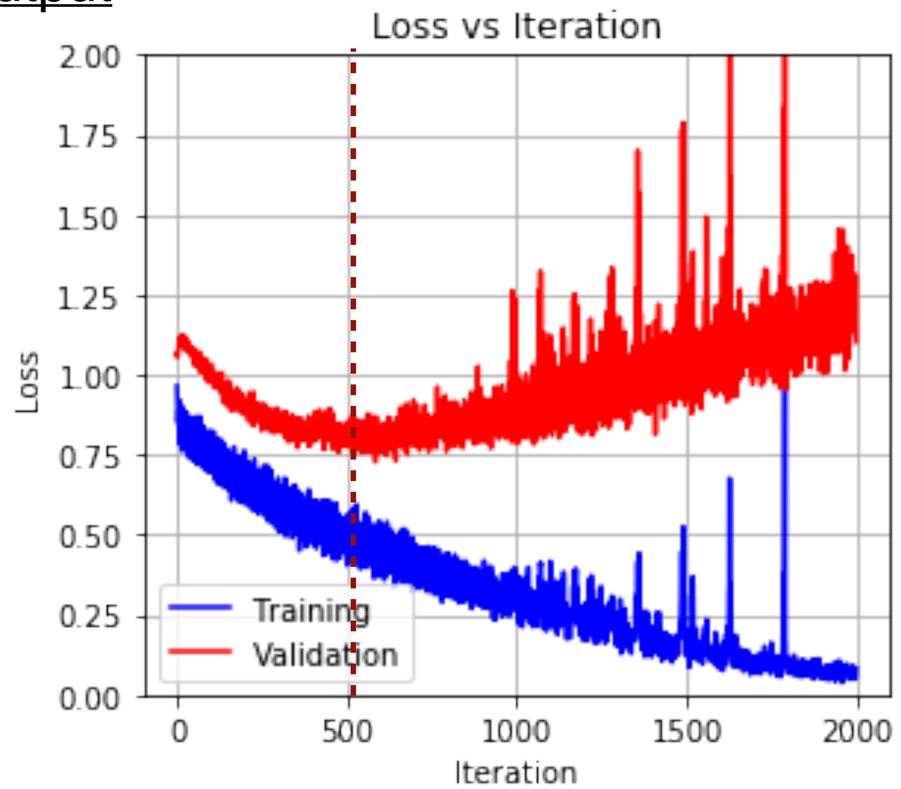
Results

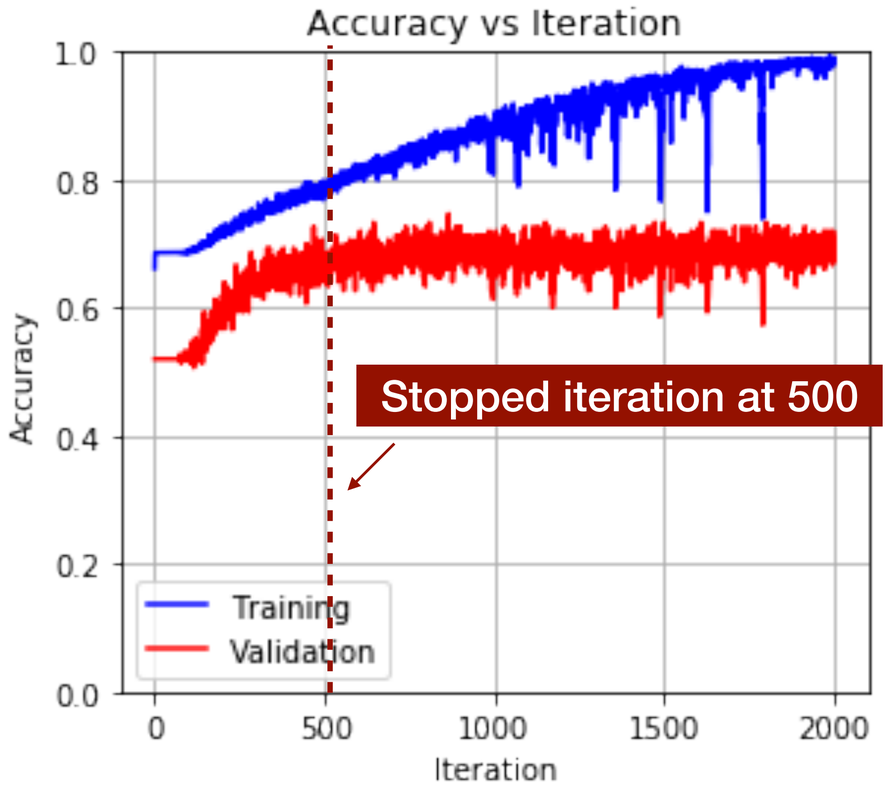
I chose the parameters as table below and got results as shown in figure below.
I stopped the training at 500 iteration to avoid overfitting.

Parameter

Parameter	Result
Learning rate	**0.005**
Drop out ratio	**0.75**
Initial weight	**0.015**
Number of fully connected layer	**3**
Shuffle the Training data or	**Shuffle**
Number of trading data set	**All data**

Output

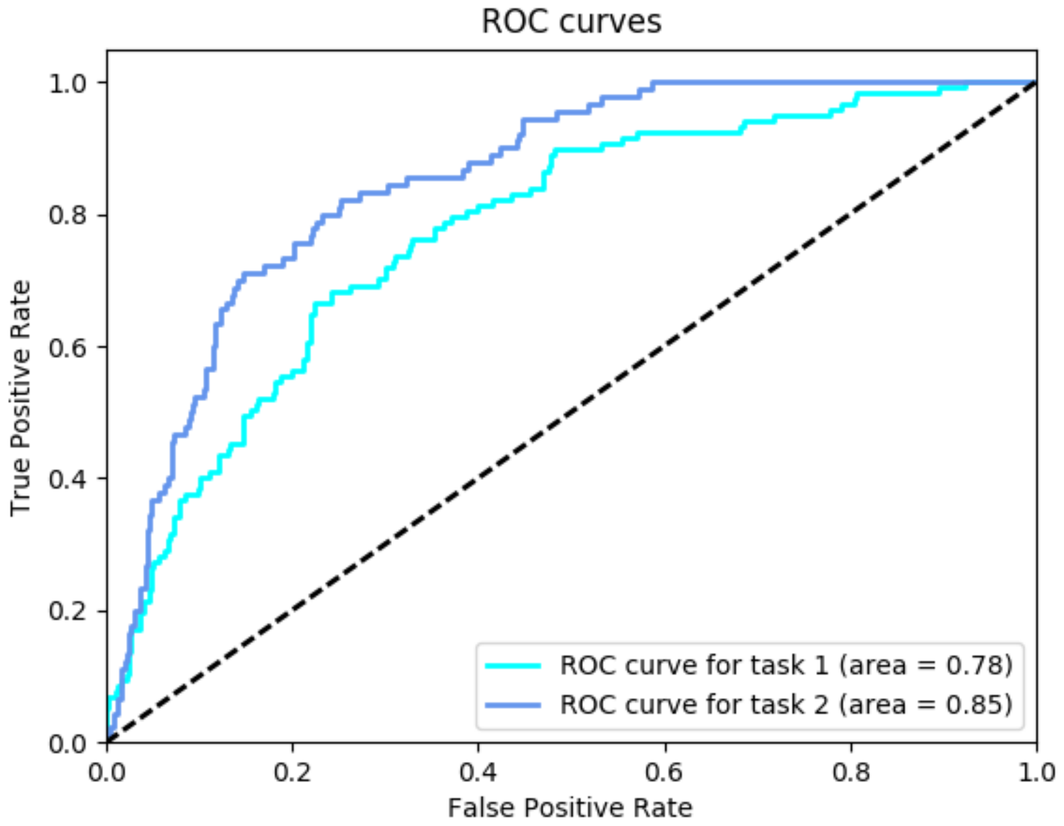




Results

ROC curve and Confusion matrix were as below. And the scores for each category was “Category 1 Score: 0.775”, “Category 2 Score: 0.853”, “Category 3 Score: 0.814”

ROC curve



Confusion matrix