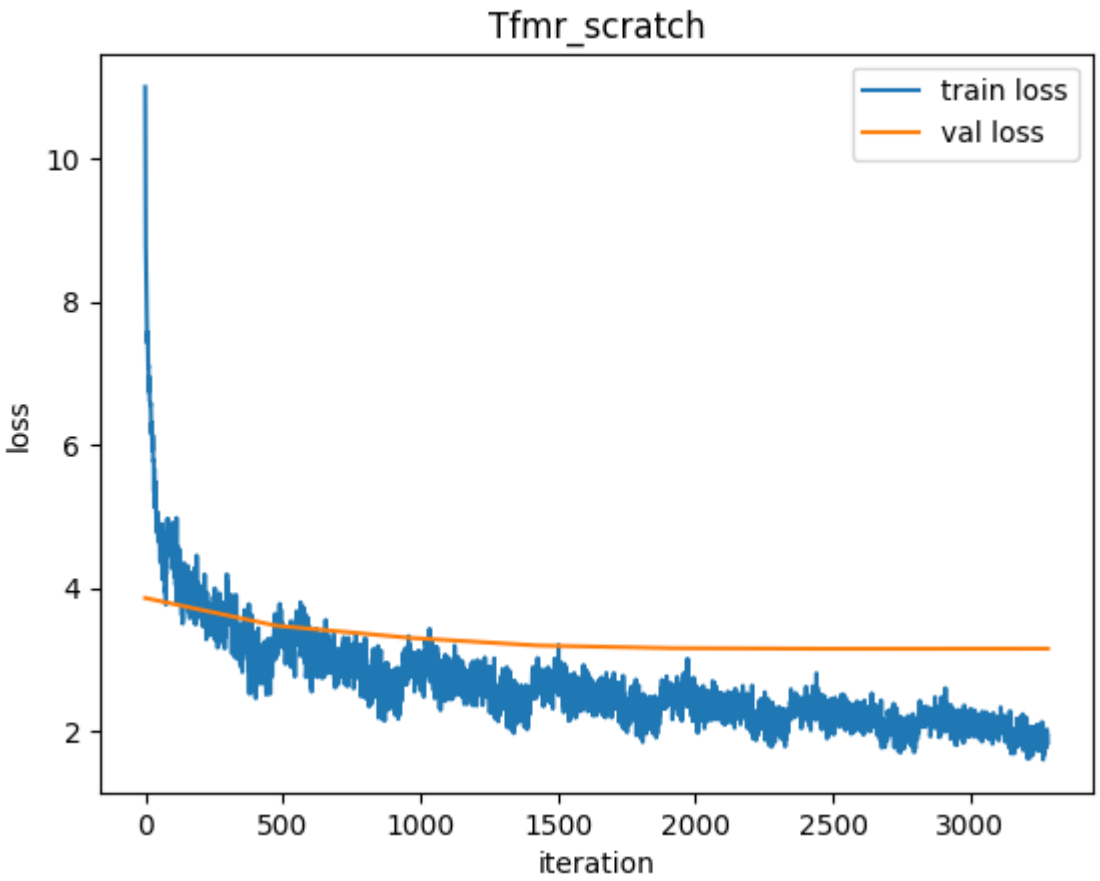
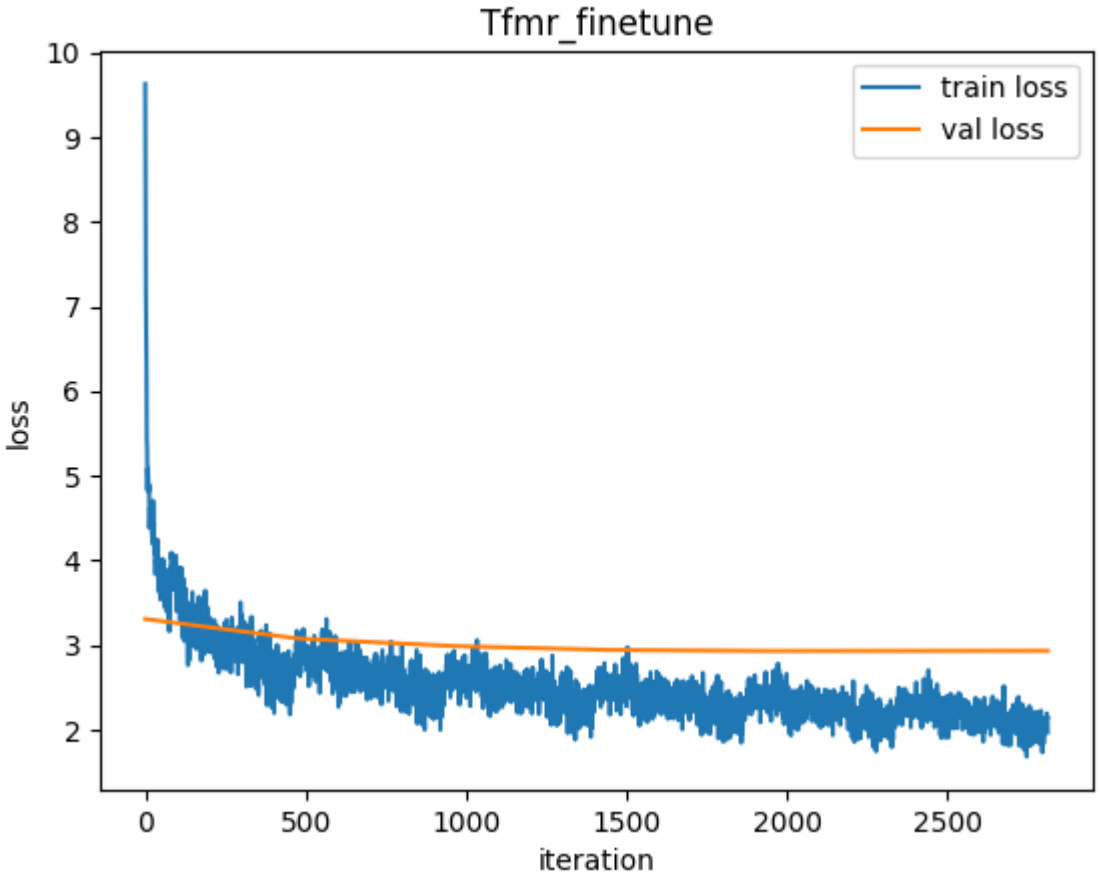
HW3

张驰 2022010754 zhang-ch22@mails.tsinghua.edu.cn

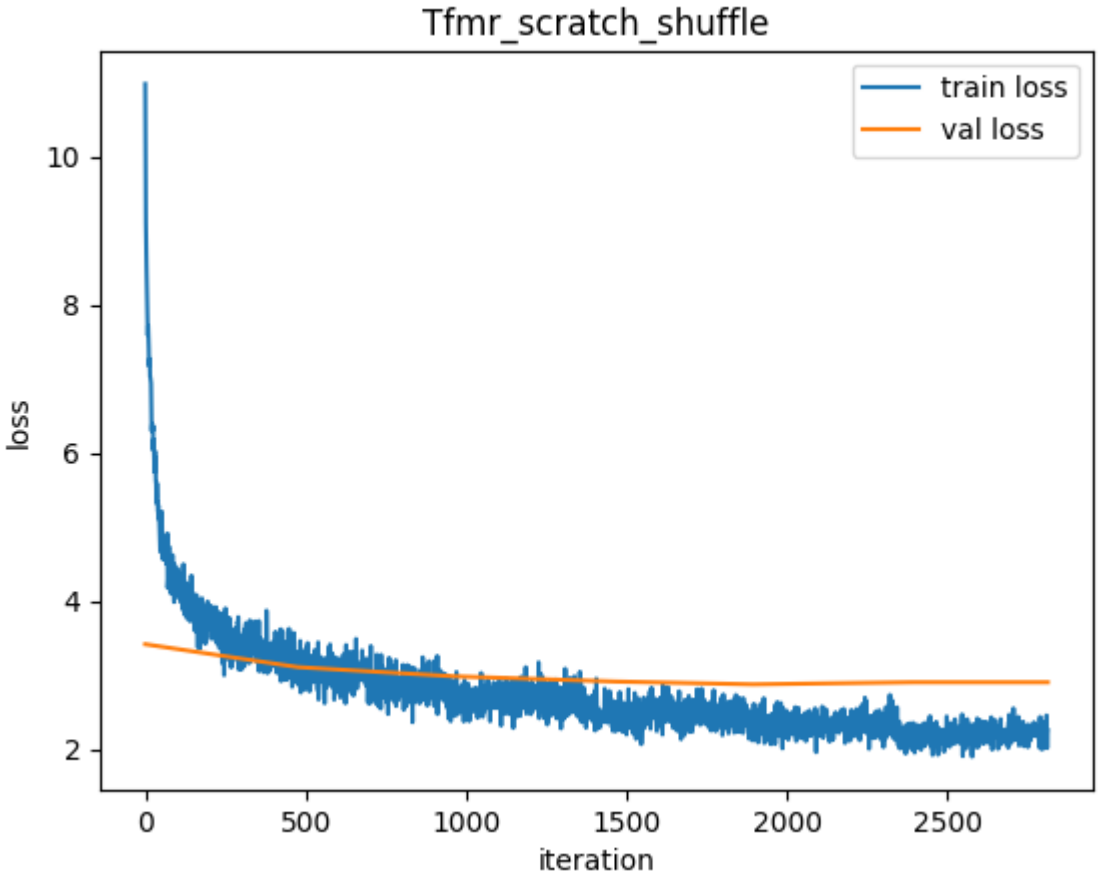
1. Adjustment to the training process

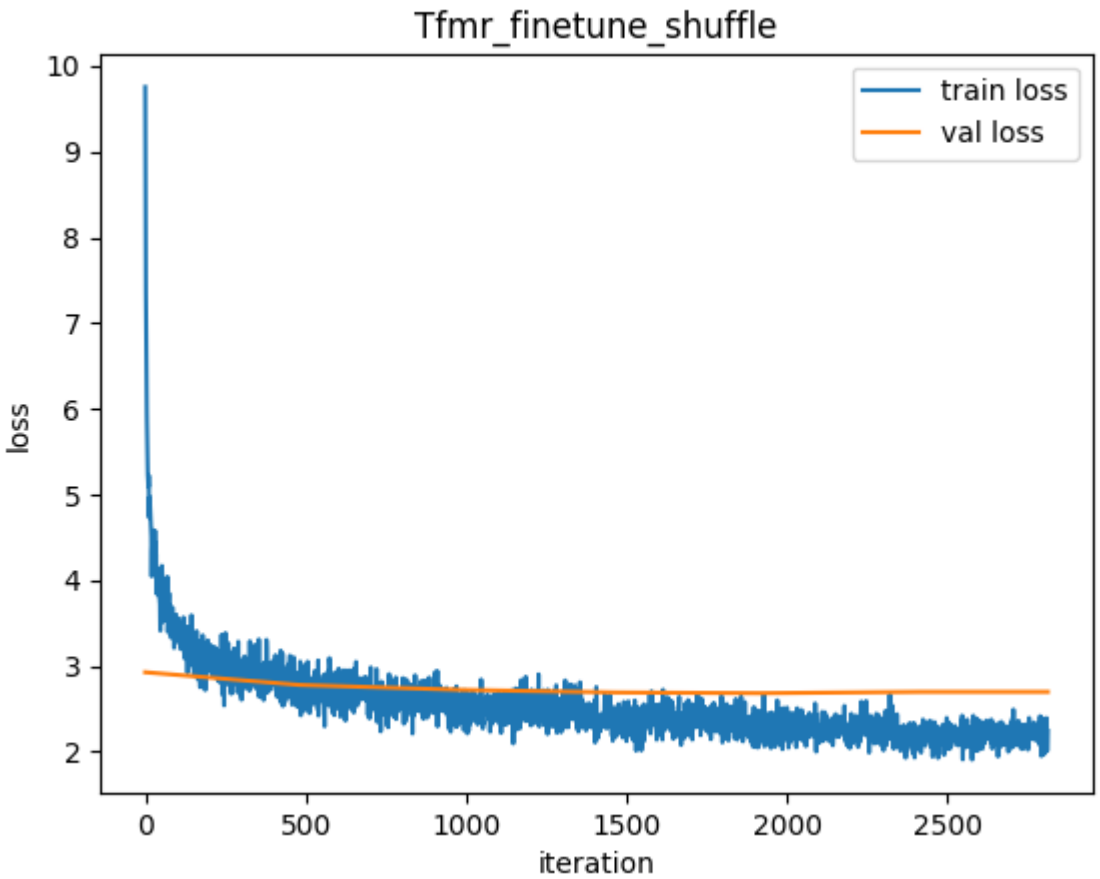
Using the default configurations of the model, the train/validation losses are as follows:



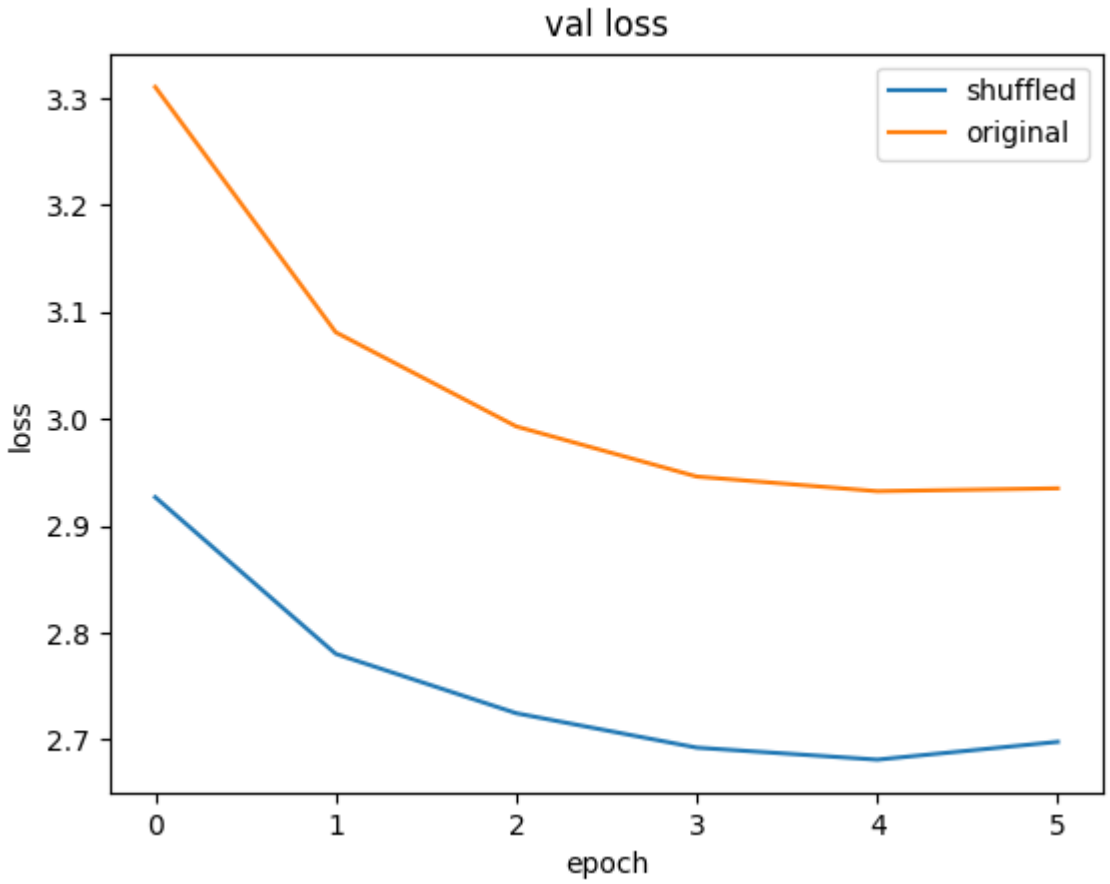


In each epoch, the training loss is decaying normally. But at the beginning of each epoch, the loss displays a sudden increase. From discussions with other classmates, I was inspired that this phenomenon may arise from the unbalanced distribution of the training data. To verify this, I randomly shuffled the training data before giving it to the model at each epoch. The results are as follows:


Tfmr_scratch_shuffle
train loss
val loss
loss
iteration
10
8
6
4
2
0
500
1000
1500
2000
2500


Tfmr_finetune_shuffle
train loss
val loss
loss
iteration
10
9
8
7
6
5
4
3
2
0
500
1000
1500
2000
2500

The significant leap at the onset of each epoch is gone. Meanwhile, after shuffling, the final validation loss and perplexity of the model display a notable decrease.



Therefore, in the experiments below, I will use the shuffled training data.

2. Effect of pretraining

The loss curve has been previously presented. Below, I will further assess the performance of different models by employing perplexity and BLEU scores. The results are as follows:

> The BLEU scores are computed using the default inference settings, with random decode strategy and temperature 1.0.

Model	Perplexity	Forward-BLEU	Backward-BLEU	Harmonic-BLEU
Scratch	18.774	0.575	0.428	0.491
Scratch-Shuffled	15.462	0.570	0.505	0.535
Finetune	15.334	0.573	0.426	0.489

Model	Perplexity	Forward-BLEU	Backward-BLEU	Harmonic-BLEU
Finetune-Shuffled	13.103	0.572	0.514	0.542

It can be observed that after shuffling the input, the model experiences a reduction in perplexity and an increase in BLEU scores, indicating a significant performance improvement, either in fluency or diversity. Moreover, the finetuned model exhibits better performance compared to the scratch-trained model, especially on the diversity metric. This is reasonable, as:

1. During pretraining the model has already learned general language patterns and structures. More knowledge has been stored in the parameters of the model, thus facilitating a faster adaptation to specific tasks. The pre-trained model may see data outside the training set, resulting in better diversity during inference.
2. Fine-tuning typically starts with a model pre-trained on extensive data, rather than initializing parameters randomly. This initial parameter setting aids in preventing the model from getting stuck in local optima, making it more likely to converge to better solutions.

3. Decoding strategies & Temperature

In this section, I will compare the performance of 4 different decoding strategies: random/top-p (p=0.9) combined with temperature 1.0/0.7. The four strategies are applied to the scratch/finetune models (trained by shuffled input). The results are as follows:

Model	Metric	Random-1.0	Random-0.7	Top-p-1.0	Top-p-0.7
Scratch	Forward-BLEU	0.570	0.815	0.693	**0.878**
	Backward-BLEU	**0.505**	0.478	0.504	0.405
	Harmonic-BLEU	0.535	**0.602**	0.584	0.554
Finetune	Forward-BLEU	0.572	0.797	0.684	**0.865**
	Backward-BLEU	**0.514**	0.477	0.512	0.411
	Harmonic-BLEU	0.541	**0.597**	0.586	0.557

Note that for the scratch/finetune model, The decoding strategy and temperature for any given evaluation metric to reach its maximum is the same. The Random decoding strategy gives a better Backward-BLEU and Harmonic-BLEU, indicating better diversity in sentence generating, whereas the Top-p decoding strategy gives a better Forward-BLEU, which means better fluency. This is also explainable: With the random decoding strategy, all tokens, no matter their calculated probabilities, are possible to be selected, which results in more diverse and rich content but comparatively lower fluency. The top-p strategy restricts the tokens to be generated to a subset with a certain probability, which guarantees the fluency of the generated sentences but may lead to a lack of diversity.

Meanwhile, the temperature's effect is also significant. Temperature 0.7 gives a larger Forward-BLEU, and temperature 1.0 gives a larger Backward-BLEU. The Harmonic-BLEU's relative performance is affected by the decoding strategy. The reason is that the temperature parameter primarily controls the sharpness of the probability distribution. A larger temperature parameter corresponds to a smoother distribution, while a smaller temperature parameter amplifies differences between logits, resulting in a sharper distribution.

A 0.7 (smaller) temperature gives a tendency during generation to favor the selection of a small set of high-probability tokens, thereby enhancing the fluency of the generated content and thus the Forward-BLEU. Conversely, when the temperature is set to 1.0, the higher temperature parameter to an overall smoother distribution. Reduced differences in probabilities between different tokens allow tokens with lower logits to be relatively amplified in probability, resulting in a more diverse sampling.

4. Generation examples

I randomly picked 10 sentences from the different models (scratch/finetune) with different decoding strategies and temperatures. Note that, the sentences from all the models are in the same position, to make the comparison clearer. All grammar mistakes are marked by "*.....*". The results are as follows:

Scratch model

> Random decoding, temperature 1.0

```
A woman standing on a stone bench under the statue.
A green double decker bus traveling down the street.
Red, white and green *field* on the side of the road.
A kitten is *laying* down on top of a bench.
A small couple of young planes waiting to take off.
*Dog* standing next to a man in front of clothing a couple of *computer*.
A photo of a busy street with white red and white wings.
A dog carrying a pink bowl and a piece of food.
A city street at night with a red fire hydrant.
The toilet seat *in* the stall next to the wall.
```

5 grammar mistakes

> Random decoding, temperature 0.7

A woman standing on a stone bench with her baby.
A green double decker bus traveling down a street.
A woman eats food in front of a group of sheep.
A man is holding a hat and sitting on a bench.
A small plane is on the grassy hillside.
A bathroom with a toilet, sink and shower in it.
A man sitting on a motorcycle next to a side truck.
A dog with a pink shirt and brown lap in front of the door.
A city street at night with *a traffic* light lights.
A toilet in a restroom stall next to a sink.

1 grammar mistakes

> Top-p decoding, temperature 1.0

A woman standing on a stone bench under the statue.
A green double decker bus traveling down the street.
A giraffe standing by some trees in a fenced area.
A kitten is *laying* down on top of a bench.
A small couple of young planes waiting to take off.
A bathroom with a toilet, sink and shower in it.
A photo of a busy street with a red and white fire hydrant.
A dog carrying a pink bowl and a piece of food.
A city street at night with a red fire hydrant.
The toilet seat *in* the stall next to the wall.

2 grammar mistakes

> Top-p decoding, temperature 0.7

A woman standing on a stone bench with her baby.
A green double decker bus traveling down a street.
A woman in a green field with a group of sheep.
A man is holding a hat and sitting on a bench.
A small plane is on the runway at an airport.
A bathroom with a toilet, sink and shower in it.
A man sitting on a motorcycle next to a side truck.
A dog with a pink shirt and brown lap in front of a laptop computer.
A city street at night with *a traffic* light lights.
A toilet in a restroom with a blue lid open.

1 grammar mistakes

Finetune model

> Random decoding, temperature 1.0

```
A woman beside a bench next to a stone statue.
A green double decker bus traveling down a street.
A space shuttle in a building kept on the end of a field.
A full of three men riding a motorcycle by a monument next to shrubbery.
A lone man on a motorcycle next to a car.
*Dog* standing next to a man on a sidewalk between a curb of *a traffic*.
A toilet with a small seat over *it's* wall and debris on it.
A dog carrying a horse tied to a car and carriage.
A city street filled with traffic with lots of traffic.
A toddler boy is kneeling down an old road with a dog.
```

3 grammar mistakes

> Random decoding, temperature 0.7

```
A woman standing next to a *stove top* oven in front of a chain.
A green double decker bus traveling down a street.
A woman eats food with her tongue out of the toilet.
A man on his motorcycle is driving down a city street.
A small plane sitting on top of a grass covered field.
A man holding a skateboard while standing on a sidewalk.
A man sitting on a motorcycle next to a woman.
A dog carrying a leash tied to a car and carriage.
A man walking across a snowy street with a bus on it.
A toilet in a bathroom stall next to a wall.
```

1 grammar mistakes

> Top-p decoding, temperature 1.0

```
A woman standing by a stone bench with *feet* in front of it.
A green double decker bus traveling down a street.
A woman and woman sitting in a brown leather jacket.
A man on his motorcycle riding next to traffic lights.
A small crowd of people next to a group of people.
A bathroom with a toilet, sink and shower in it.
A photo of a street and traffic lights, a street and a bus stop.
A dog carrying a horse tied to a car and carriage.
A city street filled with traffic with lots of traffic.
The toilet *in* a restroom stall next to the wall.
```

2 grammar mistakes

> Top-p decoding, temperature 0.7

```
A woman standing next to a *stove top* oven in front of a stove.
A man wearing a helmet sitting on a motorcycle and *woman*.
A woman and woman sitting on a bench reading a book.
A man on a motorcycle riding down a street in front of a crowd.
A small plane sitting on top of a grass covered field.
A man holding a skateboard while standing on a sidewalk.
A man sitting on a motorcycle next to a woman.
A dog with a leash tied to a car and carriage.
A man walking across a street with a fire hydrant.
A toilet in a bathroom with a blue lid up.
```

2 grammar mistakes

Basically, more grammar mistakes occur in Random decoding and high-temperature models. This aligns with the low Forward-BLEU scores of these models. Meanwhile, these models and decoding strategies give out more diverse sentences, as indicated by the high Backward-BLEU scores.

Though after finetuning, the perplexity of the model dropped a lot, the BLEU score doesn't make any significant improvement. After examining the sentences generated by both models, I subjectively agree that the finetuned model did not achieve better performance in sentence making. This indicates that compared with perplexity, the BLEU score is a more reliable metric for evaluating the performance of a language model from the human perspective.

5. Final results

I finally chose the finetuned model with Random decoding and temperature 0.7 as my final model. Other hyperparameters are set as default. The metrics are as follows:

Perplexity	Forward-BLEU	Backward-BLEU	Harmonic-BLEU
13.103	0.797	0.477	0.597

The output of the final model is stored in `outputs.txt`.

6. Analysis of Transformer decoder

1. comparison between Transformer and RNN

Time Complexity:

Transformer:

Say the sequence length is n, the hidden dimension is d, the number of heads in multi-head attention is h, then the transformer block with multi-head attention requires three steps:

1. Linear projection of the input sequence from $n \times d$ to $n \times \frac{d}{h}$, which takes $O(nd^2)$ time.
2. Perform scaled dot-product attention on each head, which is a matric calculation with sizes $n \times \frac{d}{h}$ and $\frac{d}{h} \times n$ which takes $O(n^2 d)$ time.
3. Linear projection of the output sequence from $n \times (\frac{d}{h} h)$ (concatenated) to $n \times d$, which takes $O(nd^2)$ time.

So the total time complexity is $O(nd^2 + n^2 d)$

RNN

The RNN needs to compute the hidden state at each time step, which takes $O(d^2)$ time. So the total time complexity is $O(nd^2)$.

In comparison, the Transformer has a higher time complexity than RNN of $O(n^2 d)$, when the sequence length goes higher, the Transformer will be more time-consuming.

However, we have to note that, the RNN requires sequential completion of n sequence operations, whereas self-attention and convolutional operations can concurrently execute n sequence operations. This distinction arises because RNN relies on the output of the hidden layer from the previous time step, while other models solely depend on the input.

Space Complexity:

The RNN network needs to store the hidden state at each time step, which takes $O(nd)$ space. The Transformer needs to store the $O(nd)$ hidden states and the attention weights of $O(n^2)$, so the total space complexity is $O(n^2 + nd)$. Similarly, as the sequence length increases, the Transformer will take more space.

Performance:

The Transformer network usually has a better performance than the RNN network. An explanation is that the attention mechanism of the Transformer can capture long-term dependencies of the language sequence, allowing the model to focus on the logic within the whole sequence, while the RNN network can only capture short-term dependencies.
Meanwhile, the RNN network has a longer gradient chain and is more likely to suffer from the gradient vanishing/exploding problem, which makes it difficult to train the model.

Positional Encoding:

Transformers do not inherently capture the sequential order of tokens in the input. To provide positional information, positional encodings are added to the input embeddings. These encodings are learned and injected into the model, allowing it to consider the order of tokens. RNNs naturally capture the sequential order of input tokens. Each token is processed one at a time, and the hidden state at each step incorporates information from the previous steps. Positional information is implicitly encoded in the sequential processing of the RNN.

2. Inference time complexity

1. `use_cache`

The `use_cache` parameter is used to speed up decoding. During inference, the key and value calculated in each step can be inherited in the next step by concatenating the new key/value behind.

2. inference time complexity

When decoding a single token l_t, the time complexity is composed of the following components:

1. For each Transformer block:

- Feedforward: complexity $O(4d^2)$.
- Other linear projections: the linear projections in all stages in the transformer block have a time complexity of no more than $O(d^2)$.
- Multi-head attention: the attention mechanism only attends to the t tokens before l_t, each head the calculation complexity is $O(t \times \frac{d}{n}), and adding the n heads the total time complexity is O(t d)$.

2. The decoder has B Transformer blocks, so the total time complexity is $O(B(d^2 + td))$.
3. Translating the output logits to words, we need a linear projection of $O(Vd)$, where V is the vocabulary size.

So the total time complexity is $O(B(d^2 + td) + Vd)$.

The total inference complexity can be calculated by summing the complexity of each token together. The result is $\sum_{t=0}^{T}(B(d^2 + td) + Vd) = O(Bd^2T + BdT^2 + VdT)$.

3. Attention & feedforward

The time complexity of the self-attention module and feedforward layer is $O(Bd^2T)$ and $O(BdT^2)$ respectively. So we can see that the self-attention module dominates the complexity when $T >> d$ and the feedforward layer dominates when $d >> T$.

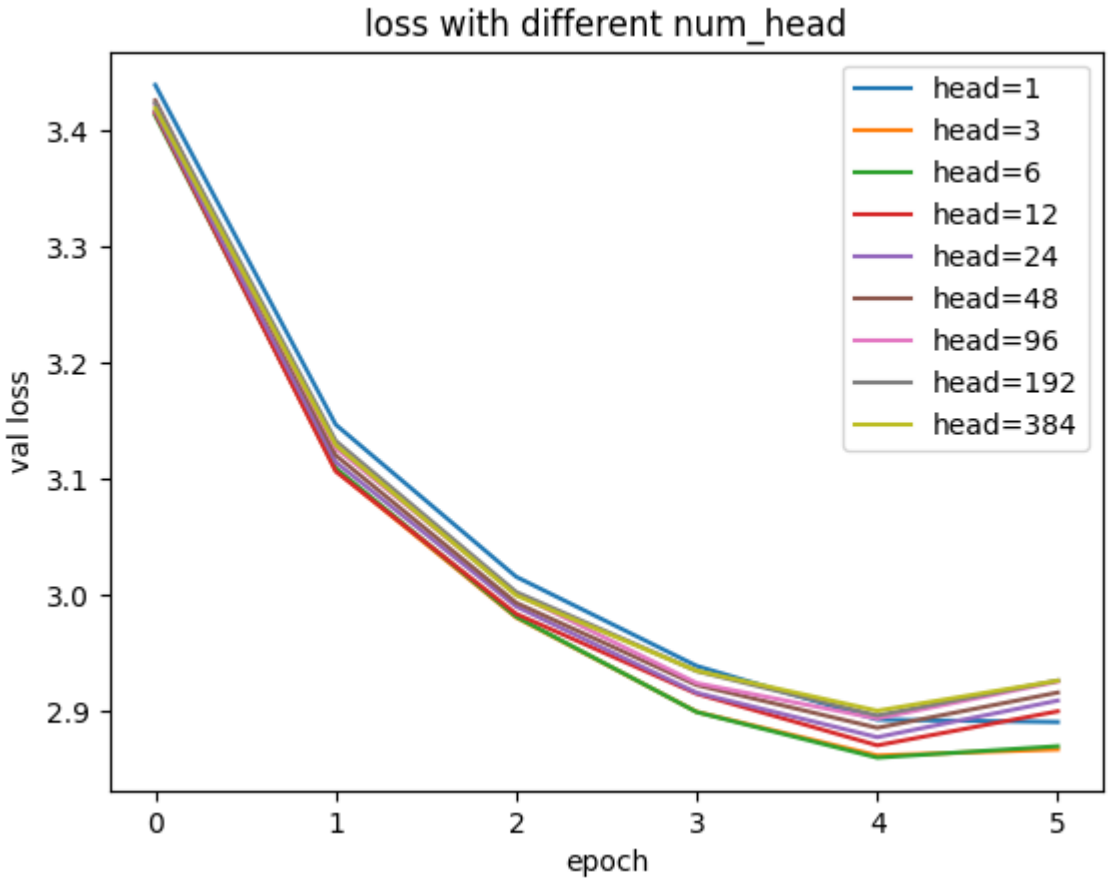
3. Influence of pretraining

From the experimental results, it is evident that training based on pre-trained models exhibits a faster convergence rate than training from scratch (6 epochs vs. 7 epochs). Meanwhile, it results in lower loss and perplexity after convergence. This phenomenon can be attributed to the fact that the model has acquired the ability to extract information and relationships from text during the pretraining process. Language knowledge has been stored in the parameters during pre-training and the model can leverage this knowledge to adapt to downstream tasks more quickly.

However, the BLEU scores of the two models are not significantly different, and the quality of the generation is relatively similar. From this perspective, the effect of pretraining does not reach my expectations. I believe that this is attributed to the limited size of our dataset and the simplicity of the task. The finetuning process can itself learn enough knowledge from the dataset to handle the task, thus weakening the pre-training's influence.

7. Effect of attention head numbers

In this section, I will compare the performance of the model with different numbers of attention heads. The results are as follows:



Head number	Perplexity	Forward-BLEU	Backward-BLEU	Harmonic-BLEU
1	**15.264**	0.805	0.477	0.599
3	15.368	**0.819**	0.472	0.599
6	15.326	0.816	0.475	0.601
12	15.462	0.815	**0.478**	**0.603**
48	15.667	0.819	0.474	0.600
96	15.764	0.818	0.472	0.599
192	15.803	0.818	0.474	0.600
384	15.878	0.819	0.473	0.600

> Note: the model using 1 head takes more epochs to converge than the others, so the perplexity's calculation isn't at the same training steps with the other models.

Experiments indicate that the most suitable head number should be around 6-12, where the validation loss and test perplexity reach the lowest and the BLEU scores are relatively high.

During the training process, different heads facilitate the capture of various features in the input sequence, accelerating the model's learning process and enhancing its feature-capturing capabilities within a given parameter count. However, if the number of heads is excessive, it may lead to a low-dimensional split vector with insufficient information. In such cases, individual heads can not attend to the correct vectors, resulting in a decrease in the model's learning effectiveness. Therefore, if the goal is to increase the number of attention heads, it is essential to simultaneously increase the hidden layer dimensions to prevent excessively low-dimensional splits.

Meanwhile, the influence of the head numbers on the model is rather limited. This may be attributed to the limited size of the training dataset. The small set of learnable features of the data is possible to capture using only a few attention heads. Therefore, the model's performance is not significantly affected by the number of attention heads.