UNITED STATES MILITARY ACADEMY

PROJECT 3

MA477: THEORY AND APPLICATION OF DATA SCIENCE
SECTION C1
LTC CHRISTOPHER WELD

By

CADET HURAM-ABI NZIA YOTCHOUM '23, CO D4
CADET ESTHER KANG '22, CO I2

WEST POINT, NEW YORK

22 APR 2022

hny MY DOCUMENTATION IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.

____ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

SIGNATURES: ____________________

MA477 Project 3

CDT Esther Kang, CDT Huram-Abi Nzia Yotchoum

April 2022

Abstract

The purpose of this report is to accurately predict the digit value of a handwritten digit. The dataset is already split into a test and train dataset, with the training dataset including 42000 different labeled handwritten digits, and the test dataset including 28000 unlabeled handwritten digits. The columns include the label values, or the digit values, and its corresponding unique values for 784 pixels. To predict value of a handwritten digit we used fitted our training dataset to a Convolutional Neural Network. Overall, the goal of this report is to describe how we designed the best model we could produce that will predict the correct value of a handwritten digit.

Introduction

For this study, we look at the supervised machine learning methods of neural networks and decision trees. Supervised learning is recommended when the goal of an analysis of a dataset is to predict a variable value. On the other hand, unsupervised machine learning is used when the goal is not to predict a certain value but to analyze a dataset to look for certain patterns. In this case, since we are looking to predict the digit value of a handwritten digit, supervised machine learning is used. Of the two methods, a convolutional neural network is used to approach the task at hand. Convolutional neural networks take in an image as an input and separate aspects of the image into unique values. This allows the network to differentate images based on how they differ in these unique values. As an image is inputted into the network, the image matrix goes through parsing until a more simplified two dimensional feature map is produced. The individual values in the feature map are then passed through ReLU and repeatedly pooled and simplified until the feature maps are one dimensional. The one dimensional feature maps are then correlated to classified as a certain output. This method will be used in this study to determine how we can best predict what digit the handwritten image represents.

Data

The training data had 42000 labeled handwritten digits, and was shaped as only a 2D tensor with the dimensions of (42000,784) without including the label column. The test data of unlabeled handwritten digits was shaped the same way. So the first this we did, was to reshaped our data into a 4D tensor. The new dimensions for our data then became (42000, 28,28,1), which tells us that each of the 42000 pictures have 28x28 pixels and are all grey. Now we know that the input shape of our Convolutional Neural Network will have to be (28,28,1) .

Methods

Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network that is used to process data that has a spatial structure, such as images. This means that the data is arranged in a grid-like fashion, which is easy for the CNN to process. CNNs are designed to take advantage of the spatial nature of data by using convolutional layers, which are layers that perform convolution operations on the data. Convolution operations are able to extract features from data, and CNNs are able to learn to recognize patterns in data. More specifically, CNNs are able to learn to recognize patterns in data that are invariant to translation, rotation, and other types of transformations. Convolution is a mathematical operation that is used to combine two signals to create a third signal. The convolution operation is performed by taking the dot product of the two signals. The convolution operation is often used in signal processing applications, such as image processing, to extract features from data. In a CNN, each convolutional layer has a set of kernels that are used to used to perform convolutional operations on the data. The kernels are convolved with the data to create feature maps. The feature maps are then input into the next layer of the CNN. The components of a CNN include the input layer, the convolutional layer, the pooling layer, the fully connected layer, and the output layer. The input layer is where the data is fed into the CNN. The convolutional layer is where the convolution operations are performed. The pooling layer is where the data is down sampled. The fully connected layer is where the final classification is performed. The output layer is where the results of the classification are output. Figure 1 is the design of one our Convolutional Neural Networks

Preventing Over fitting

The following are the methods we used to prevent against over fitting our data.

0.0.1 Early Stopping

We need to sometimes stop training a CNN early to prevent overfitting. Overfitting is when the neural network has learned the training data too well and is not able to generalize to new data. This is a problem because we want the neural network to be able to learn from new data, not just the data that it was trained on. To prevent overfitting, we can use a technique called early stopping. Early stopping is when we stop training the neural network before it has had a chance to learn the entire training set. This forces the neural network to generalize to new data because it has not had a chance to overfit the training data. One of The best conditions for early stopping is when the validation accuracy is no longer increasing. This means that the neural network has learned the training data and is now overfitting. Once the validation accuracy starts to decrease, it is an indication that the neural network is no longer learning from the training data and is starting to learn from the validation data. This is not what we want because we want the neural network to learn from new data, not just the data that it was trained on.

0.0.2 Batch Normalization

Batch normalization is a technique that is used to normalize the activations of the neurons in a neural network. Batch normalization is used to improve the training of deep neural networks. It is typically used as a regularization technique. It mitigates overfitting by constraining the activations of the neurons to a standard distribution. It is important for CNNs to have batch normalization because the CNNs often have a large number of layers, which can make the training process unstable and can lead to overfitting. The difference between batch normalization and normal normalization is that batch normalization normalizes the activations of the neurons across a batch of data, while normal normalization normalizes the activations of the neurons across the entire dataset.

0.0.3 L2 Regularization

L2 regularization is a type of regularization that is used to prevent overfitting. It is also known as weight decay. L2 regularization is used to penalize the weights of the neural network. The penalty is a function of the sum of the squares of the weights. The penalty is added to the cost function of the neural network. The cost function is then minimized during the training process. The effect of L2 regularization is to reduce the magnitude of the weights. This has the effect of reducing the complexity of the neural network and making it less likely to overfit the training data.

0.0.4 Dropouts

Dropouts are a regularization technique that is used in neural networks. Dropouts are used to prevent overfitting by randomly dropping out neurons during the training process. This forces the neural network to learn to function without the dropped out neurons. It is important because it prevents the neural network from becoming too reliant on any one neuron. The logic behind dropouts is that if a neuron is dropped out, it is less likely to overfit the training data because it cannot rely on the other neurons to do the work. Dropout also has the benefit of reducing the training time of the neural network.

Results

According to Kaggle our best accuracy score on the test data set was 99.014%. Out main was reduce overfitting while allowing the model to learn the data. If you compare figure three and four, we have attempted to reduce the gap between validation loss and training loss. That was our indicator that the model was not over fitting. When we included l2 regularization and early stopping figure five shows us how the validation and train loss stay fairly similar. There is not set science on how to tune a Convolutional network but with practice and multiple iteration we as data scientist start to get a feel for what the best models should look like.

Discussion

The other possible model was a decision tree, but that method was less applicable for this problem as there were several characteristics of decision trees that did not fit the dataset. One such characteristic was that the input values were not categorical, meaning there weren't multiple classes within each value. The way in which the images consisted of matrices was not as fitting for decision trees, since the images consisted of binary values that indicated which pixels were present in the image. One significant limitation of this study was the computation speed. With each trial, the program took about 2 to 3 minutes to produce the final outcome. Improving the technological aspect of neural networking and increasing the computational power could take away that limitation, making the use of neural networks in data science much more efficient in the future. We could have gotten a better score had we leveraged the technique called transfer learning. Transfer learning is when we create a new model starting with a pre-trained model to make learning new data of similar nature much faster.

Works Cited

Classifying Fashion with a Keras CNN (achieving 94% accuracy) - Part 2, *Medium*, July 14, 2019, https://medium.com/@mjbhobe/classifying-fashion-with-a-keras-cnn achieving-94-accuracy-part-2a5bd7a4e7e5a#:~:text=L2%20Regularization%3A%20where%20the%20cost%20added%20is%20proportional,%28implying%20that%20no%20egularization%20is%20applied%20by%20default%29.

Decision Trees Compared to Regression and Neural Networks, *DTREG*, https://www.dtreg.com/methodology/view/decision-trees-compared-to-regression-and neural-networks

Deep Learning, Overfitting and regularization, *Deep Learning*, https://atcold.github.io/pytorch-Deep-Learning/en/week14/14-3/

Evaluate the Performance Of Deep Learning Models in Keras, *Machine Learning Mastery*, August 27, 2020, https://machinelearningmastery.com/evaluate-performance-deep learning-modelskeras/#:~:text=Keras%20can%20separate%20a%20portion%20of%20your%20training,percentage%20of%20the%20size%20of%20your%20training%2dataset.

MNIST_CNN_with_Grid_Search, *Kaggle*, March 14, 2022, https://www.kaggle.com/code/suraz11/mnist-cnn-with-grid-search/notebook#Define Build-Model-Function

MNIST image classification with CNN & Keras, *Blog - Mohit Rathore*, https://mohitatgithub.github.io/2018-03-28-MNIST-Image-Classification-with-CNN-&Keras/#:~:text=MNIST%20is%20dataset%20of%20handwritten%20digits%20and%20cntains,is%20given%20here.%20Best%20accuracy%20achieved%20is%2099.79%25.

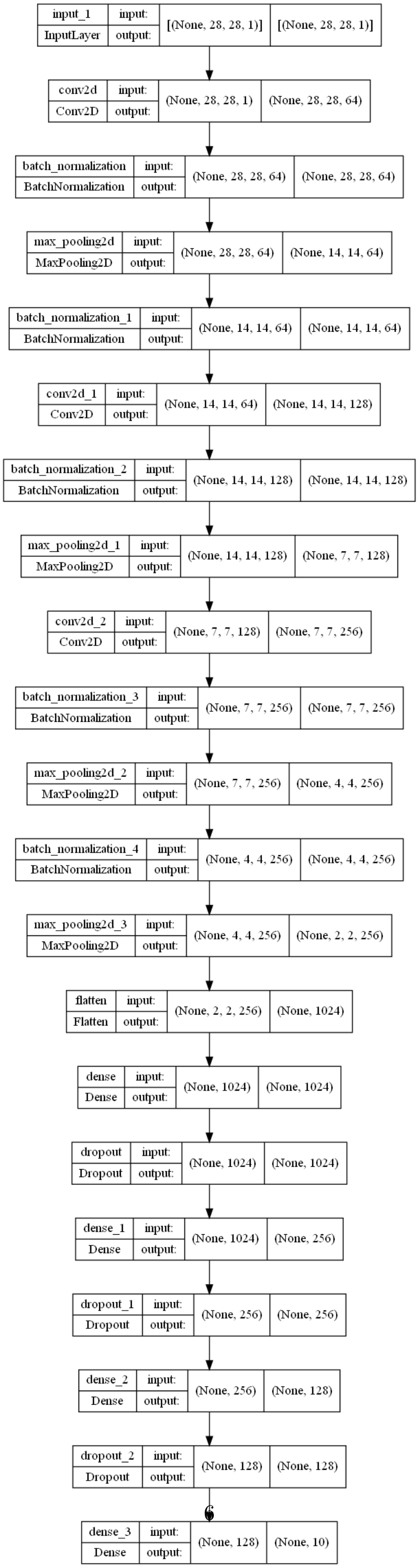


Figure 1: CNN Design

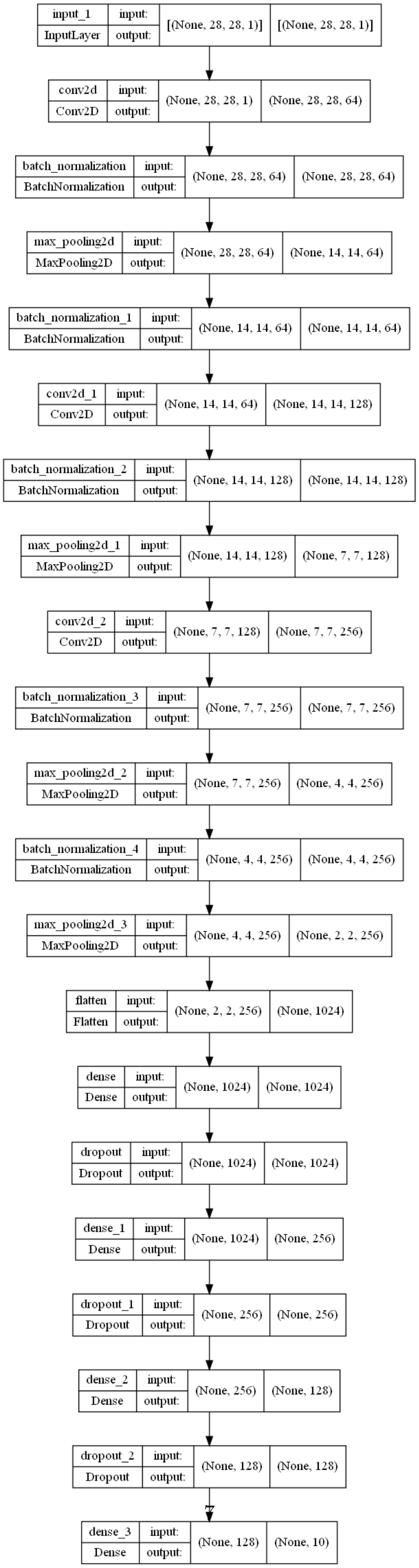

input_1 | InputLayer | input: | output: | [(None, 28, 28, 1)] | [(None, 28, 28, 1)]
conv2d | Conv2D | input: | output: | (None, 28, 28, 1) | (None, 28, 28, 64)
batch_normalization | BatchNormalization | input: | output: | (None, 28, 28, 64) | (None, 28, 28, 64)
max_pooling2d | MaxPooling2D | input: | output: | (None, 28, 28, 64) | (None, 14, 14, 64)
batch_normalization_1 | BatchNormalization | input: | output: | (None, 14, 14, 64) | (None, 14, 14, 64)
conv2d_1 | Conv2D | input: | output: | (None, 14, 14, 64) | (None, 14, 14, 128)
batch_normalization_2 | BatchNormalization | input: | output: | (None, 14, 14, 128) | (None, 14, 14, 128)
max_pooling2d_1 | MaxPooling2D | input: | output: | (None, 14, 14, 128) | (None, 7, 7, 128)
conv2d_2 | Conv2D | input: | output: | (None, 7, 7, 128) | (None, 7, 7, 256)
batch_normalization_3 | BatchNormalization | input: | output: | (None, 7, 7, 256) | (None, 7, 7, 256)
max_pooling2d_2 | MaxPooling2D | input: | output: | (None, 7, 7, 256) | (None, 4, 4, 256)
batch_normalization_4 | BatchNormalization | input: | output: | (None, 4, 4, 256) | (None, 4, 4, 256)
max_pooling2d_3 | MaxPooling2D | input: | output: | (None, 4, 4, 256) | (None, 2, 2, 256)
flatten | Flatten | input: | output: | (None, 2, 2, 256) | (None, 1024)
dense | Dense | input: | output: | (None, 1024) | (None, 1024)
dropout | Dropout | input: | output: | (None, 1024) | (None, 1024)
dense_1 | Dense | input: | output: | (None, 1024) | (None, 256)
dropout_1 | Dropout | input: | output: | (None, 256) | (None, 256)
dense_2 | Dense | input: | output: | (None, 256) | (None, 128)
dropout_2 | Dropout | input: | output: | (None, 128) | (None, 128)
dense_3 | Dense | input: | output: | (None, 128) | (None, 10)

Figure 2: CNN Design 2

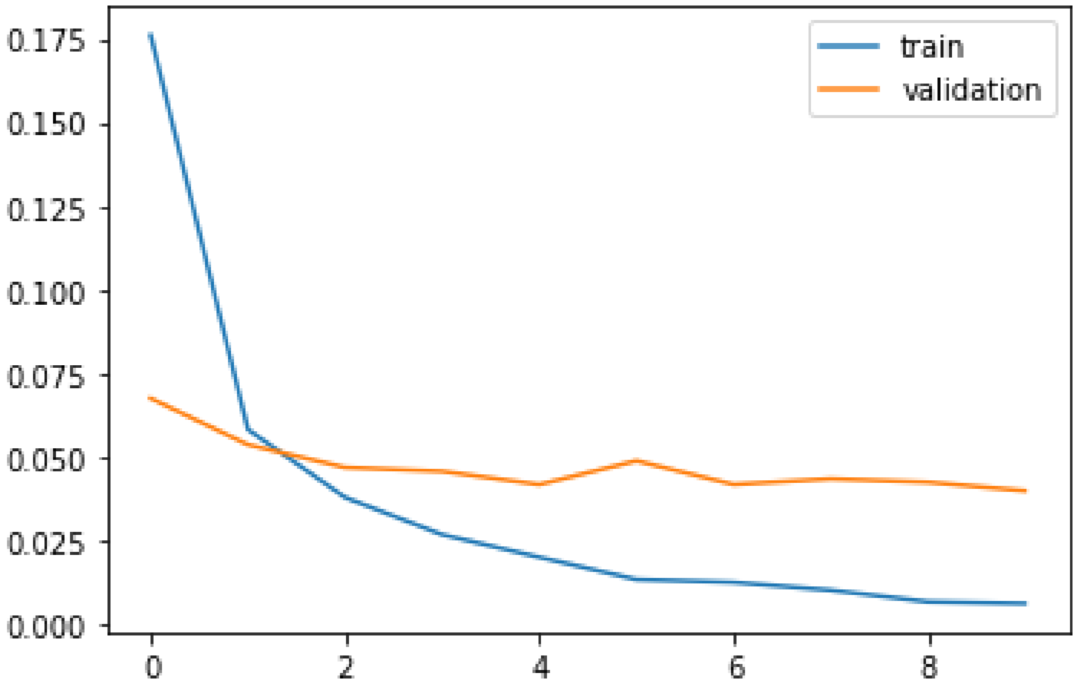


Figure 3: Train loss and Validation loss

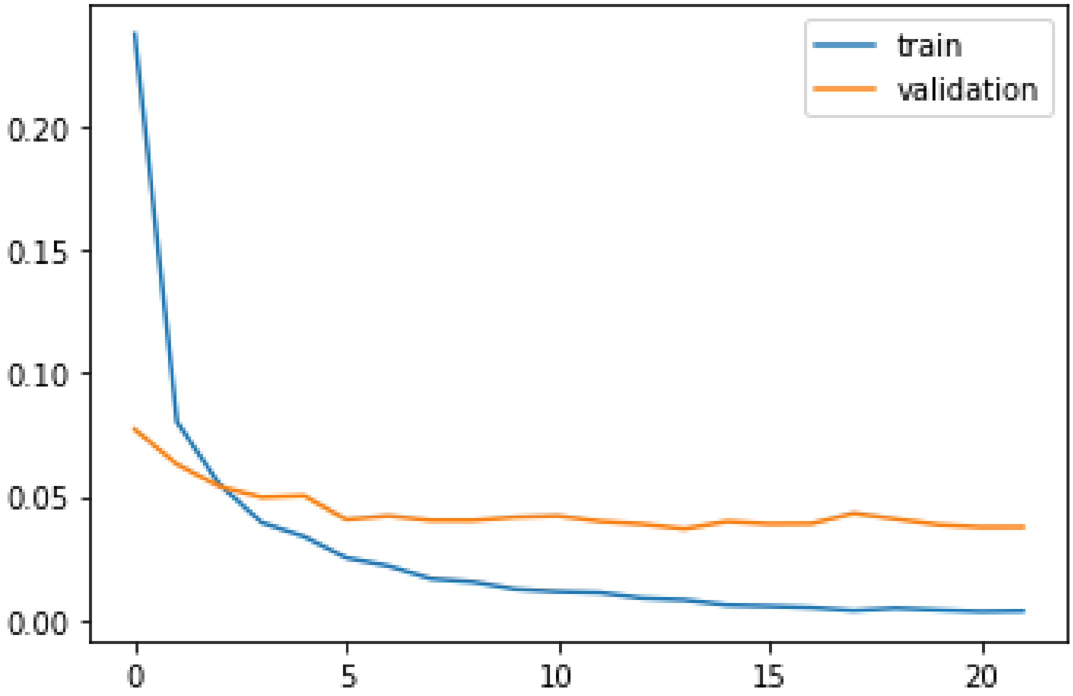


Figure 4: Train loss and Validation loss 2

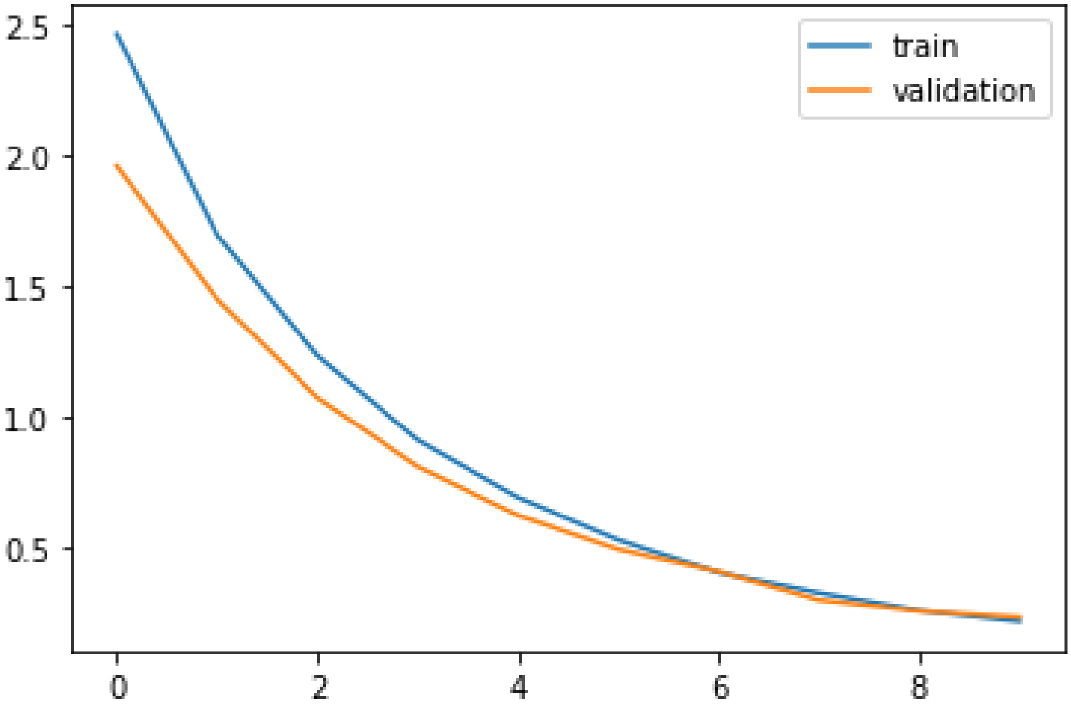
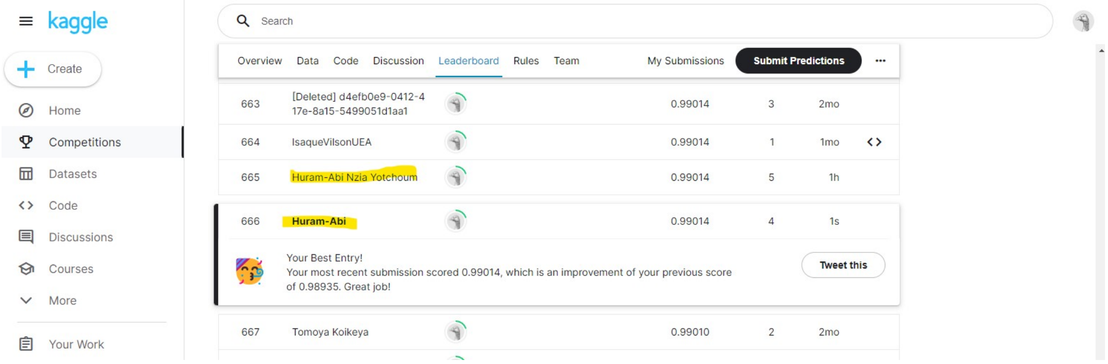


Figure 5: Train loss and Validation loss with l2 regularization and early stopping



663	[Deleted] d4efb0e9-0412-417e-8a15-5499051d1aa1	0.99014	3	2mo
664	IsaqueVilsonUEA	0.99014	1	1mo
665	Huram-Abi Nzia Yotchoum	0.99014	5	1h
666	**Huram-Abi**	0.99014	4	1s
667	Tomoya Koikeya	0.99010	2	2mo

Your Best Entry!
Your most recent submission scored 0.99014, which is an improvement of your previous score of 0.98935. Great job!

Figure 6: Kaggle Output