

## Assignment 3

Student Details: Name: Puspak Chakraborty

Roll No: DA24S002

WANDB Link: [DA6401 Assignment 3 | DA6401 Assignment 3 – Weights & Biases](#)

Github Link: [https://github.com/da24s002/DA6401\\_Assignment\\_3\\_Submission](https://github.com/da24s002/DA6401_Assignment_3_Submission)

# DA6401 Assignment 3

Use recurrent neural networks to build a transliteration system.

Puspak Chakraborty da24s002

Created on May 14 | Last edited on May 20

## ▼ Instructions

- The goal of this assignment is fourfold: (i) learn how to model sequence to sequence learning problems using Recurrent Neural Networks (ii) compare different cells such as vanilla RNN, LSTM and GRU (iii) understand how attention networks overcome the limitations of vanilla seq2seq models (iv) visualise the interactions between different components in a RNN based model.
- Collaborations and discussions with other groups are strictly prohibited.
- You must use Python (numpy and pandas) for your implementation.
- You can use any and all packages from keras, pytorch, tensorflow
- You can run the code in a jupyter notebook on colab by enabling GPUs.
- You have to generate the report in the same format as shown below using wandb.ai. You can start by cloning this report using the clone option above. Most of the plots that we have asked for below can be (automatically) generated using the apis provided by wandb.ai. You will upload a link to this report on gradescope.

- You also need to provide a link to your github code as shown below. Follow good software engineering practices and set up a github repo for the project on Day 1. Please do not write all code on your local machine and push everything to github on the last day. The commits in github should reflect how the code has evolved during the course of the assignment.
- You have to check moodle regularly for updates regarding the assignment.

## Problem Statement

In this assignment you will experiment with the [Dakshina dataset](#) released by Google. This dataset contains pairs of the following form:

x. y

ajanabee अजनबी.

i.e., a word in the native script and its corresponding transliteration in the Latin script (the way we type while chatting with our friends on WhatsApp etc). Given many such  $(x_i, y_i)_{i=1}^n$  pairs your goal is to train a model  $y = \hat{f}(x)$  which takes as input a romanized string (ghar) and produces the corresponding word in Devanagari (घर).

As you would realise this is the problem of mapping a sequence of characters in one language to a sequence of characters in another language. Notice that this is a scaled down version of the problem of translation where the goal is to translate a sequence of **words** in one language to a sequence of words in another language (as opposed to sequence of **characters** here).

Read these blogs to understand how to build neural sequence to sequence models: [blog1](#), [blog2](#)

## Question 1 (15 Marks)

Build a RNN based seq2seq model which contains the following layers: (i) input layer for character embeddings (ii) one encoder RNN which sequentially encodes the input character sequence (Latin) (iii) one decoder RNN which takes the last state of the encoder as input and produces one output character at a time (Devanagari).

The code should be flexible such that the dimension of the input character embeddings, the hidden states of the encoders and decoders, the cell (RNN, LSTM, GRU) and the number of layers in the encoder and decoder can be changed.

(a) What is the total number of computations done by your network? (assume that the input embedding size is  $m$ , encoder and decoder have 1 layer each, the hidden cell state is  $k$  for both the encoder and decoder, the length of the input and output sequence is the same, i.e.,  $T$ , the size of the vocabulary is the same for the source and target language, i.e.,  $V$ )

(b) What is the total number of parameters in your network? (assume that the input embedding size is  $m$ , encoder and decoder have 1 layer each, the hidden cell state is  $k$  for both the encoder and decoder and the length of the input and output sequence is the same, i.e.,  $T$ , the size of the vocabulary is the same for the source and target language, i.e.,  $V$ )

## Answer:

Given:

$V$ : Vocabulary size

$m$ : Embedding size

k: Hidden state size

T: Sequence length

Consider a sequence-to-sequence model with:

Embedding layer (input)

1-layer encoder and decoder (RNN/LSTM/GRU cell)

Decoder output layer (linear layer to vocab size)

Part (a): Total Number of Computations

Total computations:

$T \cdot m$  (for both encoder and decoder, but counted once per sequence)

1. Encoder and Decoder Cell Computations (per timestep)

Let  $g$  be the number of gates:

RNN:  $g=1$

GRU:  $g=3$

LSTM:  $g=4$

Each gate requires:

Input-to-hidden:  $k \cdot m$

Hidden-to-hidden:  $k \times k$

Bias:  $k$  (negligible in computation count)

Per timestep, per cell :  $g \times (k \times m + k \times k)$

For all timesteps (encoder + decoder) :  $2 \times T \times g \times (k \times m + k \times k)$

## 2. Decoder Output Layer

At each timestep, the decoder maps hidden state to vocabulary:  $k \rightarrow V$

Computation per timestep:  $k \times V$

For all timesteps:  $T \times k \times V$

## 3. Total Computations (Approximate, ignoring bias and softmax):

Total Computations =  $2Tg(km + k^2) + T(kV)$

Total Computations =  $2Tg(km + k^2) + T(kV)$

Where:

For RNN:  $g=1$

For GRU:  $g=3$

For LSTM:  $g=4$

## Part (b): Total Number of Parameters

### 1. Embedding Layers

Encoder:  $V \times m$

Decoder:  $V \times m$

## 2. Encoder and Decoder Cell Parameters

For each cell (encoder or decoder):

Parameters per cell =  $g \times [k \times m + k \times k + k]$

( $k \times m$ : input-to-hidden weights,  $k \times k$ : hidden-to-hidden weights,  $k$ : biases)

Total for encoder + decoder:  $2g(km + k^2 + k)$

## 3. Decoder Output Layer

Weights:  $k \times V$

Biases:  $V$

## 4. Total Parameters

Total Parameters =  $2(Vm) + 2g(km + k^2 + k) + kV + V$

# ▼ Question 2 (10 Marks)

You will now train your model using any one language from the [Dakshina dataset](#) (I would suggest pick a language that you can read so that it is easy to analyse the errors). Use the standard train, dev, test set from the folder `dakshina_dataset_v1.0/hi/lexicons/` (replace `hi` by the language of your choice)

Using the sweep feature in wandb find the best hyperparameter configuration. Here are some suggestions but you are free to decide which hyperparameters you want to explore

- input embedding size: 16, 32, 64, 256, ...
- number of encoder layers: 1, 2, 3
- number of decoder layers: 1, 2, 3
- hidden layer size: 16, 32, 64, 256, ...
- cell type: RNN, GRU, LSTM
- dropout: 20%, 30% (btw, where will you add dropout? you should read up a bit on this)
- beam search in decoder with different beam sizes:

Based on your sweep please paste the following plots which are automatically generated by wandb:

- accuracy v/s created plot (I would like to see the number of experiments you ran to get the best configuration).
- parallel co-ordinates plot
- correlation summary table (to see the correlation of each hyperparameter with the loss/accuracy)

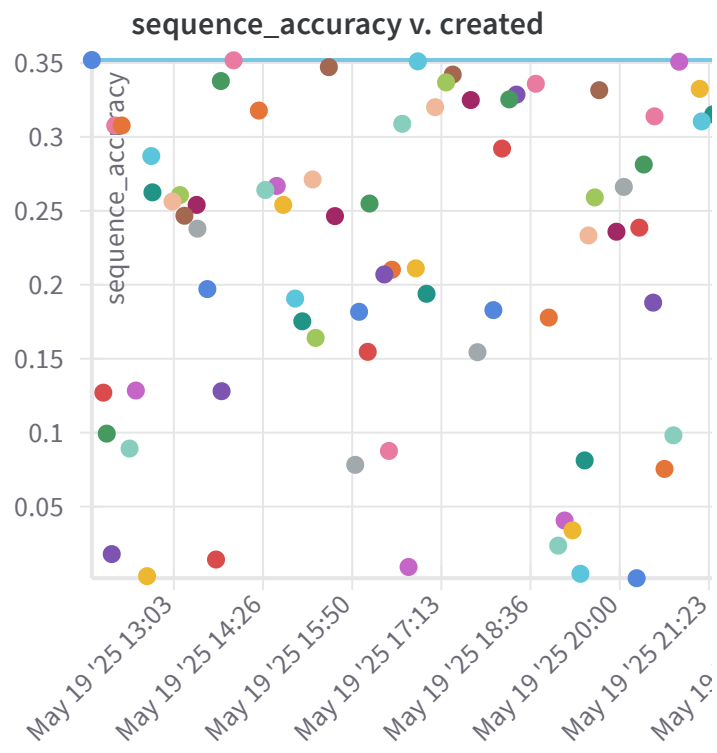
Also write down the hyperparameters and their values that you swept over. Smart strategies to reduce the number of runs while still achieving a high accuracy would be appreciated.

Write down any unique strategy that you tried for efficiently searching the hyperparameters.

## Answer:

--	--

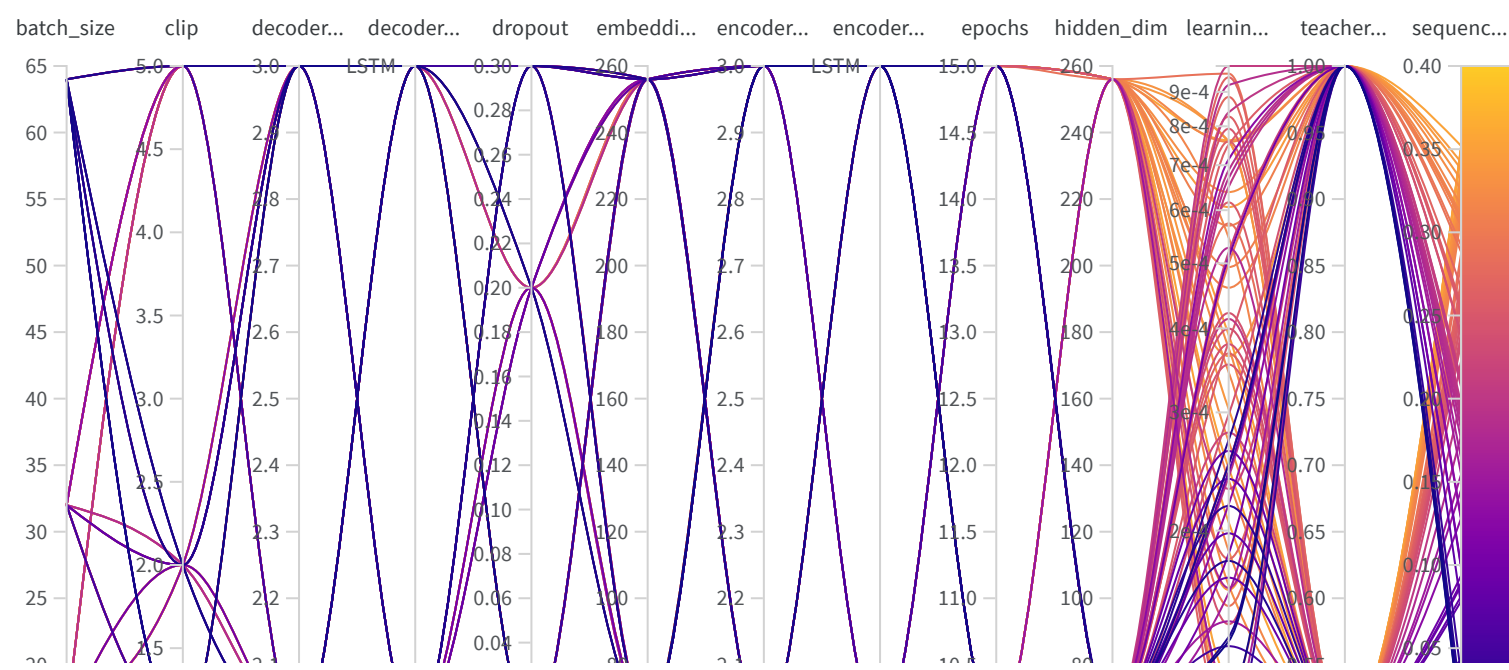


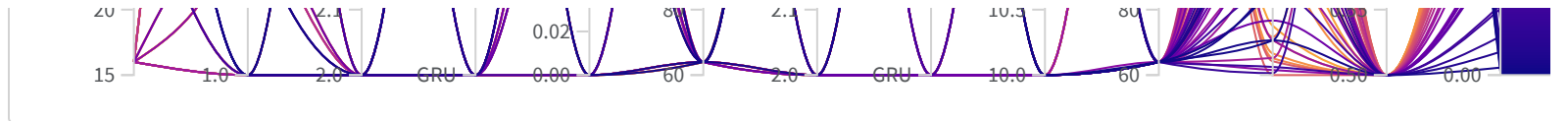


Parameter importance with respect to

sequence\_acc...

Search	Parameters	
Config parameter	Importance	Correlation
hidden_dim		
batch_size		
learning_rate		
Runtime		
clip		
encod..._GRU		





Best set of hyperparameters:

batch\_size:32

beam\_width:5

clip:5

decode\_method:"greedy"

decoder\_layers:2

decoder\_type:"LSTM"

dropout:0.2

embedding\_dim:64

encoder\_layers:3

encoder\_type:"LSTM"

epochs:15

hidden\_dim:256

language:"hi"

learning\_rate:0.0007625795353002294

length\_penalty\_alpha:1

model\_type:"basic"

teacher\_forcing\_ratio:1

Initial set of hyperparameter values swept over (in bn dataset, here I have done only for attention)

(I started working on the 'bn' dataset, but the dataset being too large I didn't think I'd be able to complete all experiments on it, so I used it to see which hyperparameters were working properly, and used a smaller set of hyperparameter for the later experiments on 'hi' dataset) (the sequence accuracy graphs for the bn dataset is attached below as well)

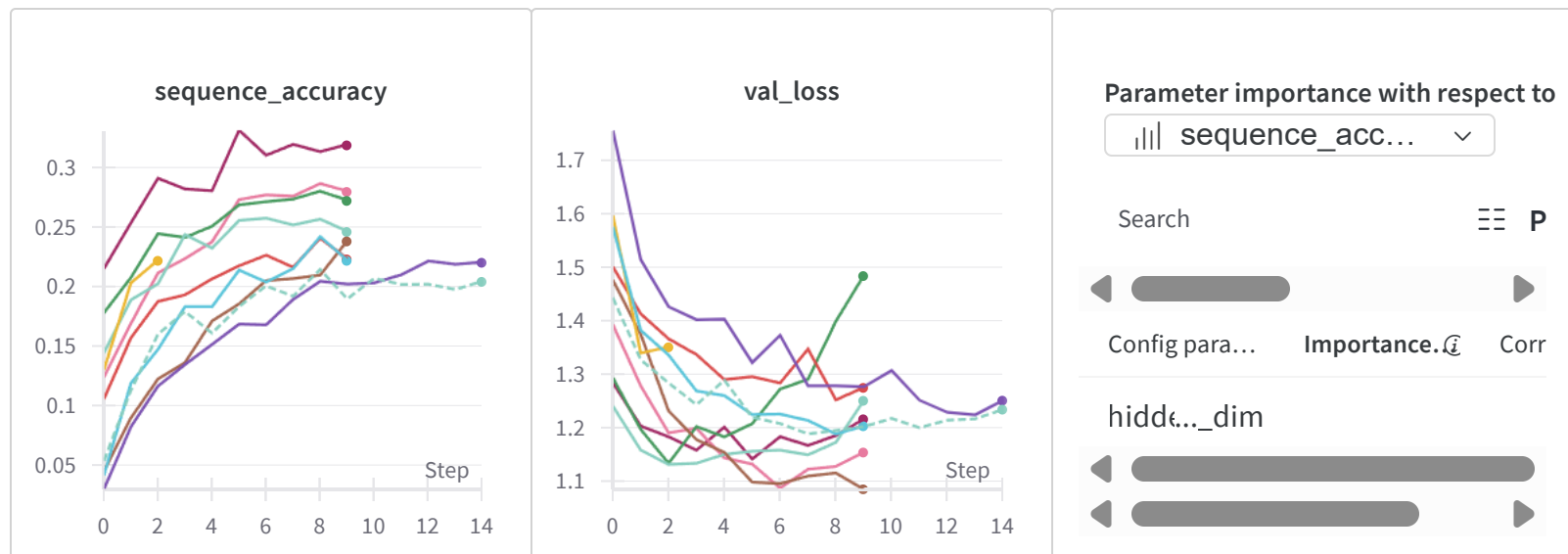
```
program: "train.py"
name: "DA6401_Assignment_3"
method: "bayes"
metric:
  goal: maximize
  name: validation_accuracy
parameters:
  epochs:
    values: [10,15]
  model_type:
    values: ["basic", "attention"]
  encoder_rnn_type:
    values: ["LSTM", "RNN", "GRU"]
  decoder_rnn_type:
    values: ["LSTM", "RNN", "GRU"]
  embedding_dim:
    values: [16,32,64,256]
  hidden_dim:
    values: [16,32,64,256]
```

```

encoder_layers:
  values: [1,2,3]
decoder_layers:
  values: [1,2,3]
batch_size:
  values: [16, 32, 64]
dropout:
  values: [0,0.2,0.3]
learning_rate:
  max: 0.001
  min: 0.0001
  distribution: log_uniform_values
clip:
  values: [1.0, 2.0, 5.0]
teacher_forcing_ratio:
  values: [0.5, 1.0]

```

This is the plot for the initial experiments on the 'bn' dataset, from the hyperparameters used here, I used a smaller set of below hyperparameters for the 'hi' dataset.



Hyperparameter values swept over for 'hi' dataset(this is a smaller set of values based on previous experiment done in 'bn' language)

```
program: "question2.py"
name: "DA6401_Assignment_3"
method: "bayes"
metric:
  goal: maximize
  name: validation_accuracy
parameters:
  epochs:
    values: [10, 15]
  encoder_rnn_type:
    values: ["LSTM", "GRU"]
  decoder_rnn_type:
    values: ["LSTM", "GRU"]
  embedding_dim:
    values: [64, 256]
  hidden_dim:
    values: [64, 256]
  encoder_layers:
    values: [2, 3]
  decoder_layers:
    values: [2, 3]
  batch_size:
    values: [16, 32, 64]
  dropout:
    values: [0, 0.2, 0.3]
  learning_rate:
```

```
max: 0.001
min: 0.0001
distribution: log_uniform_values
clip:
  values: [1.0, 2.0, 5.0]
teacher_forcing_ratio:
  values: [0.5, 1.0]
```

Unique strategies for efficiently searching hyperparameters:

1. The beam search is prohibitively expensive in terms of time and compute requirement, so I haven't included that in the hyperparameter search, but used beam search at the end with the best hyperparameter values.
2. Used bayes search for hyperparameter search instead of random, so that it finds good configurations in lesser number of runs. (Bayesian search kind of works like binary search in higher dimensional space)
3. Initially I have used a broader set of hyperparameters, then based on the sequence accuracy on the 'bn' dataset, I have ran the sweep on a smaller set of hyperparameters to get best possible sequence accuracy in the 'hi' dataset.

Best validation accuracy on sequence with non attention model: ~35%

## ▾ Question 3 (15 Marks)

Based on the above plots write down some insightful observations. For example,

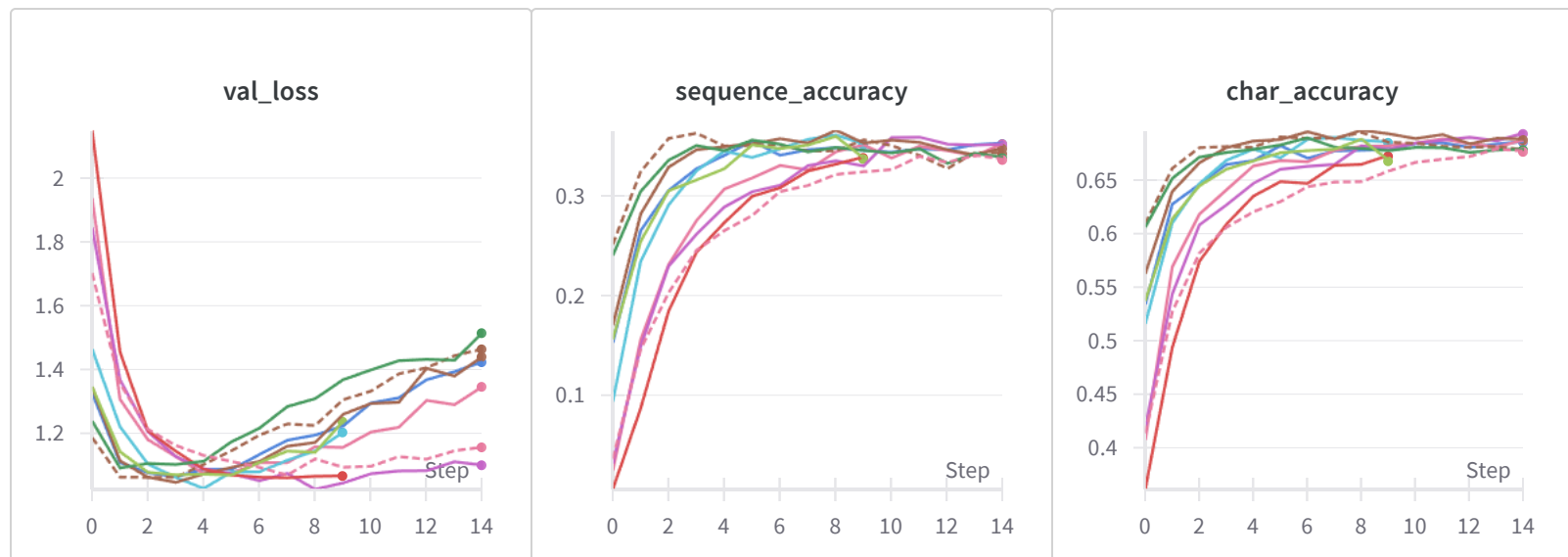
- RNN based model takes longer time to converge than GRU or LSTM
- using smaller sizes for the hidden layer does not give good results

- dropout leads to better performance

(Note: I don't know if any of the above statements is true. I just wrote some random comments that came to my mind)

Of course, each inference should be backed by appropriate evidence.

## Answer:



Some insightful observations based on the above plots (and the plots in question 2):

- 1) From the parameter correlation chart we can clearly see that hidden dimension has the highest positive impact on the sequence validation accuracy.
- 2) LSTM is the best performing architecture in decoder and GRU in encoder, vanilla RNN seems to fall behind.

- 3) Sequence accuracy is very low compared to character accuracy, giving a sense that this approach is not the best approach for transliteration (char to char translation among languages).
- 4) Taking a look at the validation loss graph, it seems like the model tends to overfit after 5-6 epochs, so early stopping would've given even better results.
- 5) Also since the model is overfitting, so we could use a higher hidden dimension to increase the number of parameters for a better result.
- 6) Gradient clipping has a positive correlation, giving the idea that gradient exploding is happening, may be some kind of input normalization could help it a little bit.
- 7) Dropout is negatively correlated to the sequence accuracy.
- 8) Embedding size of 64 gives the best sequence accuracy.
- 9) Higher learning rate has a high correlation with the sequence accuracy, this is a little counter intuitive to me, because gradient clipping is also giving good results, so why did the model not work good with low learning rate and no gradient clipping in the first place.

## ▼ Question 4 (10 Marks)

You will now apply your best model on the test data (You shouldn't have used test data so far. All the above experiments should have been done using train and val data only).

- (a) Use the best model from your sweep and report the accuracy on the test set (the output is correct only if it exactly matches the reference output).



(b) Provide sample inputs from the test data and predictions made by your best model (more marks for presenting this grid creatively). Also upload all the predictions on the test set in a folder **predictions\_vanilla** on your github project.

(c) Comment on the errors made by your model (simple insightful bullet points)

- The model makes more errors on consonants than vowels
- The model makes more errors on longer sequences
- I am thinking confusion matrix but may be it's just me!
- ...

## Answer:

Using the best model for the test dataset along with beam search for the test data:

a) Sequence accuracy : 37.38%

hyperparameters used :

```
batch_size:32
beam_width:5
clip:5
decode_method:"greedy"
decoder_layers:2
decoder_type:"LSTM"
dropout:0.2
embedding_dim:64
encoder_layers:3
encoder_type:"LSTM"
```

```
epochs:15
hidden_dim:256
language:"hi"
learning_rate:0.0007625795353002294
length_penalty_alpha:1
model_type:"basic"
teacher_forcing_ratio:1
```

b) Sample input and prediction values with ground truth from test data:

Hindi Transliteration Results (LSTM Seq2Seq Model)

Latin Input	Model Output	Ground Truth
frost	फ्रोस्ट	फ्रॉस्ट
utaney	उताने	उतने
bakhshne	बक्षने	बख्शने
aajmana	आजमना	आजमाना
shmil	शमली	शमलि
jaatakon	जातकों	जातकों
khajraana	खजराना	खजराना
shbana	शबना	शबाना
nikaalni	नकिानी	नकिलानी
charon	चररों	चारों
lindon	लडिों	लडिन
piyegaa	पयिगा	पयिगा
shud	शुड़	शुड
chhutane	छुताने	छूटने
masur	मसूर	मसूर

c) Insightful comments for errors made by the model:

Confusion matrix for test data

[illegible]

Ground Truth



## 1. Issues with Vowels

1. "anka" → "अंका" instead of "अंक"
2. "antaha" → "अंताहा" instead of "अंतः" (extra vowel)
3. "akapulko" → "अकापुलक" instead of "अकापुल्को" (missing vowel)

## 2. Consonant Substitutions

1. "ankhon" → "अंखों" instead of "अंकों" (ख/क confusion)
2. "angkor" → "अंगकर" instead of "अंकोर" (ग/क confusion)

### 3. Common English-Hindi Transliteration Challenges

1. "a" at the end is often transliterated as "आ" when it should be silent
2. "w" is sometimes confused with "व" or "उ"

#### 4. Missing Half-Letters

1. "antarmukh" → "अंतरमुख" instead of "अंतर्मुख" (missing र्)

## 5. Short Words vs. Long Words

1. The model generally performs better on shorter words (2-4 characters) than longer ones, where errors tend to compound.

▼ Question 5 (20 Marks)

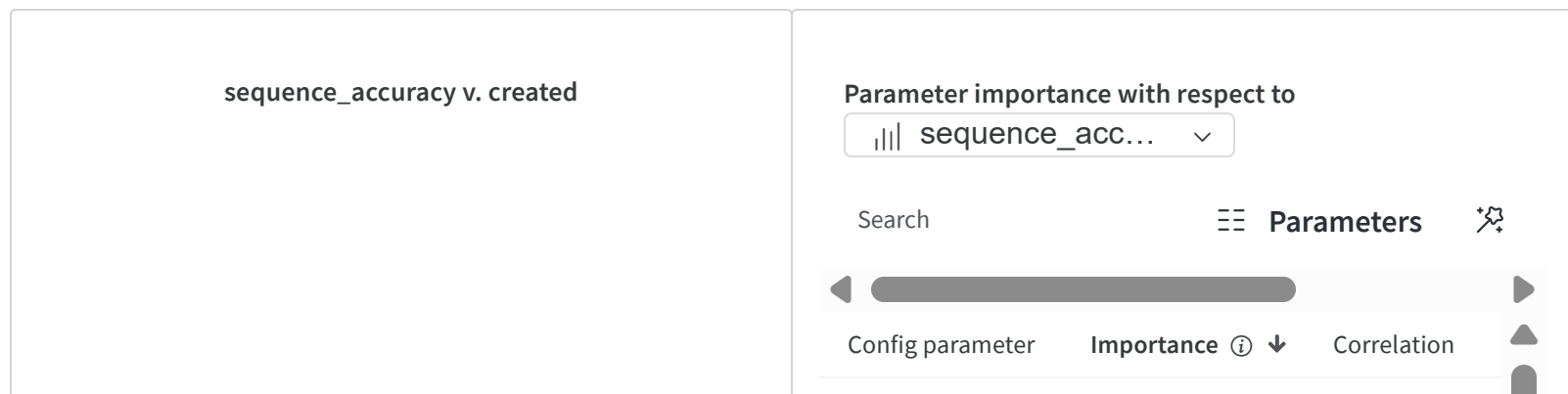
Now add an attention network to your basis sequence to sequence model and train the model again. For the sake of simplicity you can use a single layered encoder and a single layered decoder (if you want you can use multiple layers also). Please answer the following questions:

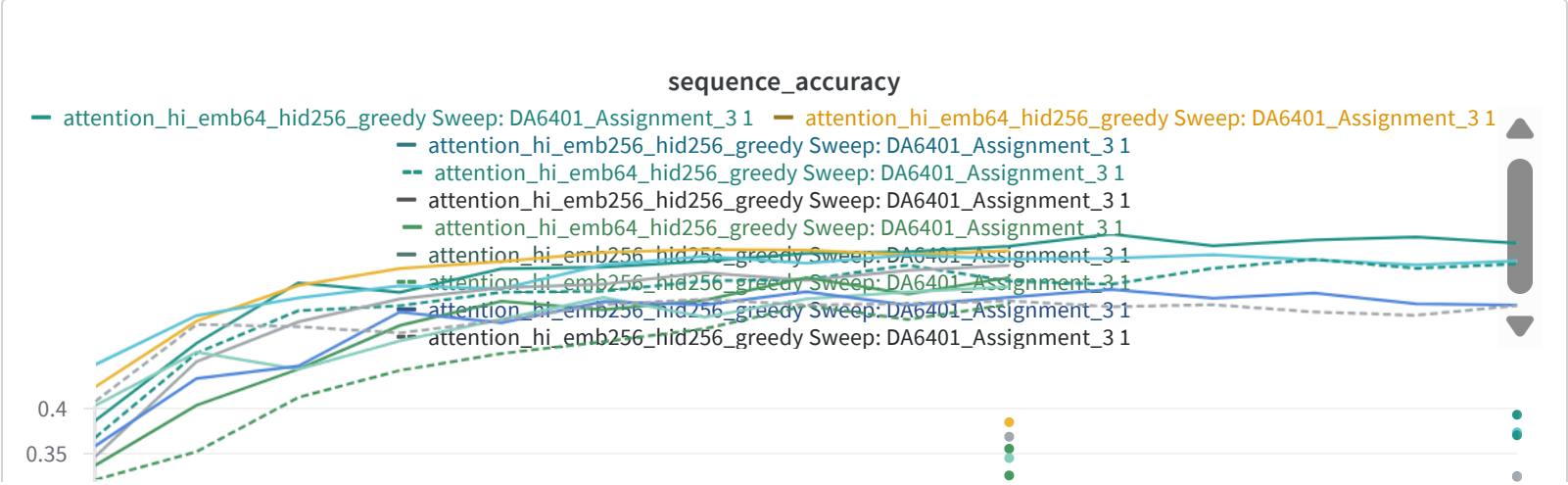
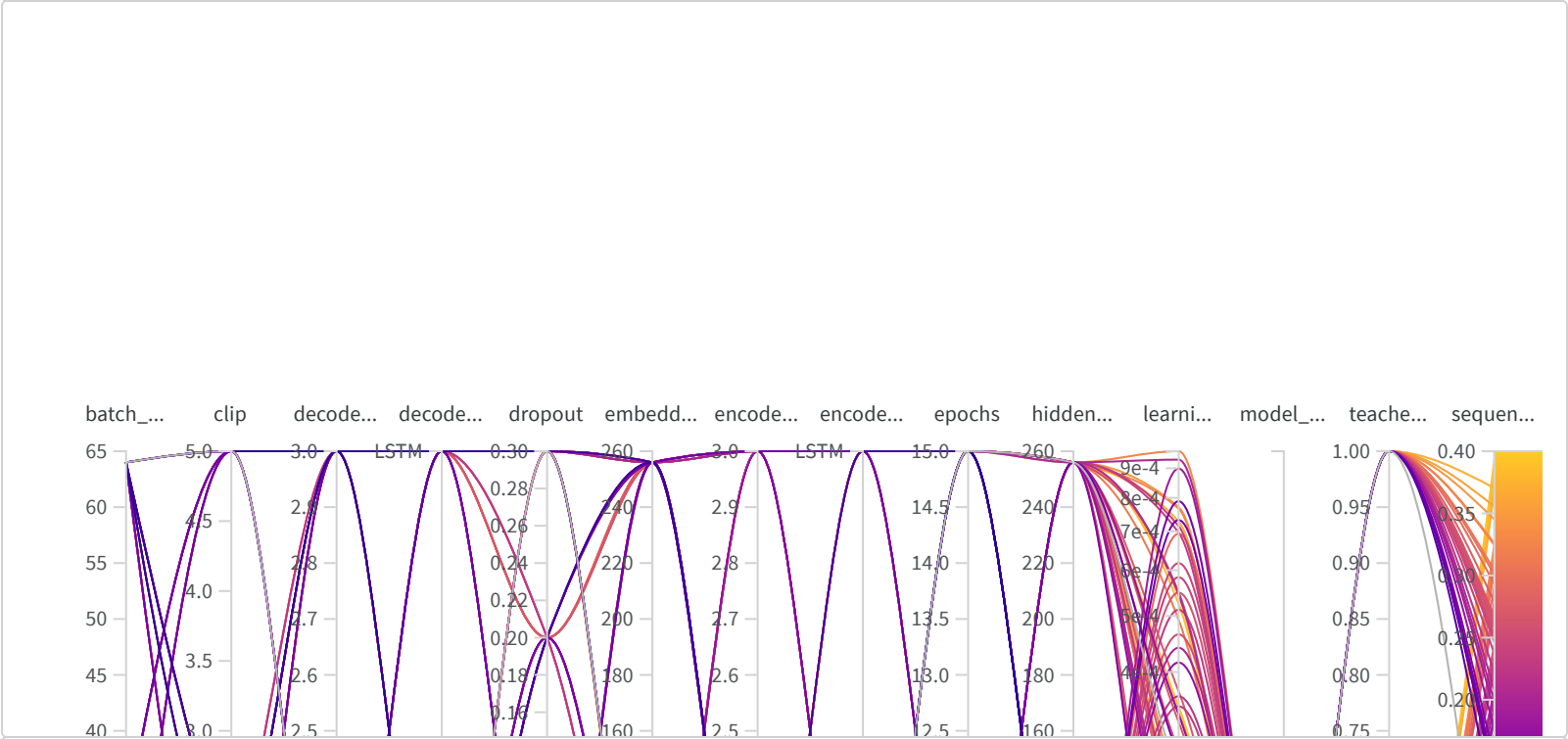
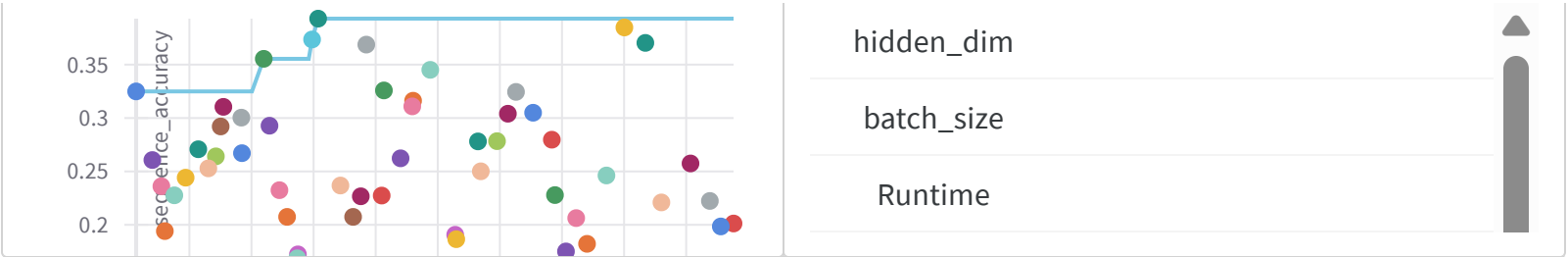
- (a) Did you tune the hyperparameters again? If yes please paste appropriate plots below.
- (b) Evaluate your best model on the test set and report the accuracy. Also upload all the predictions on the test set in a folder **predictions\_attention** on your github project.
- (c) Does the attention based model perform better than the vanilla model? If so, can you check some of the errors that this model corrected and note down your inferences (i.e., outputs which were predicted incorrectly by your best seq2seq model are predicted correctly by this model)
- (d) In a 3 x 3 grid paste the attention heatmaps for 10 inputs from your test data (read up on what are attention heatmaps).

## Answer:

a) Yes I did hyperparameter tuning on a smaller set of parameters for attention.

Best Validation Sequence accuracy obtained: ~40% (since I am saving the model with highest validation score after each epoch, so the model with validation score around 40.1% will get saved.)





Best set of hyperparameters for attention:

```
batch_size: 16
clip: 2
decoder_layers: 2
decoder_type: "LSTM"
dropout: 0.3
embedding_dim: 64
encoder_layers: 3
encoder_type: "GRU"
epochs: 15
hidden_dim: 256
language: "hi"
learning_rate: 0.000348286197910419
length_penalty_alpha: 1
model_type: "attention"
teacher_forcing_ratio: 0.5
```

b) Test accuracy obtained by model with attention: 38.69% (without beam search but higher number of parameters)

c) The attention based model performed a bit better than the vanilla model

1) Better at maintaining character consistency

Vanilla: "asurkshit" -> "असर्क्षित" (jumbled characters)

Attention: "asurkshit" -> "असुरक्षित" (correct)



## 2) Improved handling of compound words

Vanilla: "aparivartanshali" -> "अपरिवर्णशाली" (incorrect)

Attention: "aparivartanshali" -> "अपरिवर्तनशील" (correct)

## 3) Better with English loanwords

Vanilla: "email" -> "एमाइल" (incorrect)

Attention: "email" -> "ईमेल" (correct)

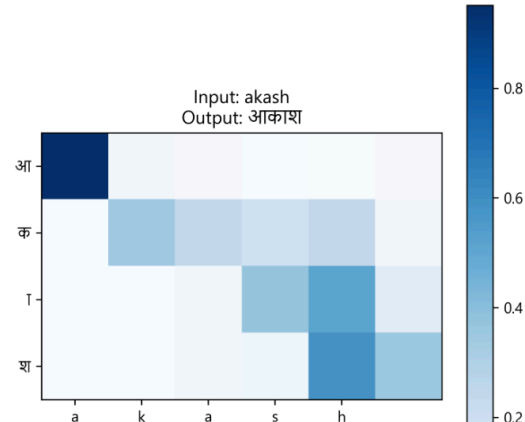
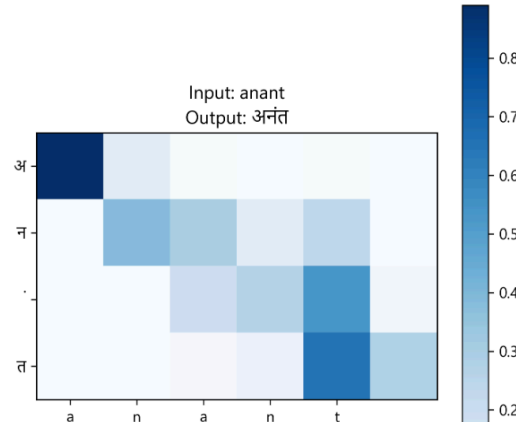
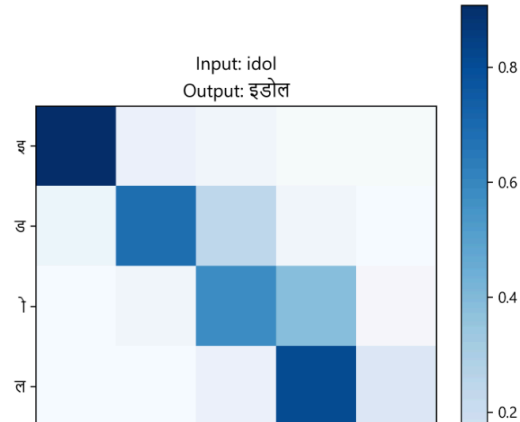
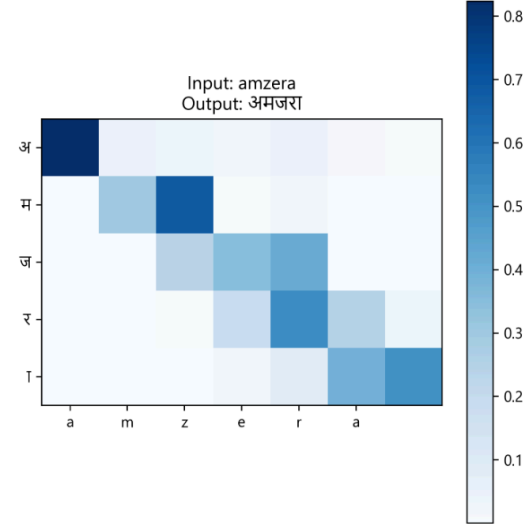
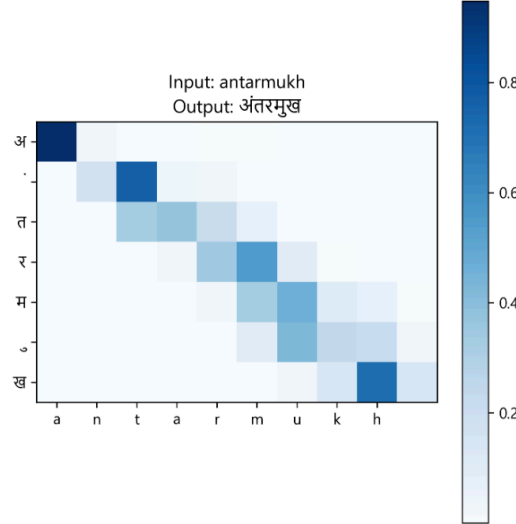
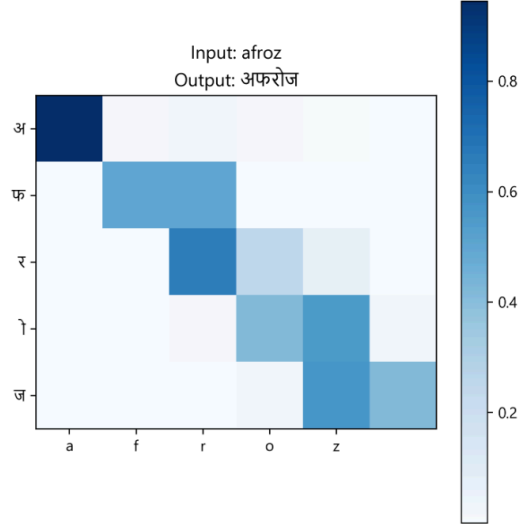
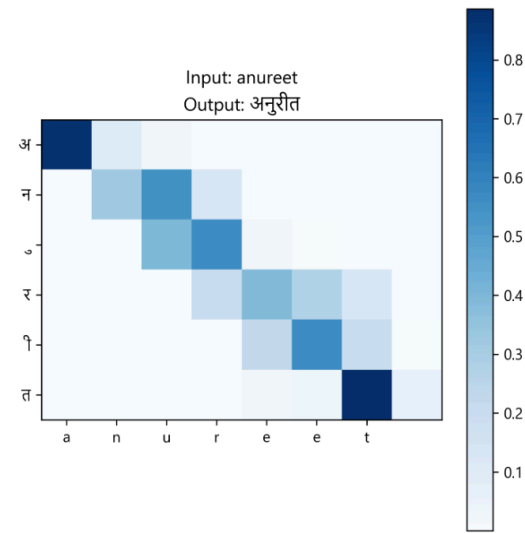
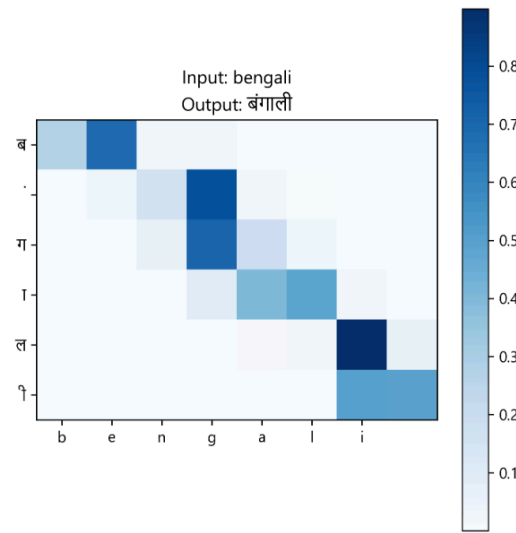
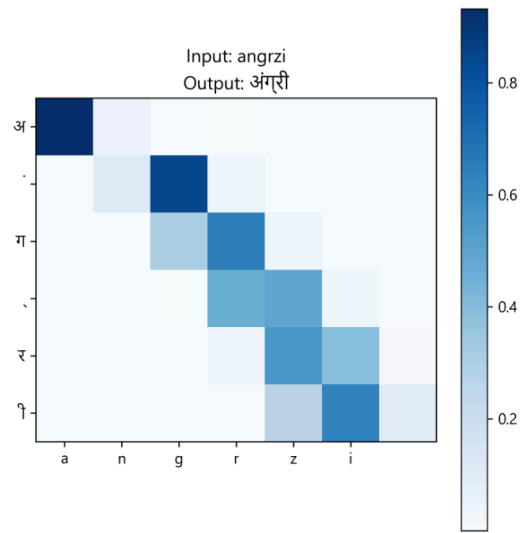
## 4) More accurate with doubled consonants

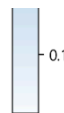
Vanilla: "addon" -> "आड़ों" (incorrect)

Attention: "addon" -> "अड्डों" (correct)

These improvements demonstrate how attention helps the model focus on relevant parts of the input when generating each output character, leading to more accurate transliterations, especially for complex or uncommon words.

## d) 3x3 attention heatmap





## Question 6 (20 Marks)

This is a challenge question and most of you will find it hard.

I like the visualisation in the figure captioned "Connectivity" in this [article](#). Make a similar visualisation for your model. Please look at this [blog](#) for some starter code. The goal is to figure out the following: When the model is decoding the  $i$ -th character in the output which is the input character that it is looking at?

Have fun!

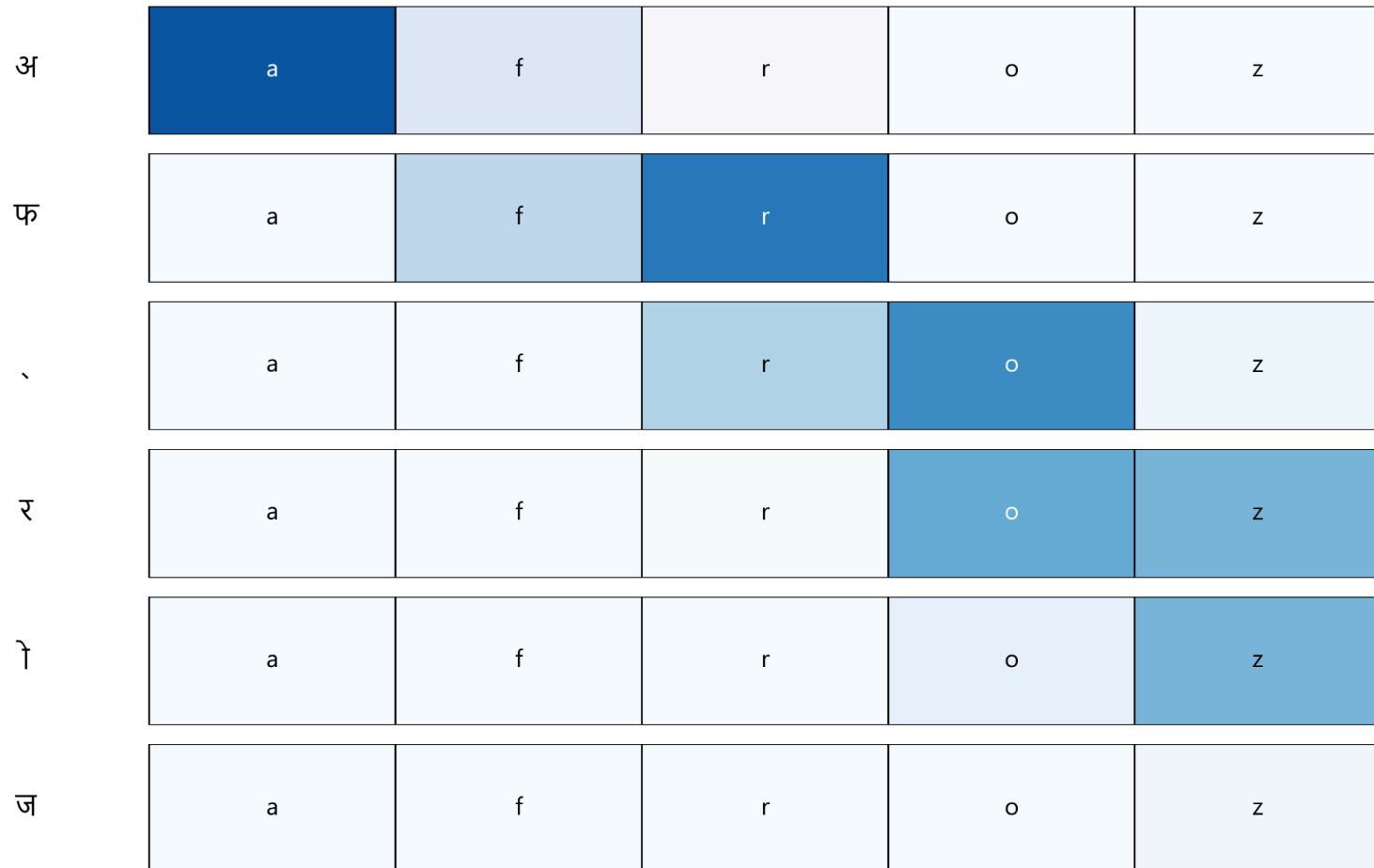
## Answer:

From the question provided above, we have to find the character/characters being focussed upon when the  $i$ -th character in the output is being generated. This is essentially similar to the attention heatmap (Note: the examples in the blogs are different to this because, 1) in the first example, they are doing it for an auto-completion task, for which the previous output is the current input, so they are focussing on the previous part of the sentence. 2) In the second blog they are discussing about which characters/patterns the cells learn/are attentive to, which is different from the task given to us).

In our task, we have to find the English character/characters being mostly focussed upon when outputting a Hindi character. I have represented it similarly to the heatmap.


In the below plot, for each output character in Hindi, we can see the important input characters being focussed on.

Input: afroz □ Output: अफ़रोज़



Input: anureet □ Output: अनुरीत

अ	a	n	u	r	e	e	t
न	a	n	u	r	e	e	t
उ	a	n	u	r	e	e	t
र	a	n	u	r	e	e	t
ी	a	n	u	r	e	e	t
त	a	n	u	r	e	e	t

 Drag an image here

Input: antarmukh □ Output: अंतरमुख

अ

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

.

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

त

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

र

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

म

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

ु

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

ख

a	n	t	a	r	m	u	k	h
---	---	---	---	---	---	---	---	---

Input: anant □ Output: अनंत

अ	a	n	a	n	t
न	a	n	a	n	t
.	a	n	a	n	t
त	a	n	a	n	t

Input: idol □ Output: इडोल

इ

i	d	o	l
---	---	---	---

ड

i	d	o	l
---	---	---	---

ो

i	d	o	l
---	---	---	---

ल

i	d	o	l
---	---	---	---



Input: akash □ Output: अकाश

अ	a	k	a	s	h
क	a	k	a	s	h
।	a	k	a	s	h
श	a	k	a	s	h

Input: bengali □ Output: বাংলা

ব	b	e	n	g	a	l	i
.	b	e	n	g	a	l	i
গ	b	e	n	g	a	l	i
ল	b	e	n	g	a	l	i
ী	b	e	n	g	a	l	i

## ▼ Question 7 (10 Marks)

Paste a link to your github code for Part A

Example: [https://github.com/da24s002/DA6401\\_Assignment\\_3\\_Submission](https://github.com/da24s002/DA6401_Assignment_3_Submission);

- We will check for coding style, clarity in using functions and a README file with clear instructions on training and evaluating the model (the 10 marks will be based on this).
- We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarised).
- We will check the number of commits made by the two team members and then give marks accordingly. For example, if we see 70% of the commits were made by one team member then that member will get more marks in the assignment (**note that this contribution will decide the marks split for the entire assignment and not just this question**).
- We will also check if the training and test splits have been used properly. You will get 0 marks on the assignment if we find any cheating (e.g., adding test data to training data) to get higher accuracy.

## Answer:

github link : [https://github.com/da24s002/DA6401\\_Assignment\\_3\\_Submission](https://github.com/da24s002/DA6401_Assignment_3_Submission)

### ▼ Question 8 (0 Marks)

Note that this question does not carry any marks and will not be graded. This is only for students who are looking for a challenge and want to get something more out of the course.


Your task is to finetune the GPT2 model to generate lyrics for English songs. You can refer to [this blog](#) and follow the steps there. This blog shows how to finetune the GPT2 model to

generate headlines for financial articles. Instead of headlines you will use lyrics so you may find the following datasets useful for training: [dataset1](#), [dataset2](#)

At test time you will give it a prompt: "I love Deep Learning" and it should complete the song based on this prompt :-) Paste the generated song in a block below!

## Self Declaration

I, Puspak Chakraborty (Roll no: DA24S002), swear on my honour that I have written the code and the report by myself and have not copied it from the internet or other students.

Created with  on Weights & Biases.

[https://wandb.ai/da24s002-indian-institute-of-technology-madras/DA6401\\_Assignment\\_3/reports/DA6401-Assignment-3--VmIldzoxMjc3MTUwNQ](https://wandb.ai/da24s002-indian-institute-of-technology-madras/DA6401_Assignment_3/reports/DA6401-Assignment-3--VmIldzoxMjc3MTUwNQ)