COMP 551 Final Project

Group 59
Yuchen Chen, Yun-fei Cheng and Jiayao Zhang
McGill University COMP 551
yuchen.chen@mail.mcgill.ca,
yun-fei.cheng@mail.mcgill.ca,
jiayao.zhang@mail.mcgill.ca

Abstract

It has become common knowledge for convolutional neural networks (CNN) to perform well on image data, so, intuitively, CNNs should be used for word vectorization of logographic characters such as those from Chinese. The paper introduces three CNN-based innovations to obtain better Chinese word embeddings for use in natural language processing (NLP) tasks. Our ablation studies will focus on the first two innovations: "Using historical scripts to enrich the pictographic evidence in characters" and using "CNN structures tailored to Chinese character image processing". We will verify that it does indeed produce better results. We then generalize and experiment whether the findings in this paper apply to another logographic but subtly different language- Japanese.

1 Introduction

Chinese characters are logograms which were developed from real-life images and thus encode abundant information of their meanings. Inspired by the assumption that logographic information should help semantic modeling, many researchers have tested Chinese-embeddings using CNN. To many's surprises, however, it was found that this approach not only did not improve the language model consistently but in fact harmed some language modeling tasks.

The authors hypothesized that this was due to the loss of information from years of textual simplification of Chinese characters. They enriched the pictographic information using historical scripts, contemporary scripts, and scripts of different writing styles. Their model improved common NLP tasks including tagging and classification.

Figure 1: Evolution of Chinese characters. The images are copied from Wikipedia.

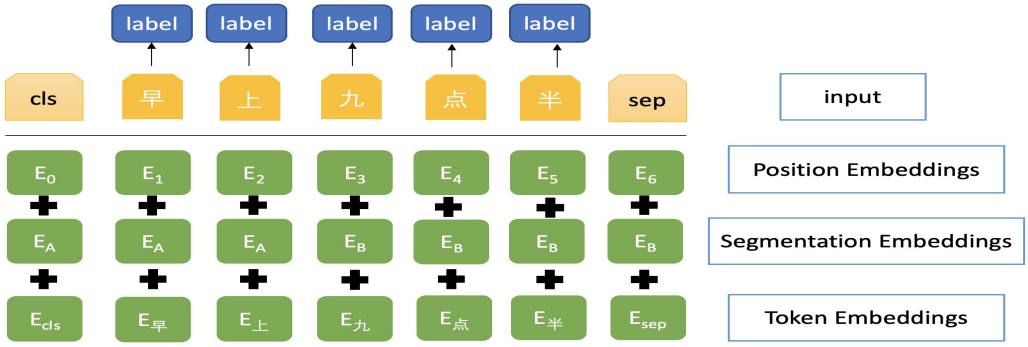
Our ablation studies aim to replicate their results, narrowing our focus on the language model tasks of Name Entity Recognition (NER), Part of Speech (POS) tagging, and Chinese Word Segmentation (CWS). We will also place our spotlight on the aspect that various historical fonts, which are assumed to preserve information, are vital.

2 Related Work

BERT[1] stands for Bidirectional Encoder Representations from Transformers which is robust for a wide range of NLP tasks. It outperforms unidirectional language models by alleviating their constraints. Particularly, during the prextraining process, it extends the choice of possible architectures by allowing every token to attend to previous and next tokens in the self-attention layers of the Transformer.

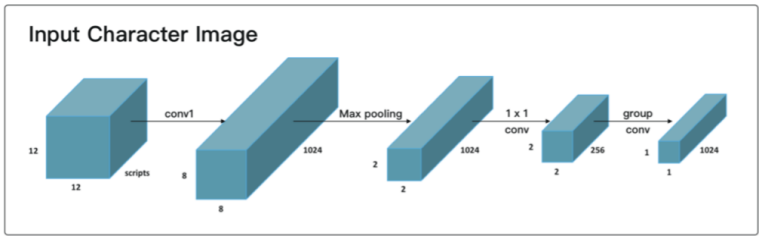
The mechanism of BERT is as follows. BERT constructs the input representation of each token by summing the corresponding token embeddings, segment embeddings, and position embeddings. During the pretraining process, rather than doing a shallow concatenation of two independently trained left-to-right and right-to-left models, it uses masked word prediction and next sentence prediction. During the fine-tuning process, it uses the self-attention mechanism in the Transformer.

Figure 2: The input of BERT transformer is the sum of Position embeddings, Segmentation embeddings, and Token embeddings.



The paper we aim to perform ablation studies proposed a new method for Chinese NLP tasks named GLYCE[2]. GLYCE stands for GLYph-vectors for Chinese character representations. Unlike typical ImageNet[3] images with the size of 800×600, Chinese character logos are significantly smaller. The CNN structures designed for ImageNet combined with glyph-vector did not perform well on Chinese characters in previous research[4][5]; it sometimes worsened the performance[6]. By treating Chinese characters as smaller images, the authors suggest a new CNN structure that can combine with glyph-vectors to perform well.

Figure 3: Image of proposed CNN in original paper[2].



3 Data Setup

The nature of the experiment requires 2 sets of data. The first data is the various Chinese scripts. We used the prepared 7 Chinese scripts: cjk, regular, seal, cursive, bronzeware, clerical, and tablet scripts.

Within each script, there are 1 to 8 styles for a total of 28 different writing styles, each stored as a .ttf file. The number of styles used to train the model is a hyperparameter, with a constraint of using up to 10 styles only out of the given 28.

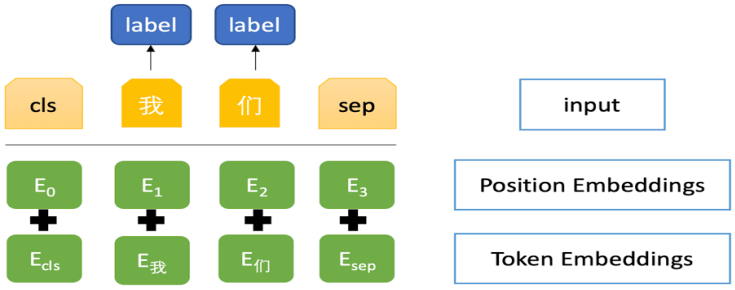
The second data is task-specific labeled data for NER, POS, and CWS. Each task has data collected from various sources, ranging from university and research institutes to web-scraped data. For example, CWS has data collected from Beijing University, Microsoft Research China, Weibo (a popular Chinese blogging website), etc. For each source, data is pre-split into 3 files -train, dev, and test. The size of these data range anywhere from a few MB to 30MB per source.

The embedding layers are trained on arbitrarily chosen N writing styles. Since it is computationally not possible for us to run our models on all data, training and testing are only done on a single source at a time.

4 Architecture of Model

1. **BERT Layer**
BERT is designed to pretrain deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right contexts in all layers. The authors used the Chinese BERT model as a layer in the proposed model.
Chinese BERT model is a large scale pretrained model and was fine-tuned to produce better results in the Chinese-only data set. In the Sequence Labeling Tasks, each input sequence S (such as the name entities) is concatenated with a prefix `[CLS]` and a suffix `[SEP]`.

Figure 4: The input of BERT transformer is the sum of Position embeddings and Token embeddings.



2. **Glyph Embedding Layer**
The authors propose a new CNN structure, called Tianzige-CNN, for Chinese NLP tasks by training CNN with single Chinese characters with different scripts and writing styles. In the first convolutional layer, font_channels is the input channel. Therefore by adjusting font_channels, features from different scripts and writing styles used in Chinese are extracted. The max-pooling layer resizes images of the Chinese character to 2×2 and extracts features efficiently. Remarkably, the authors use group convolutions [7][8] instead of the conventional convolutional layer to reduce the likelihood of overfitting.

3. **Joint Glyce-BERT Layer**
This is the proposed model that achieved the best performance in Tagging tasks. Instead of feeding Position embeddings and Token embeddings to BERT directly, Position embeddings are first added to glyph embeddings and then combined with BERT to obtain the full Glyce representation.

4. **Task-specific Output Layer**
The task-specific output layer is used to encode contextual-aware glyph representations. The outputs of Glyce-BERT are passed to the task-specific output layer. The outputs of the task-specific output layer are passed to prediction layer and obtain the prediction.

5 Limitations

The computational resource available to us was Google Colab's K80 GPU for up to approximately 12 hours of consecutive use, largely short of the author's suggested NVIDIA TITAN Xp with 12G RAM.

In reality, however, we could not afford to run the entire 12 hours because it will lead to Google blacklisting us for half a day. Besides, our training and evaluations were often terminated suddenly; this is something we could not control and lead to results that are difficult to compare because each was trained for unequal durations.

6 Ablation Studies and Results

6.1 Chinese Word Segmentation (CWS)

Results on CWS task are mentioned first because configurations used for achieving the reported precision, recall, and F1 score were provided. Due to our computational limitations, however, not a single model was run to completion at the default 20 epoch. The Cityu dataset was selected since it provided the largest numerical difference between BERT and Glyce+BERT across Precision, Recall, and F1.

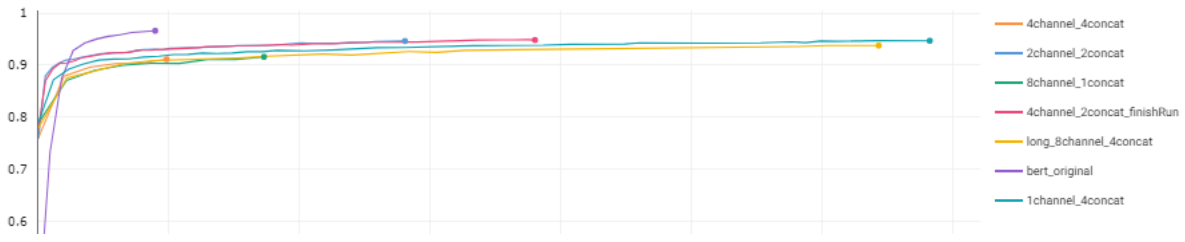
The most exemplary model is the one run with default configuration for the longest time of 11 hours. Although the results for this model are short of the reported results, the model's loss was still decreasing and thus we imagine that it is possible to reach the reported results after longer training. The loss when it terminated was 0.02, so there is still some room for improvement.

The different configurations and their best results are summarized in Table 1. The F1 for each configuration as it was trained is shown in Figure 5.

Table 1: Results are not directly comparable due to different training duration. Reported results are shown in the first 2 rows (highlighted); "Original" mean that original configurations were used. Channel is the hyperparameter font_channel and Cocat is the hyperparameter num_font_concat.

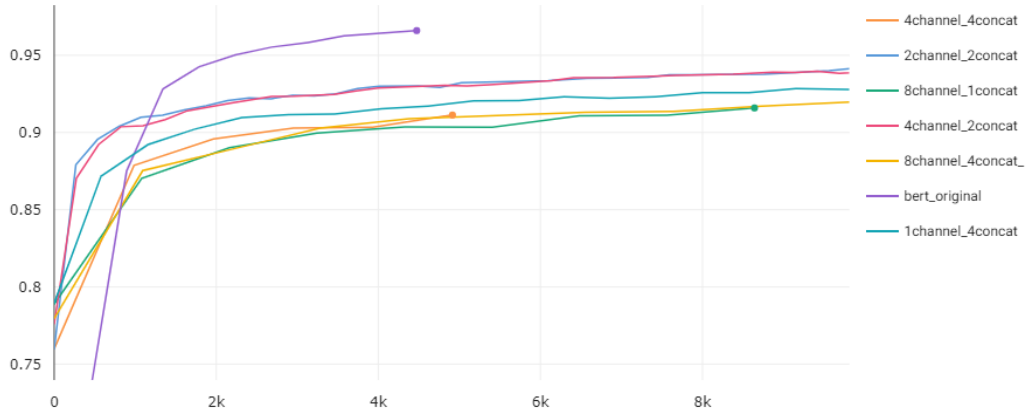
Model	Channel	Concat	Loss	Precision	Recall	F1
Glyce+BERT (reported results)	8	4	NA	97.9	96.8	97.9
BERT (reported results)	NA	NA	NA	97.5	97.7	97.6
Original (not graphed)	8	4	0.0253	95.53	95.64	95.59
Original (graphed)	8	4	0.0341	94.16	93.36	93.76
Glyce+BERT	8	1	0.0454	92.09	91.09	91.59
Glyce+BERT	4	4	0.0481	91.36	90.9	91.13
Glyce+BERT	4	2	0.0416	94.58	95.09	94.83
Glyce+BERT (not graphed)	4	1	0.0394	93.03	92.28	92.65
Glyce+BERT	2	2	0.0299	94.66	94.61	94.63
Glyce+BERT (not graphed)	2	1	0.0431	92.33	92.06	92.2
Glyce+BERT	1	4	0.0293	94.55	94.81	94.68
BERT	NA	NA	0.0275	96.35	96.85	96.6

Figure 5: F1 vs Steps. full training. Notice that they all have varying training duration resulting from our unstable training environment.Train step is on X-axis and F1 score on Y-axis.



As seen in Table 1, no substantial improvement or deterioration was observed by changing the hyperparameters; they all performed above 90%. The concern here is that results were all over the place, and no statement such as "increasing font_channels tend to improve results" can be declared.

Figure 6: F1 vs Steps. Zoomed in to initial 9000 steps. Train step is on X-axis and F1 score on Y-axis.

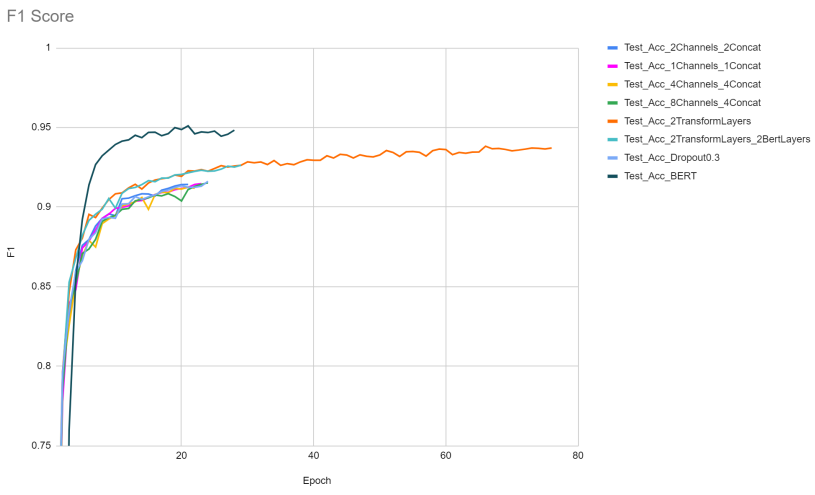


The results at the beginning of the training and evaluation phase though, were drastically different as seen in Figure 6. Depending on the configuration, the model quickly reaches a certain value, and from there improved extremely slowly. Generally, a simpler model (i.e. the model with smaller *font_channels* and *num_font_concat*) achieved higher metrics much faster than more complex models and improved from thereon. The prime example is standalone BERT without Glyce added on top; it achieved very high results in the shortest time. However, we could not determine the results in the long-run. Perhaps simpler models reach a higher F1 faster but plateaus quickly, and a complex model starts slow but is able to continue improving and ultimately will reach a higher result.

6.2 Part Of Speech tagging (POS)

In contrast to our assumption of "using more fonts leads to better performance", the results of altering *font_channels* and *num_font_concat* demonstrate that the model performance does not necessarily improve with a larger number of fonts. Experiment results for POS are shown in Figure 7. A multitude of hyperparameter combinations was tested, but they all lead to similar results with little improvement. Notice again the different training durations.

Figure 7: The F1 of different hyperparameters on the POS task.



Despite the difficulty finding the extent of *font_channels* and *num_font_concat*'s influence on the final result, *glyph_ratio* was found to positively affect the results by increasing the ratio.

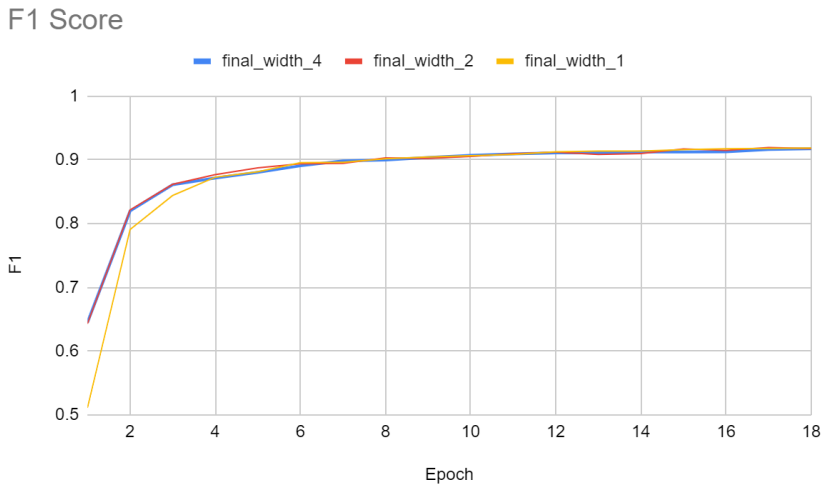
In addition to hyperparameter tuning, the CNN architecture was modified to perform the POS task. We found that increasing the hidden layer number of transformers in the task-specific output layer from 1 to 2 would noticeably improve the model performance. However, further increasing from 2 to 4 would significantly decrease accuracy. We assumed we could optimize the model by adjusting how much it depended on the scripts, but high dependency possibly resulted in high sensitivity to the bias caused by the scripts. The results for POS task are summarized in Table 2.

Table 2: Results of POS. Results are not directly comparable due to different training duration.

Model	Hidden Layer	Channel	Concat	Loss	Precision	Recall	F1
Glyce+BERT (reported results)	NA	NA	NA	NA	95.56	95.26	95.41
BERT (reported results)	NA	NA	NA	NA	94.91	94.63	94.77
Glyce+BERT	2	2	2	0.0518	92.8	92.5	92.65
BERT	NA	NA	NA	0.0603	94.5	94.71	94.61
Glyce+BERT	1	1	1	0.0689	90.2	89.24	89.72
Glyce+BERT	1	2	2	0.0594	91.19	91.4	91.29
Glyce+BERT	1	4	4	0.0611	90.83	90.79	90.81
Glyce+BERT	1	8	4	0.639	90.74	90.36	90.55

Additionally, we justified the architecture of Tianzige-CNN by evaluating on POS task. "Tianzige" signifies a 2×2 format. Experimental results show that it performed better than 4×4 or more, but only by a small margin. We assume this is due to the input fonts having a small dimension.

Figure 8: The F1 of different CNN architectures on the POS task.



6.3 Named Entity Recognition (NER)

We used the resume dataset for the NER task. The result in the paper is 96.54 for F1 by using Glyce+BERT model. The authors proved that Glyce can outperform the BERT model by improving 0.76 on F1. The authors do not provide the configurations used to obtain the results. We experimented on *learning rate*, *font_channels* and *num_font_concat* to replicate the results.

With a fixed number of epoch (epoch = 12) and learning rate (lr = 3e-5), the change of F1 does not exceed $\pm$ 1.

The different configurations and results are listed in Table 3. Graphs of experiments are shown in Figures 11, and 13. We use the F1 metric to score and compare our results.

Table 3: Configuration and results for NER.

Model	Channel	Concat	Loss	Precision	Recall	F1
Glyce+BERT (reported results)	NA	NA	NA	96.12	95.45	95.78
BERT (reported results)	NA	NA	NA	96.62	96.48	95.82
Glyce+BERT	8	4	0.0013	88.31	88.32	87.92
Glyce+BERT	8	2	0.0015	88.53	88.32	87.91
Glyce+BERT	4	4	0.0049	87.75	88.07	87.91
Glyce+BERT	4	2	0.0051	89.49	88.38	88.93
Glyce+BERT	2	2	0.0090	89.71	88.50	88.51

We hypothesized that we did not run the experiment long enough to replicate the result of Glyce+BERT model mentioned in the paper. To test this, we picked the configuration of Glyce+BERT model that performed best, with *font_channels*=4, *num_font_concat*=2, and ran 100 epochs on it. The loss stayed at 0.0045 and F1 value started to decrease. The finding contradicts our claim. Longer training does not yield better results.

A point to note is that the loss of NER is the lowest among the three tagging tasks despite the dataset for NER being the smallest. We were not able to decrease loss further and boost performance by fine-tuning the hyperparameters. We suppose that the performance of NER may be correlated to *glyph-ratio* and *glyph-decay*, but it will remain as future work due to time constraints for this project.

7 Generalization

So far the discussion has been centered around Chinese as the logographic language of choice. However, the author suggests as part of their conclusion that their approach is general enough to apply to other logographic languages, so we chose to apply the same experimental procedure to Japanese POS tagging.

Japanese was selected not only due to its logographic nature but because it has a complex history of script transformation. The current Japanese language is composed of Hiragana(平仮名), Katakana(片仮名), and Kanji(漢字), where Hiragana and Katakana both have one corresponding pronunciation/sound per character, and Kanji has the Kun-yomi(訓読み) and On-yomi(音読み), which are respectively Japanese and Chinese-based pronunciations, used differently depending on the context[9].

The three different sets of characters are used in a mixture in today's Japanese, but it all originated from the Kanji, which was imported from China. Hiragana is a cursive transformation of the Kanji as seen in Figure 9. Katakana, on the other hand, is a simplification of the Kanji as seen in Figure 10. The origin Kanji and corresponding Hiragana and Katakana are shown for the sounds "a", "i", "u", "e", "o" from the top respectively.

Figure 9: Hiragana transformation.

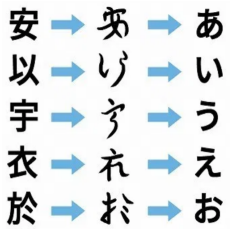
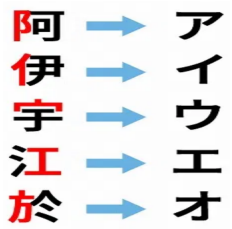


Figure 10: Katakana transformation.



We observe that similar to Chinese, Japanese has simplified greatly by the creation of Hiragana and Katakana. However, that is only the case if Kanji was no longer used. The mixed utilization of all

three sets of characters in today's Japanese is believed to preserve semantic information if not add more complexity.

Again, historical scripts were chosen for Japanese font training. They include tensho, reisho, gyousho, kaisho, and sousyo, and were obtained online[10] as .ttf files.

Text data was obtained from a website[11] which stores copyright-expired novels. The novels "Kokoro" by Souseki Natsume and "Ningen Shikkaku" by Osamu Dazai were picked due to their high reputation and orthodox use of the Japanese language. Then Janome [12] was used to process the text data and create POS tags for each tokenized word. BERT and Glyce+BERT were used to perform the POS task. A sample segmentation and its POS tags is shown below:

'筆', 'を', '執っ', 'て', 'も', '心持', 'は', '同じ', '事', 'で', 'ある'
'名詞', '助詞', '動詞', '助詞', '助詞', '名詞', '助詞', '連体詞', '名詞', '助動詞', '助動詞'

The main difference between Chinese and Japanese is the way in which words are segmented. In Chinese, character-level-segmentation is used in Tagging tasks, but there may be multiple characters in Japanese segmentation as shown above.

We suspect this to be one reason why POS tagging on Japanese failed. The best results were obtained when the loss was still 0.2954. Precision, Recall, and F1 were respectively 6.22, 5.88, and 4.85 when the loss was 0.2954. Further training only worsened the results. Another suspect for poor performance is the multilingual BERT model. Unlike the Chinese BERT model, no tuning was performed. Modifying the code almost completely was an extraordinary challenge for us, one that could not have been done given the time span of this project. Improving the result further shall remain as future work.

8 Conclusion and Future Work

Our experimental environment greatly reduced the ability to verify the author's results. The best we may do is extrapolate from the graphs, but doing so does not allow us to ascertain the superiority of Glyce+BERT over simple BERT. Hyperparameter tuning of *font_channel* and *num_font_concat* as well as some other parameters did not significantly improve Tagging task results as we hypothesized; *glyph_ratio* did improve the results. As for the CNN structure, we were able to verify that Tianzige-CNN performs the best but by an extremely small margin; adding a single transformer layer also proved beneficial.

For Japanese generalization, we were able to generalize, apply the same experiment on Japanese, and get the model running, but the results were too poor to consider it a success.

9 Statement of Contribution

Yuchen Chen: Ablation study on POS; research on the paper; write the report.

Yunfei Cheng: Ablation study on CWS; collect and process Japanese dataset; write the report.

Jiayao Zhang: Ablation study on NER; set up codes and configurations; write the report.

10 Appendix

Figure 11: Impact of font_channels on F1 (F1 vs steps)

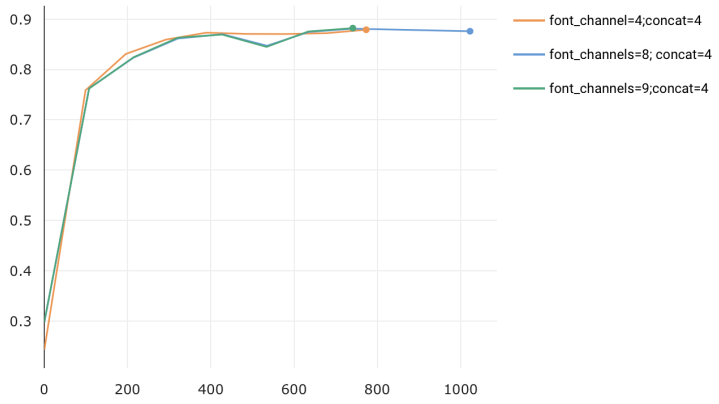


Figure 12: Impact of num_font_concat and maxpool sizes on F1 (F1 vs steps)

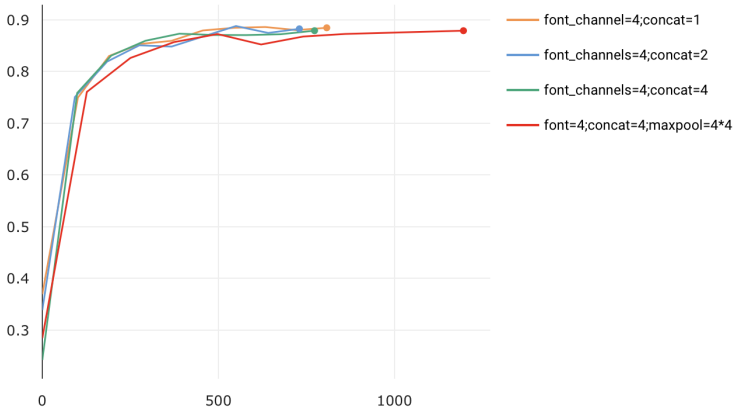
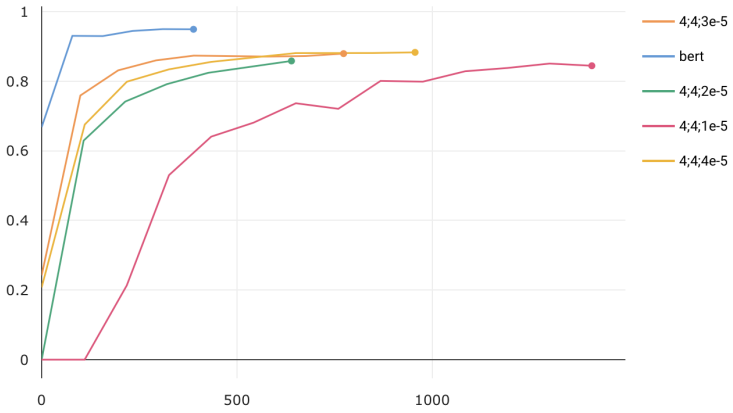


Figure 13: Impact of learning rates on F1 (F1 vs steps)



Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[2] Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations, 2019.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. Learning character-level compositionality with visual features, 2017.

[5] Xiang Zhang and Yann LeCun. Which encoding is the best for text classification in chinese, english, japanese and korean?, 2017.

[6] Falcon Z. Dai and Zheng Cai. Glyph-aware embedding of chinese characters, 2017.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[8] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions for deep neural networks. *CoRR*, abs/1707.02725, 2017.

[9] Nabeel Al Jahan. Hiragana and katakana. 2017.

[10] Hakusyu fonts. Japanese fonts obtained from http://www.hakusyu.com/download_education.htm/.

[11] Aozora Bunko. Welcome to the internet's electronic dictionary, aozora bunko. Available at `http://www.aozora.gr.jp/`.

[12] Tomoko Uchida. Janome v0.3 documentation. Janome API can be found at `https://mocobeta.github.io/janome/`.