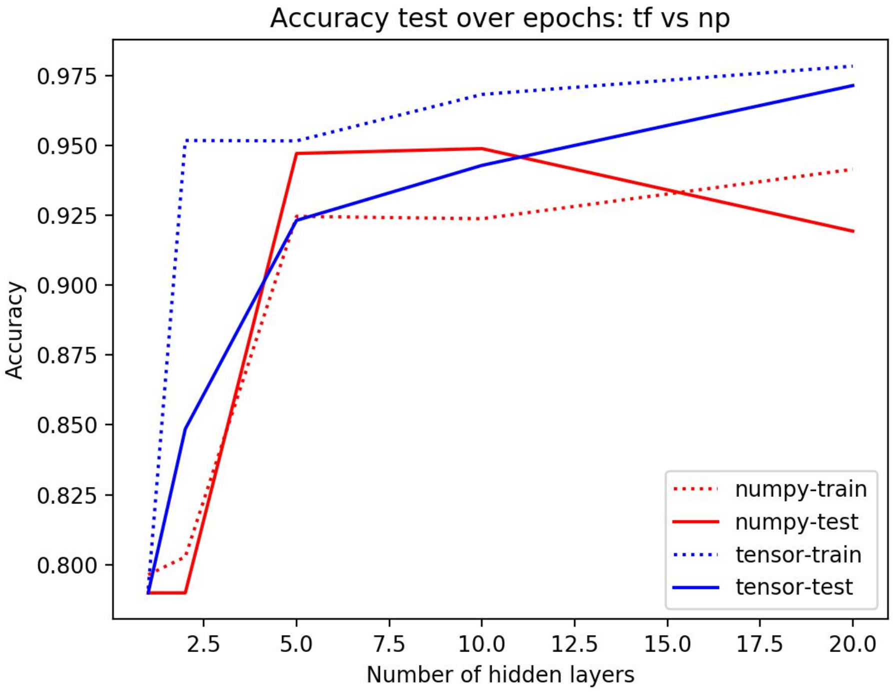
*HOMEWORK 3 - CSCI 5922 - **Neural Networks and Deep Learning***

Name : Ashwin Sankaralingam
SID : **108593584**

1.a) The following graph depicts the plot comparing best test and train of using numpy(self-implementation) vs tensorflow implementation for hidden layers [1,2,5,10,20]

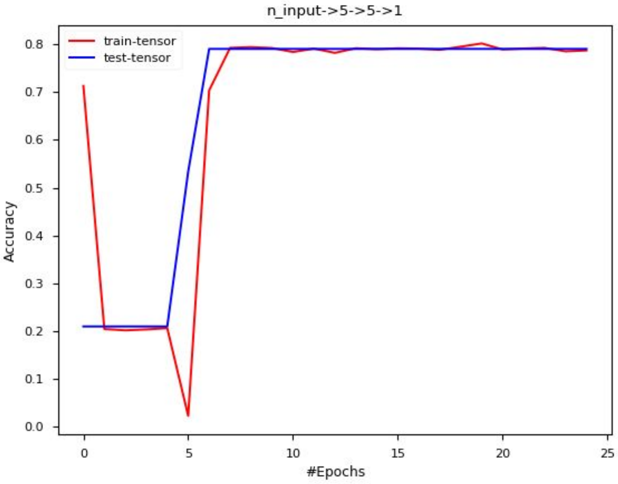
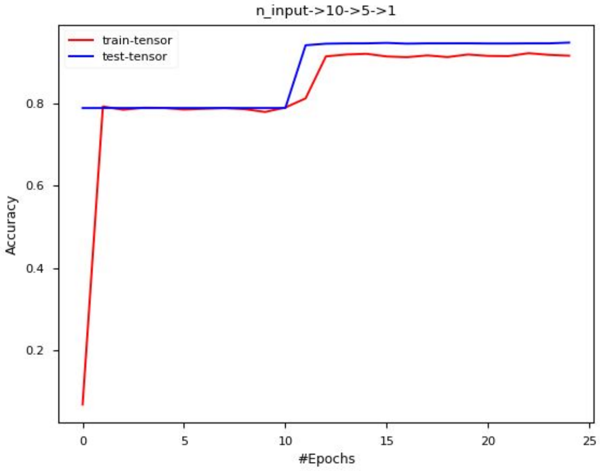


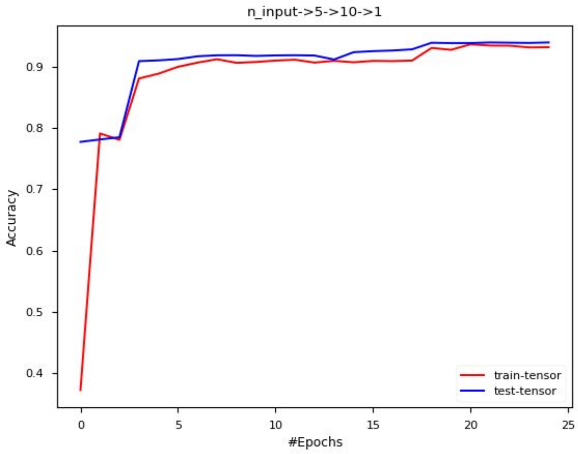
1.b) Tensorflow works better for having more number of hidden units. It handles the weight initialisations and propagation of the derivation across the layers more efficiently than using numpy self implementation. Moreover more complex loss functions can be used in tensorflow easily compared to hard coding, as Tensorflow has all the derivatives inbuilt. I also noticed that tensorflow works faster than numpy operations and it will be useful for larger iterations.

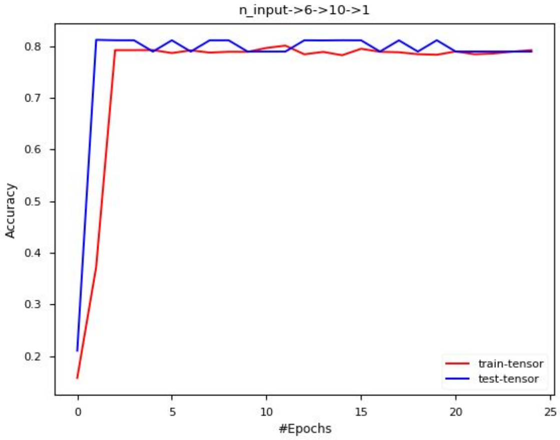
1.c) I trained over few architectures and the resulting accuracy graph are as follows:

From the previous experiments, having the architecture in->5->out gives Mean accuracies:

- Test = 92.25 %
- Train= 94.42 %

Architecture	Accuracy Graph	Mean Accuracy
in->5->5->out		Test : 78.83% Train : 82.32%
in->10->5->out		Test : 85.32 % Train : 88.21 %

in->5->10->out		Test : 94.13 % Train : 94.25 %

in->6->10->out		Test = 77.77%
		Train = 74.32 %

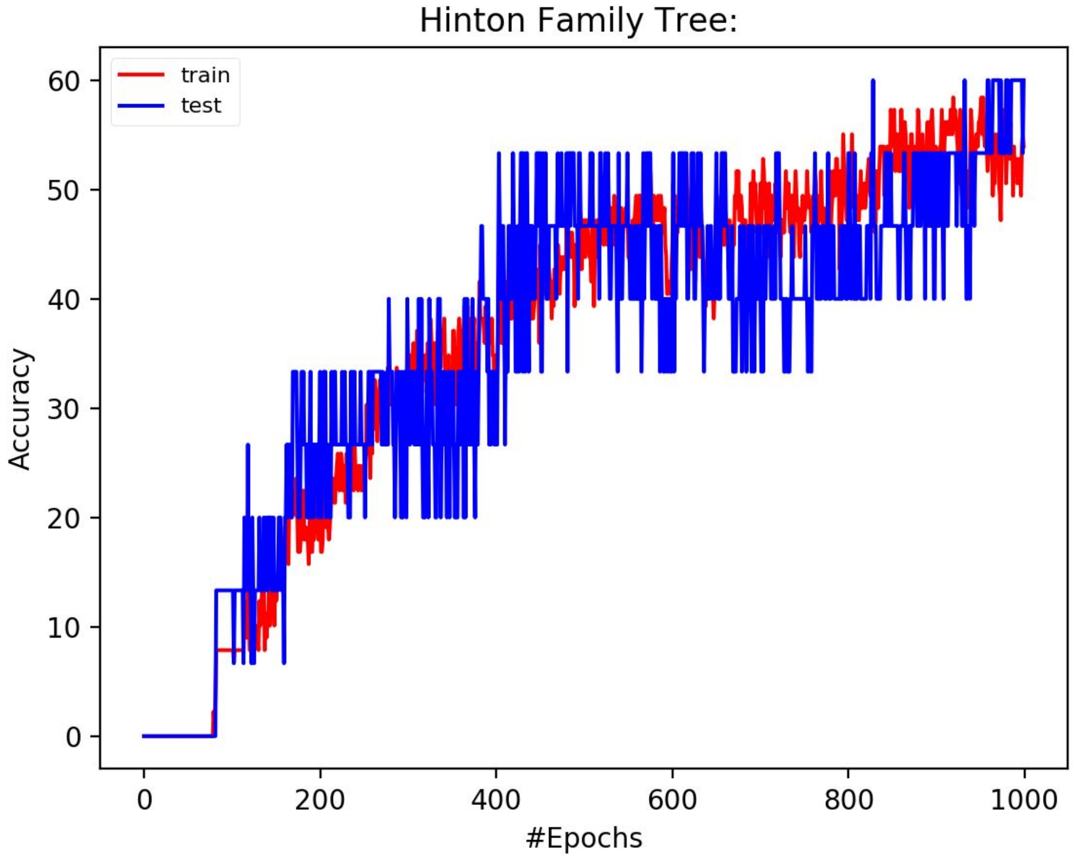
From the above experiments it is clear that single layer with 5 neurons outperforms most of the other architecture. One of the architecture in->5->10->out seems to perform almost better than a single layer in test case. My intuition this works better is because, the first 5 units in hidden layer 1 adds on a distributive representation of the 5 input layers and passing on to a larger number of units in hidden layer 2 seems to learn the patterns well. Considering this fan-out technique, I tried on in->6->10->out, and the hidden layer 1 representation did not efficient pass on the information learnt over the time.

None of the multi-layer architecture worked better than a single layer with 20 units. So it depends on the problem, if there should be a large number of layers with less number of units or less number of layers with large number of units.

2. a) Reporting the Accuracy for Hinton Family Tree run over 1000 epochs over 20 random splits:

Over 20 splits: **Mean Test Accuracy:** 33.4026% **Standard Deviation:** 0.04117

Graph: Graph for one of the splits using Hinton Family Tree architecture and 6 hidden units and activation as sigmoid and Optimizer as RMSProp Optimizer with constant learning rate 0.01



Analysing the accuracy in one of the split across epochs:

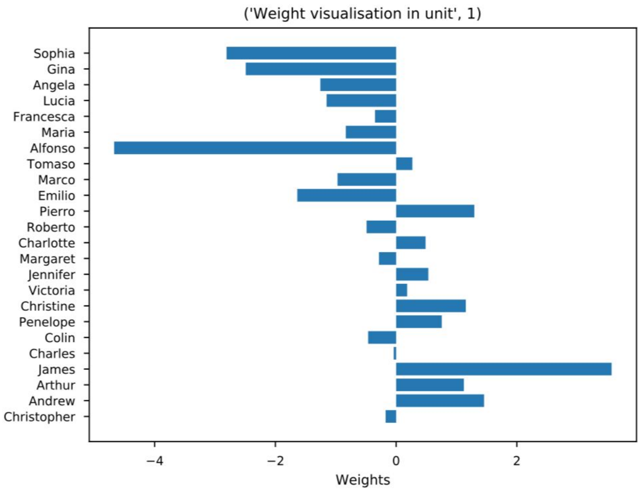
Table : Mean Accuracy Percentage for one of the splits:

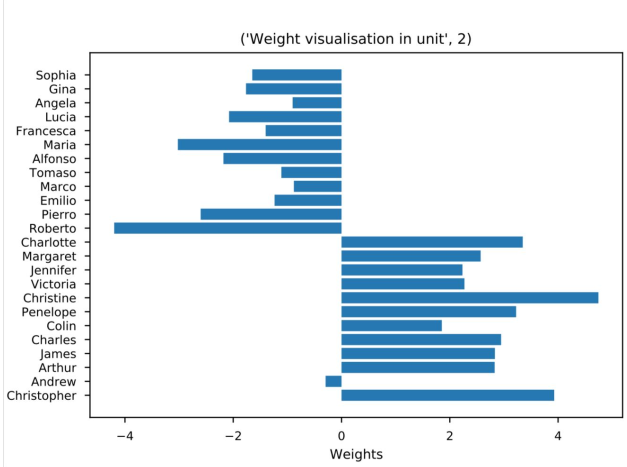
	First 500 Epochs	**Last 500 Epochs**	**Total 1000 Epochs**
Training	23.683	49.402	40.234
Testing	25.053	46.213	44.64

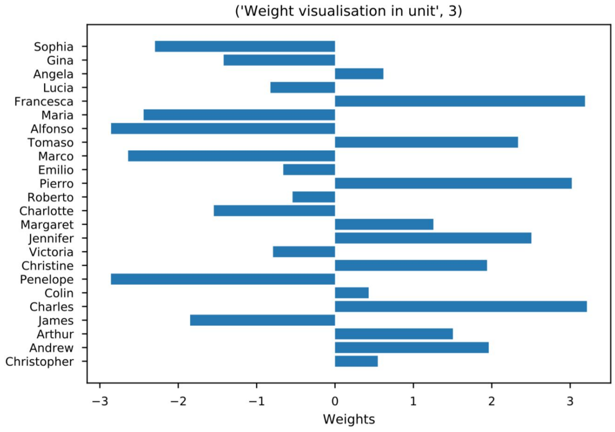
From the above table we can clearly see that the first 500 epochs maybe stuck in the plateau resulting in a lower accuracy standard. As we reach to the second half of the set, weights get a value that results in a larger value. Till the first 100 epochs, the test accuracy is 0.00% and later it starts to increase. As there are very less number of information, the test accuracy seem to do better in certain cases. To explain, for a train accuracy to get a 25% accuracy, at least 22 cases must be satisfied, while for test case to get 25% accuracy, it just needs at least 3 cases to be predicted correctly. Considering relatively, it is easier to get 3 cases right than 22 cases, hence there might be situations where test outperforms train.

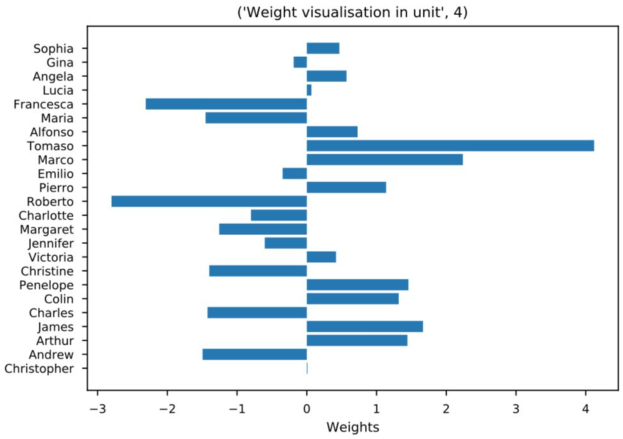
b) To represent the weights from one hot person 1 layer to the distributed person layer 1, I used vertical bar charts. The x axis are the weights and the y axis are the names of the entities.

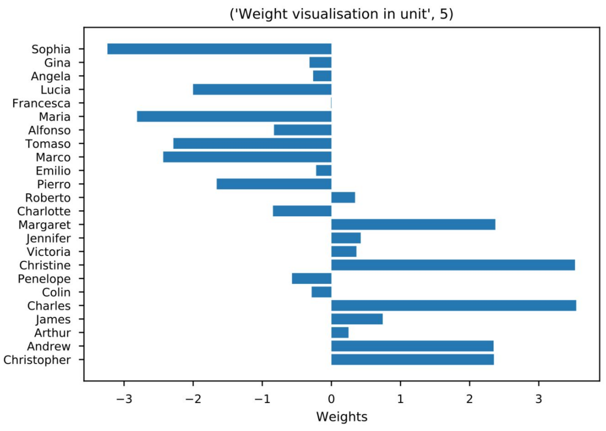
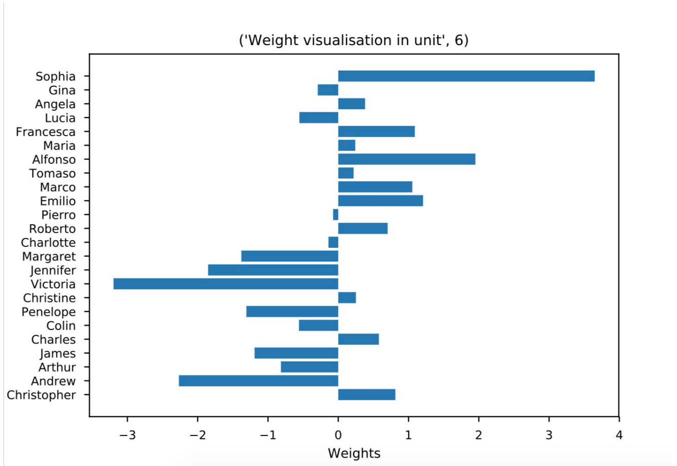
The 6 weight units looks like the following:





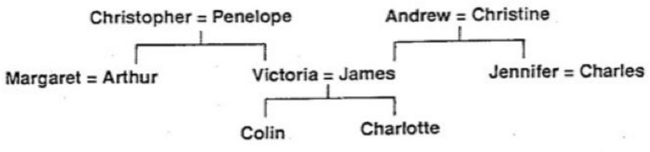


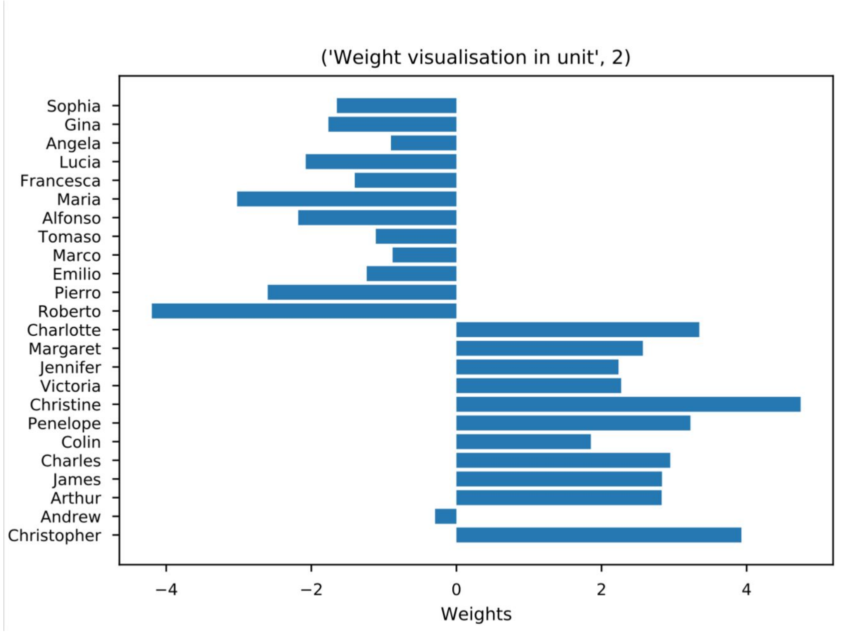


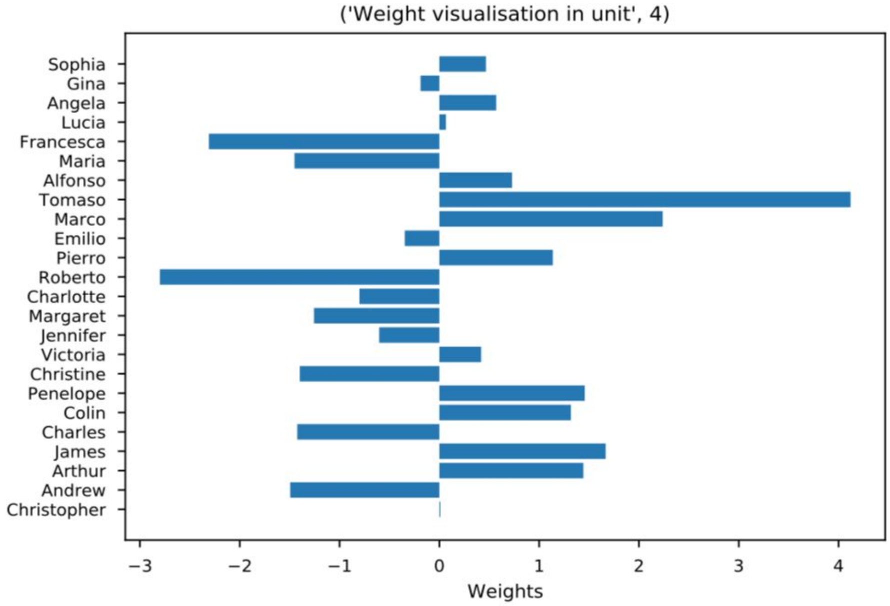
2.c) I was able to interpret the three weight matrix their hidden meaning . At each run, due to the random weight initialisation, the weights were not constant and the weights that occurred in the previous run reappeared in a different unit. So from this we can understand the system selects different features during various running instances.

Using English Family for reference:



Learning from Unit 2: Family Type	
	We can visualize that the positive weights (except Andrew) corresponds to English Family and negative weights correspond to Spanish Family. Maybe weight value for 'Andrew' in this final epoch may be set like this and would have resulted in a error.

Learning from Unit 4 : Type of Branch



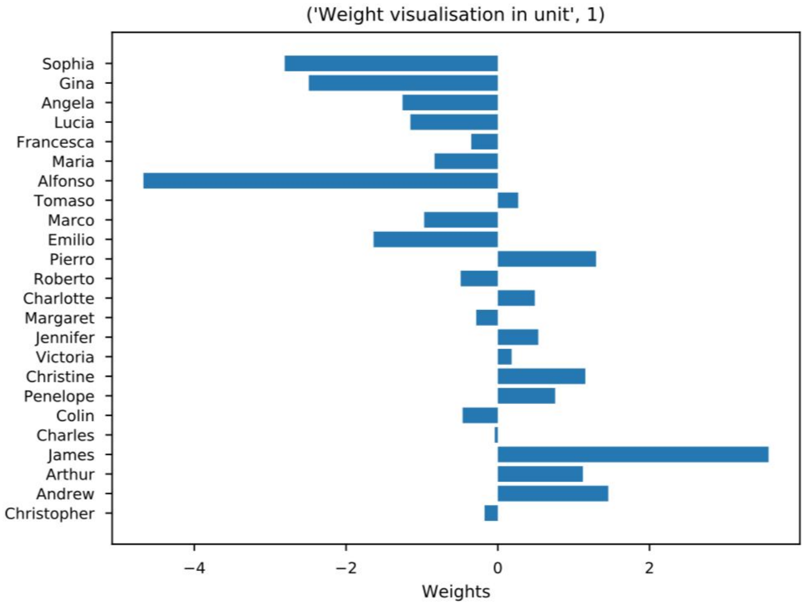
In most of the cases for English Family the right branch has negative weights and the positive weights corresponds to names that are in right branch. The following gives the anomalies in the names:

Left Branch	Right branch names:
Christopher Charles* Victoria* Arthur James Colin	Andrew Charles *Margaret* Charlotte Christine Jennifer

* : Victoria and Charles are left as well as right subtrees and it might be a confusion for classification.

Margaret is falsely classified with these weights.

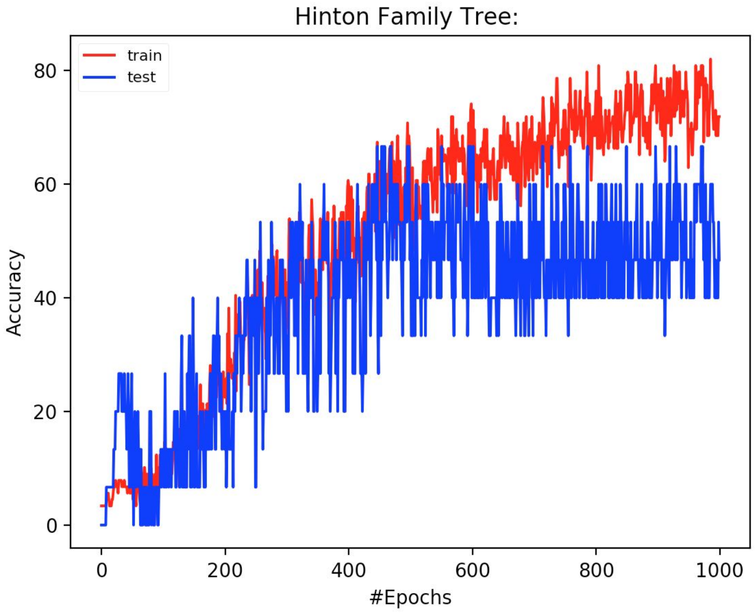
Learning Unit 1 : Youngest in English and Spanish Family

	In Spanish family Alfonso and Sophia are the youngest and correspondingly their weights are the lowest. In the same way considering the English family, Colin and Charlotte are the youngest, but only Colin has a negative weight and Charlotte has a very small positive weight. Considering that Charlotte weight assumption is wrong, then we can consider this unit learns the youngest person.

3.a)

<u>Over 20 splits:</u> **Mean Test Accuracy:** 48.326741573, **Standard Deviation of Test:** 0.0871072601501

Graph for one of the splits for 12 hidden units:



Mean Accuracy Percentage for one of the splits:

	First 500 Epochs	Last 500 Epochs	Total 1000 Epochs
Training	23.683	49.402	40.234

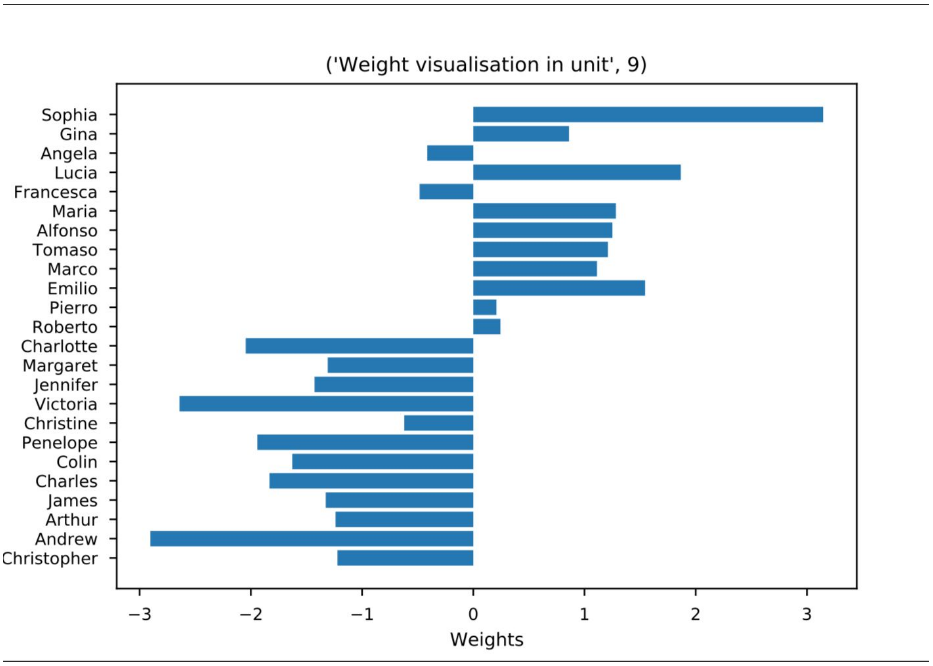
Testing	25.053	46.213	44.64

Like the 2 a), the last 500 epochs has a better mean accuracy when compared to the previous 500 epochs. Around 300-600 epochs the system is able to overcome the plateau.

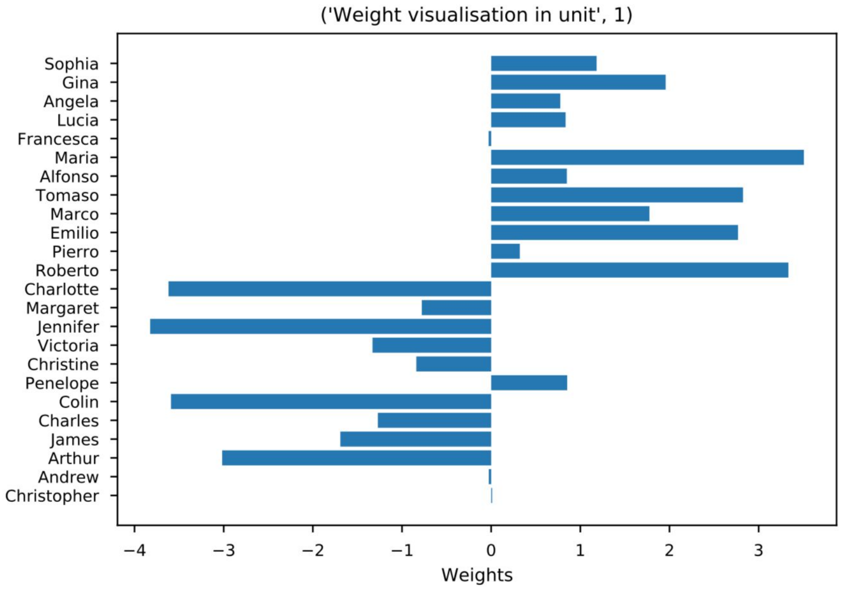
3. b) Increasing the number of hidden units seems to increase the average accuracy over 20 splits. The hidden units starts to learn new features that were not initially learnt when the hidden units were 6 in number. Now the weights are not interpretable as easily as the previous ones, but intuitively they are representing information that helps in generalisation much better than the specific 6 layer architecture. But the deviation for 12 hidden units is larger than 6 hidden units, signifying that there are a more number of variations possible due to increase value of freedom in hidden layer.

3. c) It was a difficult task interpreting what these hidden units were doing when number of hidden unit was 12. There was more generalisation and finding a pattern recognizable by man was more difficult. There were more redundant weight layers that could have meant doing the same type of feature extraction. Moreover I was able to find couple of patterns in the weight matrix.

Learning from Unit 1 and 9: Type of Family

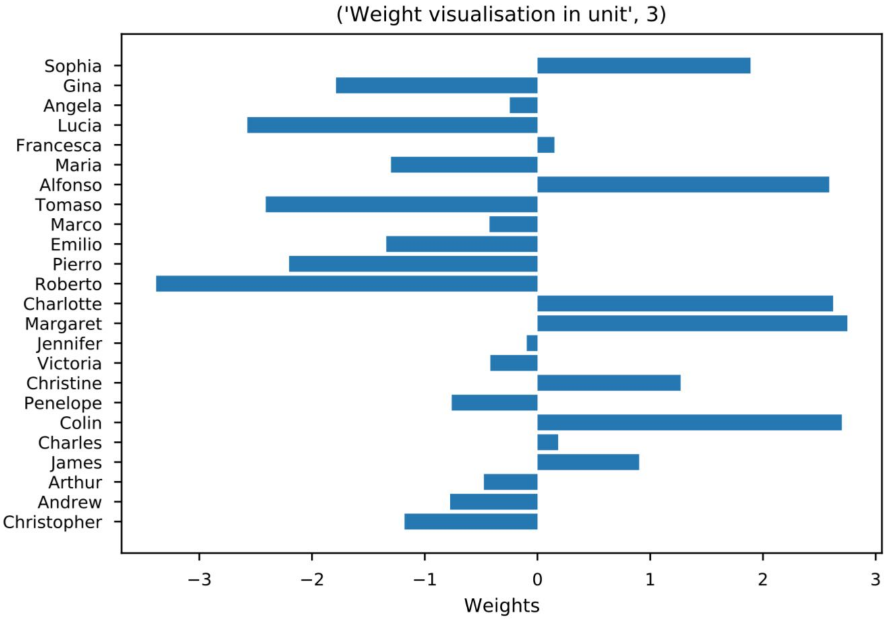


A clear distinction of positive weights referring to the Spanish Families(with two exceptions 'Angela' and 'Francesca')
And negative weight meaning English families.



Similarly another unit, Unit 1 also does the same thing with 'Colin' as exception. Therefore as the number of units increases, the features extractions can happen in redundant units.

Learning from Unit 3: Age



This is a far fetched idea, considering there might be a huge age difference between generations.Young person have positive weights and older persons have lower weights.
Considering the English Family, People in the second generation are shifting between young and old . For example, Colin might have an uncle Charles almost as old as him. But his Father James, must be older than Colin. So this is a kind of a tricky generalisation.