DEEP LEARNING & APPLICATIONS

Recurrent Neural Networks & LSTM

25th May 2022

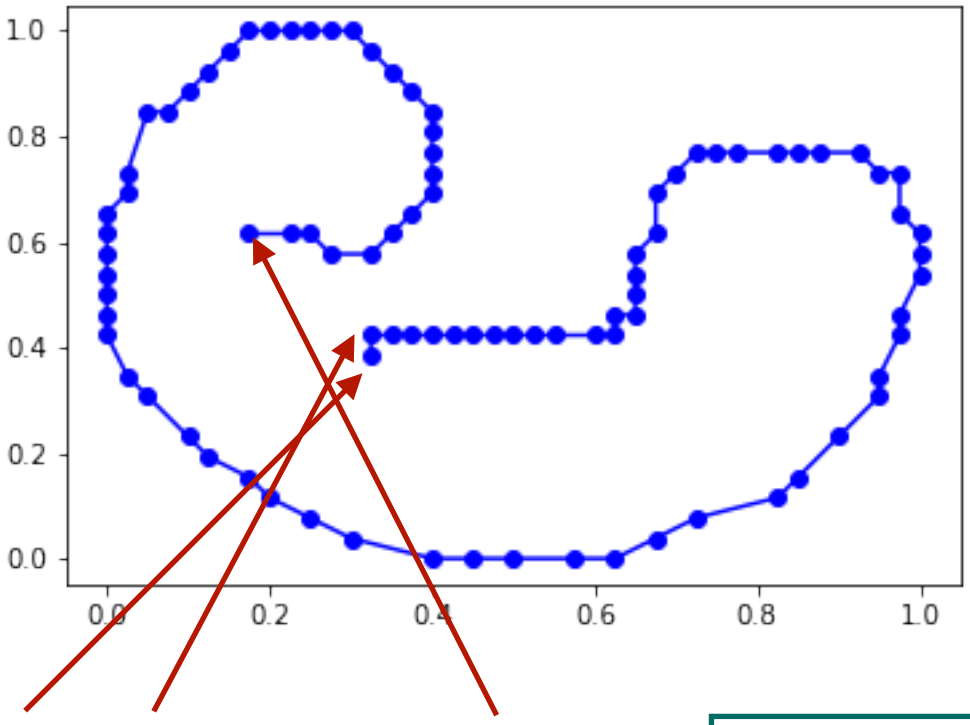
CONTENTS

DATA SETS

DATASET I: HANDWRITTEN DATA

Handwritten character dataset: This dataset consists of subset of handwritten characters from Kannada/Telugu script. Each character is seen as sequence of 2-dimensional points (x and y coordinates) of one stroke of character (pen down to pen up).
There are five character classes namely: a, ai, bA, chA, tA
Each file is of .txt extension and several preprocessing are be done.

The text files are normalised (0 to 1) and saved as images. Each letter in each of the classes is a set of 2D points, for example the character ‘a’ is plotted and shown for reference.

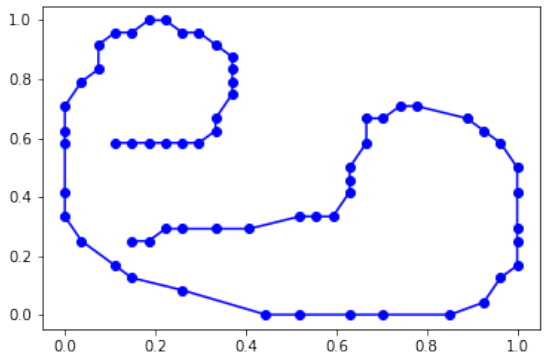
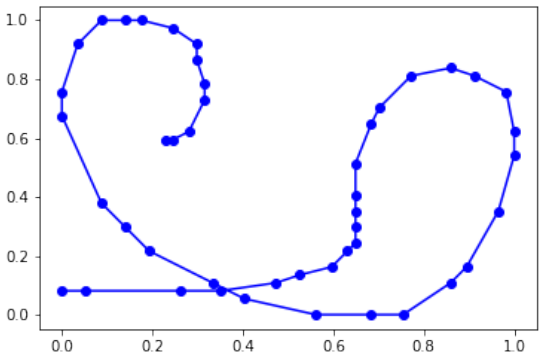
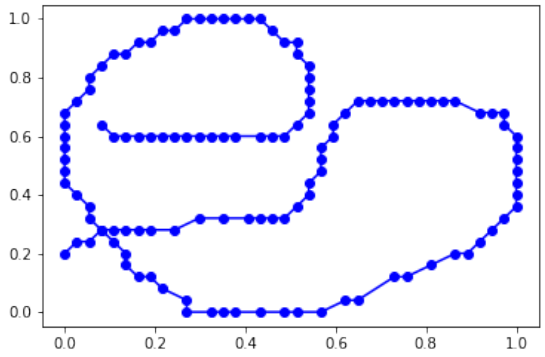
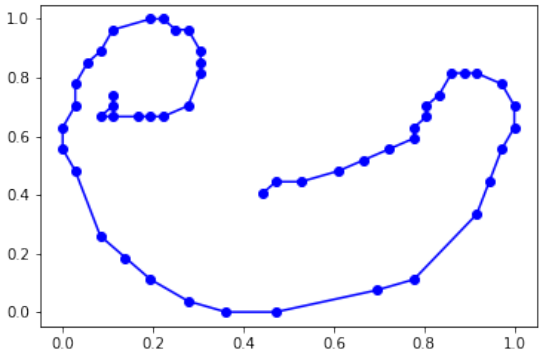
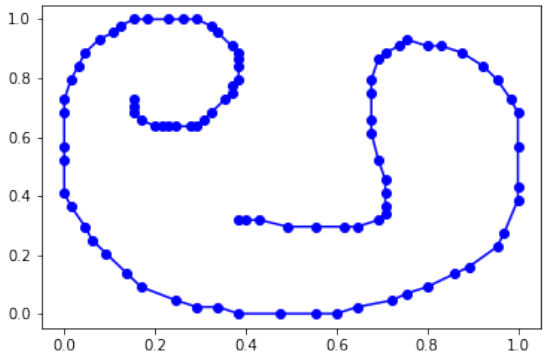
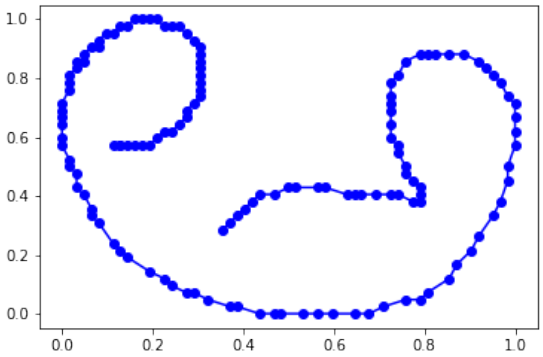
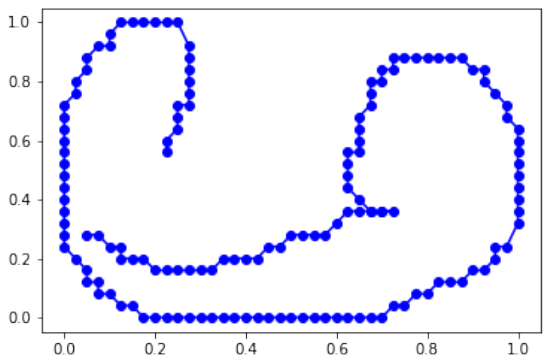
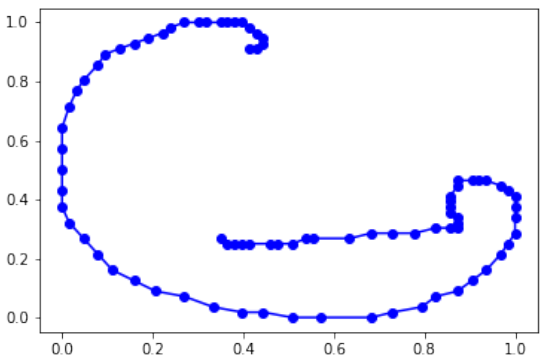


$$x_1 \quad x_2 \quad \cdots \quad x_{82}$$

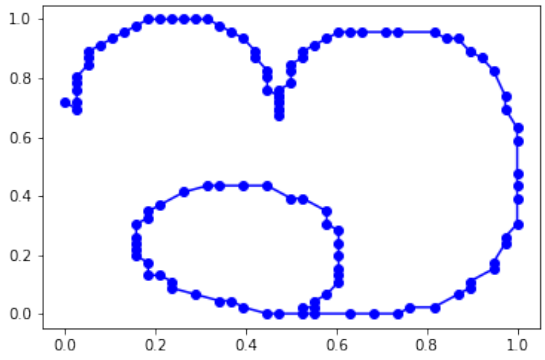
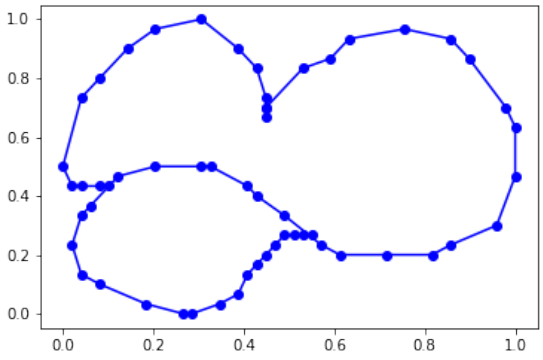
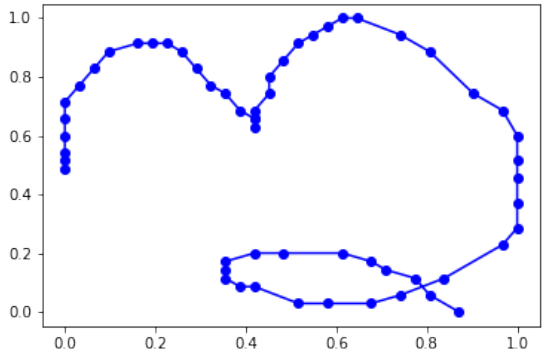
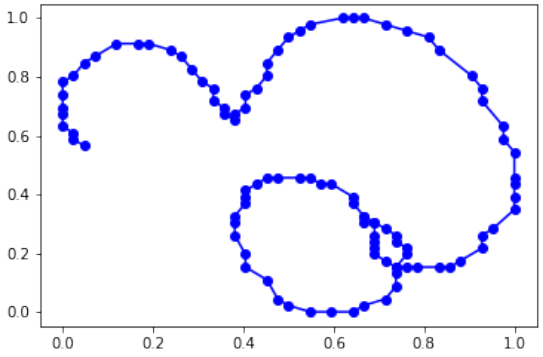
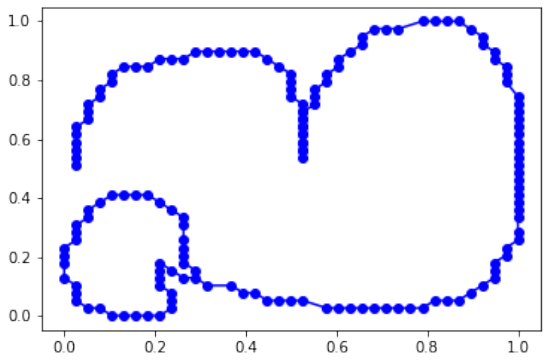
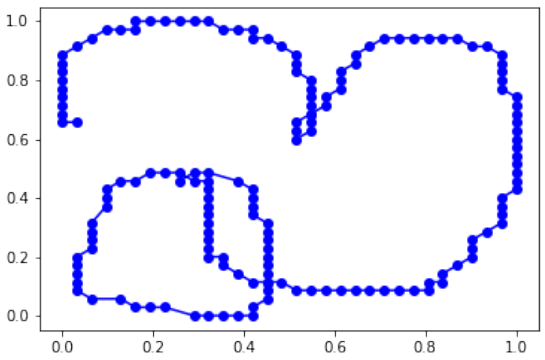
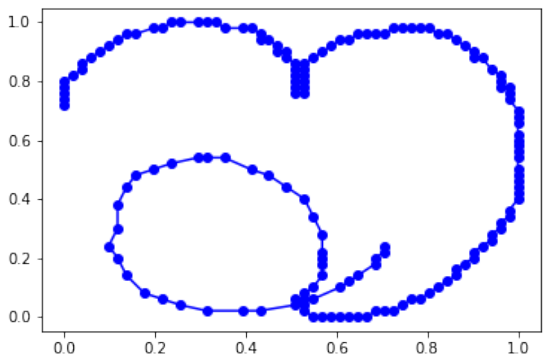
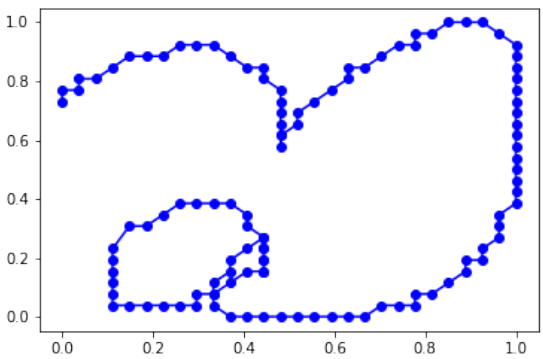
$$x_1 = \begin{bmatrix} 0.35 \\ 0.39 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0.35 \\ 0.41 \end{bmatrix} \quad \cdots \quad x_{82} = \begin{bmatrix} 0.18 \\ 0.62 \end{bmatrix}$$

Each character in dataset is of variable length.

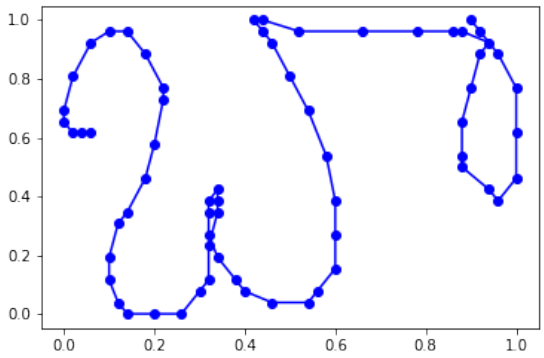
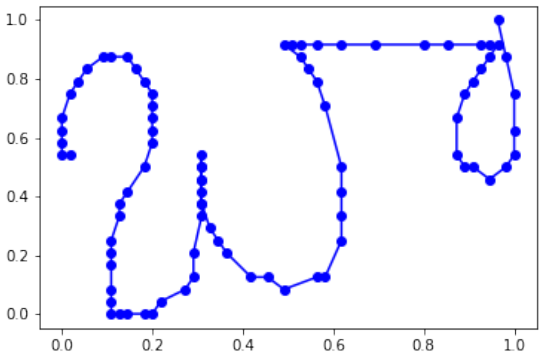
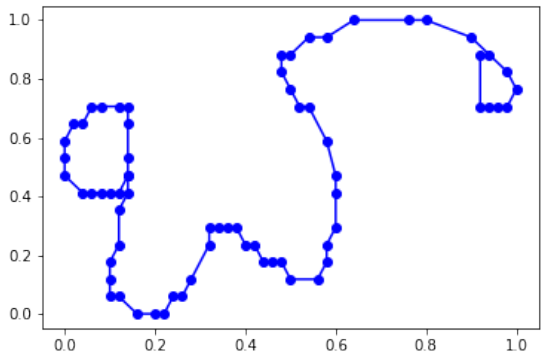
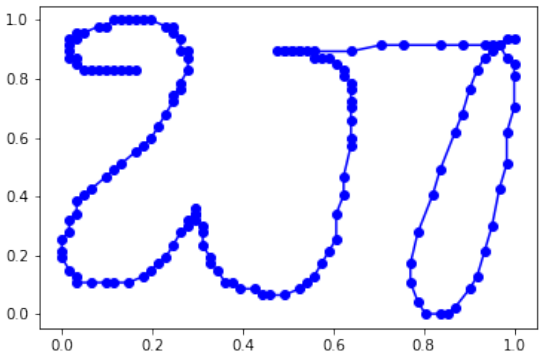
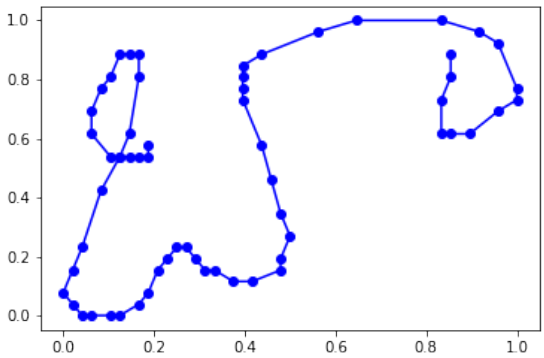
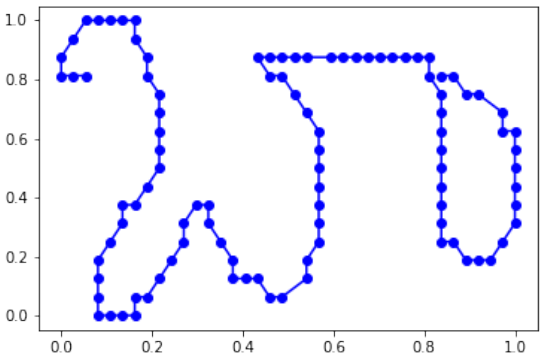
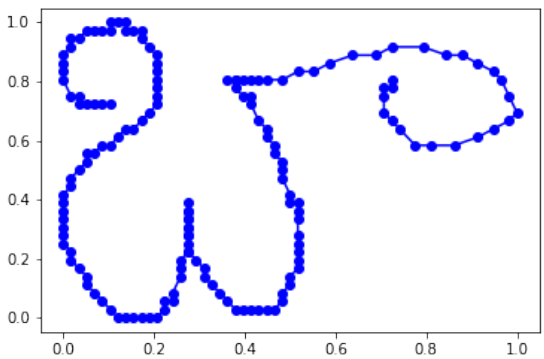
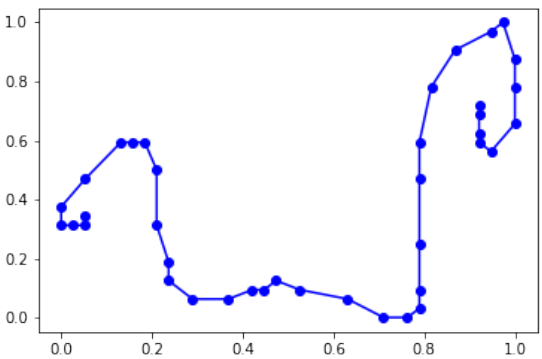
THE CHARACTER: a

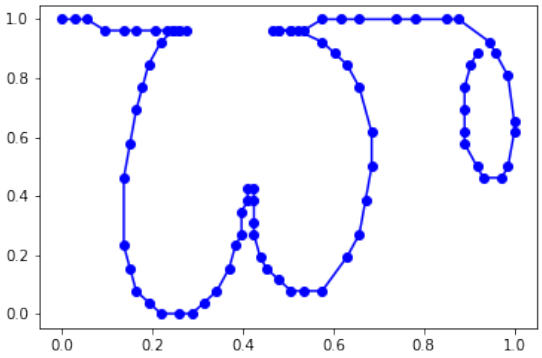
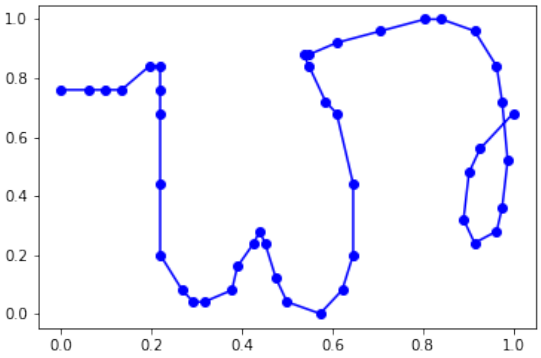
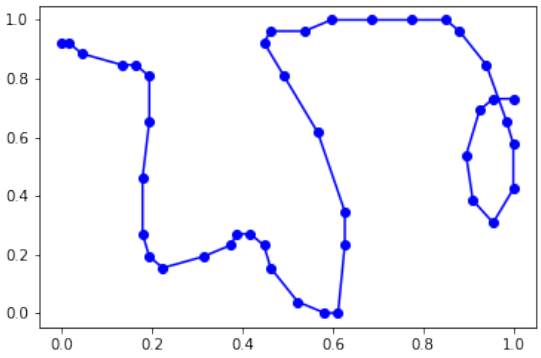
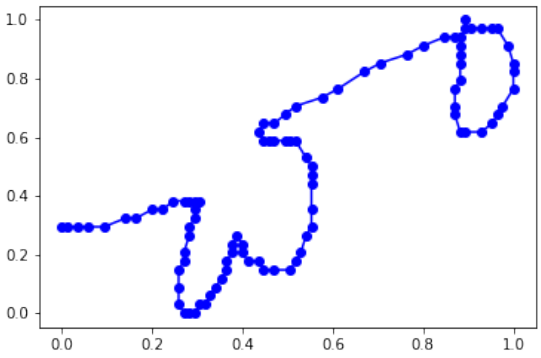
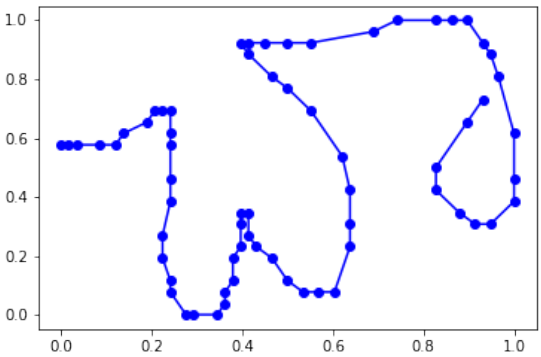
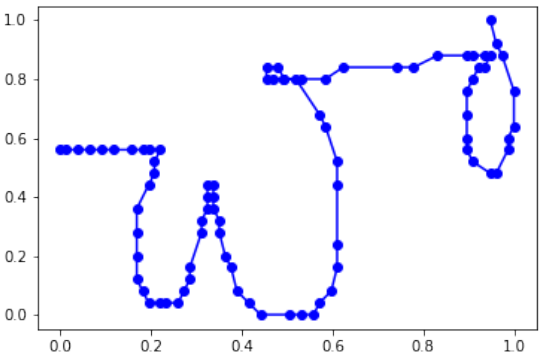
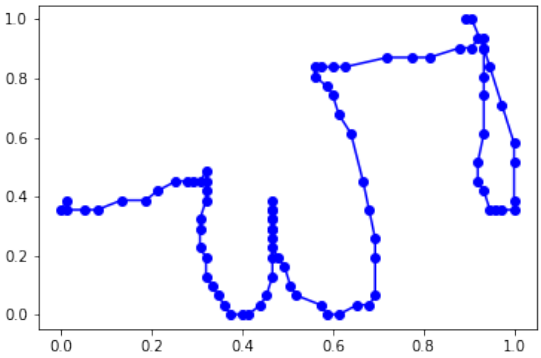
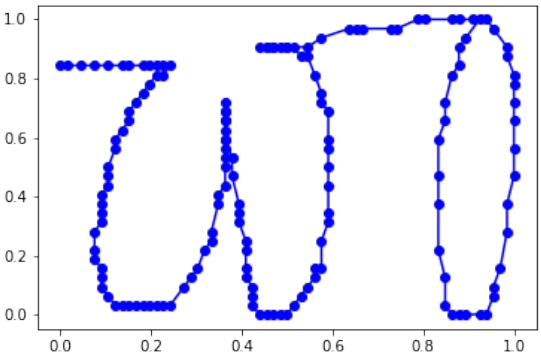
THE CHARACTER: ai

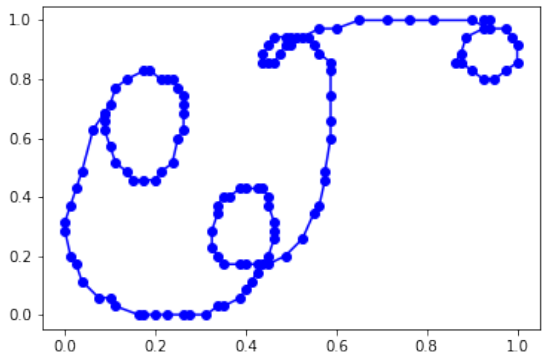
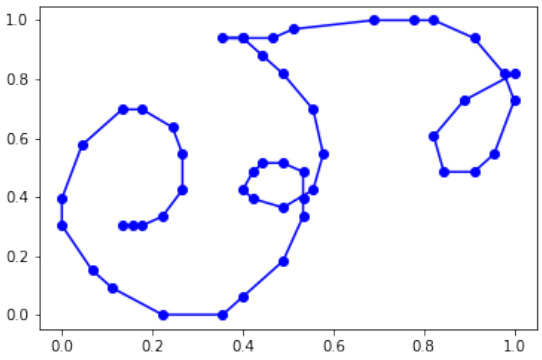
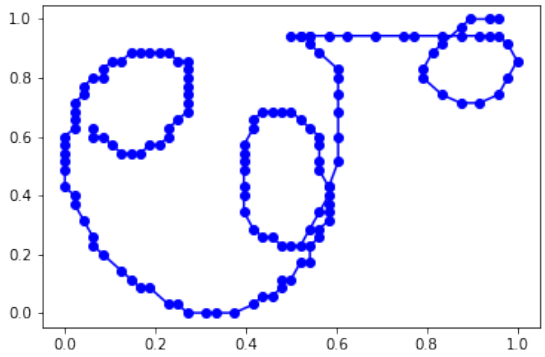
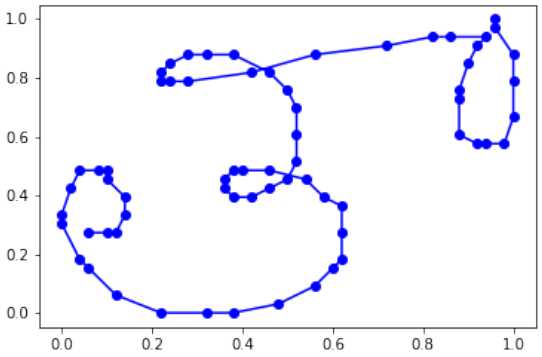
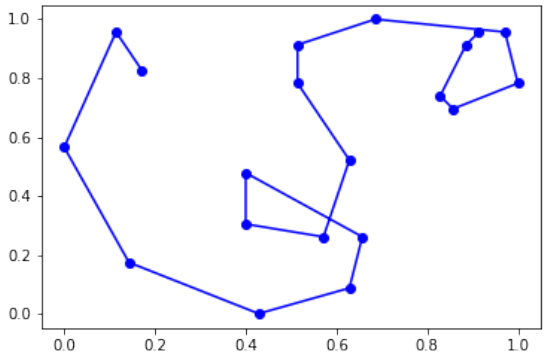
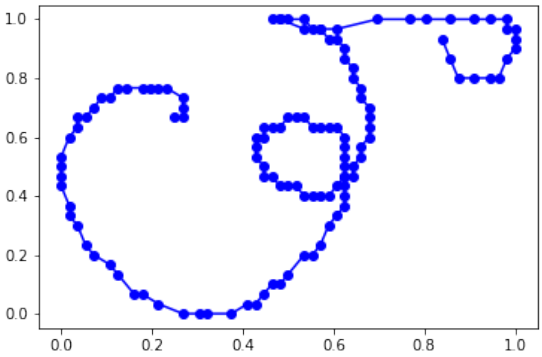
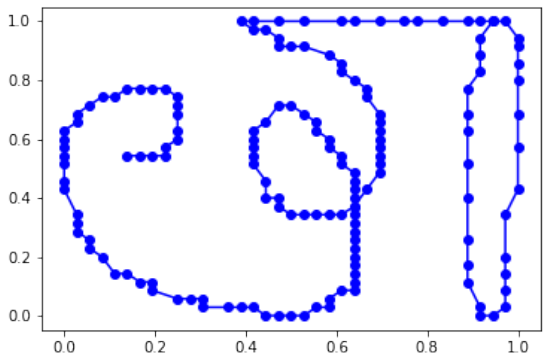
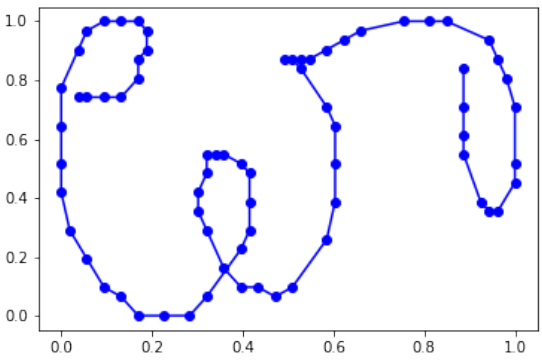
THE CHARACTER: bA

THE CHARACTER: chA

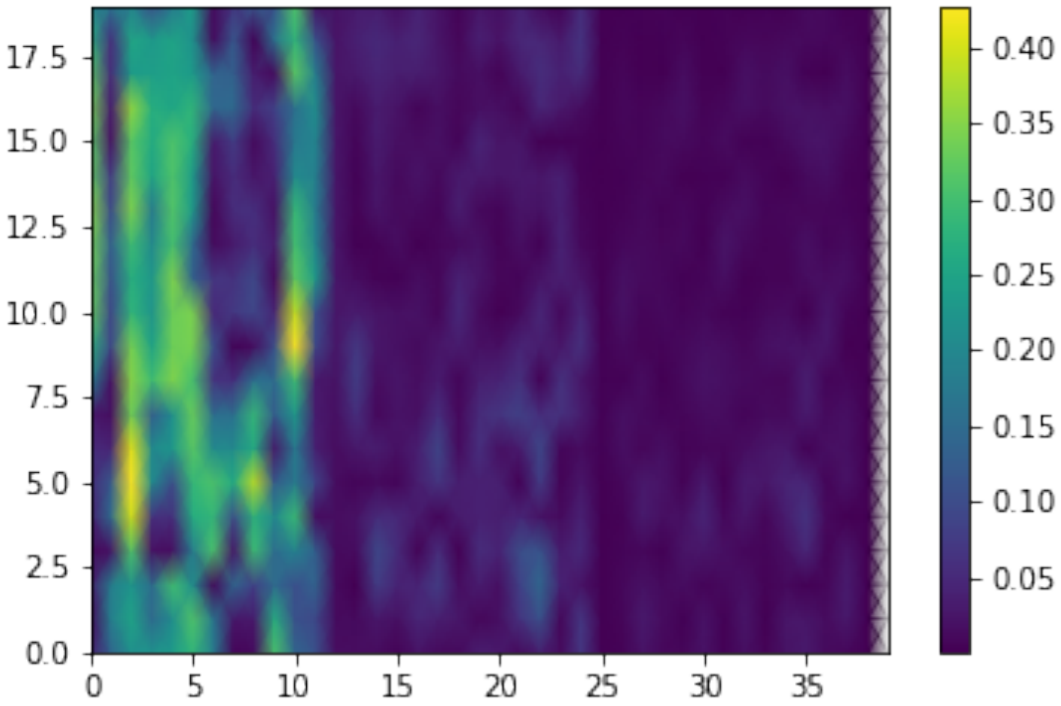
THE CHARACTER: tA

DATASET II: CONSONANT VOWEL SEGMENT DATA

This dataset consists of subset of CV segments from a conversational speech data spoken in Hindi language. There are data for 5 CV segments (5-class data).
The 39-dimensional Mel frequency cepstral coefficient (MFCC) features extracted frame-wise from utterances of a particular CV segment uttered by multiple people are provided in separate files corresponding to each class. Each data file is considered as one sample. Each row in a data file indicates one 39-dimensional MFCC feature vector. The number of such feature vectors (rows) depend on the duration of speech segment.

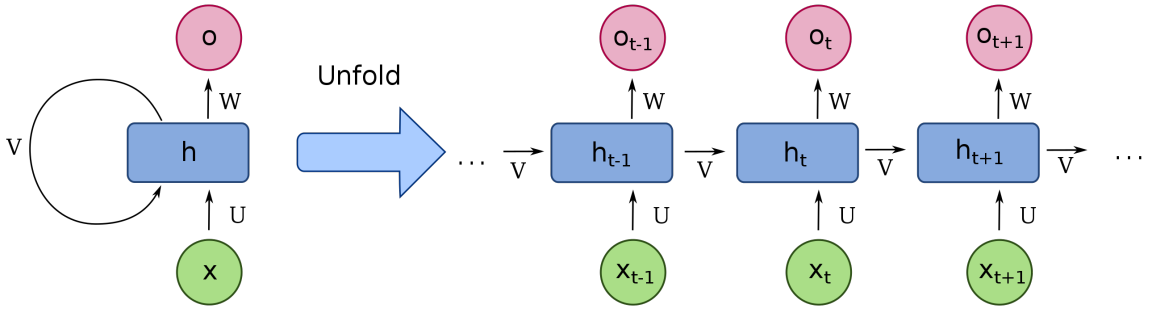
MFCC FEATURE SPECTOGRAM OF THE CHARACTER: Kaa



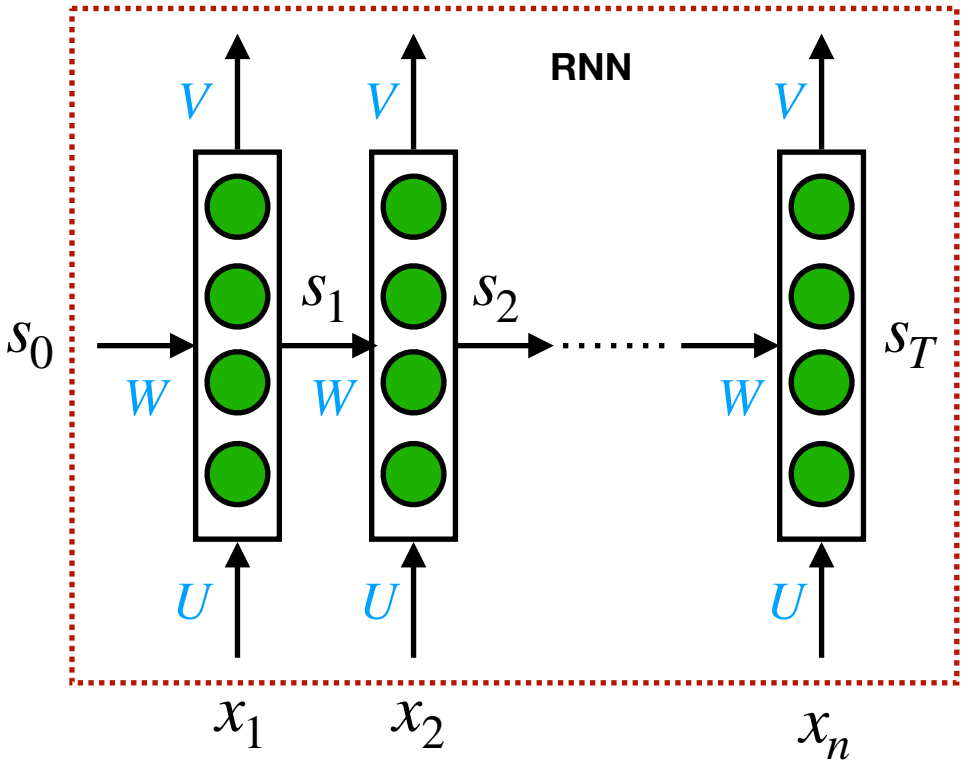
For example, the utterance of a particular segment in terms of MFCC feature is shown above. This utterance segments are used to classify data. There are five classes of data namely: Kaa, ka, sa, ba, ta.

RECURRENT NEURAL NETWORKS (RNN)

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.



https://en.wikipedia.org/wiki/Recurrent_neural_network



LEARNING: BACK PROPAGATION THROUGH TIME

Learning is performed by gradient descent method and error back propagation algorithm. In Forward computation: Inner product computation over all the time steps and activation function computation over all the time steps. In Backward operation: Error/Loss calculation and propagation over all previous time steps and modification of weights over all previous time steps. For example, classification of data set I, suitable loss and output function is cross entropy and soft max.

- Initialising the parameter set $\theta = \{U, V, W, c, b\}$ randomly and the network predicts the probabilities
- Computing the loss over each time step: $L_t(\theta)$. Total loss is the sum of the loss over all time steps $L(\theta) = \sum_{t=1}^{T} L_t(\theta)$ where T is the number of time steps.
- Back propagate the total loss and update the parameter set θ of the network. For back propagation we need to compute the gradients w.r.t. U, V, W , c, b.
- Then the change in the weight matrix V, $\Delta V = -\eta \Delta_v L(\theta)$ where $\Delta_v L(\theta) = \frac{\partial L(\theta)}{\partial V} = \sum_{t=1}^{T} \frac{\partial L_t(\theta)}{\partial V}$. Each term in the summation is simply the derivative of the loss w.r.t. the weights in the output layer. This gradient computation is same as that of the back propagation in fully connected neural network (between hidden and output layer).
- Updating the weight V, $V = V + \Delta V$
- Then the change in the weight matrix U, $\Delta U = -\eta \Delta_u L(\theta)$ where $\Delta_u L(\theta) = \frac{\partial L(\theta)}{\partial U} = \sum_{t=1}^{T} \frac{\partial L_t(\theta)}{\partial U}$. Each term is the summation is the derivative of the loss w.r.t. the weights in the input/hidden layer. This gradient computation is same as that of the back propagation in fully connected neural network(between hidden layers and hidden layer to input layer)
- By chain rule, assuming single hidden layer,

$$\frac{\partial L_t(\theta)}{\partial U} = \frac{\partial L_t(\theta)}{\partial y_t} \frac{\partial y_t}{\partial a_{2t}} \frac{\partial a_{2t}}{\partial s_t} \frac{\partial s_t}{\partial a_{1t}} \frac{\partial a_{1t}}{\partial U}$$

$$y_t = \phi(a_{2t}),\ a_{2t} = Vs_t + c,\ s_t = \Omega(a_{1t}),\ a_{1t} = Ux_t + Ws_{t-1} + b$$

- Updating the weight U, $U = U + \Delta U$

- Then the change in the weight matrix W, $\Delta W = -\eta \Delta_w L(\theta)$ where

$$\Delta_w L(\theta) = \frac{\partial L(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t(\theta)}{\partial W}$$

$$\Delta_w L(\theta) = \frac{\partial L(\theta)}{\partial W} = \sum_{k=1}^{T} \frac{\partial L_t(\theta)}{\partial s_t} \sum_{k=1}^{t} \frac{\partial s_t}{\partial s_k} \frac{\partial^+ s_k}{\partial W}$$

- Updating the weight W: Back propagate over all previous time steps

$$W = W + \Delta W$$

PROBLEMS IN RNN

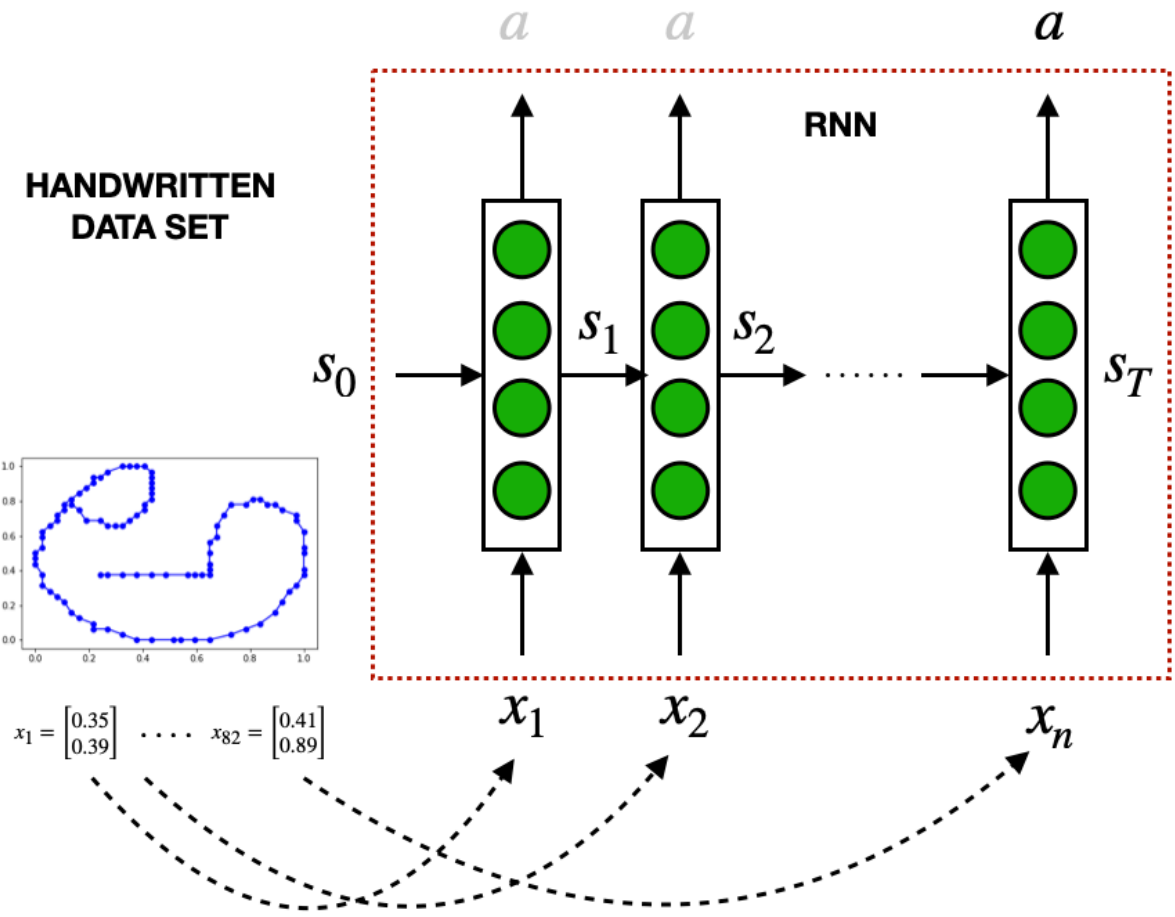
Problem of Exploding and Vanishing Gradients:

- One simple way of avoiding this is to use truncated back propagation where we restrict the product to less than t – k terms
- Other ways to avoid exploding and vanishing gradient is by using LSTM and GRU

ARCHITECTURE DETAILS

Architecture	**I**	**II**	**III**
No of RNN Layers	1	2	3
No of Hidden Layers	1	1	1
Hidden layer dimension	100	100	100

RESULTS: DATASET I



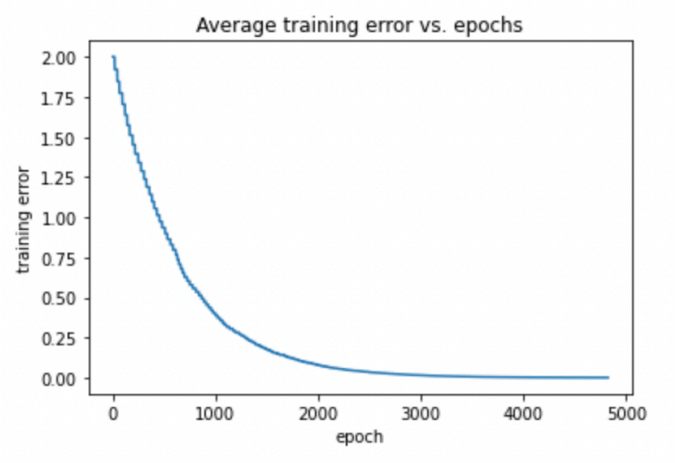
The sequence of variable length points (x,y) is passed to the RNN and the network is trained. We built a model using PyTorch as well as Keras.

For passing the variable length input in PyTorch we have used pack padded sequence available in the PyTorch module in the utils library as TORCH.NN.UTILS.RNN.PACK_PADDED_SEQUENCE.
In Keras for variable length input we can zero pad the data initially and using Embedding module we can create a mask to process the zero padded terms to work with variable data input.

The model is trained and tested using different architectures and result is presented below.

ARCHITECTURE I

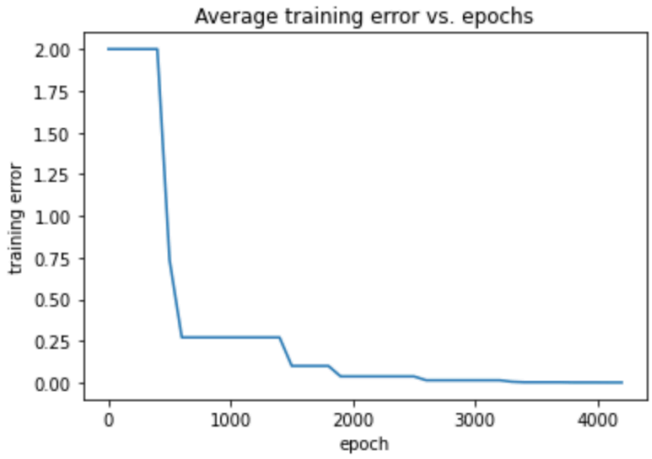
No of RNN Layers	1



Training Accuracy	0.7992
Test Accuracy	0.7321

ARCHITECTURE II

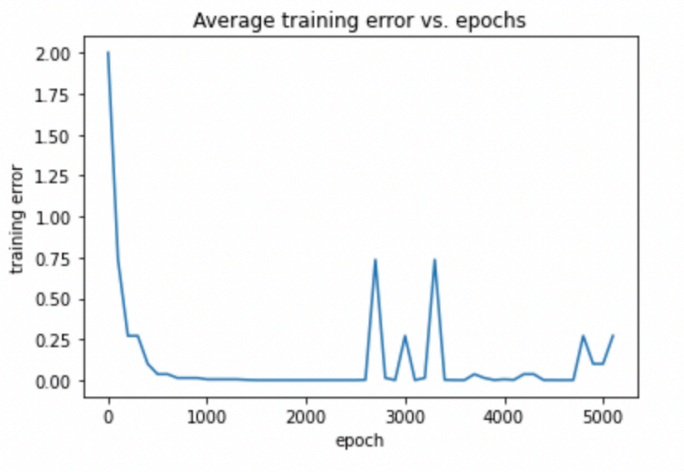
No of RNN Layers	2



Training Accuracy	0.5122
Test Accuracy	0.5089

ARCHITECTURE III

No of RNN Layers	3



Training Accuracy	0.3121
Test Accuracy	0.2221

BEST ARCHITECTURE

The architecture I performs well in terms of accuracy for this data set with number of RNN layer as 1.
Its training accuracy is obtained as 79.92% and test accuracy as 73.21%

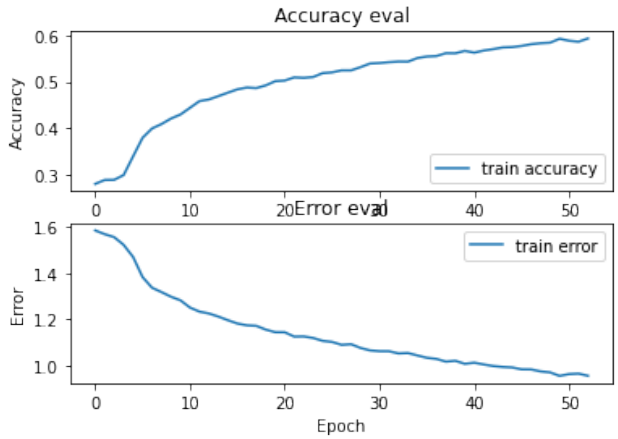
CONFUSION MATRIX

		ACTUAL CLASS				
		a	ai	bA	chA	tA
PREDICTED CLASS	a	51	12	2	0	4
	ai	4	45	3	2	0
	bA	7	6	29	31	5
	chA	4	3	25	35	4
	tA	3	4	8	2	56

RESULTS: DATASET II

ARCHITECTURE I

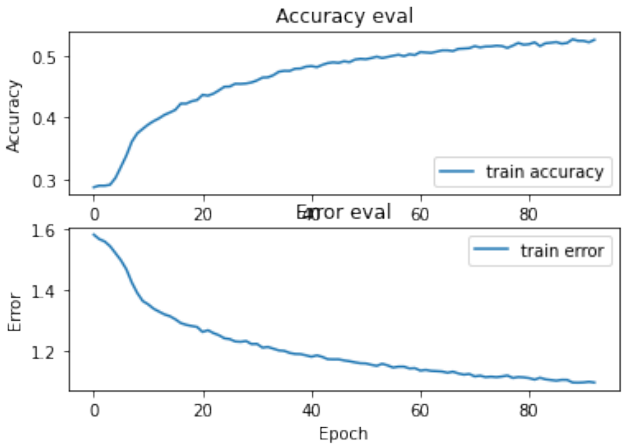
No of RNN Layers	1



Training Accuracy	0.5112
Test Accuracy	0.4551

ARCHITECTURE II

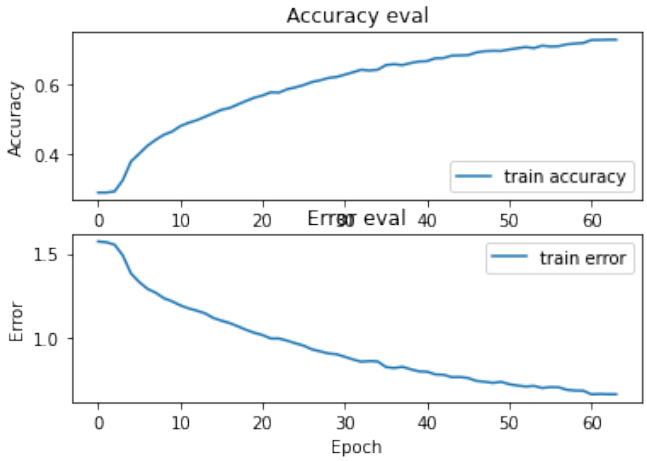
No of RNN Layers	2



Training Accuracy	0.5002
Test Accuracy	0.4646

ARCHITECTURE III

No of RNN Layers	3



Training Accuracy	0.5633
Test Accuracy	0.4748

BEST ARCHITECTURE

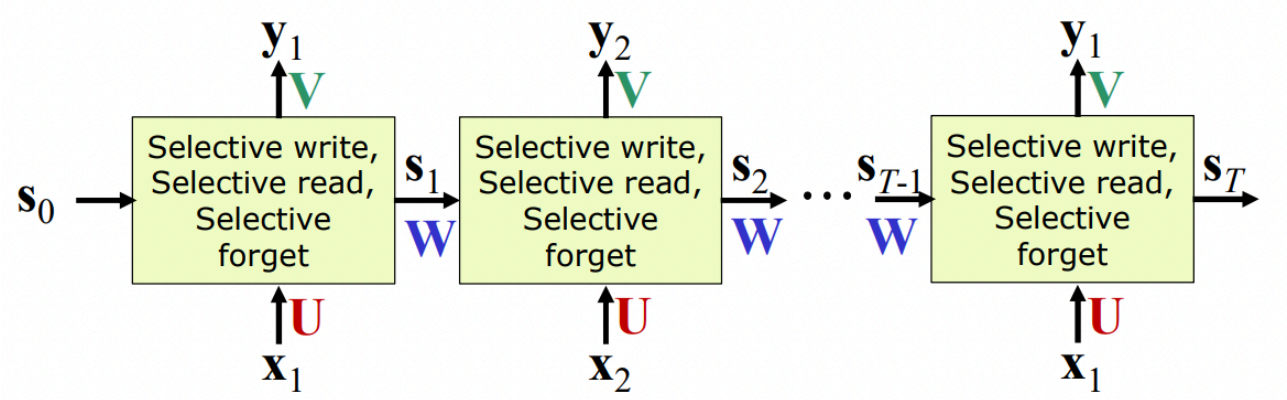
All the architecture performed very bad accuracy for this data set. Out of these the architecture III with number of RNN layer as 2
Its training accuracy is obtained as 56.33% and test accuracy as 47.48%

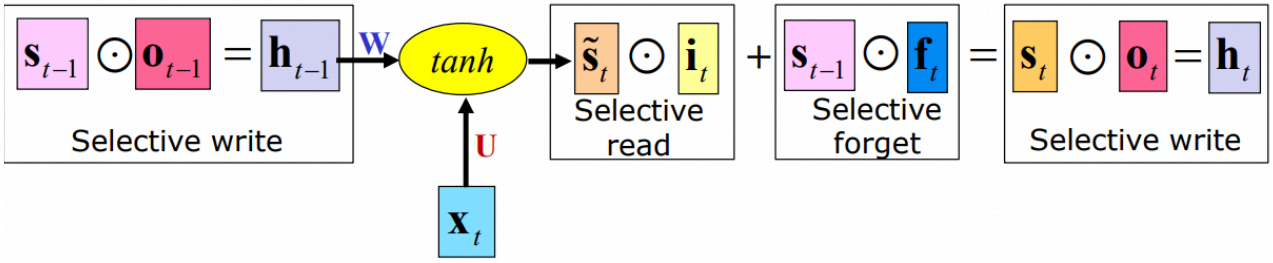
CONFUSION. MATRIX

	ACTUAL CLASS					
PREDICTED CLASS		sa	ka	kaa	tA	ba
	sa	876	102	230	105	249
	ka	253	608	840	161	275
	kaa	303	512	2118	237	748
	tA	196	155	288	1289	377
	ba	277	224	977	322	1266

LONG SHORT TERM MEMORY (LSTM)

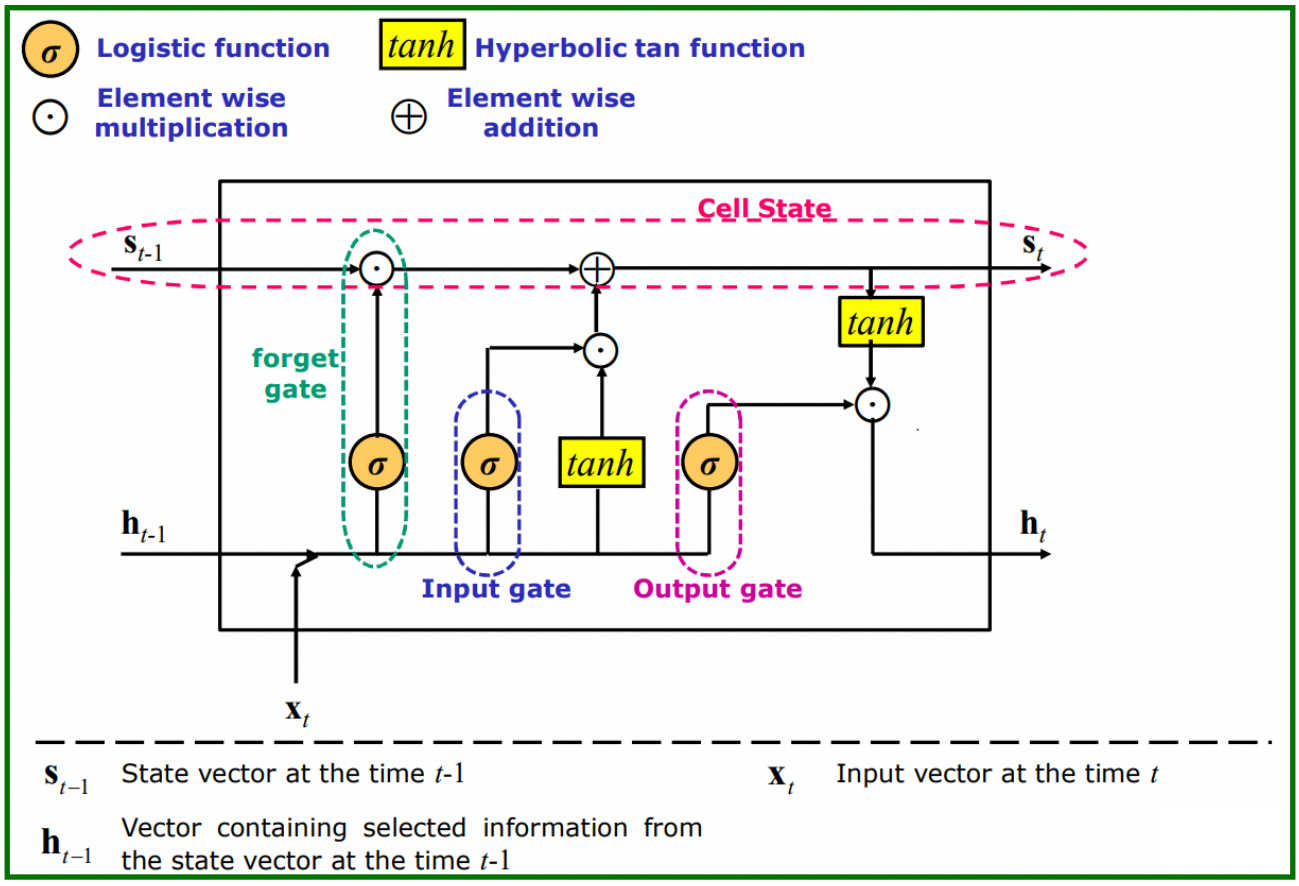
In order to overcome the disadvantages of RNN, enhanced version of RNN is introduced as LSTM. It uses the notion of selective read, selective write and selective forget as a solution to address the issues in RNN. It has many variants which include different number of gates and also different arrangement of gates.





- Previous state: $\mathbf{s}_{t-1}$
- Output gate: $\mathbf{o}_{t-1} = \text{logistic}(\mathbf{U}^{(o)}\mathbf{x}_{t-1} + \mathbf{W}^{(o)}\mathbf{h}_{t-2} + \mathbf{b}^{(o)})$
- Selective write: $\mathbf{h}_{t-1} = \mathbf{o}_{t-1} \odot \tanh(\mathbf{s}_{t-1})$
- Current (temporary) state: $\tilde{\mathbf{s}}_t = \tanh(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b})$
- Input gate: $\mathbf{i}_t = \text{logistic}(\mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{W}^{(i)}\mathbf{h}_{t-1} + \mathbf{b}^{(i)})$
- Selective read: $\mathbf{i}_t \odot \tilde{\mathbf{s}}_t$
- Forget gate: $\mathbf{f}_t = \text{logistic}(\mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{W}^{(f)}\mathbf{h}_{t-1} + \mathbf{b}^{(f)})$
- Selective forget: $\mathbf{f}_t \odot \mathbf{s}_{t-1}$
- Current (final) state: $\mathbf{s}_t = (\mathbf{f}_t \odot \mathbf{s}_{t-1}) + (\mathbf{i}_t \odot \tilde{\mathbf{s}}_t)$

ONE LSTM CELL



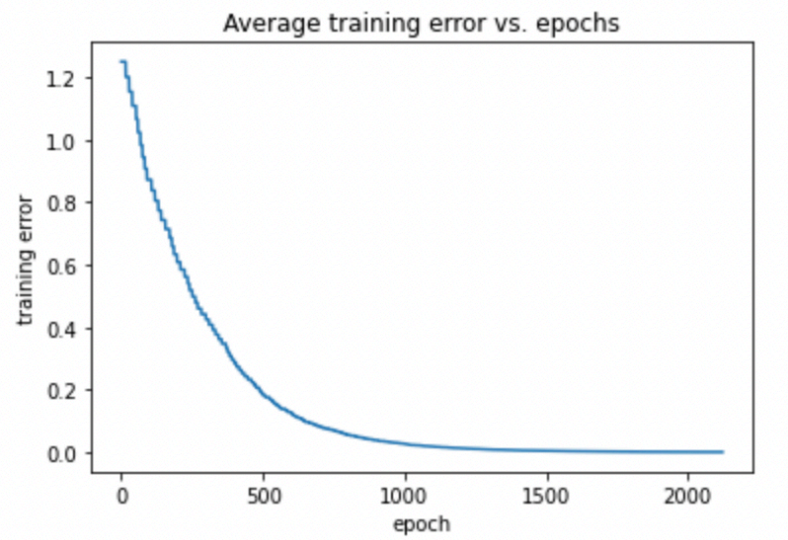
ARCHITECTURE DETAILS

Architecture	I	II	III
No of RNN Layers	1	2	3
No of Hidden Layers	1	1	1
Hidden layer dimension	100	100	100

RESULTS: DATASET I

ARCHITECTURE I

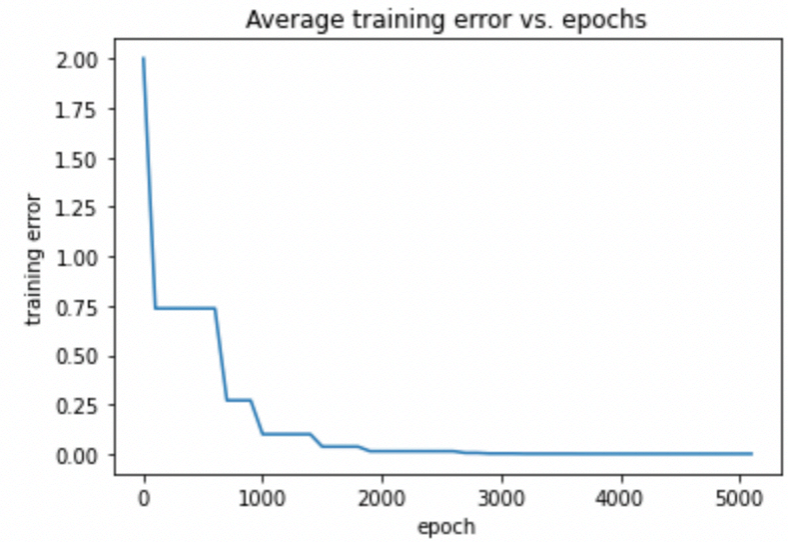
No of RNN Layers	1



Training Accuracy	0.8012
Test Accuracy	0.7194

ARCHITECTURE II

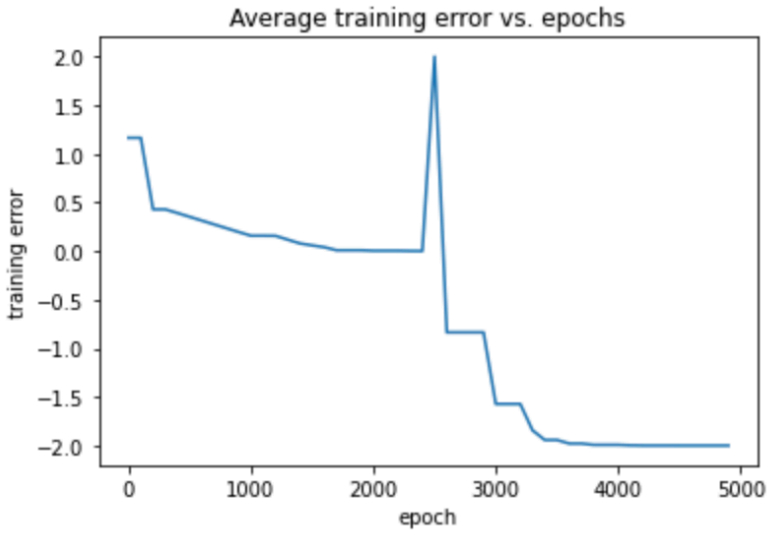
No of RNN Layers	2



Training Accuracy	0.6533
Test Accuracy	0.4317

ARCHITECTURE III

No of RNN Layers	3



Training Accuracy	0.4121
Test Accuracy	0.2010

BEST ARCHITECTURE

The architecture I performs well in terms of accuracy for this data set with number of RNN layer as 1.
Its training accuracy is obtained as 80.12% and test accuracy as 71.94%

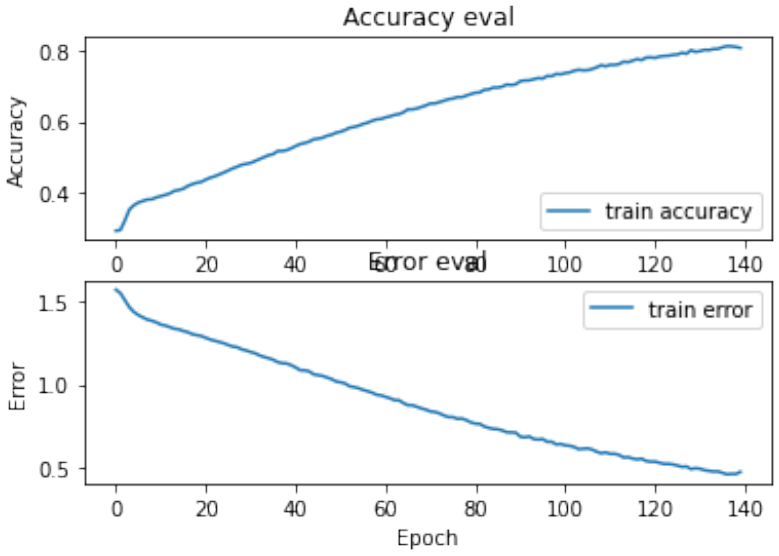
LSTM CONFUSION MATRIX

		ACTUAL CLASS				
		a	ai	bA	chA	tA
PREDICTED CLASS	a	55	13	2	2	5
	ai	3	46	5	4	1
	bA	7	4	32	27	6
	chA	3	6	26	34	2
	tA	2	3	8	3	55

RESULTS: DATASET II

ARCHITECTURE I

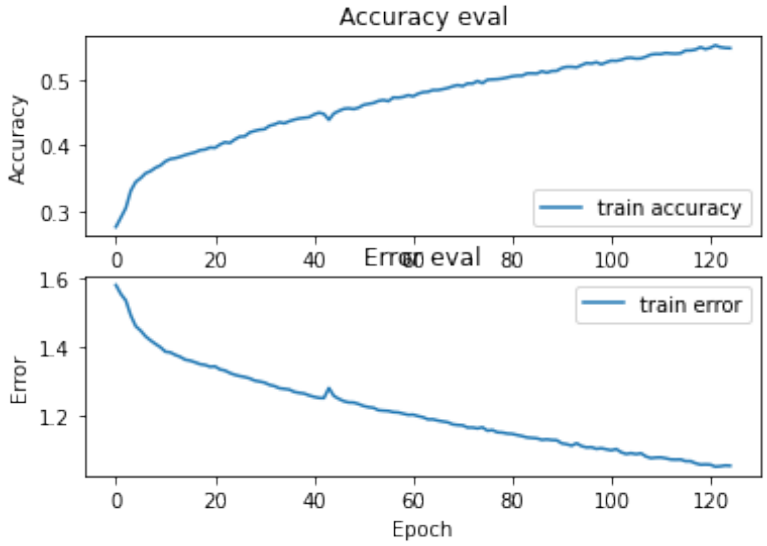
No of RNN Layers	1



Training Accuracy	0.3912
Test Accuracy	0.3901

ARCHITECTURE II

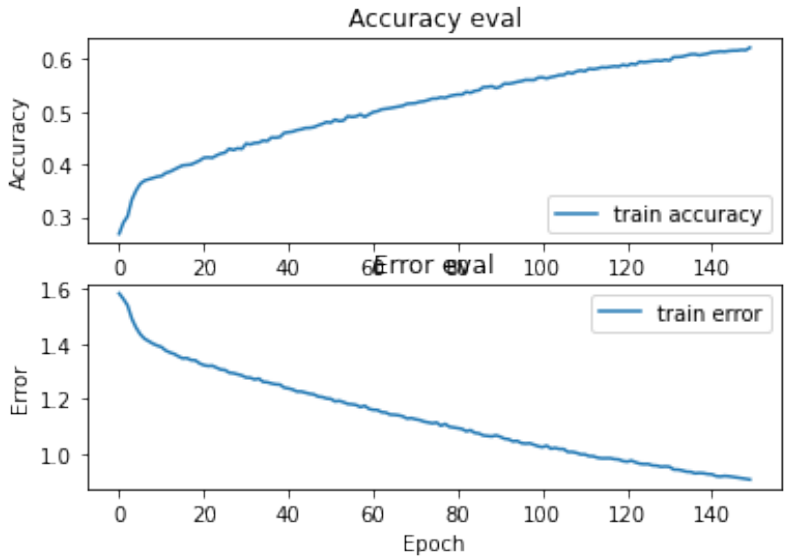
No of RNN Layers	2



Training Accuracy	0.2109
Test Accuracy	0.1972

ARCHITECTURE III

No of RNN Layers	3



Training Accuracy	0.2329
Test Accuracy	0.1872

BEST ARCHITECTURE

All the architecture performed very bad accuracy for this data set. Out of these the architecture I with number of RNN layer as 1
Its training accuracy is obtained as 39.12% and test accuracy as 39.01%

LSTM CONFUSION. MATRIX

		ACTUAL CLASS				
		sa	ka	kaa	tA	ba
PREDICTED CLASS	sa	711	164	262	114	311
	ka	167	522	865	181	382
	kaa	261	429	2029	249	950
	tA	165	166	427	1167	380
	ba	255	218	1037	242	1314

GITHUB LINK

https://github.com/its-rajesh/Deep-Learning

TEAM DETAILS
(Group 22)

Name	Rollno	Mailid
Rajesh R	S21005	s21005@students.iitmandi.ac.in
Naisarg Pandya	S21012	s21012@students.iitmandi.ac.in
Ashok Kumar	D19042	d19042@students.iitmandi.ac.in