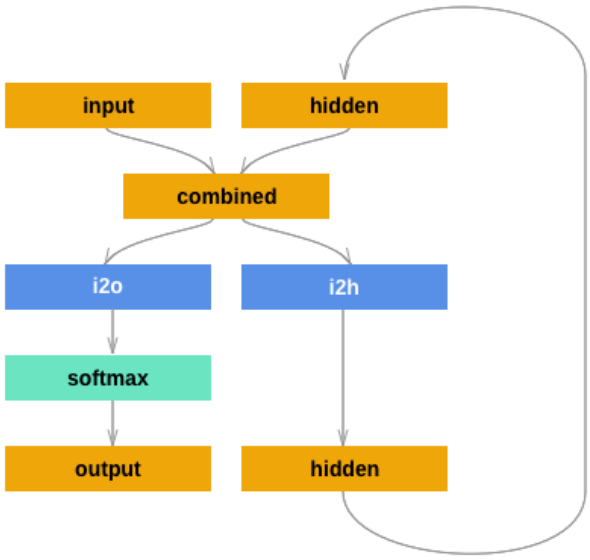
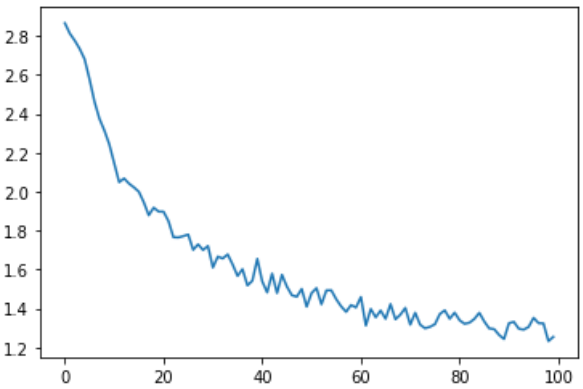
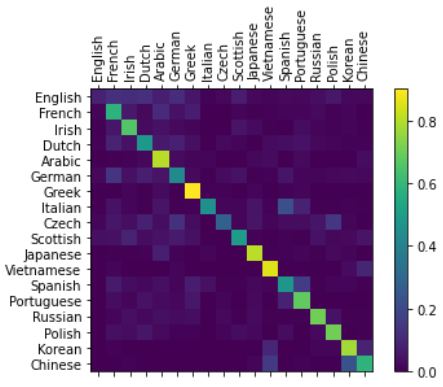
Q1(a). The main difference between RNN and normal feedforward neural network is that it stores the previous states in a network. In this model we are trying to classify words into their respective languages. The input to the feedforward neural network is the hidden states from the last iteration and the actual input combined. The two outputs of the RNN are the combined data passed through the feedforward network and the hidden states. As there are no such hidden states available in the first iteration, the hidden states in the first iteration are kept zero. The feedforward network output is the predication that is which class the word belongs to.
Each file in the given data has a bunch of names, one name per line which is converted to dictionary of lists of names per language.
I have tried by adding a greater number of layers, increasing the number of units in each layer in the feedforward portion of the network, and also changing the hyperparameters like the learning rate and number of iterations.
This model is a many to one application of RNN. Other such applications can be found in sentiment classification of a sentence.



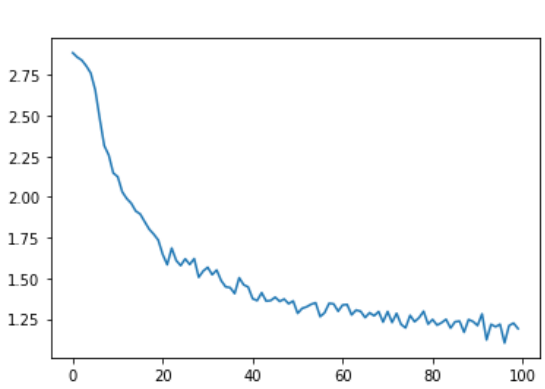
[Run - 1] No of Hidden Units in each layer - (Input+hidden, output) [original model]
Learning Rate – 0.005
No. of Iteration – 100000

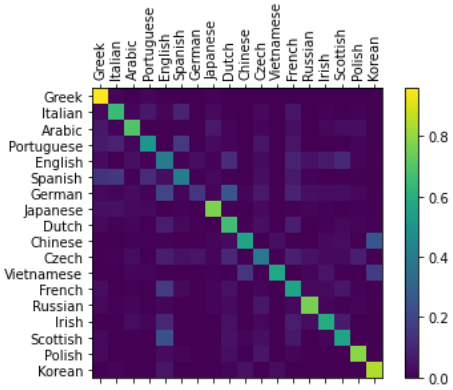




The English is not predicted as English as we can understand from the confusion matrix, probably because of similarity in the languages.

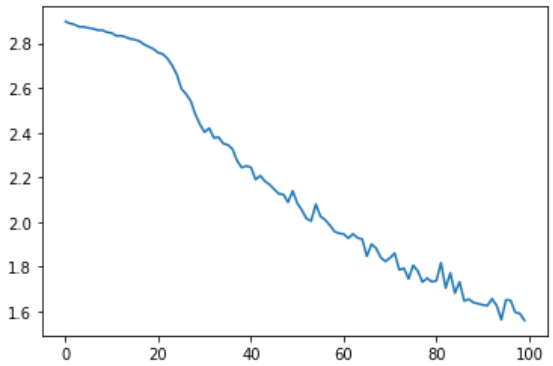
[Run - 2] No of Hidden Units in each layer - (Input+hidden, 60, output) [1 linear layer increased from the previous]
Learning Rate – 0.005
No. of Iteration – 100000

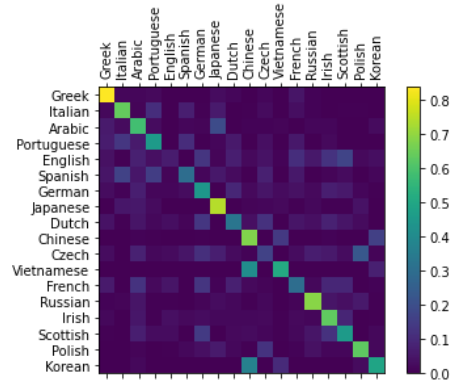




The English is better predicted as English in this model although similar problem can be seen in German as actual German is not being predicted as any other languages as such, but other languages are being predicted as German. In addition to that, some of the yellow spots that were previously there, like Vietnamese and Japanese seems to be green shifted.

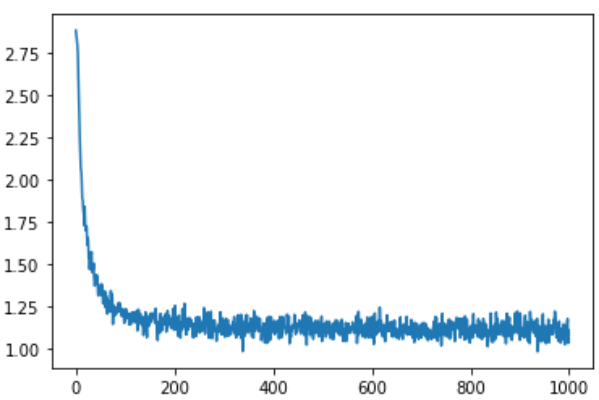
[Run – 3] No of Hidden Units in each layer - (Input+hidden, 60, output)
Learning Rate – 0.001 [LR reduced from 0.005 – model should take more time to learn]
No. of Iteration – 100000

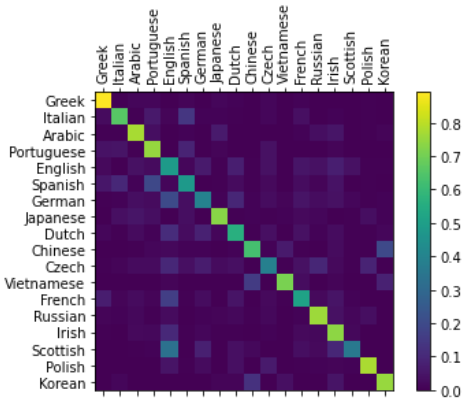




As expected, the model takes higher time to learn, as the decrease rate of the loss is reduced. But again, there arises the problem with English, and the number of yellow spots along the diagonal are also reduced.

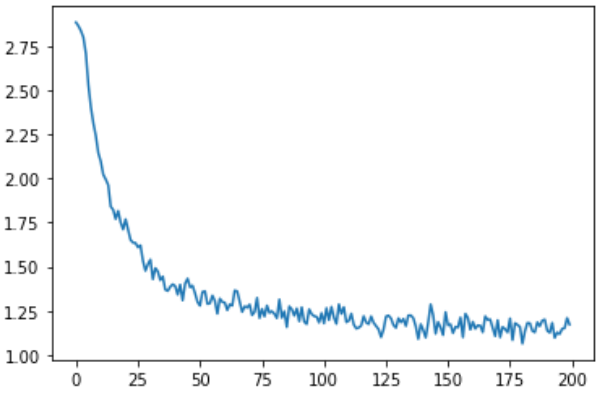
[Run -4] No of Hidden Units in each layer - (Input+hidden, 60, output)
Learning Rate – 0.005 [Changed back to the previous value]
No. of Iteration – 1000000 [increased by 10 times – this takes almost 20mins to train]

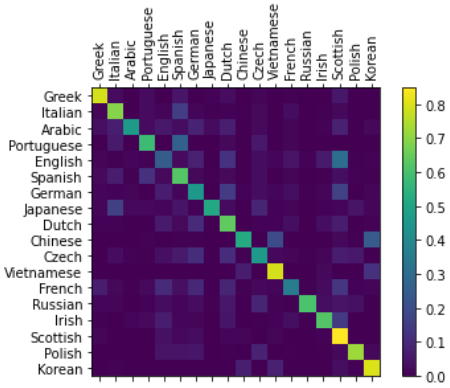




The loss does not seem to change as such although it goes up and down throughout within a certain range. But all the diagonal spots are much brighter than the rest of the graph in the confusion matrix with English being predicted as English at higher percentage. Although, there are still some English names predicted as Scottish.

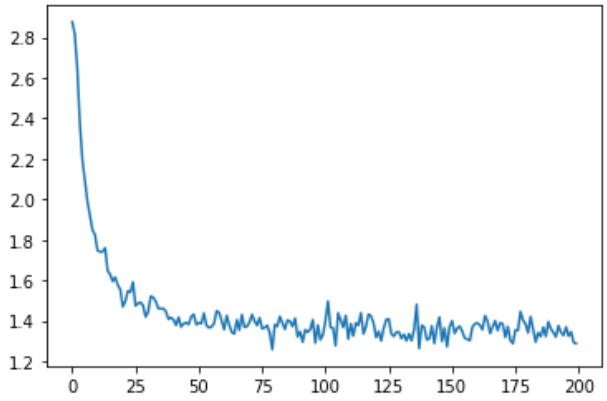
[Run - 5] No of Hidden Units in each layer - (Input+hidden, 60, output)
Learning Rate – 0.005
No. of Iteration – 200000 [reduced from the previous value as there was no significant change in the loss]

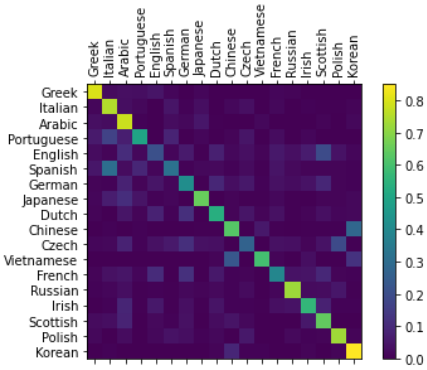




The Loss function shows a steady decrease although there are still ups and downs. The diagonal in the confusion matrix still remains bright although some of the spots outside the diagonal becomes brighter than the previous graph which might go away increasing the number of iterations.

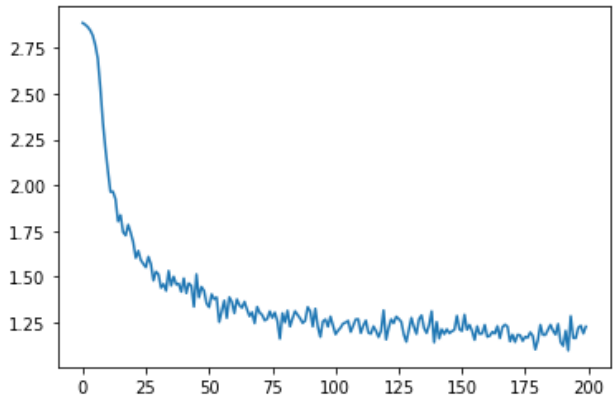
[Run – 6] No of Hidden Units in each layer - (Input+hidden, 60, output)
Learning Rate – 0.01 [LR increased from 0.001 – model should take lesser time to learn]
No. of Iteration – 200000

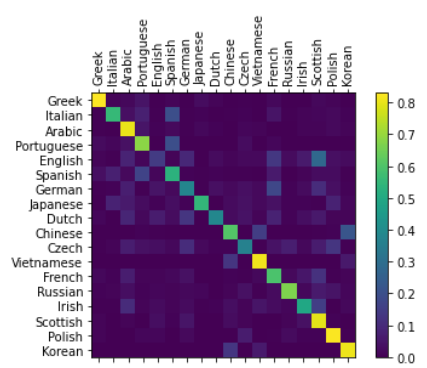




The learning rate is increased and as a result the loss curve faces a steep down as the model learns faster although the saturation happens almost at the same region. The yellow spots in the diagonal does not reduce that much thus accuracy is not that compromised even though the run takes a lot less time.

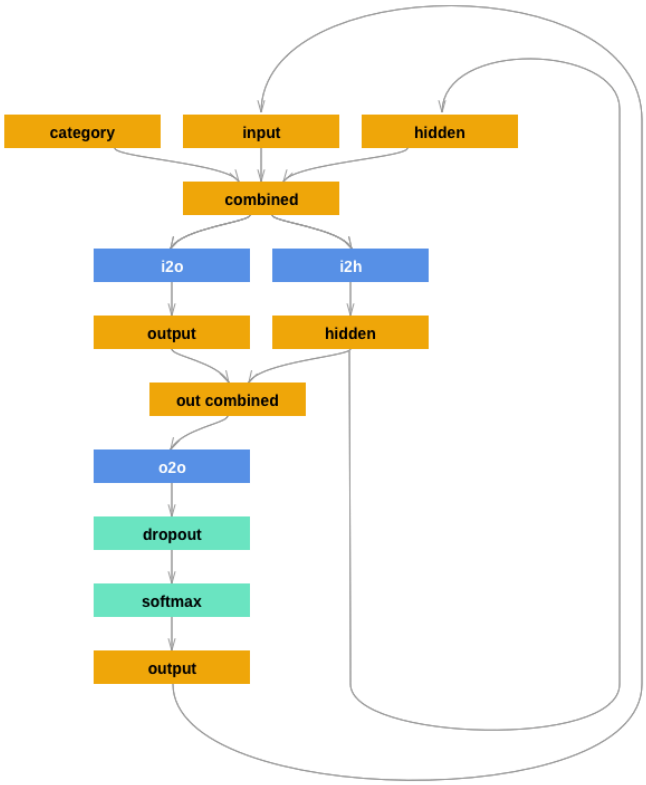
[Run – 7] No of Hidden Units in each layer - (Input+hidden, 128, 64, output) [One extra linear hidden layer introduced]
Learning Rate – 0.005
No. of Iteration – 200000



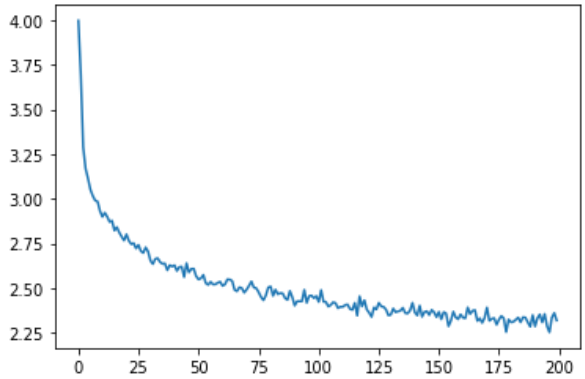


The model seems to learn faster although the learning rate is 0.005. There are quite a few numbers of yellow spots along the diagonal although the prediction of English is again merged with Scottish.
Interestingly Greek is well predicted in all of the models may be because it has no similarity with the other languages.

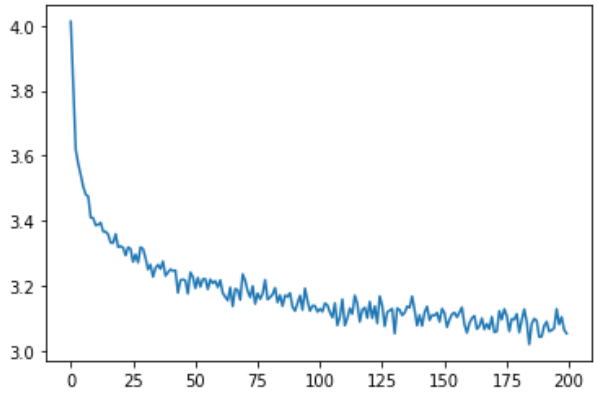
Q1(b). This model is application of one-to-many application of RNN. Previously we used RNN to classify names into its respective languages, this time we turn around and generate names from different languages. Thus, the input to this model is a bit different in the way that instead of redirecting a category after reading in all the letters of a name, we input a category and output one letter at a time.
The data provided is again a bunch of plain text files with a name per line which are converted into arrays and finally into dictionaries.
The extra argument that is passed on here is the category which is combined with others. We are interpreting the output probability of the net layer. An extra layer was already added to the original model, I tried to increase the number of units in the hidden layer after combining the hidden and the output. The dropout was fixed at 0.1, I tried changing it to higher values, but the loss seems to increase considerably more than lowering the time of computation. The dropout is used basically to fuzz the inputs to prevent overfitting.
One to many RNN have applications in music generations.



[Run – 1] No of Hidden Units in middle layer – 128 [original model]
Dropout – 0.1
Learning Rate – 0.005
No. of Iteration – 100000

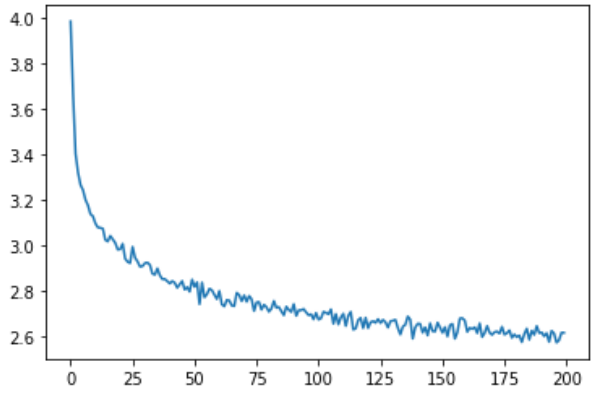


[Run – 2] No of Hidden Units in middle layer – 128
Dropout – 0.5 [dropout is increased 5 times]
Learning Rate – 0.005
No. of Iteration – 100000



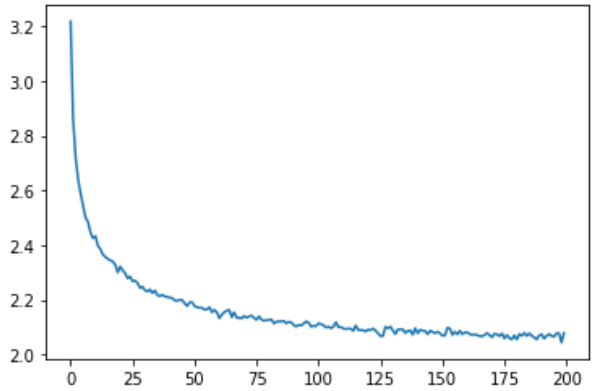
The scale of loss shows that the loss is not decreasing after a certain high value which is not good.

[Run – 3] No of Hidden Units in middle layer – 128
Dropout – 0.25 [half of the previous value]
Learning Rate – 0.005
No. of Iteration – 100000



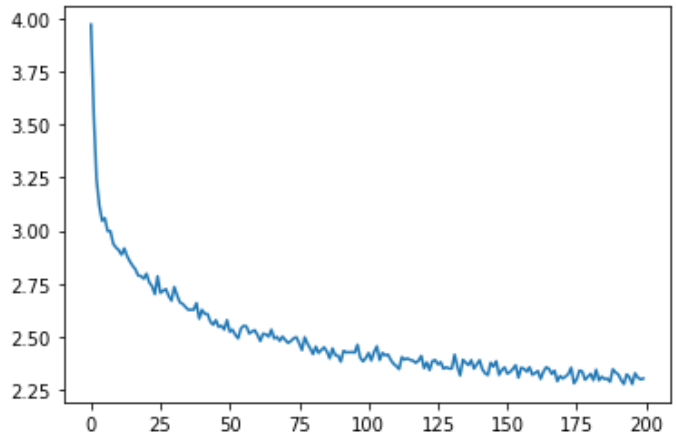
The scale of loss shows that the loss is not decreasing after a certain high value which is not good.

[Run – 4] No of Hidden Units in middle layer – 128
Dropout – 0.1
Learning Rate – 0.005
No. of Iteration – 1000000 [no of iterations increased 10 times, this takes almost an hour to compute]



The loss is decreased considerably although after certain iterations it does not decrease much.

[Run – 5] No of Hidden Units in middle layer – 256 [doubled]
Dropout – 0.1
Learning Rate – 0.005
No. of Iteration – 100000



There is not much significant change in the loss.