CSE-256 PA1 Writeup

Tianyi Chen

Janurary 30 2026

Contents

1 Part 1: Deep Averaging Network (DAN) (50 Points)

The model first maps input tokens into word embedding space, averages the embedding, and feeds the final representation into hidden layers for final prediction on classification. The optimal configuration obtained through ablation study (in the following section) is summarized in Table 1. With this configuration, the model achieves **an optimal validation accuracy of 0.807**.

Table 1: Optimal configuration for implemented DAN Model

Training

Parameter	Value
epochs	100
lr	1e-4
batch_size	32
weight_decay	1e-3
optimizer	Adam
activation	ReLU

Model

Parameter	Value
emb_dim	300
num_hidden_layers	2
hidden_dim	300
dropout_word	True
dropout_hidden	True
dropout_rate	0.3
freeze_embedding	True

1.1 Implementation of DAN

All the experiments are logged on WandB: *Link-to-WandB*

1.1.1 Number of Hidden Layers

As shown in Fig. 1, as the number of hidden layers increase, the training metrics tend to grow accordingly. On the validation metrics, when the number of layers is set to 3, the dev loss exhibits larger overall fluctuations. In the later stage of training, the dev loss increases and the dev accuracy decreases, indicating overfitting. This suggests that large depth results in memorization rather than improved generalization. **The best performance is shown when number of layers is set as 2, with a highest accuracy of 0.807.**

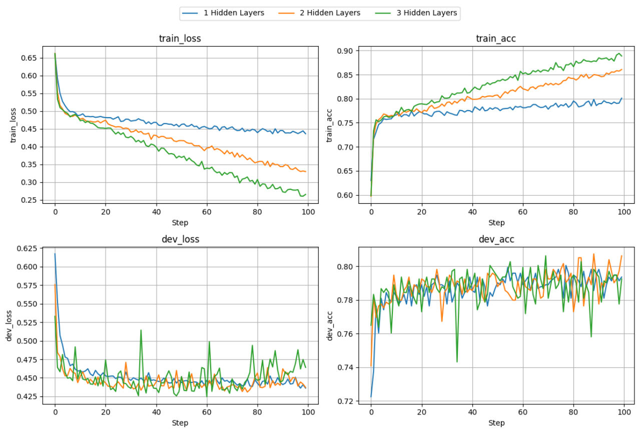

Figure 1: Ablation result of varying number of hidden layers

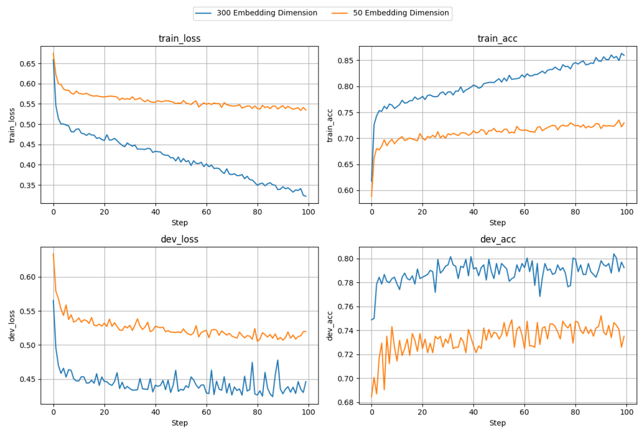

Figure 2: Ablation result of varying dimension of embedding layer

1.1.2 Dimension of Embedding

According to Fig. 2, larger embedding dimension brings notable advantage in both training and validation performance. **This demonstrates that larger embedding space has a stronger capability of representation of words, acting as the primary bottleneck of the model.**

1.1.3 Optimizer

Considering different scaling effect, SGD and AdaGrad uses a 1e-2 learning rate, compared to 1e-4 of Adam. As shown in Fig. 3, SGD performs slightly worse than others. Adam and AdaGrad is similar in dev metrics, while Adam shows a more stable convergence during training.

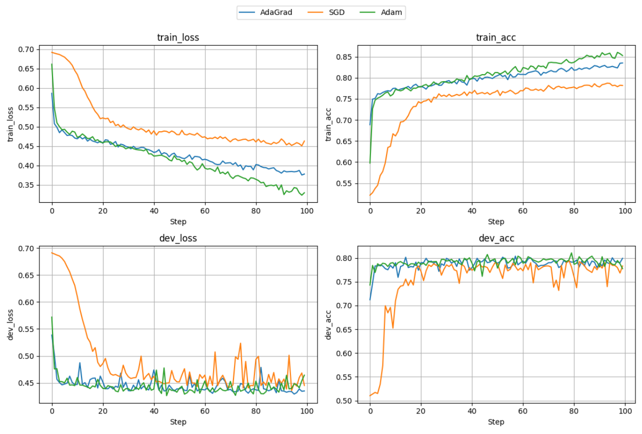

Figure 3: Ablation result of different optimizers

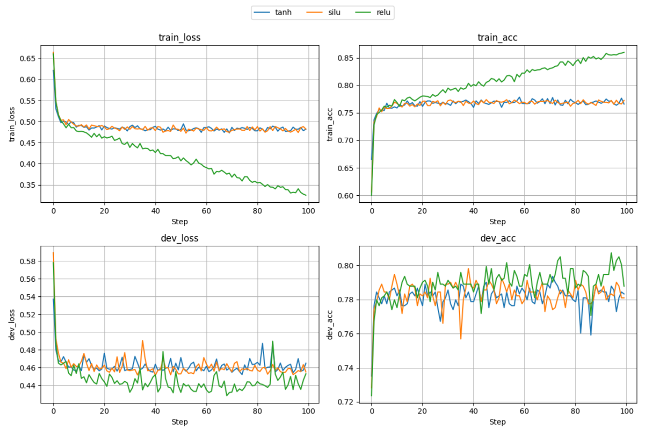

Figure 4: Ablation result of different activation function

1.1.4 Non-linearity

As shown in Fig. 4, ReLU exhibits notable training and validation performance compared to Tanh and SiLU. This shows that on simple model and task, ReLU is the optimal choice for non-linearity.

1.1.5 Dropout

In this experiment, dropout layer is tested after embedding layer and hidden layer individually and together. The result shows that embedding dropout (i.e., drop some words before the model) is more dominant in affecting the final result. On the contrary, hidden layer dropout shows weak regularization, leading to severe overfitting. **This demonstrates that forcing the model not to rely on specific words at first, is much more effective to dropout hidden states. The final validation accuracy shows that with both dropout, the model achieves the peak performance of 0.809 accuracy.**

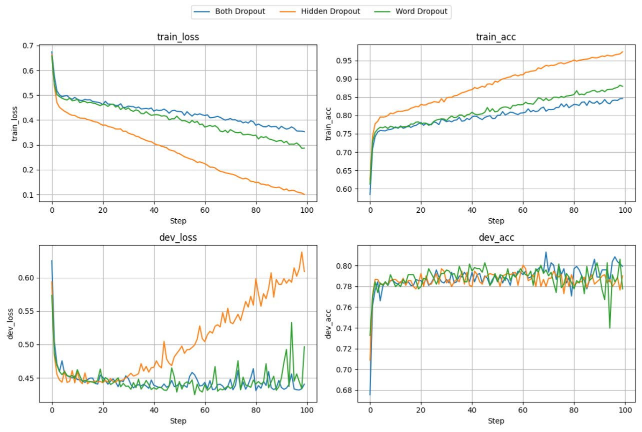

Figure 5: Ablation result of varying position of drop out layer

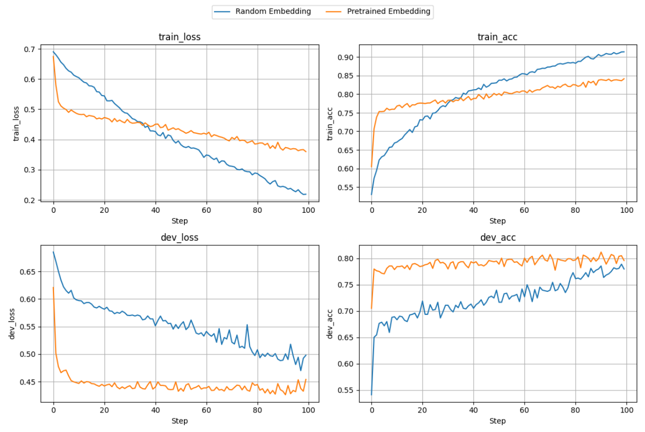

Figure 6: Contrastive results between pretrained and randomly-initialized embeddings

1.1.6 Conclusion

Overall, these ablation studies show that the generalization capability of DAN is mainly contained by representational capability at the embedding level rather than by deeper or wider MLP architectures. Larger hidden layer mainly increases model scale without introducing deeper understanding in semantics, leading to overfitting. In contrast, enlarging embedding dimensionality consistently improves both training and validation performance, highlighting the importance of rich semantic representations for text classification. Regularization techniques such as dropout and weight decay mitigate overfitting to some extent but do not fundamentally reverse the trend.

1.2 Randomly initialized embeddings

The contrastive results between pretrained and randomly-initialized embeddings are shown in Fig. 6. Pretrained embedding has a higher starting point, and shows stable performance during both training and dev process. While random embedding seems better in training, it converges much slower than pretrained ones. **Pretrained embedding shows better generalization capability.**

2 Part 2: Byte Pair Encoding (30 Points)

As shown in Fig. 7, larger vocab size leads to higher performance on training. However, as the vocab size increase to 20,000, dev loss exhibits an increase, indicating an overfitting. **A medium vocab size of 10,000 reaches the peak dev accuracy of 0.778.**

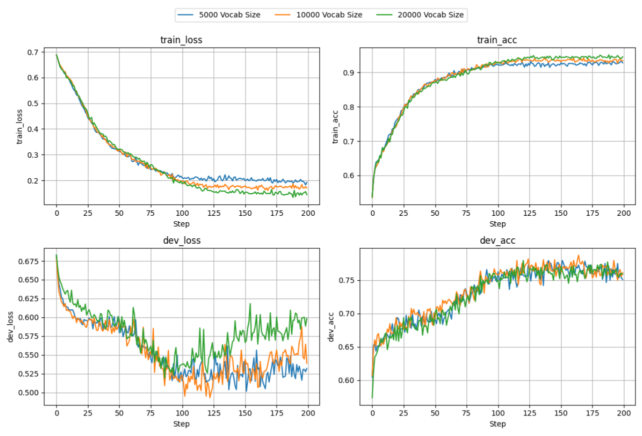

Figure 7: Contrastive results between different BPE vocab size

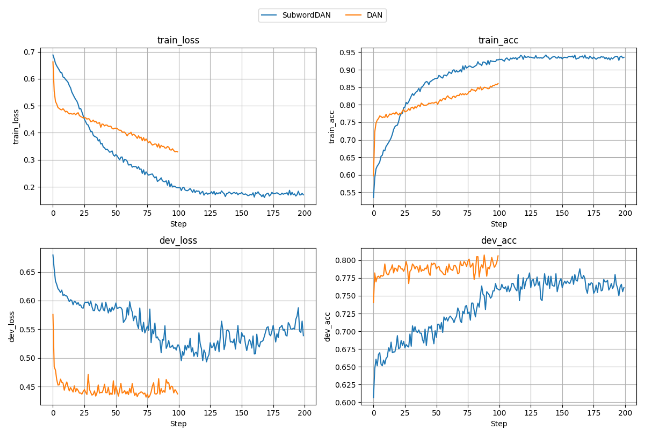

Figure 8: Contrastive results between SubwordDAN and DAN

Compared with pretrained embeddings (Fig. 8), SubwordDAN has a lower starting point due to randomly-initialized embedding, thus requiring more steps to train. Besides, the dev loss indicates that SubwordDAN is harder to converge. The optimal dev accuracy of SubwordDAN is 0.781, slightly lower than DAN by 0.02.

3 Part 3: Understanding Skip-Gram (20 Points)

3.1 Q1

(a) **Answer:**

$$P(y \mid \text{the}) = \begin{cases} 0.5, & y = \text{dog} \\ 0.5, & y = \text{cat} \\ 0, & \textit{otherwise} \end{cases}$$

(b) **Answer:** Let $v_{\text{the}} = (0, M)$. Then

$$v_{\text{the}} \cdot c_{\text{dog}} = v_{\text{the}} \cdot c_{\text{cat}} = M, \quad v_{\text{the}} \cdot c_{\text{a}} = v_{\text{the}} \cdot c_{\text{the}} = 0.$$

After the softmax, the resulting probabilities satisfy

$$P(\text{dog} \mid \text{the}) = P(\text{cat} \mid \text{the}) = \frac{e^M}{2e^M + 2}.$$

To control the final result within an error of 0.01, M needs to be larger than about 3.89

3.2 Q2

(c) **Answer:** (the, dog) (dog, the) (the, cat) (cat, the) (a, dog) (dog, a) (a, cat) (cat, a)

(d) **Answer:** Suppose M is large enough.

Word Vectors: $v_{the} = v_a = (M, 0)$; $v_{dog} = v_{cat} = (0, M)$

Context Vectors: $c_{the} = c_a = (0, M)$; $c_{dog} = c_{cat} = (M, 0)$