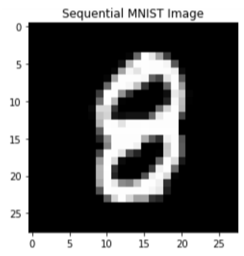
Drishya Uniyal MT21119
Prashant Sharma MT21227

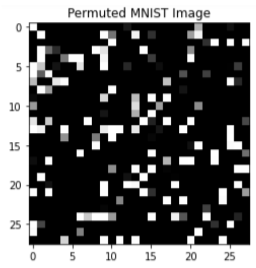
DL Assignment 3

Sequential and Permuted MNIST:
MNIST has 28 x 28-pixel images (of a handwritten digit) into one of ten digits (0 to 9). We first turn it into a stream of 784 (28x28) individual pixels, presented to the network one at a time. We then apply a fixed permutation to all of the pixel sequences. Overall the train dataset has an input 784 sequential (unordered) for a given image per label. The aim is to input this sequence and predict class labels.
Task: Develop an RNN-based model to classify the digits (ONE EACH FOR SEQ AND PERMUTATED DATASET). Perform your own splitting of train data in training, validation sets with an 80:20 ratio (random stratified) and the following setup for the 2 datasets:

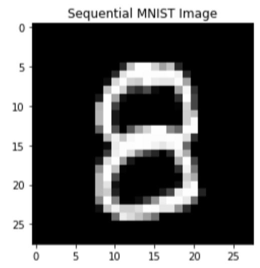
1. Visualize 3 random images with their corresponding sequence. [3 marks]



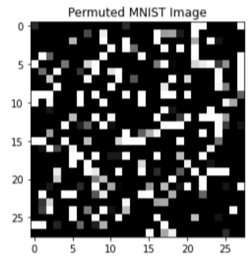
```
Sequence: [0, 0, 0, 0, 15, 15, 15, 12, 11, 10, 10, 19, 18, 10, 10, 11, 11, 10, 10, 10, 10, 10, 10, 12, 0, 0, 0, 0]
Label: 8
```



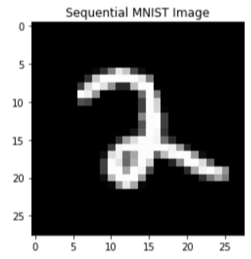
```
Sequence: [0, 1, 23, 8, 1, 9, 0, 25, 1, 15, 13, 15, 7, 4, 25, 8, 23, 20, 26, 1, 12, 0, 17, 22, 6, 0, 21, 11]
Label: 6
```



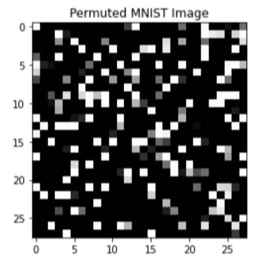
```
Sequence: [0, 0, 0, 0, 0, 13, 13, 11, 10, 9, 9, 9, 9, 9, 13, 13, 11, 10, 9, 9, 20, 19, 9, 14, 11, 0, 0, 0]
Label: 8
```



```
Sequence: [21, 16, 9, 10, 1, 0, 3, 7, 3, 0, 0, 17, 8, 4, 12, 0, 2, 5, 5, 7, 18, 8, 1, 12, 2, 5, 0, 15]
Label: 2
```



```
Sequence: [0, 0, 0, 0, 0, 0, 11, 11, 8, 6, 15, 15, 16, 16, 15, 14, 16, 14, 21, 11, 11, 12, 0, 0, 0, 0, 0, 0]
Label: 2
```



```
Sequence: [13, 3, 5, 6, 8, 26, 9, 18, 6, 24, 20, 15, 11, 3, 16, 2, 5, 0, 7, 5, 4, 0, 1, 24, 5, 27, 2, 15]
Label: 2
```

2. Propose and implement an RNN based architecture to predict the digit in the image. You can incorporate all the techniques you have studied so far of weight initialization, attention, loss functions, regularization etc as long as your base model is RNN based and your final layer is a classification head. No transformer based system allowed.

Sol: RNN has been implemented in three stages, simple, with weight initialization, and then with attention. The layers have been implemented from Layers in keras.

Sequential MNIST

	Train Acc	Val Acc	Loss
Simple RNN	0.9927	0.987	0.0463
WI + Reg	0.9933	0.9875	0.0699
WI + Reg + Attention	0.9950	0.98675	0.0516

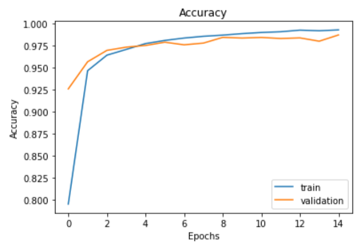
Permutated MNIST

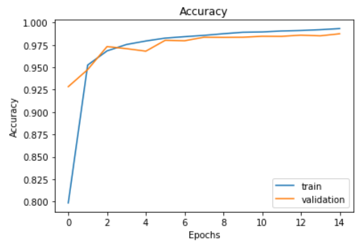
	Train Acc	Val Acc	Loss
Simple RNN	**0.98**	**0.96**	**0.0810**
WI + Reg	**0.9927**	**0.98**	**0.0770**
WI + Reg + Attention	**0.9731**	**0.96**	**0.1199**

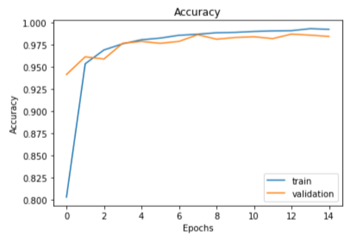
Results: The model with WI + R + Attention gave the best results in both the cases, though he accuracy may be less in Permutated but the predictions on val set are better.

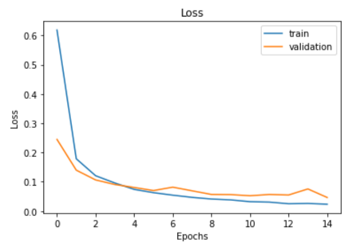
3. Show validation and training loss plots vs. the number of epochs
Sol:

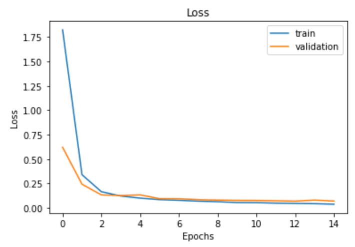
Sequential:

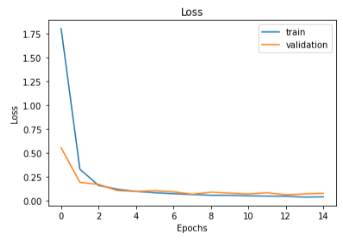




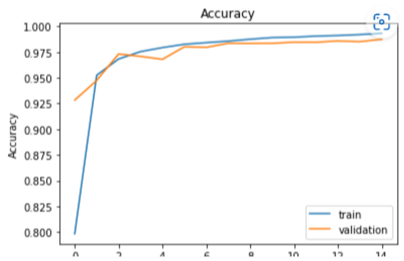


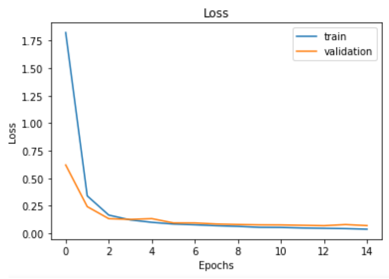


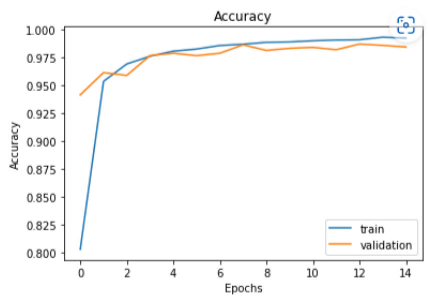


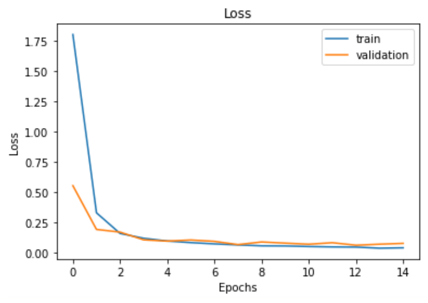


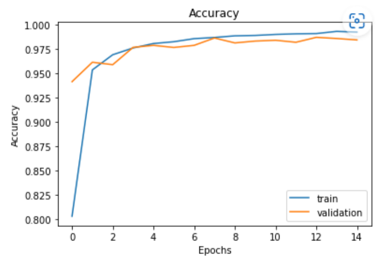
Permutated:

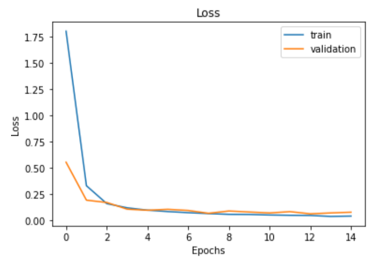








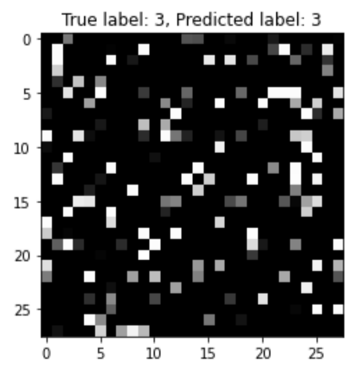


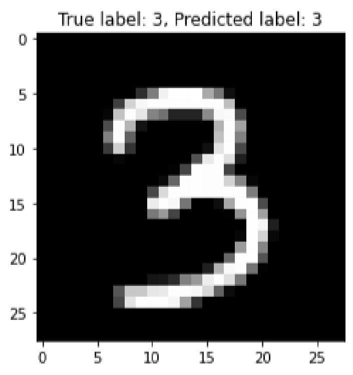


4. Report the class-wise F1 for your validation set.

Simple RNN	[0.99243697 0.98963731 0.98624427 0.9893617 0.98047538 0.98700093 0.99070946 0.99239087 .97985426 0.98098859]
WI + Reg	[0.99074074 0.99259808 0.98950021 0.98852459 0.98492029 0.98708487 0.99033207 0.987251 0.98462852 0.97858043]
WI + Reg + Attention	[0.99194574 0.99259259 0.98652064 0.97709924 0.99016674 0.97802712 0.9907563 0.9884139 0.98794143 0.982881]

Simple RNN	[0.97492563 0.98156342 0.96296296 0.95299317 0.94112516 0.95318352 0.97468354 0.96245458 0.9527897 0.92654904]
WI + Reg	[0.98902954 0.9925871 0.98501249 0.98334011 0.97758985 0.98607242 0.98205128 0.98730159 0.98329764 0.97834395]
WI + Reg + Attention	[0.97895623 0.98298817 0.96643183 0.96314496 0.96031061 0.94870588 0.97886729 0.97075548 0.94648125 0.95249897]

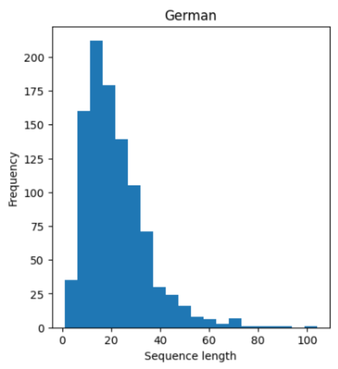


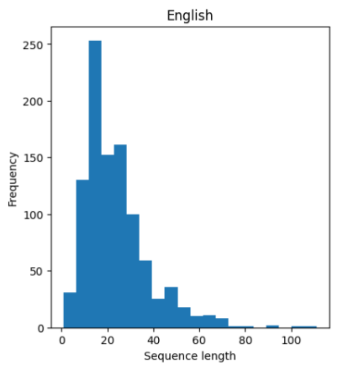


5. Prepare an inference pipeline that during the demo will load your best model and the test dataset (we will provide) and run it live. Each team will be ranked based on the test set F1. The marks out of 10 will be based on your team's rank. We can ask you to run either of the systems (SEQ or PERMUTED).
Sol: For test file (expected as sequential_test.pickle), the input has been taken and images and labels have been modified. The best model has been saved for both sequential and permutated which will be loaded.

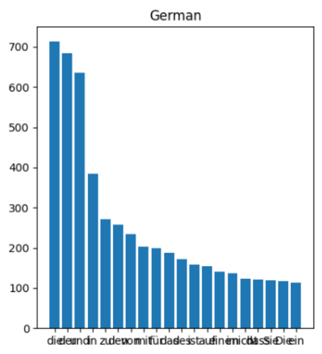
__Task 2 (Translation)__
1. Visualise the dataset using five different techniques (sequence length, frequency of words, mean token length, word cloud, etc) to show how the 2 languages differ.

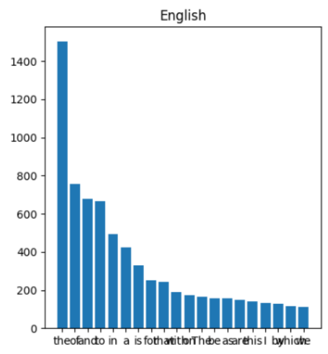
Sequence Length



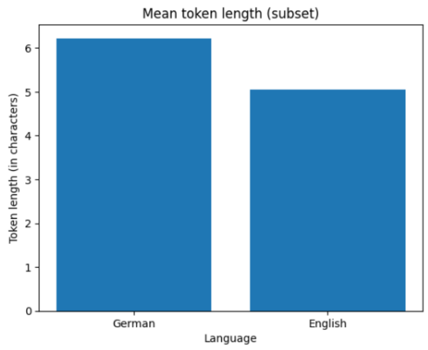


Sequence length for the German language seems to be more than that in English.
Frequency of Words



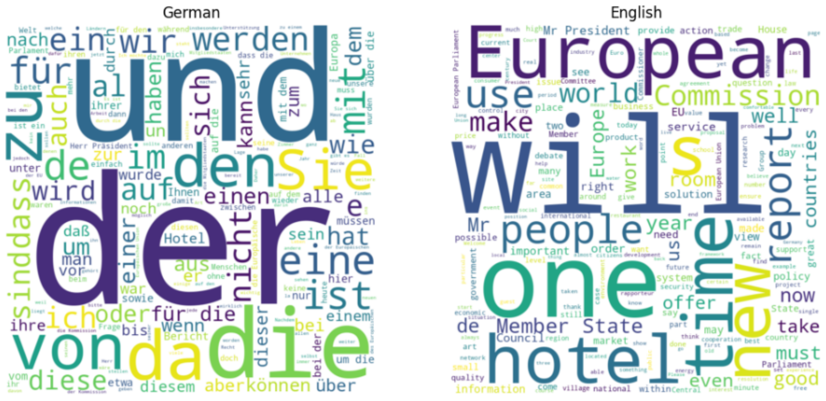


Mean Token Length

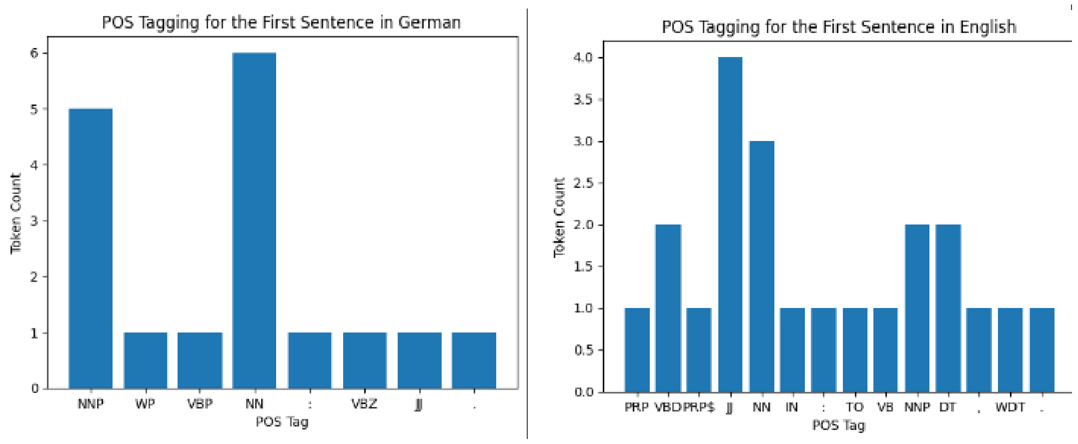


The mean token length for ENglish is lesser than that in German.

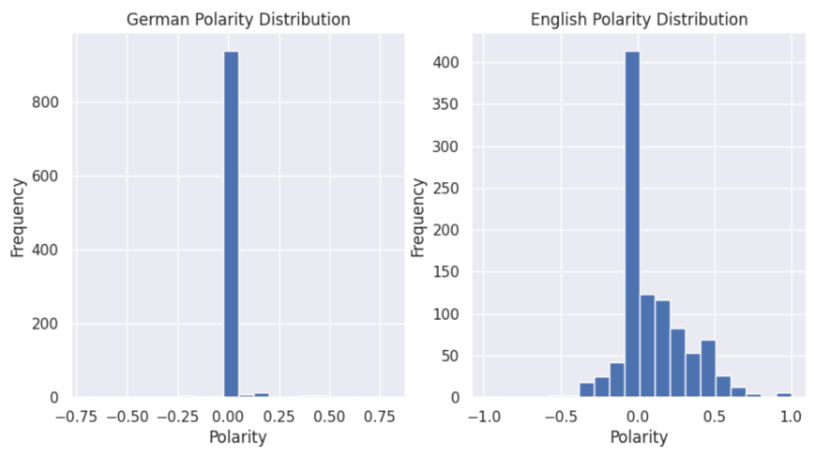
Word Cloud



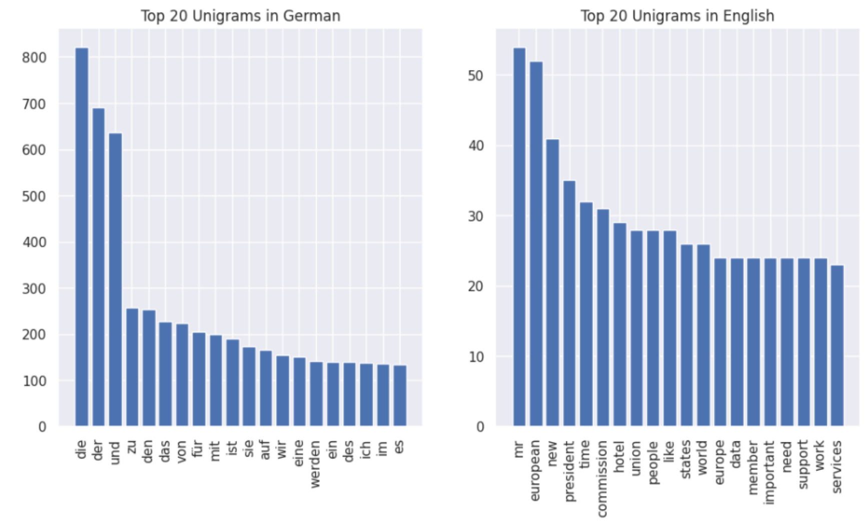
As can be observed that der and und have the most frequency in German and will one have the highest in English. We have not removed the stopwords from this that can be done.



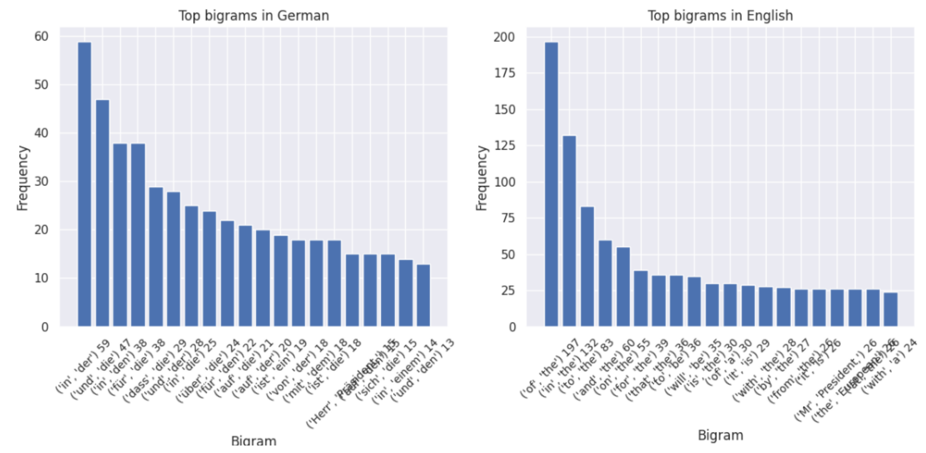
This graph suggests the POS tagging for the first sentence in both languages.



For this graph, the polarity of the sentences from each language was calculated using TextBlob and then plotted. For German we can see mostly the sentences tend to be neutral when converted, neutral sentences are more, but we can also see positive and negative.



This graph shows the unigram distribution for German we have die which has been used the most and for English we have mr.



This graph shows the bigram distribution for German we have (in, der) which has been used the most and for English we have (of, the).

.2. Implement Model A with: one with non-contextualized initialised embeddings (or random embeddings) + LSTM based

- Implemented the non-contextualized initlaised embedding, took Embedding layer and parameter Trainable is kept as True.
- Basic LSTM model used.

3. Implement Model B with: one with non-contextualized initialised embeddings (or random embeddings) + LSTM + global attention based

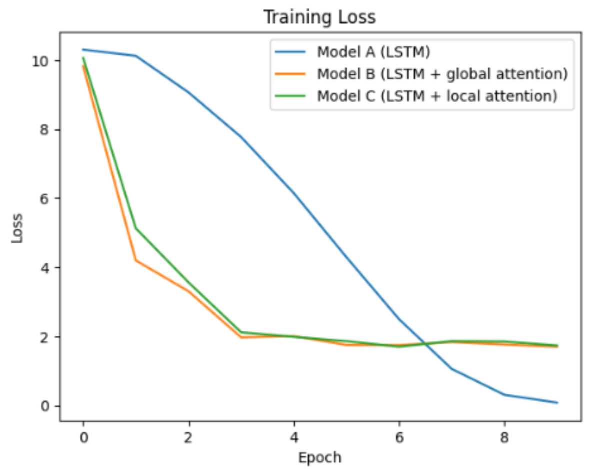
- Implemented the non-contextualized initlaised embedding, took Embedding layer and parameter Trainable is kept as True.
- Attention weight mechanism introduced
- The attention_weights are computed from the entire input sequence and used to compute a context vector for each time step of the LSTM.
- Global attention mechanism

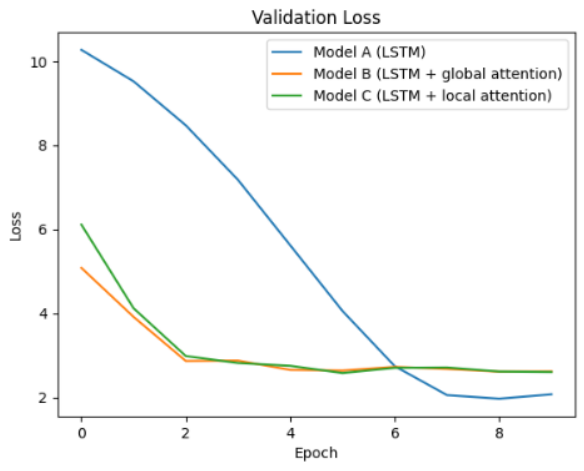
4. Implement Model B with: one with non-contextualized initialised embeddings (or random embeddings) + LSTM + local attention based

- Implemented the non-contextualized initlaised embedding, took Embedding layer and parameter Trainable is kept as True.
- Attention weight mechanism introduced
- The attention weights are calculated using a dense layer with a hyperbolic tangent activation function, then a softmax activation function is applied to get the final attention weights. The Permute layer is used to swap the second and third dimensions of the attention weights tensor, and the Dot layer performs a matrix multiplication between the attention weights tensor and the LSTM layer tensor to get the context vector. This context vector is then flattened, repeated, and reshaped before being concatenated with the LSTM output and passed through a dense layer to produce the final output.
- Local attention mechanism

The models have been tried on a subset as it was crashing for the whole dataset.

5. Show plots of validation and training loss vs. number of epochs for Models A, B, C.





6. Report the overall Rouge-L score, Bleu-1,2 score for the test set. (metrics available in Huggingface metric) for Models A, B, C.

	Rouge-L	Bleu-1	Bleu-2
Model A	0.3742	0.4727	0.3136
Model B	0.4197	0.5205	0.3633
Model C	0.3942	0.5001	0.3349

7**. Out of the 3 models, which setup performed best and why?**
Gloabl Attention performs better in general when the data has longer sequence lengths but here we observed from the plot in Q2 part 1 that for both the languages, the sequence length is approx 20 to 30 amd not very long still they might cotaine relevant informations which Model B used and performed better. Model B uses a global attention mechanism, which allows the decoder to focus on different parts of the source text when generating the output sequence. This helps the model to capture the relevant information from the source text and produce more accurate and fluent translations.Model A and Model C do not have any attention mechanism and rely solely on the hidden state of the LSTM to encode the input sequence.
The global attention mechanism introduces more parameters to the model, which allows it to learn more complex patterns in the input sequence. This increased complexity helps the model to capture more fine-grained details in the input sequence and produce better translations.

Task 3 (Transformer)
1. Atleast one layer of your fine-tuined model should be set to trainable. You are free to employ any model (even distilled ones.). Huggingface models already tuned on WNT16 dataset cannot be used. Special case should be taken for overfitting, include appropriate measures if overfitting is observed. [10 marks]
2. Show plots of validation and training loss vs. number of epochs for the model. [1 marks]
3. Report the overall Rouge-L score, Bleu-1,2 score for the test set. (metrics available in Huggingface metric) for the model. [2 marks]

Rouge-L score	0.52100
Bleu-1	0.715500
Bleu-2	0.506500

4. What can be said about the quality of output generated by models in Task 2 vs Task 3. [2 Marks]
The output with bert came out better it produced good results.BERT is pre-trained on large amounts of text data, which allows it to capture rich semantic and contextual information. This pre-training makes it easier to fine-tune the model on specific downstream tasks, such as machine translation, which results in better performance compared to models that are trained from scratch.So Task 3 generated better translations.