Final Project Report

Student Name: Sweta DAS Student ID: 18080395D

Design of task 1[[1]](#footnote-1)

*Model Selection*

The first dataset consists of 180 datapoints with 20 features each. Each datapoint is associated with one label that belongs to 3 categories. The data is therefore 1 dimensional, here we refer that to as 1D array. It is because we consider it to be stored in a 1D array. The model chosen for the training of the dataset is Artificial Neural Network (ANN). ANN is one of the models of neural networks used in deep learning. The concept is based on mimicking the human neural system, create artificial neurons and perceptron that act as a neural network same as in humans. It is a non-linear statistical model that creates and predicts complex relations between inputs and outputs. Why this model was selected because it has the capability to learn from previous inputs and sample, which enables in better predictions for classification. There can be multiple hidden layers using mathematical functions, can be different at each stage, then do analysis on the patterns defined by input. The output layer gives the result finally, in our case which datapoint belongs to what category based on multiple parameters. Since there is flexibility to include various parameters and manipulate them. The parameters are weight, bias, and activation function to name a few.

The whole model can be imagined as a structure of neurons which has multiple layers. Each input can be assumed as input to each node of the layer with certain weight, depending on how much weight is assigned. The transfer function takes the weighted sum of all inputs and forwards result to the activation function, which gives a certain output based on certain threshold. The activation function used in the project is RELU.

*Model Implementation- Essential features and Code execution*

To build the model, the datasets used are ‘data\_task1.npy’ and ‘label\_task1.npy’, each of size 180. ‘data\_task1.npy’ has 180 datapoints with 20 features each and has one label belonging to one of the three categories. The libraries used are numpy to work with arrays, sklearn for splitting data and classification, matplotlib to view accuracy of model, pytorch to help train the model. There are 7 layers used in total, the last one being the output layer. The activation function used is RELU (Rectified Linear Unit) as mentioned earlier, which means it gives value 0 if value of its input is negative, otherwise positive. Hence, no negative value is dealt with, which is a positive point of why RELU is used mostly in classification.

The model is trained in python language, therefore saved as ‘test\_task1.py’. To run the test data, put the name of the test data in single inverted commas and input at the places shown in the figure. For example, here it is shown how the datasets ‘data\_task1.npy’ and ‘label\_task1.npy’ are stored-

data = np.load('data\_task1.npy') #load data

label=np.load('label\_task1.npy') #load label

The code should be run in Python 3.8 environment, preferably Anaconda. The two files of data which are to be input as shown in above lines of code shall be run in the same folder or directory as the python file ‘test\_task1.py’. Just to give a proof of results discussed further, the files 'data\_task1.npy' and 'label\_task1.npy' are zipped in the same folder as the python file. Remember, the size of data and label should be the same. While running the program do not minimize it or open another window as it may pause its running. Else there will be runtime error. Also, the execution takes more than a minute sometimes if computer is being worked on for long hours. Therefore, it is better to run after keeping a busy computer on pause or sleep for few seconds in such cases.

***Conclusion and Discussion***

After training the model, accuracy was found to be 80%, and training time took 1.9 seconds. The value of epoch used in the code is 800. The accuracy is affected when the number of layers is increased or decreased. In this case, the optimal accuracy stays at 80% with 7 layers in total. Even one layer less brings down accuracy to 76% and increasing one layer reduces accuracy to around 74%. It happens because lesser layers cannot predict the pattern well for classification and having too many layers can cause overlap. Also, similar is the reason for value of epoch which determines the for loop. Too low value decreases accuracy, too high value also reduces it. Below is the graph used to show the values of training loss and training accuracy against the value of each epoch-

Graphical user interface

Description automatically generated

**Design of Task2**

*Model Selection*

The second dataset consists of 180 datapoints with 10\*10 features each. Each datapoint is associated with one label that belongs to 3 categories. The data is therefore 2 dimensional, here we refer that to as 2D array. It is because we consider it to be stored in a 2D array. The model chosen for the training of the dataset is Recurrent Neural Network (RNN). RNN is one of the models of neural networks used in deep learning. The concept is based on mimicking the human neural system, create artificial neurons and perceptron that act as a neural network same as in humans. It is a type of neural network in which the connection between the nodes exists based on a temporal or time-based sequence. As the name suggests, the network is recurrent. It can continuously refer to its past and learn from it, hence inferring the next output. Unlike some feed forward neural networks, it takes as input its past which is older than the current input, handles sequential data and memories previous inputs. Since the array is 2D, RNN is a good choice to handle such data structures as it is used for image and speech analysis, which are multidimensional. Within RNN, LSTM or Long Short-Term Memory network is used. This type of RNN has the default behaviour of learning and remembering long term dependencies. RNN is a chain like repeating structures having neural network modules. In LSTM, the inner layers interact in a more complex manner. There can be multiple hidden layers using mathematical functions, can be different at each stage, then do analysis on the patterns defined by input. The output layer gives the result finally, in our case which datapoint belongs to what category based on multiple parameters. Since there is flexibility to include various parameters and manipulate them. The parameters are weight, bias, and activation function to name a few.

The whole model can be imagined as a chain of repeating structure of neurons which has multiple layers. Each input can be assumed as input to each node of the layer with certain weight, depending on how much weight is assigned. The transfer function takes the weighted sum of all inputs and forwards result to the activation function, which gives a certain output based on certain threshold. The activation function used in the project is RELU.

*Model Implementation- Essential features and Code execution*

To build the model, the datasets used are ‘data\_task2.npy’ and ‘label\_task2.npy’, each of size 180. ‘data\_task2.npy’ has 180 datapoints with 10\*10 features each and has one label belonging to one of the three categories. The libraries used are numpy to work with arrays, sklearn for splitting data and classification, matplotlib to view accuracy of model, pytorch to help train the model. There are 7 layers used in total, the last one being the output layer. The activation function used is RELU (Rectified Linear Unit) as mentioned earlier, which means it gives value 0 if value of its input is negative, otherwise positive. Hence, no negative value is dealt with, which is a positive point of why RELU is used mostly in classification.

The model is trained in python language, therefore saved as ‘test\_task2.py’. To run the test data, put the name of the test data in single inverted commas and input at the places shown in the figure. For example, here it is shown how the datasets ‘data\_task2.npy’ and ‘label\_task2.npy’ are stored-

data = np.load('data\_task2.npy') #load data

label=np.load('label\_task2.npy') #load label

The code should be run in Python 3.8 environment preferably Anaconda. The two files of data which are to be input as shown in above lines of code shall be run in the same folder or directory as the python file ‘test\_task2.py’. Just to give a proof of results discussed further, the files 'data\_task2.npy' and 'label\_task2.npy' are zipped in the same folder as the python file. While running the program do not minimize it or open another window as it may pause its running. Remember, the size of data and label should be the same. Else there will be runtime error. Also, the execution takes more than a minute sometimes if computer is being worked on for long hours. Therefore, it is better to run after keeping a busy computer on pause or sleep for few seconds in such cases.

*Conclusion and Discussion*

After training the model, accuracy was found to be 93.33%, and training time took around 24 seconds. The value of epoch used in the code is 45. The accuracy is affected when the number of hidden layers or units is increased or decreased. In this case, the optimal accuracy stays at 93.33% with 64 hidden units in total. There is not much difference in accuracy when the epoch value is increased or decreased by 4-5 units. However, decreasing below by 5 units and increasing by 5 units decreases accuracy to around 86%. If hidden units are increased or decreased by even 1-2 units, there comes a big gap in accuracy. It happens because lesser hidden units cannot predict the pattern well for classification and having too many layers can cause overlap. The nature of hidden units is different than ANN as the hidden units is complexly connected. Also, similar is the reason for value of epoch which determines the for loop. Too low value decreases accuracy, too high value also reduces it. Below is the graph used to show the values of training loss and training accuracy against the value of each epoch-

A picture containing text, pencil

Description automatically generated

**Submitted Documents**

1. Folder name ‘Project\_SwetaDas\_18080395d’ contains-
   1. Folder ‘Task1’- Contains ‘test\_task1.py’. For proof of results- ‘data\_task1.npy’ and ‘label\_task1.npy’
   2. Folder ‘Task2’- Contains ‘test\_task2.py’. For proof of results- ‘data\_task2.npy’ and ‘label\_task2.npy’

**References**

Without the following references the model training and the whole project would have been not possible:

1. Lecture notes and Laboratory assignment codes of course COMP4431 of The Hong Kong Polytechnic University
2. Biswal, A. (2020, October 14). Recurrent Neural Network (RNN) Tutorial for Beginners. Retrieved December 15, 2020, from <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>
3. What are Recurrent Neural Networks? An Ultimate Guide for Newbies! (2019, July 24). Retrieved December 15, 2020, from <https://data-flair.training/blogs/recurrent-neural-networks/>
4. SarthakSarthak 94011 gold badge88 silver badges1818 bronze badges, Pho7pho7 97177 silver badges1212 bronze badges, User3693922user3693922 18388 bronze badges, BluponBlupon 43855 silver badges1212 bronze badges, & Khubaib RazaKhubaib Raza 37533 silver badges77 bronze badges. (1966, March 01). PyTorch: How to use DataLoaders for custom Datasets. Retrieved December 15, 2020, from <https://stackoverflow.com/questions/41924453/pytorch-how-to-use-dataloaders-for-custom-datasets>
5. How to run a basic RNN model using Pytorch? - #hackerday. (n.d.). Retrieved December 15, 2020, from <https://www.dezyre.com/recipes/run-basic-rnn-model-using-pytorch>
6. Brownlee, J. (2020, August 26). 3 Ways to Encode Categorical Variables for Deep Learning. Retrieved December 15, 2020, from <https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/>
7. John2xjohn2x 16.1k1616 gold badges4646 silver badges9090 bronze badges, Rogeriopvlrogeriopvl 39.5k77 gold badges5151 silver badges5858 bronze badges, PaulMcGPaulMcG 54.4k1212 gold badges8181 silver badges121121 bronze badges, Stevehasteveha 65.4k1818 gold badges8484 silver badges111111 bronze badges, Newacctnewacct 108k2626 gold badges146146 silver badges217217 bronze badges, NicojoNicojo 91377 silver badges88 bronze badges, . . . Rui CarvalhoRui Carvalho 2. (1958, November 01). How do I get time of a Python program's execution? Retrieved December 15, 2020, from <https://stackoverflow.com/questions/1557571/how-do-i-get-time-of-a-python-programs-execution>
8. ArkadipGhoshCheck out this Author's contributed articles., ArkadipGhosh, & Check out this Author's contributed articles. (2018, November 29). Numpy.load() in Python. Retrieved December 15, 2020, from <https://www.geeksforgeeks.org/numpy-load-in-python/>
9. Ved. (2016, October 05). How to improve performance of Neural Networks. Retrieved December 15, 2020, from <https://d4datascience.wordpress.com/2016/09/29/fbf/>
10. Brownlee, J. (2020, August 26). Train-Test Split for Evaluating Machine Learning Algorithms. Retrieved December 15, 2020, from <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/> GulzarGulzar 7, Wasi AhmadWasi Ahmad 28k2626 gold badges8282 silver badges139139 bronze badges, EyshikaEyshika 33033 silver badges1111 bronze badges, Ashiq ImranAshiq Imran 1, Prostiprosti 22.6k33 gold badges108108 silver badges100100 bronze badges, & Marius JohanMarius Johan 31522 silver badges88 bronze badges. (1968, February 01). How to convert a pytorch tensor into a numpy array? Retrieved December 15, 2020, from <https://stackoverflow.com/questions/54268029/how-to-convert-a-pytorch-tensor-into-a-numpy-array>

1. COMP4431 [↑](#footnote-ref-1)