HW3 Report

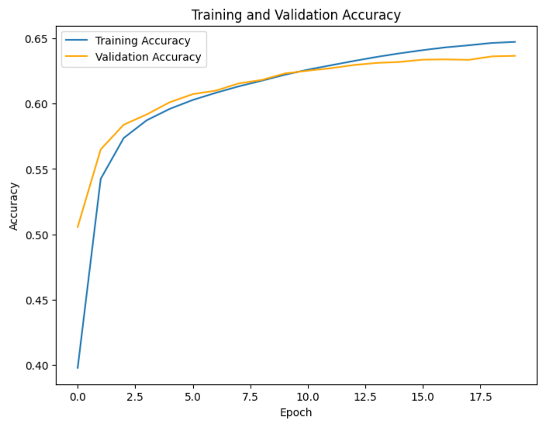
312511052 李元中

RNN

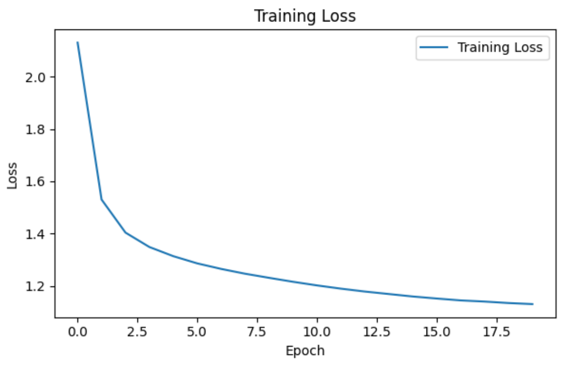
1. network architecture

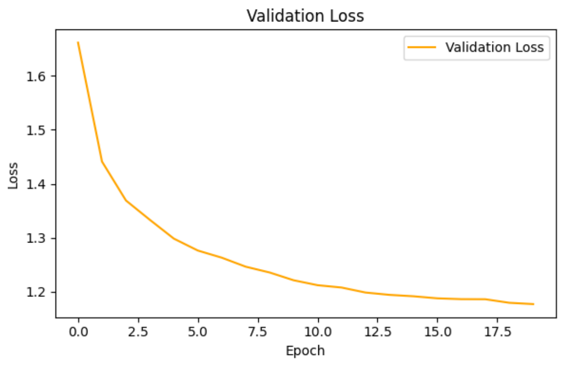
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (64, None, 256)           17152

 gru_2 (GRU)                 (64, None, 1024)          3938304

 dense_2 (Dense)             (64, None, 67)            68675

=================================================================
Total params: 4024131 (15.35 MB)
Trainable params: 4024131 (15.35 MB)
Non-trainable params: 0 (0.00 Byte)
```

2. learning curve



3. training error rate and validation error rate





4. checkpoints

   (1) ckpt_1

```
input = peo
output = u
```

   (2) ckpt_2

```
input = peo
output = k
```

   (3) ckpt_3

```
input = peo
output = u
```

   (4) ckpt_4

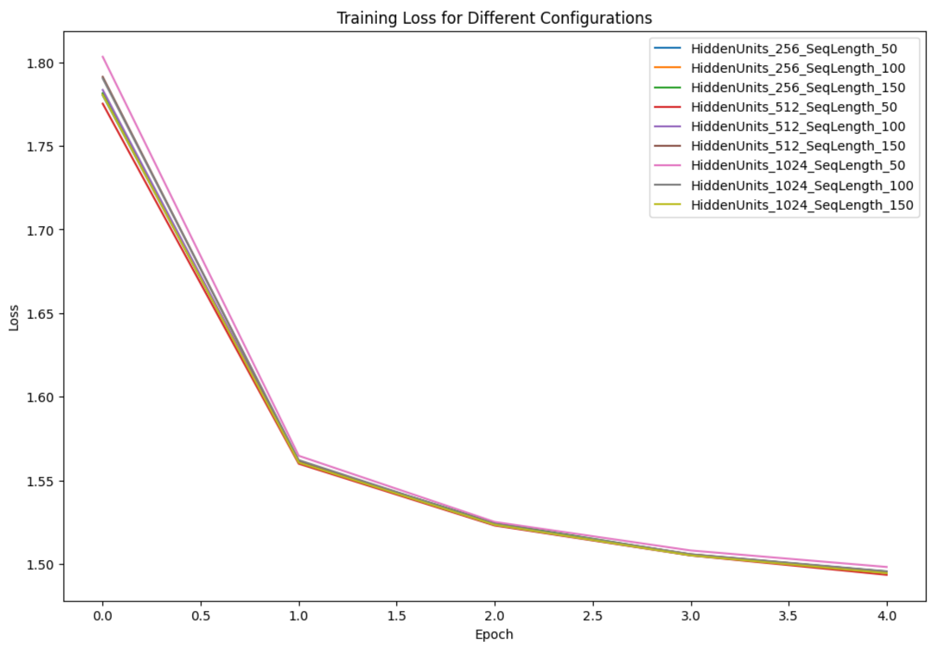
```
input = peo
output = u
```

   (5) ckpt_5

```
input = peo
output = p
```

   With different checkpoints, we can observe that the model is improving over time. Taking the input "peo" as an example, the model has learned what should be the next output, capturing the pattern such as the initial "p" in most occurrences of the word "people."

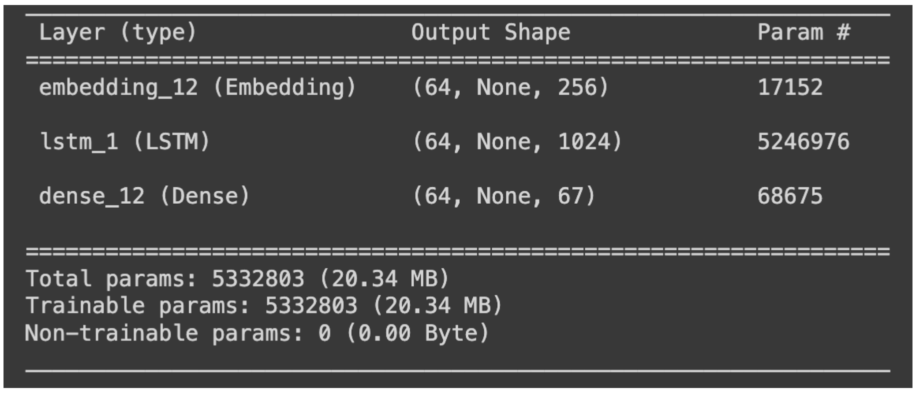
5. Different size of hidden states and sequence length



In the context of Recurrent Neural Networks (RNNs), the choice of hidden state size plays a crucial role in the model's learning dynamics. A smaller hidden state, such as 256 units, might struggle to capture intricate patterns, potentially resulting in slower convergence and a higher training loss. Opting for a medium-sized hidden state, around 512 units, strikes a balance between model complexity and training efficiency, often leading to smoother convergence. On the other hand, a larger hidden state, like 1024 units, may exhibit faster initial convergence, but there is a heightened risk of overfitting, particularly if the dataset lacks sufficient diversity. Regarding sequence lengths, shorter sequences, such as 50, might limit the model's ability to grasp long-term dependencies, whereas medium sequences (e.g., 100) often offer a pragmatic compromise between context and computational efficiency. Longer sequences (e.g., 150) provide an extended context but might demand more computational resources.

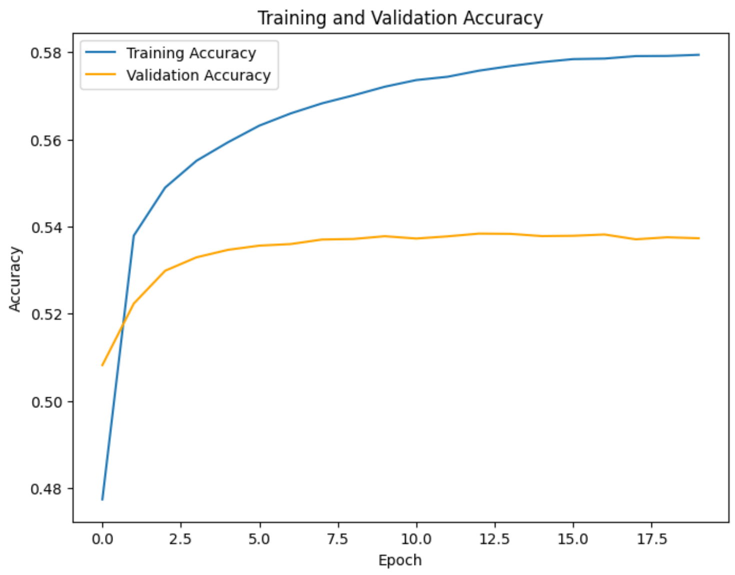
LSTM

1. network architecture

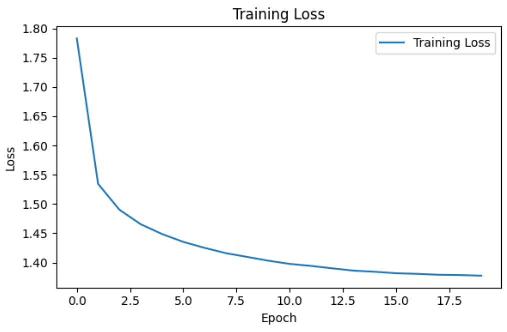


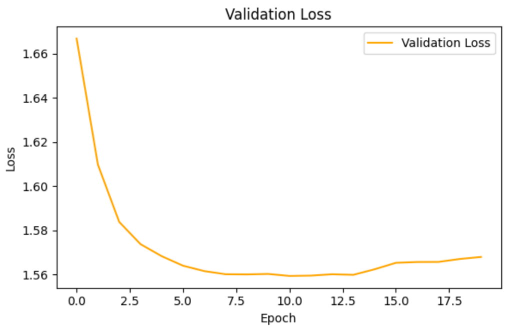
```
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_12 (Embedding)    (64, None, 256)           17152

 lstm_1 (LSTM)               (64, None, 1024)          5246976

 dense_12 (Dense)            (64, None, 67)            68675

=================================================================
Total params: 5332803 (20.34 MB)
Trainable params: 5332803 (20.34 MB)
Non-trainable params: 0 (0.00 Byte)
```

2. learning curve



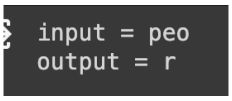
3. training error rate and validation error rate





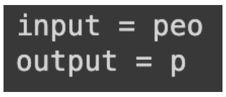
4. checkpoints

(1) ckpt_1



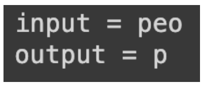
```
input = peo
output = r
```

(2) ckpt_2



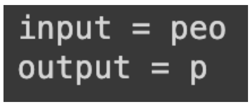
```
input = peo
output = p
```

(3) ckpt_3



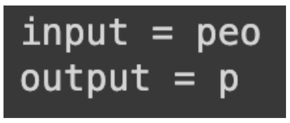
```
input = peo
output = p
```

(4) ckpt_4



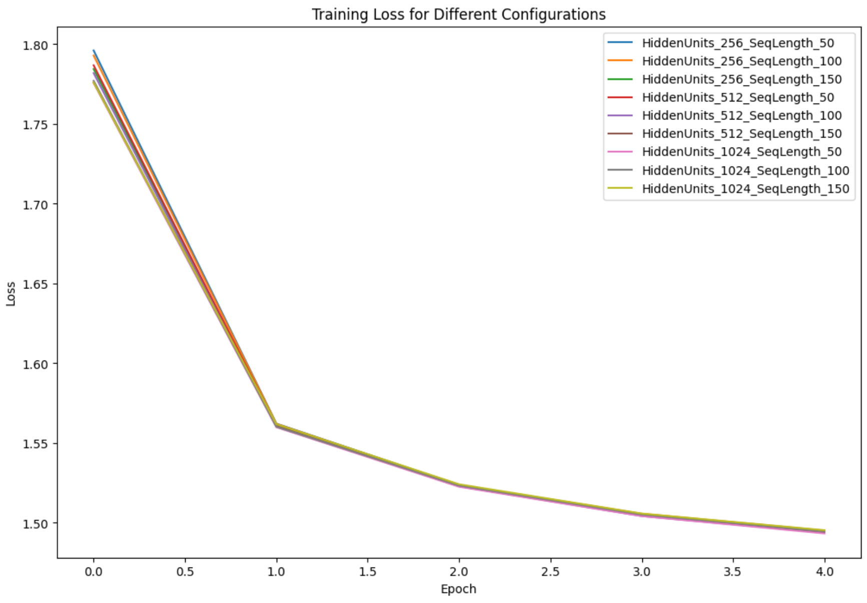
```
input = peo
output = p
```

(5) ckpt_5



```
input = peo
output = p
```

5. Different size of hidden states and sequence length



In the realm of Long Short-Term Memory (LSTM) networks, the size of hidden states influences the model's ability to capture complex temporal dependencies. Smaller hidden states, say 256 units, may offer more stable learning by mitigating vanishing gradient issues, yet might struggle with intricate pattern recognition. Medium-sized hidden states (e.g., 512 units) often strike a balance, facilitating effective learning of both short and long-term dependencies. Larger hidden states (e.g., 1024 units) enhance the LSTM's capacity to capture intricate patterns but may increase susceptibility to overfitting. Concerning sequence lengths, shorter sequences like 50 may hinder the LSTM's potential to understand longer-term dependencies. Medium sequences (e.g., 100) are versatile, efficiently balancing context and computational efficiency, while longer sequences (e.g., 150) extend the context, aiding the LSTM in capturing dependencies over more extensive spans, albeit at a higher computational cost.

COMPARISON

The distinction between standard Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks becomes evident in their capacity to handle temporal dependencies and challenges related to vanishing gradients. Standard RNNs often grapple with vanishing gradients, impeding their ability to effectively capture and retain information over extended sequences. Consequently, this limitation manifests as slower convergence and potential difficulty in learning intricate sequential patterns. In contrast, LSTMs are tailored to mitigate the vanishing gradient problem through the incorporation of memory cells and gating mechanisms. This architectural enhancement equips LSTMs with the ability to more efficiently capture and remember long-range dependencies, resulting in faster convergence and enhanced performance, particularly in tasks requiring nuanced modeling of temporal structures.

The handling of long-term dependencies further underscores the divergence between these architectures. Standard RNNs, hindered by vanishing gradients, may struggle to maintain information across lengthy sequences, impacting their proficiency in understanding dependencies spanning distant time steps. On the contrary, LSTMs excel in managing long-term dependencies. The inclusion of memory cells allows for selective storage and retrieval of information over extended periods, positioning LSTMs as effective solutions for tasks demanding the modeling of intricate and distant dependencies, such as language modeling and sequence generation.

Consideration of overfitting and generalization reveals additional disparities. Standard RNNs, with their simpler architectures, are more susceptible to overfitting, especially when faced with complex datasets or limited training sets. Conversely, LSTMs, with their enhanced complexity and adaptive mechanisms, exhibit improved generalization, adeptly capturing intricate patterns without succumbing to overfitting too quickly. This characteristic makes LSTMs particularly suitable for tasks where robust generalization to unseen data is paramount.

Lastly, the ease of training and hyperparameter sensitivity further distinguishes these architectures. Standard RNNs, grappling with the vanishing gradient issue, pose greater challenges during training and exhibit higher sensitivity to hyperparameter choices, such as learning rates and initialization methods. In contrast, LSTMs, designed to mitigate these challenges, offer more stable training dynamics and are less sensitive to hyperparameter tuning, contributing to a smoother and more robust training process. In essence, the key differences between standard RNNs and LSTMs lie in their architectures and the adeptness of LSTMs in addressing critical challenges associated with sequential data modeling.

priming the model

```
JULIET:
In truth, ones may drown your trusten nights.
My dukedow him rusty have to dine, and on our England, whom we might.

Servant:
So this is Diomedest stand enow it was when you were worth
That Brutus have weight o'er;
And in this tongued has at this praise over
Is profit value down and make a sore within
the earth for want of pup bewith the drowsy time,
For all the patience of Venice, this had not been with, we, my best-skill be known
To meet him to yourselves;
And 'tis no slave, and west
Our quiet, which cannd with her;
But dear, and present thou mayst see him post
To this proce posseritary-mangled Titus,
You know let me hear this grace
shroubled mine aspect, Noble Macbeth, 'tis safe,
Let me not cheer it.

CRESSIDA:
O, 'tis up, I tout and heartily;
I send the time with the daintiest charm be said
Against their own disgrace ambitions:
Sellect your respect: if we have anster,
But marcleth you, we must content for that man's
saw Sir John.

FLUELLEN:
I'll have a tongue that he unvested tho
```

Using RNN or LSTM models to generate text by priming them with a specific word, such as "JULIET" in the context of a Shakespeare dataset, allows for the creation of coherent and contextually relevant output. The priming process initiates the generation with a seed word, and the model iteratively builds on its internal representations to produce subsequent lines of text. This approach showcases the model's ability to capture the linguistic style and patterns ingrained in the training data. The generated output reflects the influence of Shakespearean language, providing a glimpse into the model's capacity to continue and expand upon established literary contexts. While the quality of the output depends on the model's training and architecture, this priming technique demonstrates the potential of RNNs and LSTMs in creative text generation within a specific thematic framework.