StreamNet: A Combination Model For Answering Multi-task Reading Comprehension Problems

Lifan Wu, University of Science and Technology of China

Abstract

Today's reading comprehension problems have become more and more complex, from simple answer extraction problems (such as the SQuAD data set) to a series of problems that require strong analytical skills (such as the Drop data set). Existing natural language processing model structures often encounter great challenges in solving such problems. In this work, I build a scalable multi-output head model to classify and handle question-answer problems in natural language processing. My model achieved 0.40 em scores and 0.46 f1 score on the training dataset of Drop and 0.16 em score and 0.27 f1 score on the test dataset. It can be easily extended to more sophisticated models.

Introduction

There are many types of reading comprehension questions, such as giving a question and answering the corresponding answer. This paper is concerned with a reading comprehension problem of long texts. Which gives a piece of context, one or more questions, to get the answer to the question.

In the past, the answers to such questions were often limited to a part of the context, which greatly simplified the output form of the model. A typical example of this type of dataset is the SQuAD dataset (Dasigi et al., 2019), whose answers must be part of the context. With the advent of pre-trained models such as Bert(Devlin et al., 2019), existing models have achieved high scores on such problems, even surpassing human performance.

As a result, a series of more challenging qa datasets have emerged, the most representative of which is the Drop dataset (Dua et al., 2019). The answers to such datasets are often not part of the context, but require certain analytical derivations or mathematical operations to get the answer to the question. The area where the answer is located is no longer limited to one span of the text, but may exist in different spans of the text. An example of a problem is shown in the figure1:

Context: “Still searching for their first win .…… In the second quarter, Dallas increased its lead as QB Tony Romo completed a 4-yard TD pass to TE Jason Witten. The Bengals would end the half with kicker Shayne Graham getting a 41-yard and a 31-yard field goal. ……”
Question: “How many second quarter field goals did the Bengals get?”
Answer: 2

Figure 1: An example in the Drop dataset.

Simple span extraction models perform poorly on these types of problems, because the answers are likely not in the text, and such problems often require a certain amount of mathematical computing ability. How to effectively make the model handle math problems is also a huge challenge.

In this work, I designed a model that uses a pretrained Bert model as a preprocessing layer. According to the characteristics of the Drop data set, the problem is divided into the following categories by its answer type: span questions, arithmetic questions, count questions. The model selects the processing method according to the type of the question. And a separate output head is used to judge the problem type and the processing method that should be used to solve the problem.

Related Work

pre-trained model

Nowadays, the use of pre-training models has become a standard process in the field of natural language processing. Pre-training models are trained on larger data sets and have strong prior knowledge, which can greatly enhance the understanding of the model. Representatives of such models include the BERT model (Devlin et al., 2019) based on self-attention mechanism.

The model is pretrained with bidirectional Transformers to generate deep bidirectional linguistic representations.

After pre-training, it is only necessary to add an additional output layer for fine-tune to achieve state-of-the-art performance in a variety of downstream tasks. No task-specific structural modifications to BERT are required in this process.

Methods

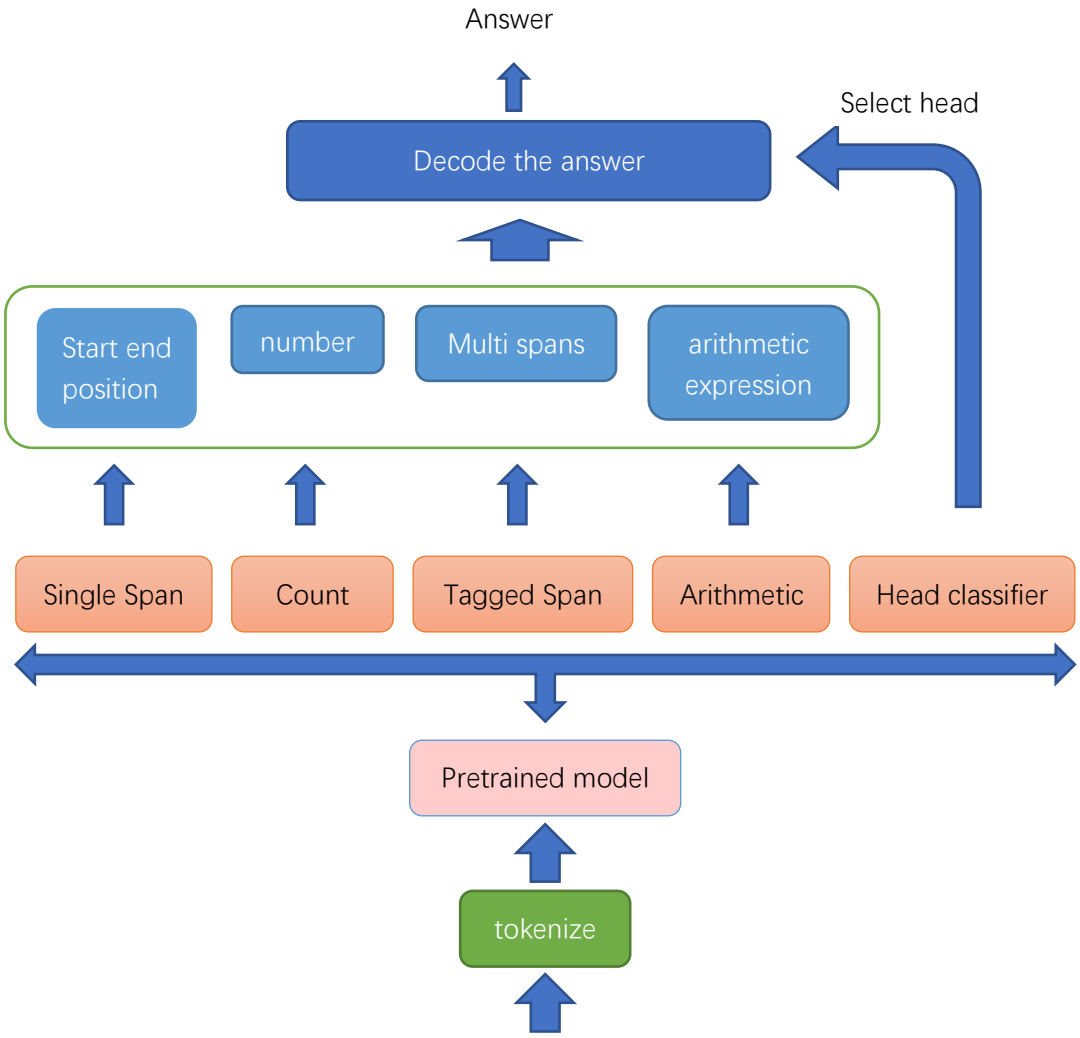


Figure 2: The framework of my model. It consists of a passage tokenizer, a Bert pretrained model and five output head.

Span questions

Span question's answers are contained in the passage. Such questions can be further subdivided into single-span questions and multi-span questions, where the answer to the former is a continuous part of the text, and the answer to the latter may be multiple parts of the text.

Passage:Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week 16 duel with the Houston Texans. Oakland would get the early lead in the first quarter as quarterback JaMarcus Russell completed a 20-yard touchdown pass to rookie wide receiver Chaz Schilens. The Texans would respond with fullback Vonta Leach getting a 1-yard touchdown run……"
question: "Who scored the first touchdown of the game?",
answer: "Chaz Schilens"

An example of span question

In the work of BERT (Devlin et al., 2019), they used an pretrained encoding layer to encode word tokens into fixed-length vectors: h = Encoder([q, c]). Send the hidden representations into two feedforward neural networks, and then go through a layer of softmax to get the probabilities of whether they are the beginning and end of the answer.

However, in the Drop dataset, many span-like problems involve multiple disconnected spans, and the above model is difficult to handle such a situation. Here I use a simple modified version of the output header. I feed the output of the pretrained model into a two-layer fully connected network, and then use the sigmoid function to calculate the probability that the corresponding token is part of the answer. I train my models by minimize the cross-entropy loss between predictions and ground-truth labels: (Ti equals 1 if the token is part of the answer, and 0 otherwise)

$$loss = -\sum T_i \times \log(\hat{T}_i)$$

In the decoding process, the problem is reduced to a binary classification problem. Decode the ones that are part of the answer to get the answer of the qestion. The special token neither participates in the calculation of loss nor in decoding.

Arithmetic questions

Arithmetic problems occupy most of the Drop dataset and are also the most difficult problems. Existing natural language processing models are difficult to understand mathematical operations, because the main components of the training set of such models are not mathematical problems. Moreover, existing tokenizers are difficult to efficiently encode digital information. In order to solve the arithmetic problems, I adopted the method in the Drop paper (Dua et al., 2019).

I simply assume that the answer to an arithmetic problem can be obtained by adding and subtracting the numbers in the text. I extract all the numbers in the text and only need to predict each number's sign to get the final answer.

In my implementation, the sign 0 means the number is not involved in the operation, 1

means the number should be added, and 2 means the number should be subtracted.

After getting the hidden representations from the pretrained model, I use the pooler-output of the hidden representations as the passage summary vector. I select the token representations where the numbers correspond to, concatenate the token representation and the summary vector of the passage. Then feed them into a two-layer fully connected network and a softmax layer to get the possibility of each sign.

In practice, I've found that 1 and 100 often appear in arithmetic expressions. This is because calculations involving percentages occur frequently, and adding or subtracting a value by one is also a common expression. Therefore, I also take these two numbers into the calculation as special numbers and predict their signs. The model is still trained by minimizing the cross-entropy loss between the predicted and actual signs.

When predicting, extract the numbers in the text, feed the text into the model and calculate the probability of the symbol of each number, select the symbol with the highest probability for addition and subtraction. The result of the operation is the predicted answer.

Count questions

However, if the answer cannot be obtained by adding and subtracting the numbers in the text, we need another solution. These kinds of math problems that cannot be answered from the numbers in the text are reduced to count questions.

An example of a count problem: There is a football, two table tennis balls, and three volleyballs in a basket. How many kinds of balls are there in total?

I feed the pooler output of the pretrained model into a two-layer feedforward network, then use a softmax layer to calculate the probability that the count value is 0, 1, 2, 3……I only consider the count value less than 10, and the part greater than 10 is skipped. The model is still trained by minimizing the cross-entropy loss between the predicted and actual numbers.

In the preprocessing stage, I extract the position and value of all numbers in the text, and try to get the answer according to the above methods. For the spans type problem, it is to find the position of the answer in the text, and for the arithmetic type problem, it is to traverse the operation combination of the numbers. If the answer is successfully obtained, the corresponding training data and test data are obtained. If the training data required by the corresponding output head cannot be generated, the data will be skipped directly.

Classification of Problems

When actually solving the problems, we need a way to choose between the various output layers. I set this problem as a classification problem separately, and use the pretrained model to judge the classification of the problem. If an answer belongs to one of the above categories, the corresponding output header can be used to predict the answer.

I designed a simple classification head consisting of a preprocessing model and a two-

layer feedforward neural network. The output of the head is a vector of length 4, which is clipped between 0 and 1 after passing through the sigmoid function. The output corresponds to the usage probability of each head.

In the actual training process, I use the above four output heads to try to generate an answer. If it can be generated, set the label of the corresponding head to 1. Obviously, the same problem may be solved by multiple output heads, for example, the problems that single span head can solve can also be solved by tagged spans head, some problems that arithmetic head can solve can also be solved by count head. The output label is in the format of 1 0 1 0. And there may be some problems that cannot be solved with the above output headers. This part of the data is skipped directly.

The head type classifier head uses mean square error to calculate the loss between the predicted value and the actual label.

The combination of models

The above five models (single span model, tagged span model, arithmetic model, count model, question classifier model) are trained separately, the pretrained models do not share parameters.

In the prediction stage of the model, we need to make a decision to select the appropriate output head. A possible selection method is to select the output head with the highest score.

But consider the following two factors: (1) Most of the span problems in the dataset are are single span problems, only a few involve multi span problems. (2) Most of the mathematical problems in the training set are solved using the arithmetic method, and only a few are solved using the count method.

Therefore, I designed a new decision-making method: first, according to the sum of the scores of the two span-related heads and the sum of the scores of the two math-related heads, determine the basic classification of the problem. If it is a span problem, first try to generate an answer through the single span head. If the answer length is greater than 7 or the predicted end position > start position, try again to predict the answer using the multi span output header. If this is a math problem, use the arithmetic head to predict the mathematical expression, if the mathematical expression cannot be predicted then use the count head to predict. The entire forecasting process is shown in Figure3:

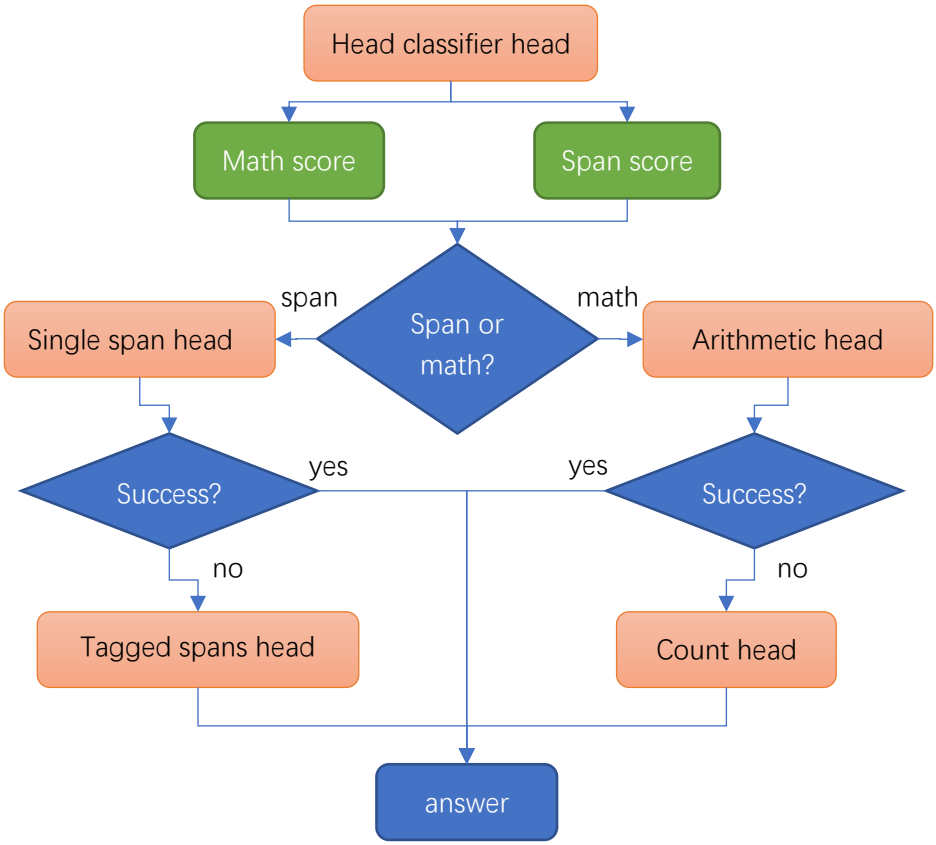


Figure3: the head choose strategy used for predicting the answer

I call the above neural network framework stream-net because the model has many branches merged like a small stream. And another reason is that the dataset name used in the project is called Drop.

Experimental Settings

I use the Bert-base-uncased model for pretraining, the hidden size of Bert-base-uncased is 768. The training rate is 1e-5,and I use the Adam optimizer to train the model. A relatively small learning rate requires more epochs of learning, I choose 30 epochs for each model, and the batch size is 16. The model is tested on the validation set at the end of each epoch. I trained the model for a total of 40 hours on a single rtx3090 graphics card.

Since the test set is small, there are many branches in the test phase, so I directly perform the test phase on the CPU.

I use the HuggingFace's Transformers library[4] to implement the Bert pretrained model. The text input is truncated to 512 tokens because positional embedding is limited to 512 tokens.

Results and Discussion

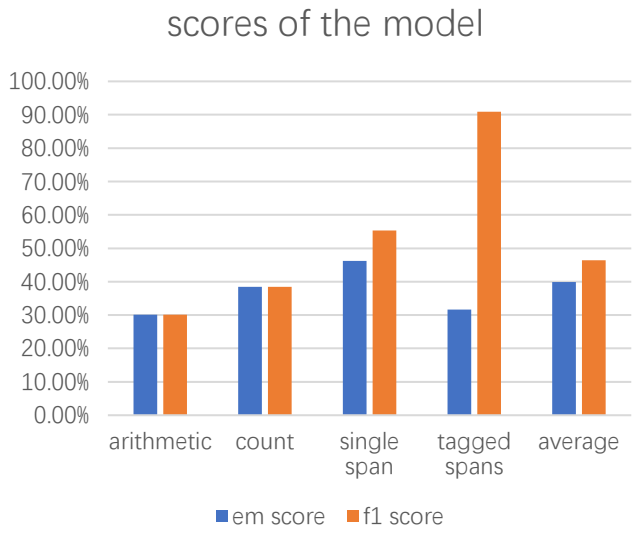
I encountered serious overfitting problems when training the model, the training curve of the model on the training set and test set is given in the appendix, as shown in Figure4.

It can be seen from the training curve that different heads require different epochs to achieve convergence. However, their loss on the training set generally keeps decreasing. The single span model requires more step to train and others get overfitted quickly. I think the reason is that other head designs are not reasonable and the training data is too small.

Evaluation

I evaluated the trained model on the training datasets and the test datasets. I used the em score (exact match score) and the f1 score to evaluate my model. Due to time reasons, I only tested part of the data of the training set. Here is the results:

	arithmetic	count	single span	tagged spans	average
em score	30.10%	38.50%	46.20%	31.67%	39.88%
f1 score	30.10%	38.50%	55.31%	90.90%	46.39%
num	402	2426	1508	275	4611



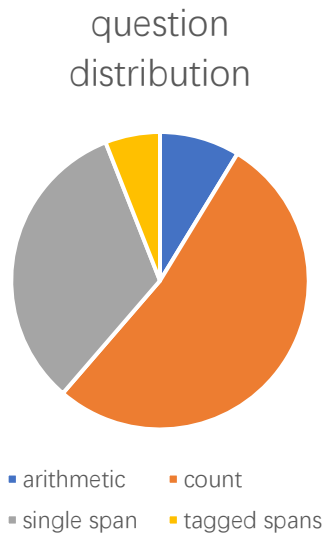
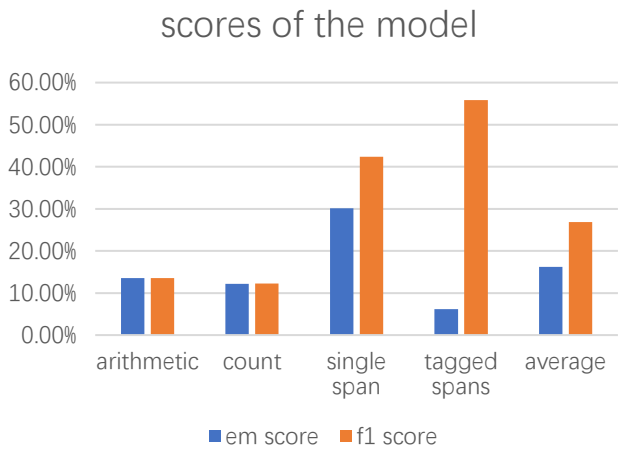


Table1:The em score and f1 score and the number of each head on the training dataset:

	arithmetic	count	single span	tagged spans	average
em score	13.58%	12.20%	30.10%	6.18%	16.22%

f1 score	13.58%	12.27%	42.35%	55.87%	26.83%
num	324	1868	1000	550	3742



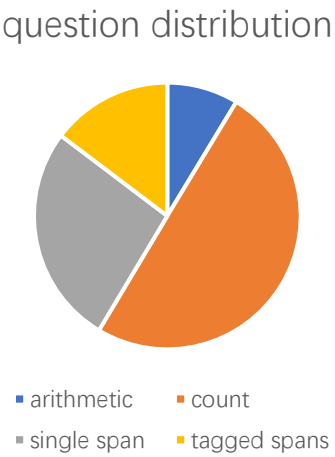


Table2:The em score and f1 score and the number of each head on the test dataset:

Error analysis

As can be seen from the model scoring table above, the single span model performs the best while the arithmetic model performs the worst. This shows that my model still has many shortcomings in solving mathematical problems.

We analyze several common errors of the model:

Context: ……Detroit's Calvin Johnson caught a 1-yard pass in the third quarter. ……

Question: "How many yards was the shortest touchdown scoring play?"

Answer: "number": "1"

The head type model choose: arithmetic

Predict result: 31

An example of a model misjudgment

We can see that it is more reasonable to treat this problem as a span problem rather than a math problem. However, during training, I did not use the solution method of span class for all math problems, which caused the model to be more inclined to use the arithmetic output header and gave a wrong answer.

Context: "In South America , the Portuguese conquered from Spain most of the Rio Negro valley, and repelled a Spanish attack on Mato Grosso . Between September 1762 and April 1763, Spanish forces led by don Pedro Antonio de Cevallos, ……Hispano-Portuguese war of 1763-1777. ……

Question: "What was the latter 1/10 of a decade that Spanish forces led a campaign against the Portuguese in Uruguay and South Brazil?",

Answer: "spans": "1763"

The head type model choose: tagged_spans

Predict result: “1763 1763”

An example of a model misjudgment

Here the model uses the tagged spans class output header, but outputs two identical correct answers, because there are two identical “correct” answers in the text. I found that the tagged spans class output head tends to output more possible answer parts, resulting in its relatively high f1 score but low em score.

Context: "……It was 17-7. Then the Texans made a 17 play drive resulting in a 2-yard Ben Tate touchdown to make it 24-7. The Texans would get a field goal to make it 27-7, and that would end the game.",

Question: "How many points were scored in total?"

Answer: "number": "31"

The head type model choose: count

Predict: 3

An example of a model misjudgment

The arithmetic class head should be used here, but the model cannot use the arithmetic head to generate a valid expression, only the count head can be used, and the head cannot output a value greater than 10, resulting in an error.

Conclusion

In this project, I simplified the complex and diverse natural language understanding problems into sub-problems by classifying the problems, and constructed a set of framework for solving natural language processing problems. My method is to deal with complex and diverse natural language processing problems by using a mixture model, and the follow-up experimental results also show the effectiveness of this method. In future work, the entire model can be further subdivided and trained distributedly, fusing multiple specialized models into a unified framework. The output head is the key to enhancing the reading comprehension ability of the model. In the future, I will explore more sophisticated output models for follow-up research, especially the arithmetic and count output headers which did not perform well on the Drop datasets.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL.

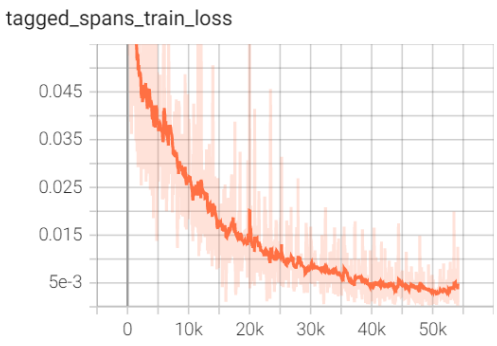
Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Proc. of NAACL.

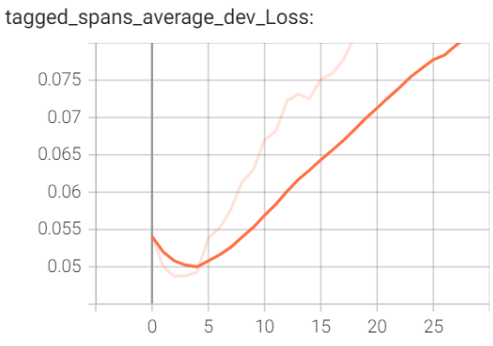
Pradeep Dasigi, Nelson F. Liu, Ana Marasovic,´ Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In Proc. of EMNLPIJCNLP.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson,and Jonathan Berant. 2020. A Simple and Effective Model for Answering Multi-span Questions

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771.

Appendix





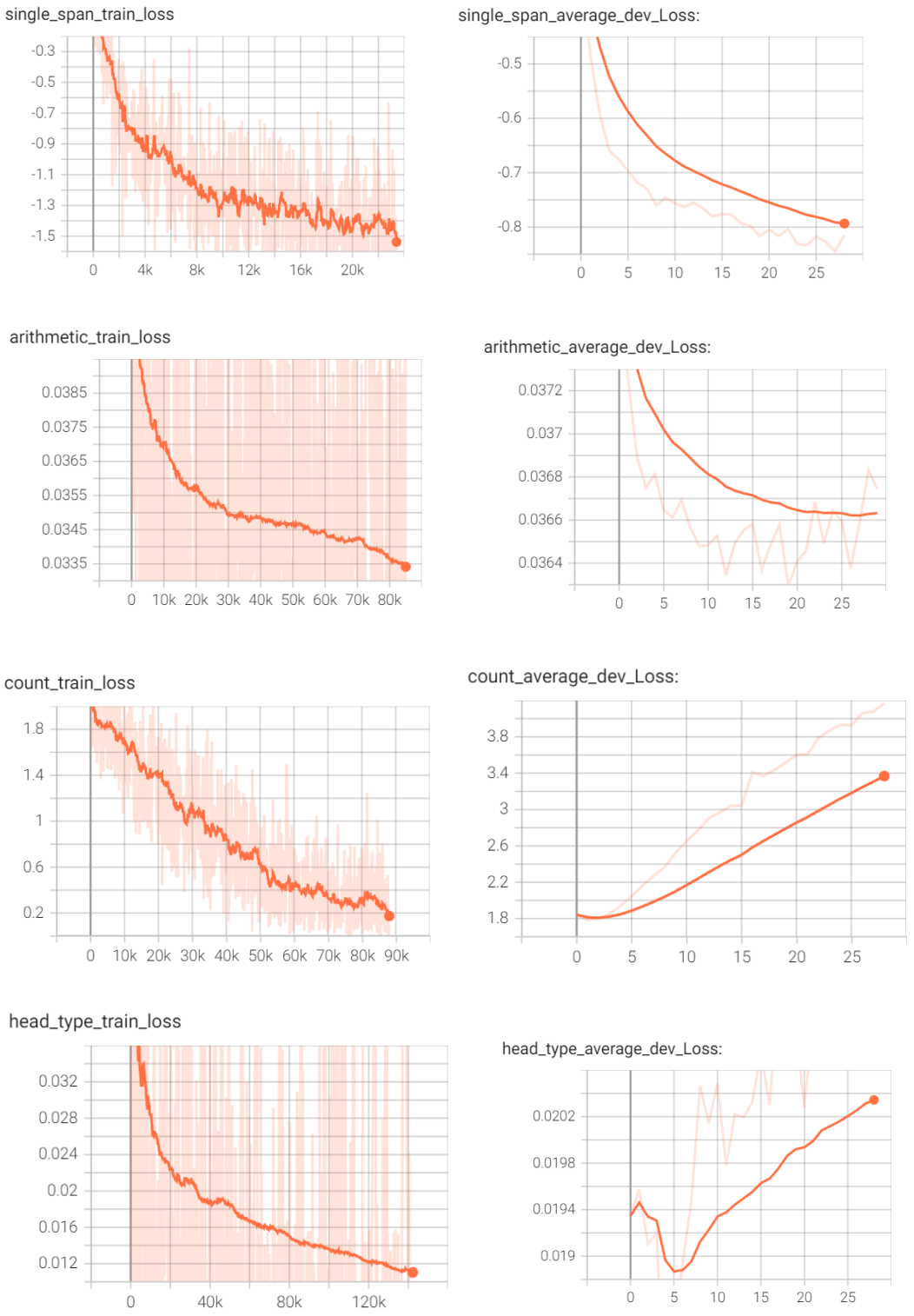


Figure4: the loss of each head on the training datasets and test datasets during training process, the horizontal axis represents the training steps, the vertical axis represents the loss.