COMP5046 Natural Language Processing
Assignment 2 – Named Entity Recognition

Li Ding(460471404) & George Wu (470528901), Group 119

University of Sydney, NSW 2006, Australia
ldin8131@uni.sydney.edu.au
xiwu6084@uni.sydney.edu.au

1 Introduction

Named Entity Recognition is a field in NLP which categories individual words into specific categories, such as organizations, names, geographical locations etc. In this paper, we will analyze performance of various techniques applied to sequential model and evaluate the performance of the models.

The source code of our base model is from COMP5046 Lab 9 – CRF NER model, which would be our benchmark when evaluating the performance of model.

2 Data preprocessing

The dataset is generally very well organized; thus, we have not applied any preprocessing techniques.

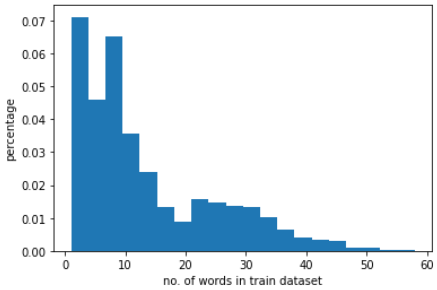
The dataset is from a segment of CONLL-2003, which comprised of English news articles from Reuters.

Table 01 - Dataset

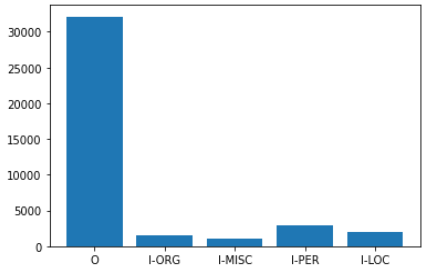
Data	**Quantity**
Train	3000
Validation	700
Test	3685 (no y label)

Distribution of length of words in sentences in train data are shown in table 2. As shown, length of words within sentences is between 1 to 60, with the distribution of dataset significantly tampering down beyond 30 words.

Distribution of composition of tags are shown in table 3. As Shown, overall majority of tags are O, with NER elements tagged in either “I-ORG” “I-MISC” “I-PER” OR “I-LOC”.



Graph 02 – Distribution of no. of words in sentence



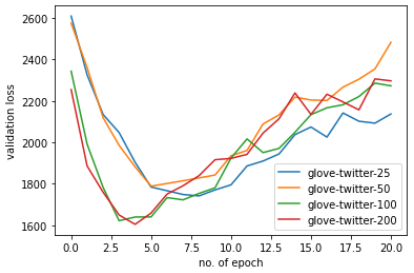
Graph 03 – Distribution of NER tags

3 Input Embedding

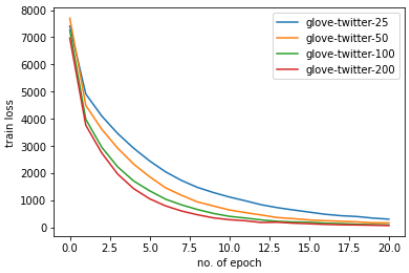
3.1 Word Embedding

In our model, we have used pre-trained glove models from gensim. The model is downloaded directly from gensim downloader. The base model uses glove-twitter-25, which converts words into 25 embedding features. We have also attempted using 50, 100 & 200 embedding to evaluate the performance.

As shown in graph below, higher dimensions of word embedding generally performs betters than lower dimensionality when tested using validation data, dim = 100 & 200 have yielded similar results. This might indicate that for data size, dimensionality of 100 is likely to produce sufficient performance, additional increase in dimensionality would not significantly boost the performance of the model. Graph 6 shows the training loss per epoch, as shown, higher dimensionality would help the model converge faster.



Graph 05 – Validation loss of different word embedding size



Graph 06– training loss of different word embedding size

3.2 POS Tagging

We have utilized the nltk package for POS tagging. The pretrained model we have selected is "averaged_perceptron_tagger". The resultant output is to classify each word into 1 of 45 categories. As shown, top 4 categories are NN (common noun): 10721, JJ (Adjective): 4329, IN(Prepositions and Subordinating Conjunctions): 3654, & CD (Cardinal Numbers): 2698.

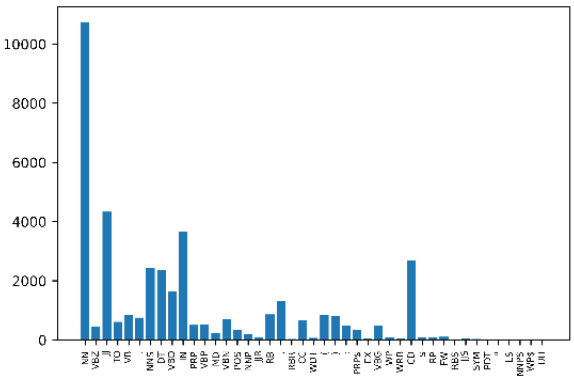
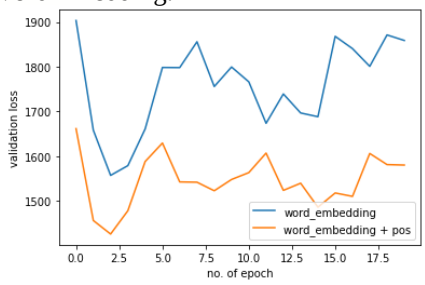


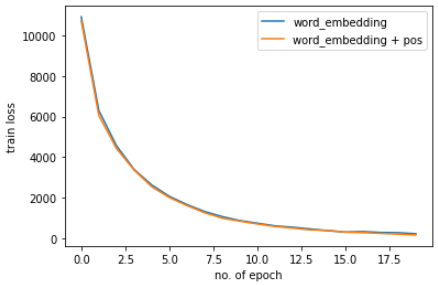
Table 7 – Distribution of POS tags

The POS output is then converted in the same way as words, which each word is coveted to a corresponding numerical figure. Because the POS information is a categorical information. The information is further encoded via one-hot-encoder via an identity matrix. During training process, the POS information would also be inputted into the model, which we then use nn.embedding to convert the POS label to the one-hot-encoder. E.g. if we use 100 dim embedding, and POS is 45 dim, the total dimension for the LSTM input would be 145, which can be combined via torch.cat function.

We would expect the model performance of having additional POS information in comparison to just word embedding to be better, due to additional valuable information this will bring. As shown in graph below, Word Embedding + POS does indeed perform better than model within Word Embedding only. In the training graph, the model converges faster with POS than without, and in the Validation loss graph, Word Embedding + POS produced a consistent lower loss than the model with just Word Emedding.



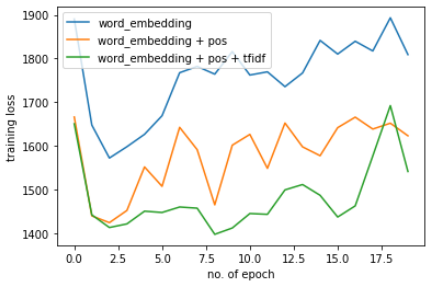
Graph 08 – Validation loss of Word_emb vs WordEmb + POS



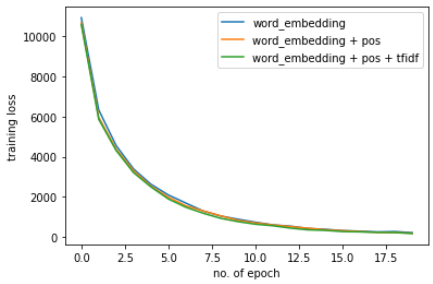
Graph 09 – Training loss of Word_emb vs WordEmb + POS

3.3 TF-IDF

In the TF-IDF section, we have referenced the code from Lab 02 to generate the TF-IDF on a sentence basis. This would compare the word occurrence within the sentence against the entire dataset. The intent is to have key words of the particular sentence to have a higher value than other words, potentially assisting in identifying non "O" value labels in NER model. The output from TF-IDF is a single value per word, which can be concatenated to the main model. This can be converted into the same shape using .unsequeeze & .expand to convert data into shape of [1,1,1].



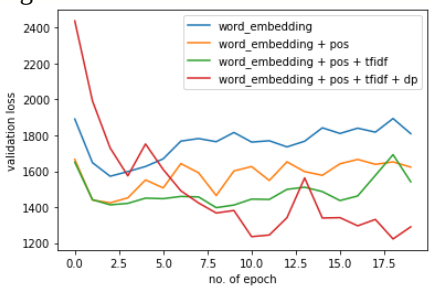
Graph 10 – Validation loss of Word_emb vs WordEmb + POS vs WordEmb + POS + TFIDF



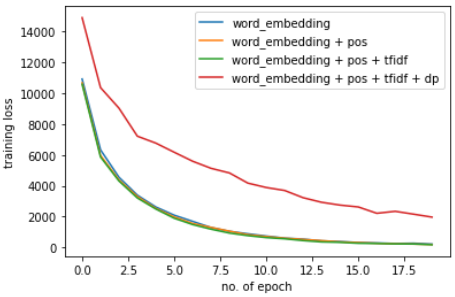
Graph 11 – Training loss of Word_emb vs WordEmb + POS vs WordEmb + POS + TFIDF

3.4 Dependency Parsing

We have used en_core_web_sm from spacy for dependency parsing training. The dependency parsing column (token.head) is then feed into the model. Similar to TF-IDF, the single value is concatenated into the main model. As shown, dependency parsing also helps with the minimizing validation loss, although the model does seem to take longer to train



Graph 12 – Validation loss of Word_emb vs WordEmb + POS vs WordEmb + POS + TFIDF + Dependency Parsing



Graph 13 – Training loss of Word_emb vs WordEmb + POS vs WordEmb + POS + TFIDF + Dependency Parsing

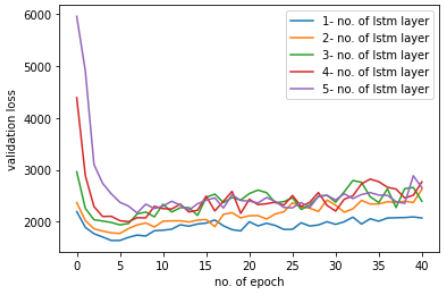
4 NER model

In the sections above, we have outlined the selection & creation of additional input features to potentially enhance the model, in this section, we would like to analyze the hyperparameters that would help with the performance of the model.

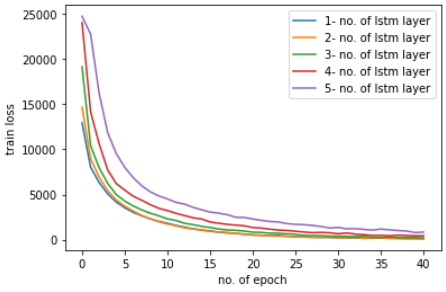
We have used the CRF-NER model as our base model. The model is built using conditional random field with single layer LSTM. In the following sections, we would like to analyze whether we can bring further improved performance from adjusting the number of layers & application of attention model.

4.1 Layers

We have attempted with LSTM layer from 1 – 5. As shown below, as the number of layers increase there's a general decrease in validation loss & speed of training. The mostly likely explanation for this is that because the dataset is relatively small, having a deeper layer would make the model harder to learn due to small amount of data.



Graph 14 – Validation Loss of different number of LSTM layer (without dropout)



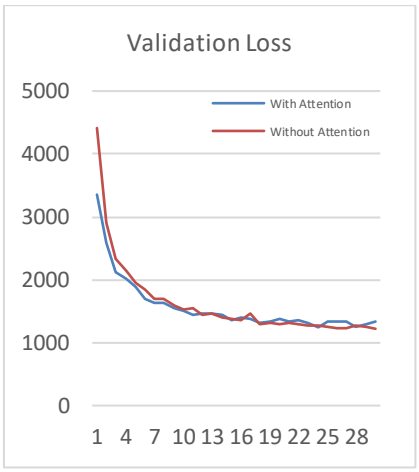
Graph 15 – Training Loss of different number of LSTM layer (without dropout)

4.2 Attention Model

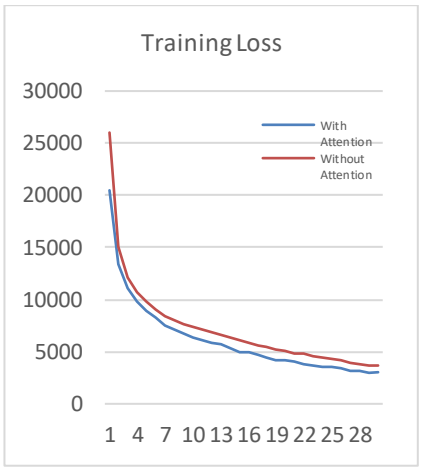
We have implemented multihead attention model and asessed the performance of the model. we have used basic scaled dot-product attention. In theory, multihead attention will increase the performance of long sentences.

The multi-head attention model is applied in the _get_lstm_feature function, after the lstm has completed the forward pass, it is then feed into the multihead attention class, which returns att_output & att_weight variables.

As shown in graph below, multi-head attention does bring about improved performance, although performance improvements are quite marginal.

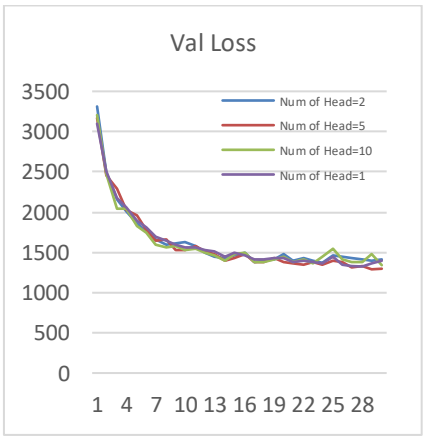


Graph 16 – Validation Loss of standard model vs model with attention

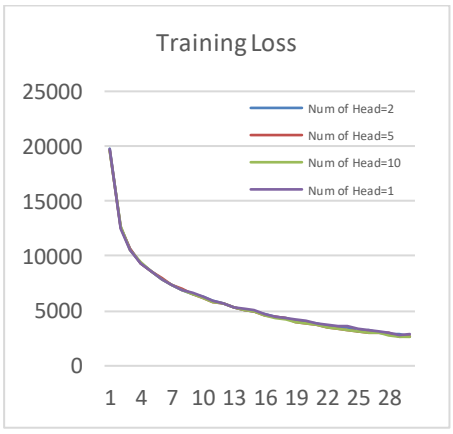


Graph 16 – Training Loss of standard model vs model with attention

Next, we would like to test the performance of model with different number of heads. Result is shown in the following graph. As shown, number of heads does not significantly impact on model performance, although Num_head = 5 seems to produce the most optimal result.



Graph 18 - Validation Loss comparing with different number of heads in attention model



Graph 19 - Training Loss comparing with different number of heads in attention model

5 Evaluation

5.1 Evaluation setup

The following evaluation would be performed with validation dataset with various model, including the following:

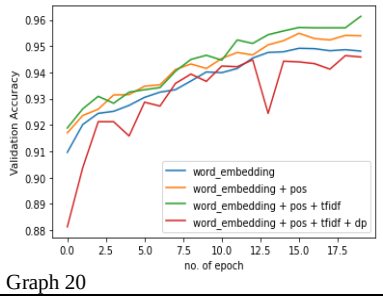
1. Ablation study of different embedding model.
2. Ablation study of different attention strategy.
3. Ablation study of different layer study.
4. Performance comparison of selectioned models.

The model would perform a fixed number of epochs, with best validation accuracy result recorded.

5.2 Evaluation result

1. Ablation study of different embedding model.

Model	**Validation Accuracy**
Baseline Lab 9 CRF LSTM Model With word embedding	0.9419
Baseline Lab 9 CRF LSTM Model With word embedding + POS	0.9548
Baseline Lab 9 CRF LSTM Model With word embedding + POS + TF-IDF	0.9613
Baseline Lab 9 CRF LSTM Model With word embedding + POS + TF-IDF + Dependency Parsing	0.9464

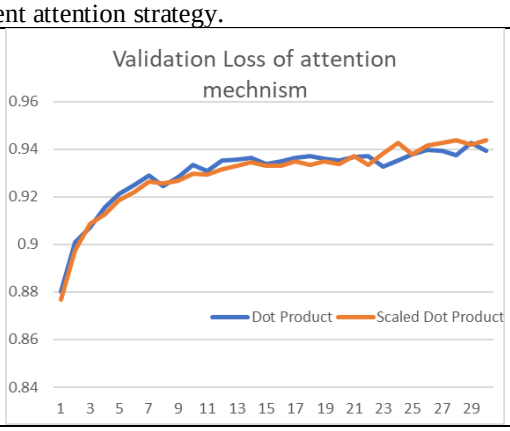


Graph 20

Note: inclusion of Dependency Parsing does not improve performance if measured against validation accuracy, although measured using validation loss, the model shows the min. loss against other models.

2. Ablation study of different attention strategy.

Model	**Validation Accuracy**
Dot Product	0.939386
Scaled Dot Product	0.943753



3. Ablation study of different layer study.

Model	**Validation Accuracy**
Baseline Lab 9 CRF LSTM Model 1 LSTM Layer	0.9483
Baseline Lab 9 CRF LSTM Model 2 LSTM Layers	0.9498
Baseline Lab 9 CRF LSTM Model 3 LSTM Layers	0.9502
Baseline Lab 9 CRF LSTM Model 4 LSTM Layers	0.9459
Baseline Lab 9 CRF LSTM Model 5 LSTM Layers	0.9367

4. Performance comparison of different models.

Model	Validation Accuracy
Baseline Lab 9 CRF LSTM Model with 1 layer	0.9465
Baseline Lab 9 CRF LSTM Model with 2 layer & dropout = 0.5	0.9443
Wordembedding (200 dim) + POS + TFIDF + Dependency Parsing – CRF LSTM Model	0.943489
Wordembedding (200 dim) + POS + TFIDF + Dependency Parsing – CRF LSTM Model with Attention (submitted code file)	**0.944283**

5.3 Discussion

Similar to what we have noticed in comparison of validation loss vs validation accuracy in testing different embedding models (comparison of inclusion/exclusion of Dependcy Parsing), we have noticed similar divergence in Validation Loss & validation accuracy in many of our other models. While validation loss reaches a minimal after 5-10 epochs, validation accuracy continues to improve slightly. This makes the evaluation much harder, whether to end training at min. loss or max. accuracy.

Possible explanation to this divergence could be some small amount of dataset is producing results of higher loss values (without impacting on accuracy), while a majority of data continues to improve in prediction. This is definitely an element that can be further investigated in order to further fine-tune the model. For our paper, we have chosen the model with the highest validation accuracy, rather than lowest validation loss.

Further, the application of attention mechanism has only made marginal improvements to the performance of model. This is might be to due to the fact that the model is not large enough to see a significant improvement. This can be further tested with much larger dataset to see if more improvements can occur.

6 Conclusion

From the analysis above, we have evaluated the performance of including various input features, comparing performance of number of layers & usage of attention model. We have chosen to implement CRF LSTM network with following parameters:

- Word Emedding using glove-twitter-100, with POS tagging with one-hot-encoder, TF-IDF value & Dependency Parsing values.
- 1 LSTM Layer
- Attention model with 1 or 5 heads using Dot Product

7 Appendix

7.1 Training log of different layers of LSTM

```
Epoch:1, Training loss: 12936.39, train acc: 0.9116, val loss: 2198.10, val acc: 0.9012, time: 202.88s
Epoch:2, Training loss: 8032.53, train acc: 0.9247, val loss: 1892.49, val acc: 0.9062, time: 202.54s
Epoch:3, Training loss: 6337.73, train acc: 0.9375, val loss: 1770.01, val acc: 0.9117, time: 201.87s
Epoch:4, Training loss: 5075.95, train acc: 0.9466, val loss: 1707.89, val acc: 0.9149, time: 201.62s
Epoch:5, Training loss: 4170.41, train acc: 0.9526, val loss: 1639.79, val acc: 0.9201, time: 201.20s
Epoch:6, Training loss: 3532.61, train acc: 0.9571, val loss: 1640.68, val acc: 0.9223, time: 200.55s
Epoch:7, Training loss: 3027.93, train acc: 0.9615, val loss: 1700.77, val acc: 0.9235, time: 200.97s
Epoch:8, Training loss: 2637.94, train acc: 0.9629, val loss: 1743.71, val acc: 0.9238, time: 200.87s
Epoch:9, Training loss: 2303.16, train acc: 0.9689, val loss: 1725.65, val acc: 0.9263, time: 202.69s
Epoch:10, Training loss: 2028.35, train acc: 0.9698, val loss: 1828.57, val acc: 0.9274, time: 203.35s
Epoch:11, Training loss: 1811.89, train acc: 0.9722, val loss: 1837.96, val acc: 0.9288, time: 204.92s
Epoch:12, Training loss: 1581.20, train acc: 0.9735, val loss: 1857.53, val acc: 0.9292, time: 206.38s
Epoch:13, Training loss: 1387.81, train acc: 0.9765, val loss: 1942.50, val acc: 0.9304, time: 207.42s
Epoch:14, Training loss: 1208.11, train acc: 0.9771, val loss: 1915.67, val acc: 0.9312, time: 209.04s
Epoch:15, Training loss: 1071.20, train acc: 0.9802, val loss: 1957.74, val acc: 0.9312, time: 209.04s
Epoch:16, Training loss: 975.09, train acc: 0.9814, val loss: 1975.10, val acc: 0.9324, time: 208.24s
Epoch:17, Training loss: 841.59, train acc: 0.9793, val loss: 2035.46, val acc: 0.9296, time: 207.41s
Epoch:18, Training loss: 740.33, train acc: 0.9852, val loss: 1922.18, val acc: 0.9358, time: 205.70s
Epoch:19, Training loss: 695.73, train acc: 0.9886, val loss: 1848.86, val acc: 0.9382, time: 204.43s
Epoch:20, Training loss: 591.66, train acc: 0.9895, val loss: 1824.32, val acc: 0.9395, time: 203.16s
Epoch:21, Training loss: 520.71, train acc: 0.9875, val loss: 2001.07, val acc: 0.9378, time: 205.00s
Epoch:22, Training loss: 498.32, train acc: 0.9920, val loss: 1920.26, val acc: 0.9412, time: 203.95s
Epoch:23, Training loss: 443.63, train acc: 0.9916, val loss: 1972.10, val acc: 0.9394, time: 203.56s
Epoch:24, Training loss: 480.65, train acc: 0.9916, val loss: 1929.38, val acc: 0.9431, time: 203.70s
Epoch:25, Training loss: 377.18, train acc: 0.9948, val loss: 1852.75, val acc: 0.9423, time: 202.40s
Epoch:26, Training loss: 322.01, train acc: 0.9943, val loss: 1855.73, val acc: 0.9456, time: 201.94s
Epoch:27, Training loss: 296.39, train acc: 0.9946, val loss: 1981.97, val acc: 0.9435, time: 201.85s
Epoch:28, Training loss: 279.14, train acc: 0.9950, val loss: 1918.32, val acc: 0.9460, time: 202.30s
Epoch:29, Training loss: 259.71, train acc: 0.9957, val loss: 1938.68, val acc: 0.9428, time: 200.52s
Epoch:30, Training loss: 246.38, train acc: 0.9959, val loss: 2001.40, val acc: 0.9465, time: 200.50s
Epoch:31, Training loss: 218.41, train acc: 0.9963, val loss: 1950.88, val acc: 0.9463, time: 202.03s
Epoch:32, Training loss: 202.02, train acc: 0.9964, val loss: 2000.91, val acc: 0.9471, time: 201.90s
Epoch:33, Training loss: 187.13, train acc: 0.9971, val loss: 2089.55, val acc: 0.9463, time: 202.00s
Epoch:34, Training loss: 169.67, train acc: 0.9976, val loss: 1956.92, val acc: 0.9477, time: 202.94s
Epoch:35, Training loss: 186.31, train acc: 0.9962, val loss: 2058.27, val acc: 0.9467, time: 202.34s
Epoch:36, Training loss: 177.52, train acc: 0.9970, val loss: 2011.68, val acc: 0.9436, time: 201.45s
Epoch:37, Training loss: 160.85, train acc: 0.9973, val loss: 2073.05, val acc: 0.9483, time: 201.85s
Epoch:38, Training loss: 146.05, train acc: 0.9981, val loss: 2078.41, val acc: 0.9461, time: 201.82s
Epoch:39, Training loss: 128.11, train acc: 0.9984, val loss: 2081.96, val acc: 0.9477, time: 202.02s
Epoch:40, Training loss: 139.86, train acc: 0.9986, val loss: 2093.52, val acc: 0.9476, time: 202.71s
Epoch:41, Training loss: 119.48, train acc: 0.9980, val loss: 2073.08, val acc: 0.9473, time: 202.95s
2 number_of_layer
Epoch:1, Training loss: 14683.49, train acc: 0.8987, val loss: 2370.74, val acc: 0.8928, time: 208.05s
Epoch:2, Training loss: 8972.04, train acc: 0.9166, val loss: 2023.41, val acc: 0.9051, time: 208.78s
Epoch:3, Training loss: 6921.65, train acc: 0.9281, val loss: 1865.21, val acc: 0.9101, time: 208.38s
Epoch:4, Training loss: 5419.69, train acc: 0.9372, val loss: 1821.99, val acc: 0.9165, time: 207.20s
Epoch:5, Training loss: 4406.83, train acc: 0.9483, val loss: 1786.76, val acc: 0.9201, time: 204.39s
Epoch:6, Training loss: 3713.19, train acc: 0.9528, val loss: 1774.20, val acc: 0.9203, time: 204.54s
Epoch:7, Training loss: 3174.26, train acc: 0.9541, val loss: 1870.06, val acc: 0.9194, time: 204.58s
Epoch:8, Training loss: 2674.80, train acc: 0.9576, val loss: 1939.58, val acc: 0.9217, time: 203.98s
Epoch:9, Training loss: 2305.29, train acc: 0.9622, val loss: 1978.05, val acc: 0.9242, time: 204.07s
Epoch:10, Training loss: 1985.04, train acc: 0.9642, val loss: 1899.15, val acc: 0.9254, time: 203.62s
Epoch:11, Training loss: 1734.03, train acc: 0.9689, val loss: 2012.46, val acc: 0.9243, time: 203.50s
Epoch:12, Training loss: 1532.68, train acc: 0.9720, val loss: 2017.50, val acc: 0.9255, time: 204.53s
Epoch:13, Training loss: 1301.83, train acc: 0.9741, val loss: 2019.42, val acc: 0.9247, time: 207.90s
Epoch:14, Training loss: 1178.79, train acc: 0.9768, val loss: 1995.63, val acc: 0.9292, time: 208.69s
Epoch:15, Training loss: 1051.38, train acc: 0.9790, val loss: 2032.22, val acc: 0.9305, time: 208.73s
Epoch:16, Training loss: 919.47, train acc: 0.9808, val loss: 2045.00, val acc: 0.9289, time: 208.43s
Epoch:17, Training loss: 836.46, train acc: 0.9845, val loss: 1906.58, val acc: 0.9369, time: 209.07s
Epoch:18, Training loss: 798.64, train acc: 0.9817, val loss: 2144.12, val acc: 0.9349, time: 207.76s
Epoch:19, Training loss: 685.34, train acc: 0.9862, val loss: 2174.92, val acc: 0.9350, time: 208.59s
Epoch:20, Training loss: 630.22, train acc: 0.9887, val loss: 2074.77, val acc: 0.9386, time: 208.99s
Epoch:21, Training loss: 521.34, train acc: 0.9882, val loss: 2117.40, val acc: 0.9378, time: 209.26s
Epoch:22, Training loss: 481.87, train acc: 0.9912, val loss: 2119.01, val acc: 0.9410, time: 208.90s
Epoch:23, Training loss: 431.74, train acc: 0.9934, val loss: 2054.14, val acc: 0.9419, time: 209.61s
Epoch:24, Training loss: 393.38, train acc: 0.9931, val loss: 2150.59, val acc: 0.9432, time: 206.78s
Epoch:25, Training loss: 377.42, train acc: 0.9947, val loss: 2192.69, val acc: 0.9444, time: 209.34s
Epoch:26, Training loss: 335.54, train acc: 0.9936, val loss: 2366.80, val acc: 0.9423, time: 210.20s
Epoch:27, Training loss: 286.22, train acc: 0.9939, val loss: 2285.80, val acc: 0.9423, time: 207.52s
Epoch:28, Training loss: 250.57, train acc: 0.9947, val loss: 2262.16, val acc: 0.9427, time: 205.10s
Epoch:29, Training loss: 254.46, train acc: 0.9957, val loss: 2199.26, val acc: 0.9453, time: 205.43s
Epoch:30, Training loss: 271.29, train acc: 0.9956, val loss: 2418.23, val acc: 0.9472, time: 205.08s
Epoch:31, Training loss: 202.75, train acc: 0.9958, val loss: 2319.02, val acc: 0.9468, time: 204.83s
Epoch:32, Training loss: 217.83, train acc: 0.9976, val loss: 2183.67, val acc: 0.9492, time: 208.49s
Epoch:33, Training loss: 241.56, train acc: 0.9975, val loss: 2249.45, val acc: 0.9488, time: 208.00s
Epoch:34, Training loss: 147.64, train acc: 0.9980, val loss: 2413.52, val acc: 0.9469, time: 207.94s
Epoch:35, Training loss: 153.38, train acc: 0.9975, val loss: 2342.22, val acc: 0.9476, time: 208.65s
Epoch:36, Training loss: 182.19, train acc: 0.9983, val loss: 2344.01, val acc: 0.9477, time: 208.47s
Epoch:37, Training loss: 155.99, train acc: 0.9982, val loss: 2386.12, val acc: 0.9472, time: 206.74s
Epoch:38, Training loss: 135.21, train acc: 0.9986, val loss: 2373.03, val acc: 0.9468, time: 205.10s
Epoch:39, Training loss: 106.76, train acc: 0.9995, val loss: 2395.75, val acc: 0.9473, time: 204.56s
Epoch:40, Training loss: 90.14, train acc: 0.9994, val loss: 2371.15, val acc: 0.9479, time: 204.63s
Epoch:41, Training loss: 92.24, train acc: 0.9992, val loss: 2629.11, val acc: 0.9498, time: 205.63s
3 number_of_layer
Epoch:1, Training loss: 19159.79, train acc: 0.8756, val loss: 2963.26, val acc: 0.8678, time: 211.11s
Epoch:2, Training loss: 10419.14, train acc: 0.9034, val loss: 2257.00, val acc: 0.8993, time: 211.76s
Epoch:3, Training loss: 7967.04, train acc: 0.9211, val loss: 2039.26, val acc: 0.9047, time: 211.28s
Epoch:4, Training loss: 6234.38, train acc: 0.9297, val loss: 2014.92, val acc: 0.9107, time: 211.84s
Epoch:5, Training loss: 5003.63, train acc: 0.9390, val loss: 1987.56, val acc: 0.9170, time: 212.09s
Epoch:6, Training loss: 4273.49, train acc: 0.9452, val loss: 1939.78, val acc: 0.9191, time: 211.01s
Epoch:7, Training loss: 3726.16, train acc: 0.9487, val loss: 1967.26, val acc: 0.9189, time: 211.50s
Epoch:8, Training loss: 3296.52, train acc: 0.9498, val loss: 2151.11, val acc: 0.9188, time: 212.52s
Epoch:9, Training loss: 2931.26, train acc: 0.9544, val loss: 2188.05, val acc: 0.9194, time: 214.73s
Epoch:10, Training loss: 2635.64, train acc: 0.9607, val loss: 2094.95, val acc: 0.9232, time: 213.93s
Epoch:11, Training loss: 2306.83, train acc: 0.9621, val loss: 2339.06, val acc: 0.9190, time: 214.57s
Epoch:12, Training loss: 2127.29, train acc: 0.9678, val loss: 2188.06, val acc: 0.9254, time: 214.85s
Epoch:13, Training loss: 1815.23, train acc: 0.9667, val loss: 2271.64, val acc: 0.9226, time: 215.76s
Epoch:14, Training loss: 1664.63, train acc: 0.9684, val loss: 2273.73, val acc: 0.9248, time: 218.84s
Epoch:15, Training loss: 1474.49, train acc: 0.9749, val loss: 2121.94, val acc: 0.9264, time: 221.86s
Epoch:16, Training loss: 1347.85, train acc: 0.9762, val loss: 2470.67, val acc: 0.9288, time: 222.95s
Epoch:17, Training loss: 1176.46, train acc: 0.9766, val loss: 2534.34, val acc: 0.9264, time: 223.95s
Epoch:18, Training loss: 1077.39, train acc: 0.9809, val loss: 2369.52, val acc: 0.9316, time: 224.22s
Epoch:19, Training loss: 1039.49, train acc: 0.9802, val loss: 2467.97, val acc: 0.9316, time: 220.59s
Epoch:20, Training loss: 955.95, train acc: 0.9823, val loss: 2418.12, val acc: 0.9304, time: 221.93s
Epoch:21, Training loss: 830.57, train acc: 0.9839, val loss: 2543.65, val acc: 0.9319, time: 222.71s
Epoch:22, Training loss: 797.70, train acc: 0.9858, val loss: 2610.65, val acc: 0.9354, time: 223.89s
Epoch:23, Training loss: 723.19, train acc: 0.9837, val loss: 2561.06, val acc: 0.9348, time: 225.23s
Epoch:24, Training loss: 709.29, train acc: 0.9887, val loss: 2365.72, val acc: 0.9411, time: 225.25s
Epoch:25, Training loss: 649.75, train acc: 0.9894, val loss: 2388.94, val acc: 0.9424, time: 225.11s
Epoch:26, Training loss: 586.24, train acc: 0.9865, val loss: 2463.23, val acc: 0.9332, time: 224.59s
Epoch:27, Training loss: 471.95, train acc: 0.9930, val loss: 2237.34, val acc: 0.9460, time: 218.53s
Epoch:28, Training loss: 459.01, train acc: 0.9905, val loss: 2323.47, val acc: 0.9436, time: 219.70s
Epoch:29, Training loss: 455.12, train acc: 0.9920, val loss: 2484.37, val acc: 0.9403, time: 216.55s
Epoch:30, Training loss: 392.99, train acc: 0.9917, val loss: 2511.17, val acc: 0.9442, time: 221.17s
Epoch:31, Training loss: 385.28, train acc: 0.9947, val loss: 2377.78, val acc: 0.9489, time: 219.70s
```

```
Epoch:31, Training loss: 385.28, train acc: 0.9947, val loss: 2377.78, val acc: 0.9489, time: 219.70s
Epoch:32, Training loss: 365.09, train acc: 0.9958, val loss: 2578.76, val acc: 0.9475, time: 222.14s
Epoch:33, Training loss: 348.30, train acc: 0.9920, val loss: 2795.41, val acc: 0.9457, time: 224.01s
Epoch:34, Training loss: 367.82, train acc: 0.9941, val loss: 2759.98, val acc: 0.9450, time: 222.86s
Epoch:35, Training loss: 254.09, train acc: 0.9970, val loss: 2482.93, val acc: 0.9457, time: 217.29s
Epoch:36, Training loss: 230.07, train acc: 0.9970, val loss: 2371.20, val acc: 0.9487, time: 222.44s
Epoch:37, Training loss: 309.61, train acc: 0.9956, val loss: 2634.47, val acc: 0.9483, time: 223.47s
Epoch:38, Training loss: 225.77, train acc: 0.9980, val loss: 2273.74, val acc: 0.9502, time: 224.57s
Epoch:39, Training loss: 236.80, train acc: 0.9972, val loss: 2638.16, val acc: 0.9471, time: 223.69s
Epoch:40, Training loss: 226.38, train acc: 0.9961, val loss: 2660.94, val acc: 0.9475, time: 225.09s
Epoch:41, Training loss: 191.88, train acc: 0.9991, val loss: 2395.73, val acc: 0.9502, time: 226.01s
4 number_of_layer
Epoch:1, Training loss: 24056.91, train acc: 0.8132, val loss: 4391.45, val acc: 0.7733, time: 230.76s
Epoch:2, Training loss: 14173.45, train acc: 0.8791, val loss: 2886.34, val acc: 0.8634, time: 228.73s
Epoch:3, Training loss: 10630.92, train acc: 0.9009, val loss: 2292.42, val acc: 0.8891, time: 221.81s
Epoch:4, Training loss: 7726.75, train acc: 0.9201, val loss: 2102.28, val acc: 0.9055, time: 220.22s
Epoch:5, Training loss: 6222.44, train acc: 0.9293, val loss: 2105.63, val acc: 0.9094, time: 220.47s
Epoch:6, Training loss: 5489.84, train acc: 0.9365, val loss: 2021.18, val acc: 0.9125, time: 220.54s
Epoch:7, Training loss: 4811.67, train acc: 0.9385, val loss: 2002.41, val acc: 0.9150, time: 220.19s
Epoch:8, Training loss: 4366.99, train acc: 0.9426, val loss: 2079.56, val acc: 0.9162, time: 222.29s
Epoch:9, Training loss: 3871.37, train acc: 0.9452, val loss: 2075.11, val acc: 0.9161, time: 219.79s
Epoch:10, Training loss: 3470.44, train acc: 0.9483, val loss: 2299.62, val acc: 0.9181, time: 220.69s
Epoch:11, Training loss: 3219.00, train acc: 0.9509, val loss: 2253.15, val acc: 0.9210, time: 219.27s
Epoch:12, Training loss: 2918.26, train acc: 0.9538, val loss: 2246.33, val acc: 0.9231, time: 219.48s
Epoch:13, Training loss: 2671.81, train acc: 0.9571, val loss: 2340.98, val acc: 0.9194, time: 219.21s
Epoch:14, Training loss: 2421.58, train acc: 0.9615, val loss: 2188.82, val acc: 0.9222, time: 219.18s
Epoch:15, Training loss: 2288.01, train acc: 0.9648, val loss: 2223.25, val acc: 0.9238, time: 219.51s
Epoch:16, Training loss: 1962.61, train acc: 0.9641, val loss: 2493.88, val acc: 0.9219, time: 219.33s
Epoch:17, Training loss: 1822.12, train acc: 0.9704, val loss: 2209.25, val acc: 0.9287, time: 219.02s
Epoch:18, Training loss: 1698.04, train acc: 0.9714, val loss: 2396.06, val acc: 0.9252, time: 218.81s
Epoch:19, Training loss: 1611.45, train acc: 0.9689, val loss: 2586.10, val acc: 0.9243, time: 219.27s
Epoch:20, Training loss: 1538.08, train acc: 0.9783, val loss: 2161.96, val acc: 0.9346, time: 218.10s
Epoch:21, Training loss: 1340.94, train acc: 0.9784, val loss: 2434.63, val acc: 0.9292, time: 218.30s
Epoch:22, Training loss: 1284.04, train acc: 0.9805, val loss: 2329.86, val acc: 0.9370, time: 221.24s
Epoch:23, Training loss: 1161.42, train acc: 0.9815, val loss: 2348.17, val acc: 0.9337, time: 219.75s
Epoch:24, Training loss: 1063.25, train acc: 0.9817, val loss: 2378.54, val acc: 0.9354, time: 218.38s
Epoch:25, Training loss: 1017.45, train acc: 0.9808, val loss: 2318.34, val acc: 0.9333, time: 219.01s
Epoch:26, Training loss: 933.31, train acc: 0.9839, val loss: 2507.46, val acc: 0.9344, time: 218.02s
Epoch:27, Training loss: 843.89, train acc: 0.9867, val loss: 2295.40, val acc: 0.9365, time: 216.99s
Epoch:28, Training loss: 775.31, train acc: 0.9854, val loss: 2374.23, val acc: 0.9408, time: 217.88s
Epoch:29, Training loss: 805.71, train acc: 0.9851, val loss: 2562.88, val acc: 0.9364, time: 218.72s
Epoch:30, Training loss: 761.80, train acc: 0.9894, val loss: 2316.30, val acc: 0.9401, time: 218.40s
Epoch:31, Training loss: 659.54, train acc: 0.9914, val loss: 2206.50, val acc: 0.9443, time: 217.17s
Epoch:32, Training loss: 740.88, train acc: 0.9849, val loss: 2434.76, val acc: 0.9393, time: 215.86s
Epoch:33, Training loss: 617.52, train acc: 0.9892, val loss: 2500.40, val acc: 0.9401, time: 215.85s
Epoch:34, Training loss: 556.34, train acc: 0.9896, val loss: 2732.08, val acc: 0.9395, time: 214.86s
Epoch:35, Training loss: 473.19, train acc: 0.9912, val loss: 2821.81, val acc: 0.9397, time: 214.93s
Epoch:36, Training loss: 477.21, train acc: 0.9890, val loss: 2772.78, val acc: 0.9415, time: 215.27s
Epoch:37, Training loss: 430.24, train acc: 0.9926, val loss: 2664.31, val acc: 0.9459, time: 214.99s
Epoch:38, Training loss: 500.04, train acc: 0.9920, val loss: 2630.77, val acc: 0.9430, time: 215.35s
Epoch:39, Training loss: 469.83, train acc: 0.9939, val loss: 2459.17, val acc: 0.9440, time: 215.96s
Epoch:40, Training loss: 431.45, train acc: 0.9935, val loss: 2510.69, val acc: 0.9418, time: 215.15s
Epoch:41, Training loss: 431.14, train acc: 0.9946, val loss: 2769.79, val acc: 0.9427, time: 215.29s
5 number_of_layer
Epoch:1, Training loss: 24751.83, train acc: 0.8121, val loss: 5956.17, val acc: 0.7662, time: 221.77s
Epoch:2, Training loss: 22778.39, train acc: 0.8103, val loss: 4902.94, val acc: 0.7634, time: 221.87s
Epoch:3, Training loss: 16220.41, train acc: 0.8638, val loss: 3103.90, val acc: 0.8445, time: 221.66s
Epoch:4, Training loss: 11866.52, train acc: 0.8881, val loss: 2740.63, val acc: 0.8701, time: 219.66s
Epoch:5, Training loss: 9555.82, train acc: 0.9072, val loss: 2538.95, val acc: 0.8849, time: 219.92s
Epoch:6, Training loss: 8010.94, train acc: 0.9168, val loss: 2375.81, val acc: 0.8845, time: 219.46s
Epoch:7, Training loss: 6861.20, train acc: 0.9281, val loss: 2302.08, val acc: 0.8965, time: 219.49s
Epoch:8, Training loss: 5958.45, train acc: 0.9346, val loss: 2170.96, val acc: 0.9047, time: 219.99s
Epoch:9, Training loss: 5299.30, train acc: 0.9343, val loss: 2339.57, val acc: 0.9043, time: 219.96s
Epoch:10, Training loss: 4838.22, train acc: 0.9392, val loss: 2254.97, val acc: 0.9058, time: 218.64s
Epoch:11, Training loss: 4517.34, train acc: 0.9394, val loss: 2302.37, val acc: 0.9059, time: 218.38s
Epoch:12, Training loss: 4136.38, train acc: 0.9372, val loss: 2395.60, val acc: 0.9090, time: 219.08s
Epoch:13, Training loss: 3943.57, train acc: 0.9460, val loss: 2323.13, val acc: 0.9144, time: 219.49s
Epoch:14, Training loss: 3579.56, train acc: 0.9490, val loss: 2224.60, val acc: 0.9117, time: 219.08s
Epoch:15, Training loss: 3311.92, train acc: 0.9507, val loss: 2347.08, val acc: 0.9101, time: 220.58s
Epoch:16, Training loss: 3053.77, train acc: 0.9518, val loss: 2415.30, val acc: 0.9105, time: 218.99s
Epoch:17, Training loss: 2937.55, train acc: 0.9517, val loss: 2460.62, val acc: 0.9156, time: 218.63s
Epoch:18, Training loss: 2770.67, train acc: 0.9623, val loss: 2264.96, val acc: 0.9117, time: 218.92s
Epoch:19, Training loss: 2479.89, train acc: 0.9623, val loss: 2520.40, val acc: 0.9137, time: 221.01s
Epoch:20, Training loss: 2454.65, train acc: 0.9645, val loss: 2409.46, val acc: 0.9120, time: 219.39s
Epoch:21, Training loss: 2289.49, train acc: 0.9677, val loss: 2398.32, val acc: 0.9198, time: 219.16s
Epoch:22, Training loss: 2134.00, train acc: 0.9705, val loss: 2363.45, val acc: 0.9219, time: 219.30s
Epoch:23, Training loss: 2018.04, train acc: 0.9675, val loss: 2466.80, val acc: 0.9203, time: 218.95s
Epoch:24, Training loss: 1961.50, train acc: 0.9717, val loss: 2393.91, val acc: 0.9238, time: 218.93s
Epoch:25, Training loss: 1783.18, train acc: 0.9749, val loss: 2276.45, val acc: 0.9203, time: 219.20s
Epoch:26, Training loss: 1693.59, train acc: 0.9762, val loss: 2268.00, val acc: 0.9234, time: 218.74s
Epoch:27, Training loss: 1662.22, train acc: 0.9744, val loss: 2372.01, val acc: 0.9289, time: 218.05s
Epoch:28, Training loss: 1577.34, train acc: 0.9762, val loss: 2273.89, val acc: 0.9226, time: 218.53s
Epoch:29, Training loss: 1453.77, train acc: 0.9790, val loss: 2496.03, val acc: 0.9304, time: 217.97s
Epoch:30, Training loss: 1295.51, train acc: 0.9808, val loss: 2508.36, val acc: 0.9300, time: 218.47s
Epoch:31, Training loss: 1359.14, train acc: 0.9790, val loss: 2420.29, val acc: 0.9283, time: 218.63s
Epoch:32, Training loss: 1195.99, train acc: 0.9768, val loss: 2540.06, val acc: 0.9271, time: 218.49s
Epoch:33, Training loss: 1224.50, train acc: 0.9829, val loss: 2446.68, val acc: 0.9330, time: 218.77s
Epoch:34, Training loss: 1131.56, train acc: 0.9823, val loss: 2525.75, val acc: 0.9375, time: 224.59s
Epoch:35, Training loss: 1067.81, train acc: 0.9838, val loss: 2562.79, val acc: 0.9297, time: 225.72s
Epoch:36, Training loss: 1172.39, train acc: 0.9802, val loss: 2514.65, val acc: 0.9293, time: 224.62s
Epoch:37, Training loss: 1088.87, train acc: 0.9844, val loss: 2510.75, val acc: 0.9367, time: 225.01s
Epoch:38, Training loss: 1009.77, train acc: 0.9843, val loss: 2388.91, val acc: 0.9349, time: 222.26s
Epoch:39, Training loss: 955.21, train acc: 0.9873, val loss: 2353.70, val acc: 0.9381, time: 220.44s
Epoch:40, Training loss: 784.73, train acc: 0.9779, val loss: 2886.21, val acc: 0.9246, time: 221.07s
Epoch:41, Training loss: 837.57, train acc: 0.9843, val loss: 2655.76, val acc: 0.9330, time: 221.10s
```