Assignment 3

CS20M059 Shibobrota Das | CS20M007 Abhishek Kumar

Mitesh Khapra, Shibobrota Das

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)

Solution

(a) The total number of computations done by our network:

1. Number of computations done by Encoder in each step is $\mathrm{O}(\mathrm{k}^2 + \mathrm{mk} + \mathrm{mV})$
2. Number of computations done by Decoder in each step is $\mathrm{O}(\mathrm{k}^2 + \mathrm{mk} + \mathrm{mV} + \mathrm{V}\,\mathrm{k})$

Therefore, total number of computations in one step = $\mathrm{O}(2\mathrm{k}^2 + 2\mathrm{mk} + 2\mathrm{mV} + \mathrm{V}\,\mathrm{k})$

For T steps, $\mathrm{O}(\mathrm{T}\,(2\mathrm{k}^2 + 2\mathrm{mk} + 2\mathrm{mV} \ + \mathrm{V}\,\mathrm{k}))$

Note: Smaller terms have been ignored.

(b) The total number of parameters in our network: $\mathrm{O}(2\mathrm{k}^2 + 2\mathrm{mk} + 2\mathrm{mV} \ + \mathrm{V}\,\mathrm{k})$

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 sweeped 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.

Solution:

Hyperparameters:

1. layer cell Type - ['LSTM', 'GRU', 'SimpleRNN']
2. Number of encoder/decoder layers = [1, 2]
3. Dropout = [0.002,0.2]
4. Units = [64,256]
5. Batch size = [128]
6. Embedded dimension = [16, 64]

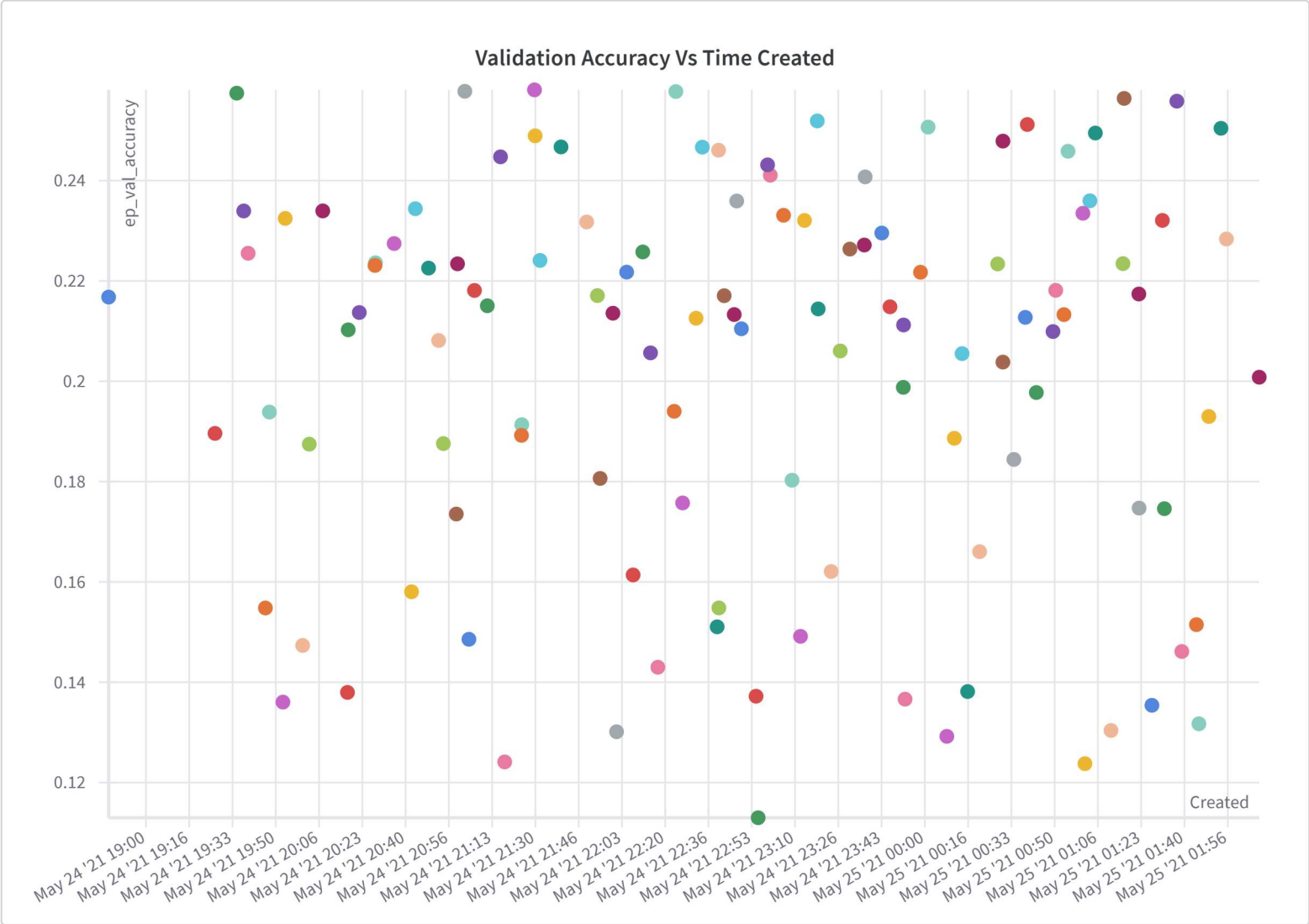
```
"parameters": {
        "layer_type": {
            "values": ["GRU", "LSTM", "SimpleRNN"]
        },
        "dropout": {
            "values": [0.0, 0.2]
```

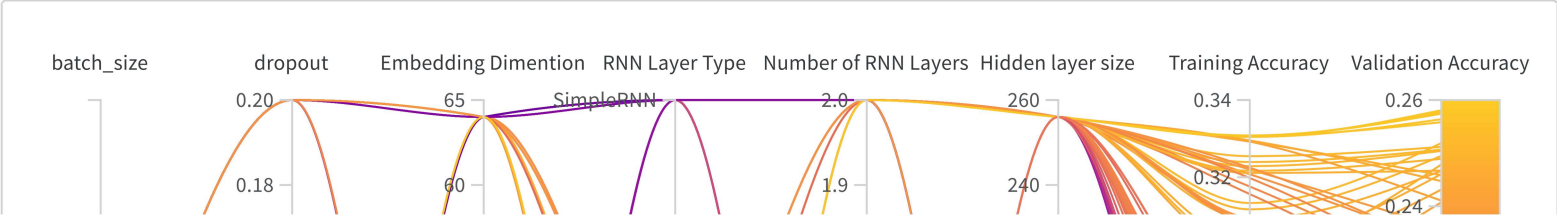
```
        },
        "units": {
            "values": [64, 256]
        },
        "embedding_dim": {
            "values": [16, 64]
        },
        "optimiser": {
            "values": ["nadam"]
        },
        "epochs": {
            "values": [20]
        },
        "batch_size": {
            "values": [128]
        },
        "num_layers": {
            "values": [1, 2]
        }
}
```

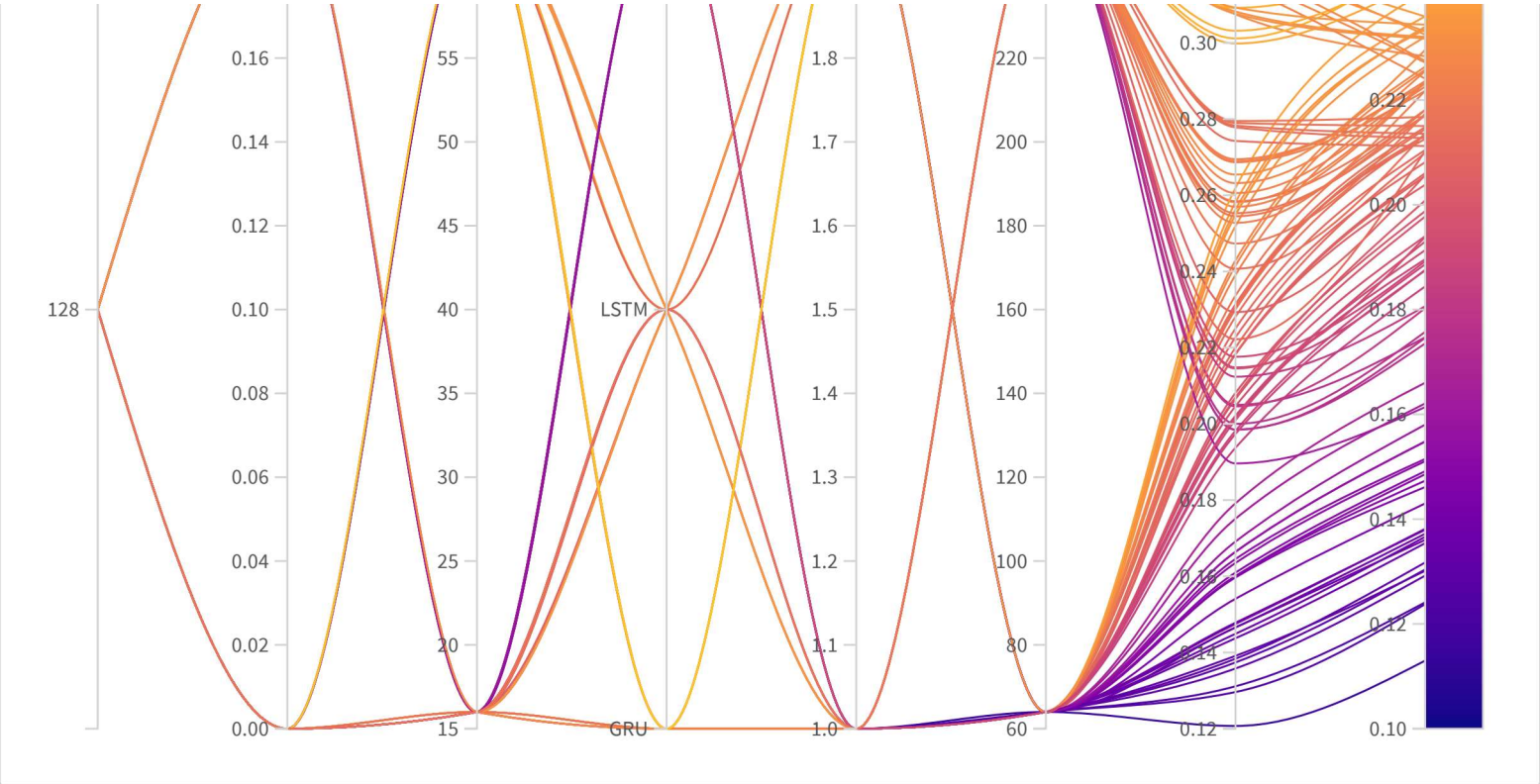
Hyperparameter search:

- Initially based on intuition, we tried some configurations offline and then decided upon the above mentioned hyperparameters. Since, training the models takes considerable amount of time, we kept the number of parameters as less as possible.
- We used the bayesian search provided by wandb to sweep over the hyperparameters, it helped in picking up the parameters which were leading to better accuracy.
- Then we observe the hyperparameters which are providing better results.

Following are the required plots:







Parameter importance with respect to ep_val_accuracy

Search

Parameters

1-10 of 11

Config parameter	Importance	Correlation
layer_type.value_SimpleRNN		
units		
num_layers		
embedding_dim		
Runtime		

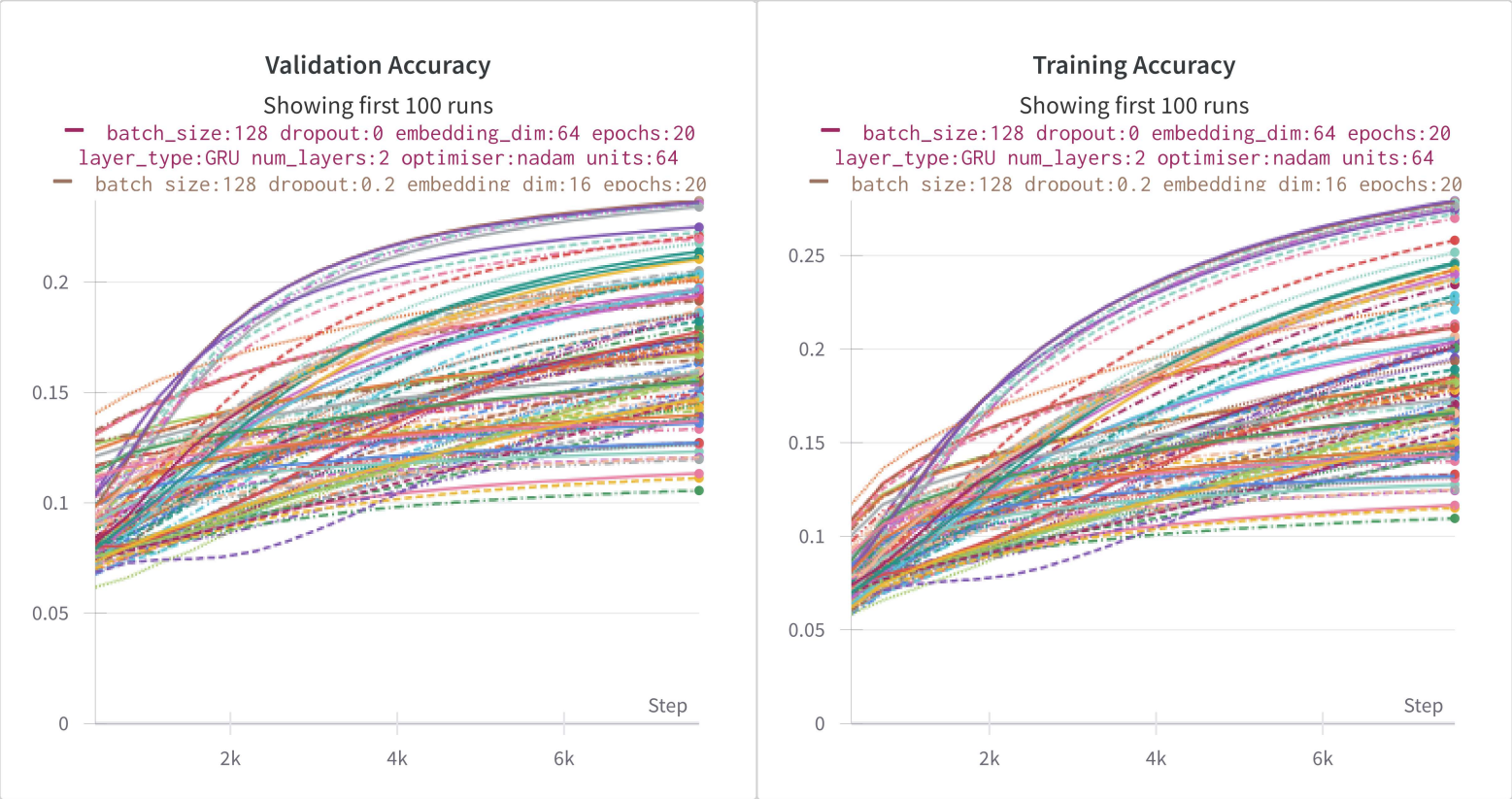
layer_type.value_GRU

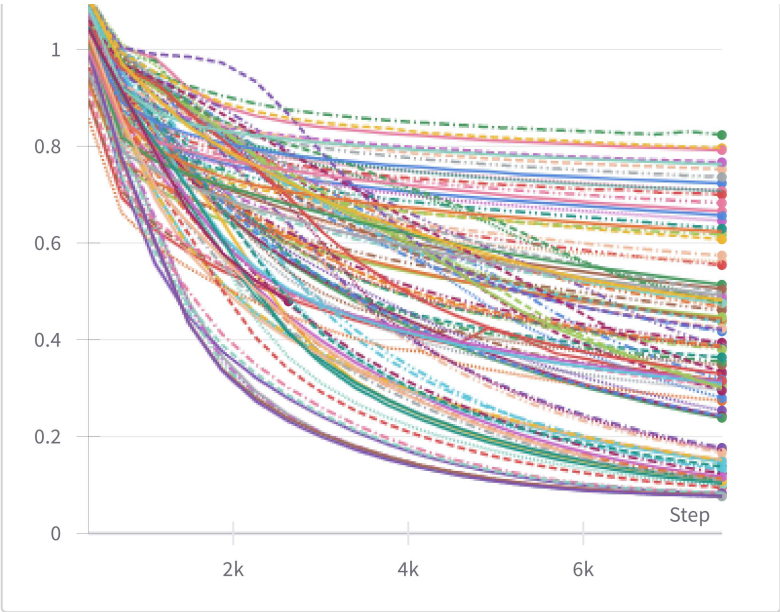
layer_type.value_LSTM

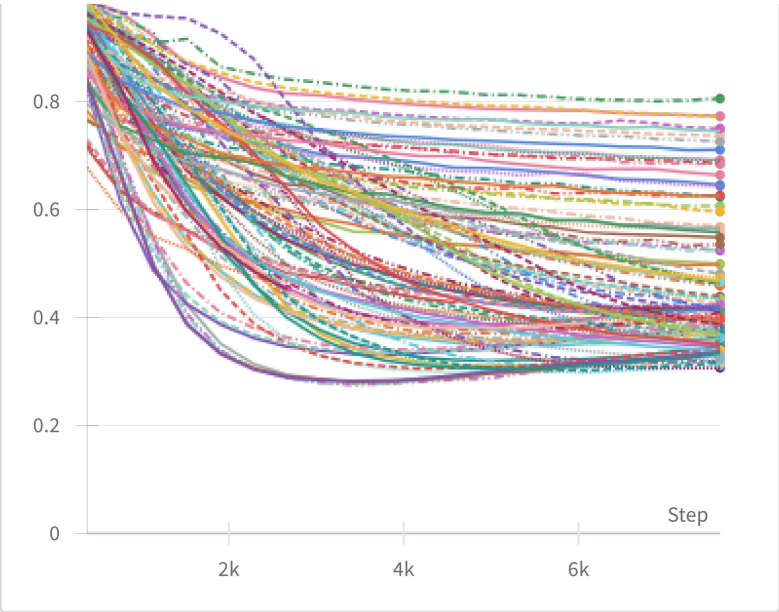
dropout

batch_size

optimiser.value_nadam







Question 3 (15 Marks)

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

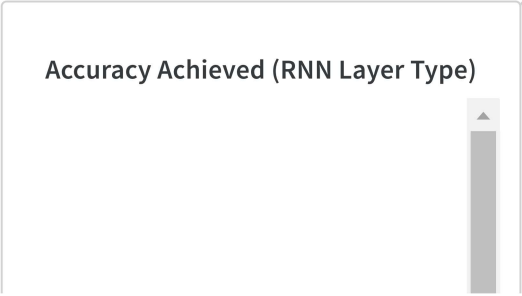
- RNN based model takes longer time to converge than GRU or LSTM
- using smaller sizes for the hidden layer does not give good results
- dropout leads to better performance

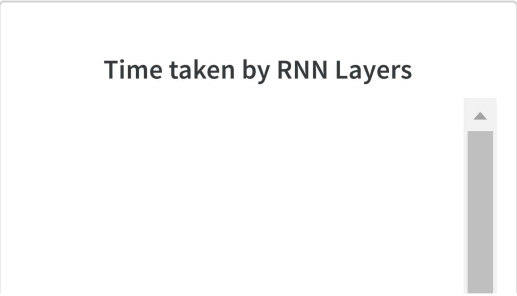
(Note: I don't know if any of the above statements is true. I just wrote some random comments that came to my mind)

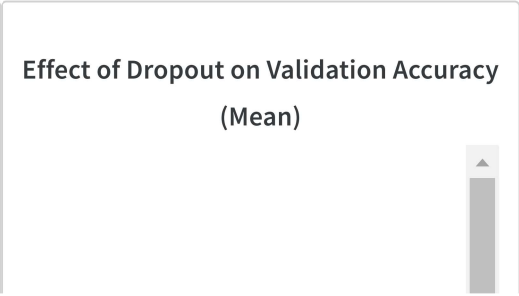
Of course, each inference should be backed by appropriate evidence.

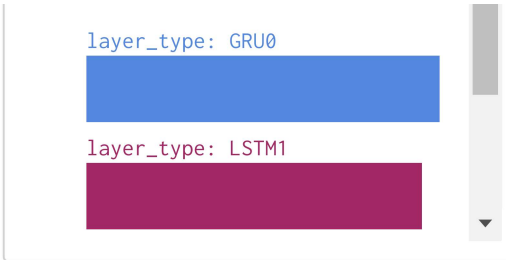
Solution:

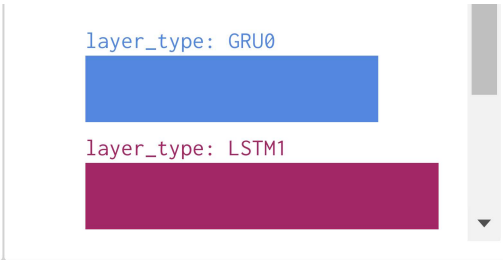
- The cell type "LSTM" and "GRU" performed better as compared to SimpleRNN for this task.
- Larger embedding size produced better results.
- Large number of hidden layer units performed better than the models having lesser number of hidden layer units. (Larger unit size performs better.)
- Batch size is an important parameter. From offline experiments, we observed that batch size of 128 performs better and takes less time. It is found to be well balanced.
- Dropout of 0.2 was providing slightly better results. We observed that higher dropout rates prevented the network's long-term memory to retain important features.
- We found that number of epochs was positively related to the performance of the model. And observed that #20 epochs provides good balance between time taken for training and accuracy.
- Time taken to train the model with LSTM layer is more as compared to GRU and SimpleRNN. Training time of model with SimpleRNN is found to be the least.
- We can observe from the bargraph, that the number of RNN layes influences the accuracy. Got better accuracy with more number of layers stacked one upon another.
- It took more time to train a model with multiple RNN layers as compared to a model with single RNN layer.

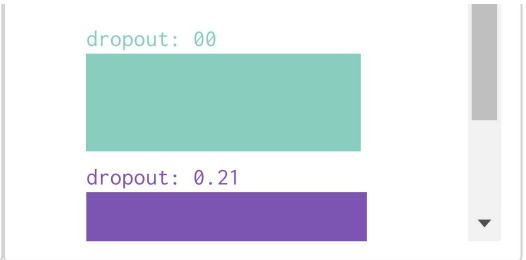




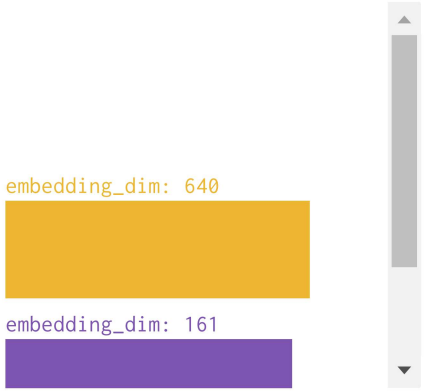




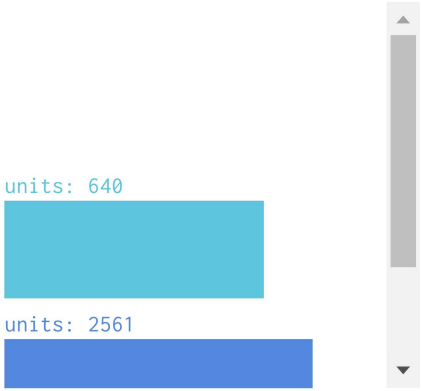




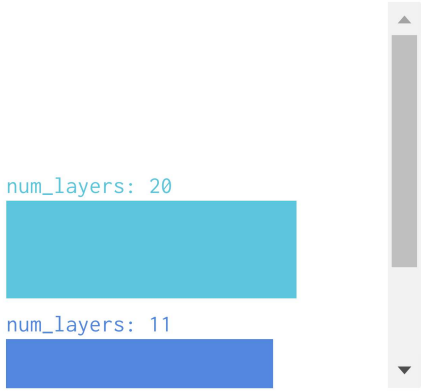
Effect of Embedding Dimension on Accuracy



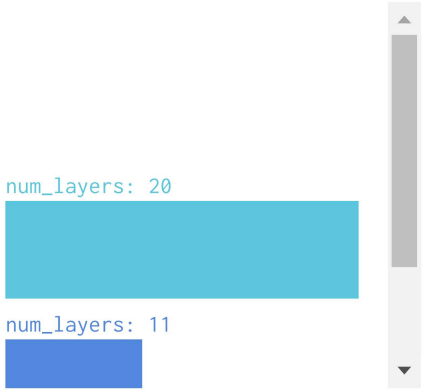
Influence of Number of Hidden layer units on Accuracy



Influence of number of layers on performance of model



Training time of model with different number of layers



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!
- ...

Solution

(a) The Accuracy is ~10% on the test set. (the output is correct only if it exactly matches the reference output)

(b) Sample inputs from the test data and predictions made by our best model

```
INPUT   PREDICTION  TRUE
ank  आंक     अंक
anka     आंका     अंक
ankit    आंकित    अंकित
anakon   अनकों    अंकों
ankhon   आं   अंकों
ankon    आं   अंकों
angkor   अंगकोंगकोंगकोंगकोंगक      अंकोर
ankor    एंकोर    अंकोर
angaarak अंगारक  अंगारक
angarak  अंगारक   अंगारक
angraji  अंग्रजी  अंग्रज़ी
angreji  अंग्रजी  अंग्रज़ी
angrzi   अंग्रजी  अंग्रज़ी
antah    अंतह     अंतः
```

Uploaded all the predictions on the test set in a folder **predictions_vanilla** on our github project.

(c) Errors:

- Our model gets confused with alphabets with multiple pronunciations. For example: ("j" and "z")

```
angraji  अंग्रजी  अंग्रज़ी
angreji  अंग्रजी  अंग्रज़ी
angrzi   अंग्रजी  अंग्रज़ी
```

- Model gets confused with vowels.
- Short words were transliterated more correctly as compared to longer words.

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)

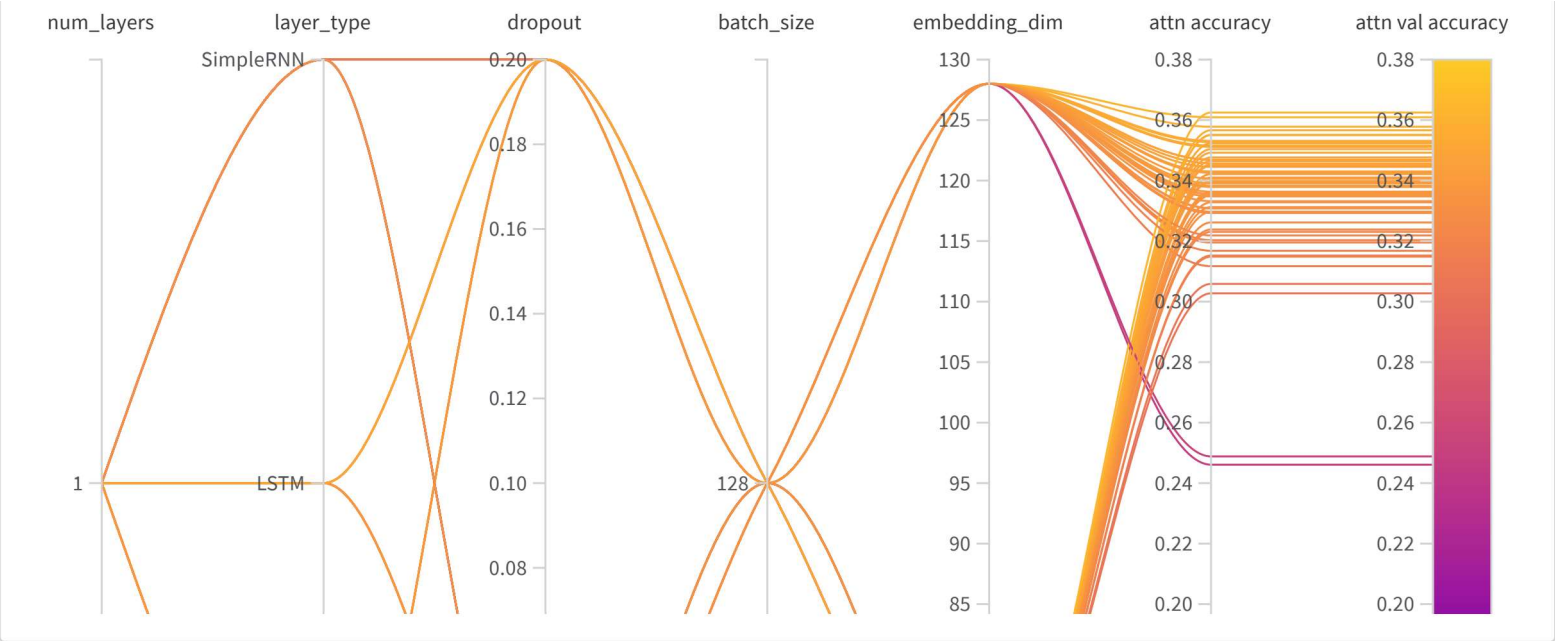
(d) In a 3 x 3 grid paste the attention heatmaps for 10 inputs from your test data (read up on what are attention heatmaps).

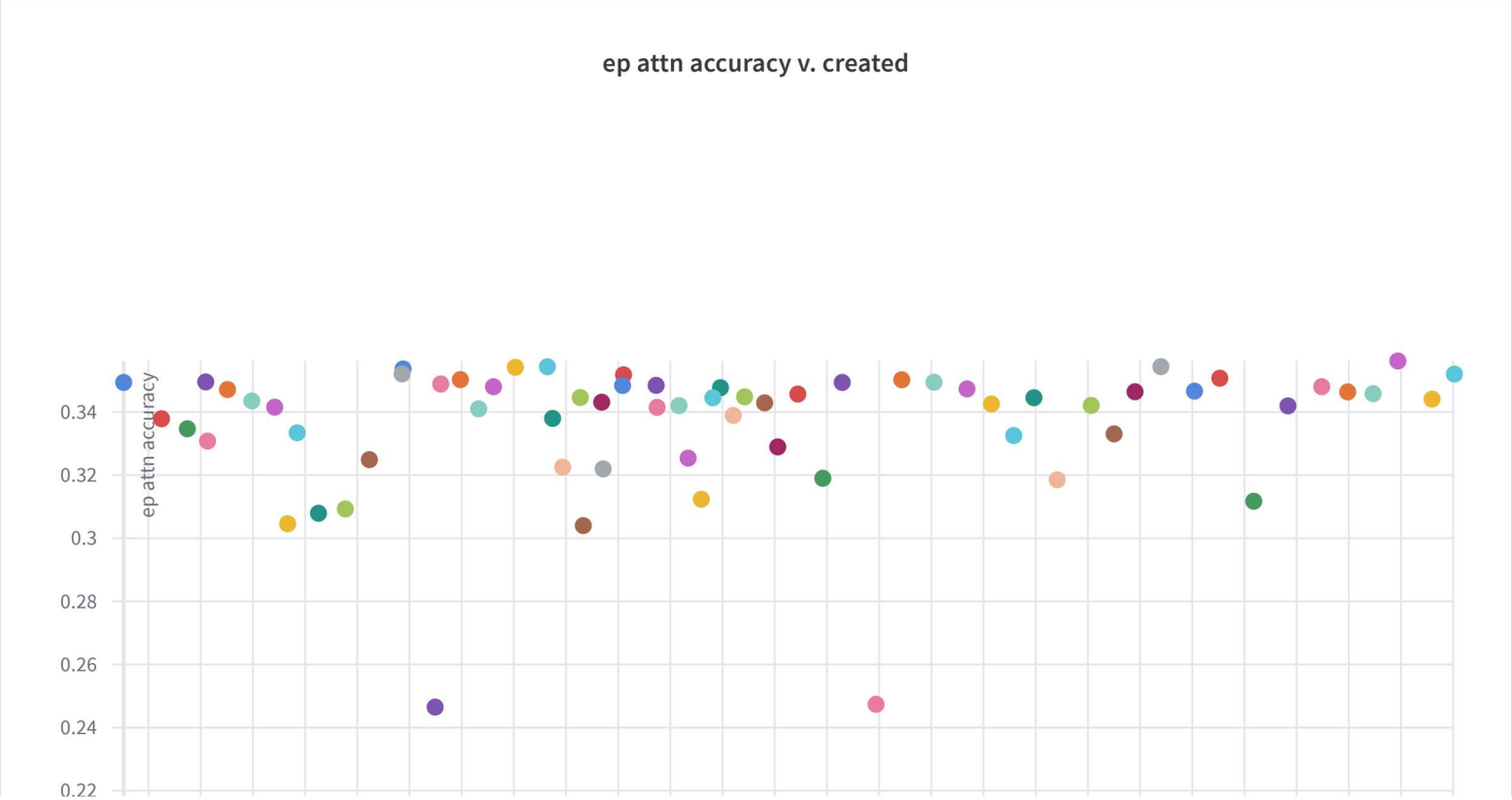
Solution:

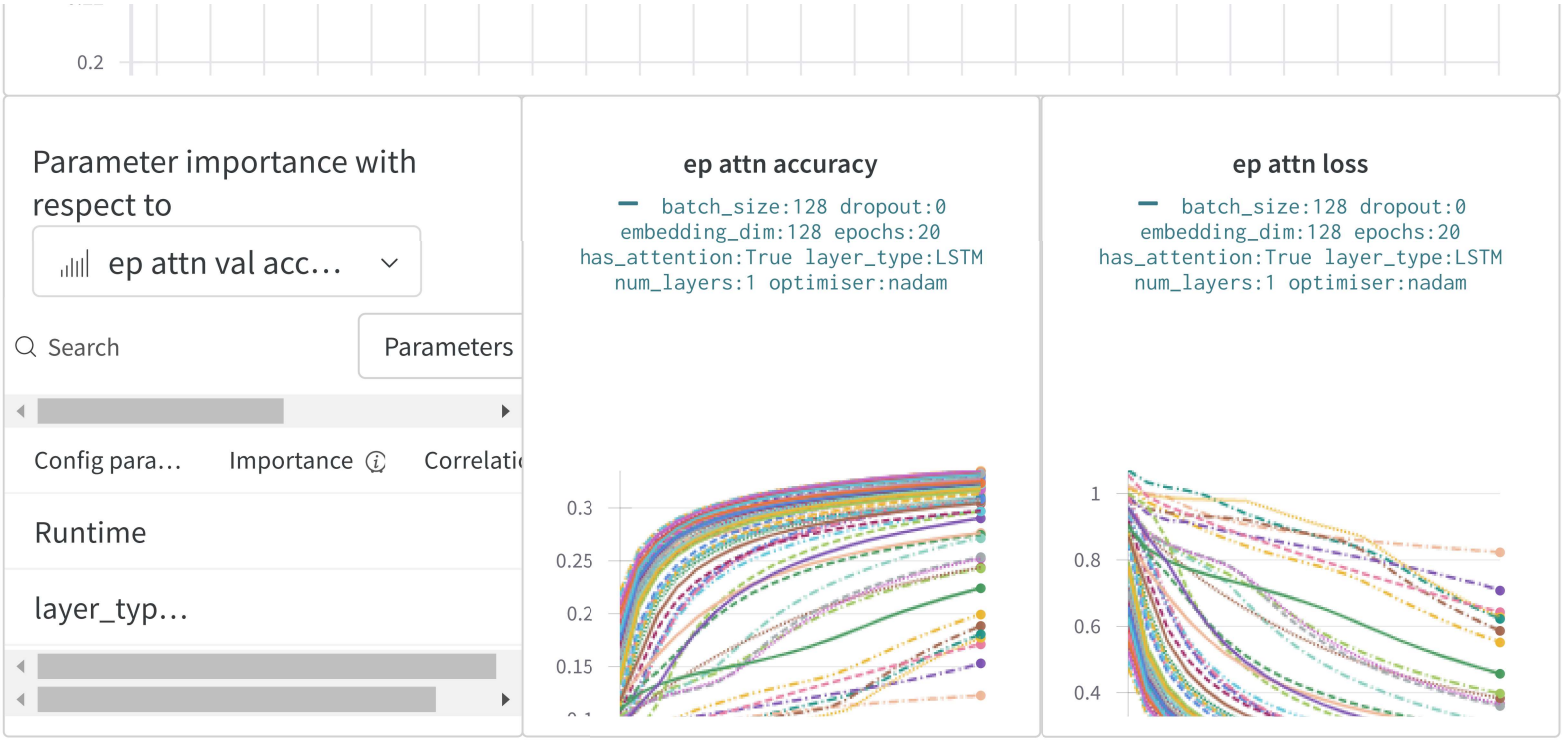
(a) Hyper parameters used:

```
"parameters": {
        "layer_type": {
            "values": ["GRU", "LSTM", "SimpleRNN"]
        },
        "dropout": {
            "values": [0.0, 0.2]
        },
        "units": {
```

```
            "values": [256, 512]
        },
        "embedding_dim": {
            "values": [64, 128]
        },
        "optimiser": {
            "values": ["nadam"]
        },
        "epochs": {
            "values": [20]
        },
        "batch_size": {
            "values": [dataset.batch_size]
        },
        "num_layers": {
            "values": [1]
        },
        "has_attention": {
            "values": [True]
        }
    }
```


num_layers
layer_type
dropout
batch_size
embedding_dim
attn accuracy
attn val accuracy
SimpleRNN
LSTM
1
128
0.20
0.18
0.16
0.14
0.12
0.10
0.08
130
125
120
115
110
105
100
95
90
85
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20


ep attn accuracy v. created
ep attn accuracy
0.34
0.32
0.3
0.28
0.26
0.24
0.22



(b) Accuracy is around ~17%. Uploaded all the predictions on the test set in a folder **predictions_attention** on our github project.

(c) Without vs with attention model's predictions.

- The attention model is able to correct errors made by out best model(without attention).
- Model with attention is able to provide accuracy similar to multi-layered model(which required more number of parameters).

Test Accuracy (Attention)	Test Accuracy (Without Attention)

0.3991

0.1816

With Attention - Predictions

Image - predictions

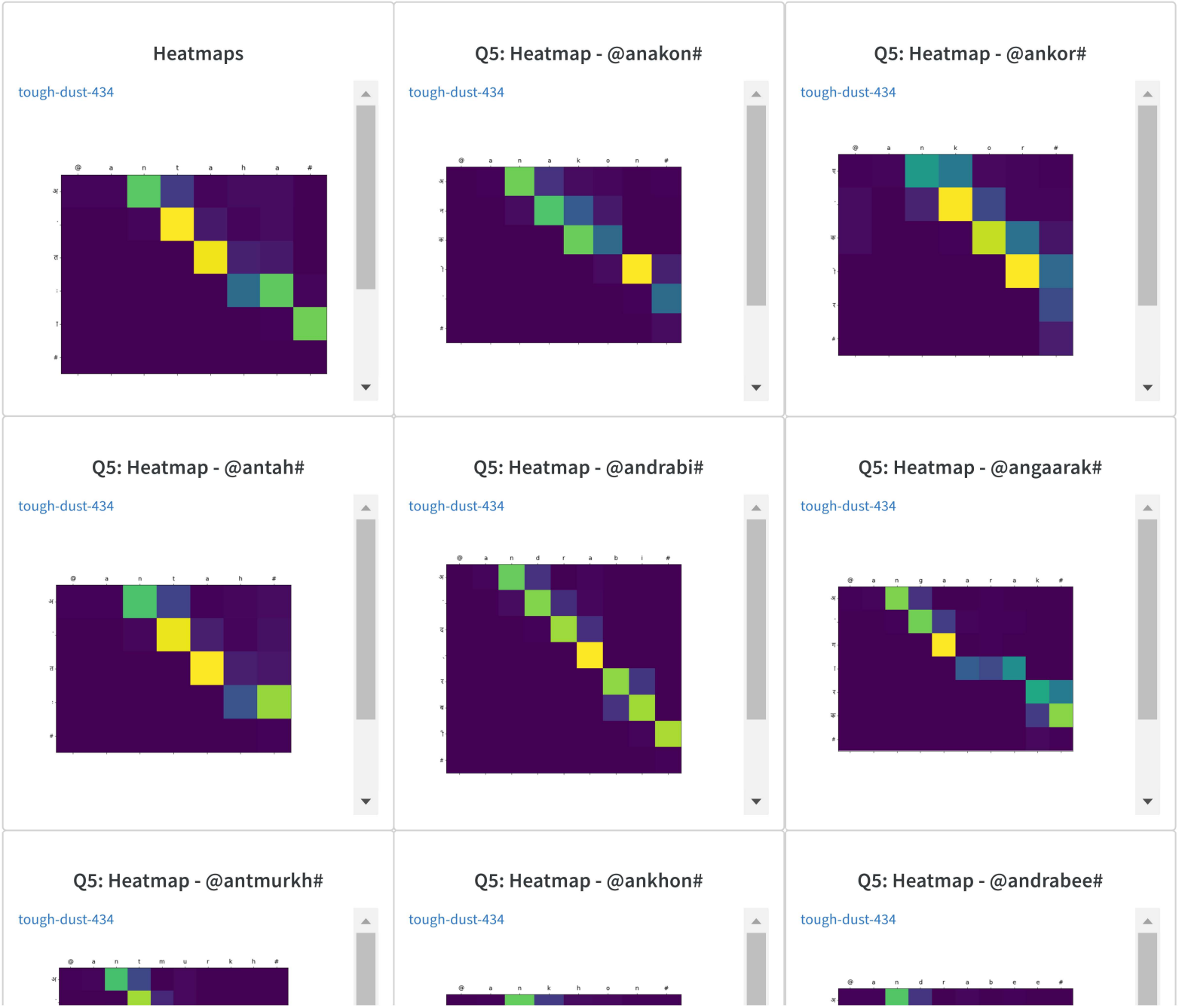
INPUT	PREDICTION	TRUE
ank	अंक	अंक
anka	अंका	अंक
ankit	अंकित	अंकित
anakon	अनकों	अंकों
ankhon	अंखों	अंकों
ankon	अंकों	अंकों
angkor	अंगकोर	अंकोर
ankor	एंकोर	अंकोर
angaarak	अंगारक	अंगारक
angarak	अंगरक	अंगारक
angraji	अंग्रजी	अंग्रज़ी
angreji	अग्रेजी	अंग्रज़ी
angrzi	अंग्रजी	अंग्रज़ी

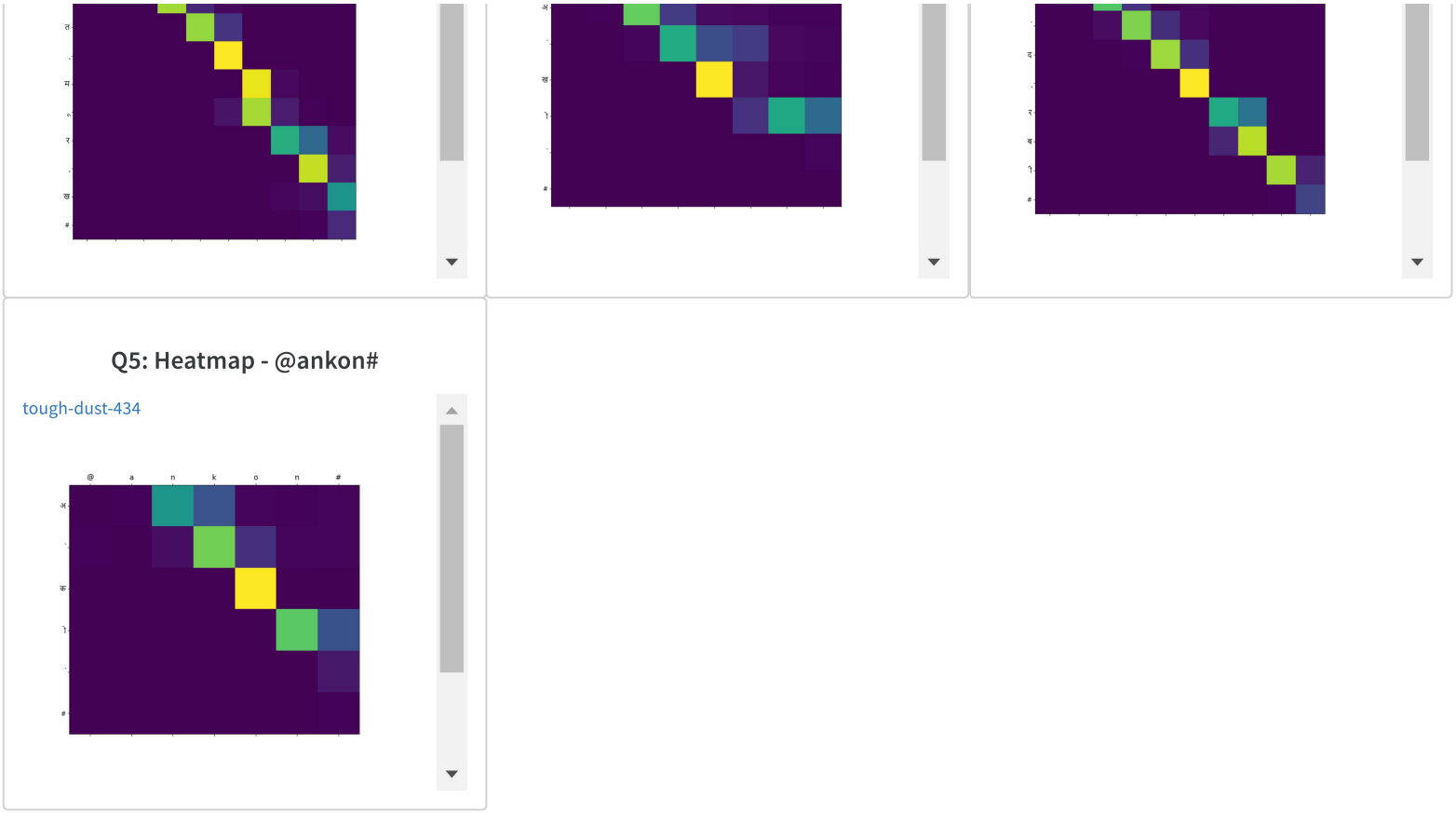
Without Attention

Image - predictions

INPUT	PREDICTION	TRUE
ank	आंक	अंक
anka	आंका	अंक
ankit	आंकित	अंकित
anakon	अनकों	अंकों
ankhon	आं	अंकों
ankon	आं	अंकों
angkor	अंगकोंगकोंगकोंगक	अंकोर
ankor	एंकोर	अंकोर
angaarak	अंगारक	अंगारक
angarak	अंगारक	अंगारक
angraji	अंग्रजी	अंग्रज़ी
angreji	अंग्रजी	अंग्रज़ी
angrzi	अंग्रजी	अंग्रज़ी

(d) Attention Heatmaps:





Question 6 (20 Marks)

This 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?

Solution

Question 7 (10 Marks)

Paste a link to your github code for Part A

Example: https://github.com/<user-id>/cs6910_assignment3/partA;

- 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.

Solution

Link: https://github.com/shibobrota/cs6910-assignments/tree/main/Assignment%203

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!

Created with ♡ on Weights & Biases.

https://wandb.ai/iitm-cs6910-jan-may-2021-cs20m059-cs20m007/Assignment%203/reports/Assignment-3--Vmlldzo1Njc0MDc