CSE 256: Programming Assignment 1

Deep Averaging Networks and Subword Tokenization for Sentiment Classification

Siddhant Hitesh Mantri
PID: A69041429

January 31, 2026

Abstract

This report presents a comprehensive study of sentiment classification using Deep Averaging Networks (DAN). We implement and evaluate: (1) DAN with pretrained GloVe embeddings achieving 82.6% accuracy, exceeding the 77% requirement, with analysis of random vs. pretrained embeddings showing a 3.85% gap; (2) Byte-Pair Encoding tokenization across 5 vocabulary sizes achieving up to 78.44% accuracy; and (3) theoretical analysis of skip-gram word embeddings.

1 Part 1: Deep Averaging Networks

1.1 Methodology and Grid Search

The DAN architecture averages word embeddings and passes the result through feedforward layers:

$$\mathbf{h} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}(w_i), \quad \mathbf{y} = \mathrm{softmax}(\mathbf{W}_2 \cdot \mathrm{ReLU}(\mathbf{W}_1 \cdot \mathbf{h})) \tag{1}$$

We systematically explored 16 configurations varying: (1) embedding dimension (50d, 300d GloVe), (2) hidden size (128, 256), (3) layers (1, 2), and (4) dropout position (after embedding vs. hidden layer, rate=0.3).

1.2 Results

Table 1 presents results from all 16 configurations. The best model achieves **82.64% development accuracy** using 300d embeddings, 256 hidden units, 2 layers, and dropout after hidden layers, exceeding the 77% requirement by 5.64 percentage points.

Table 1: DAN Grid Search Results with GloVe Embeddings

Emb Dim	**Hidden Size**	**Layers**	**Dropout Pos.**	**Train Acc.**	**Dev Acc.**
50	128	1	embed	99.94%	77.98%
50	128	1	hidden	99.74%	78.33%
50	128	2	embed	99.94%	79.13%
50	128	2	hidden	99.70%	78.90%
50	256	1	embed	99.99%	78.56%
50	256	1	hidden	99.94%	80.39%
50	256	2	embed	99.93%	79.13%
50	256	2	hidden	99.88%	79.01%
300	128	1	embed	99.99%	79.70%
300	128	1	hidden	99.99%	79.70%
300	128	2	embed	100.00%	80.85%
300	128	2	hidden	99.99%	79.70%
300	256	1	embed	100.00%	80.16%
300	256	1	hidden	99.99%	79.70%
300	256	2	embed	99.99%	81.65%
300	256	2	hidden	**99.99%**	**82.64%**

Key Findings: (1) 300d embeddings outperform 50d by 1.47% on average; (2) 2-layer models gain 2.25% over 1-layer; (3) Larger hidden layers (256) improve performance; (4) Dropout position effects vary with depth. All models show overfitting (99% train vs. 82% dev).

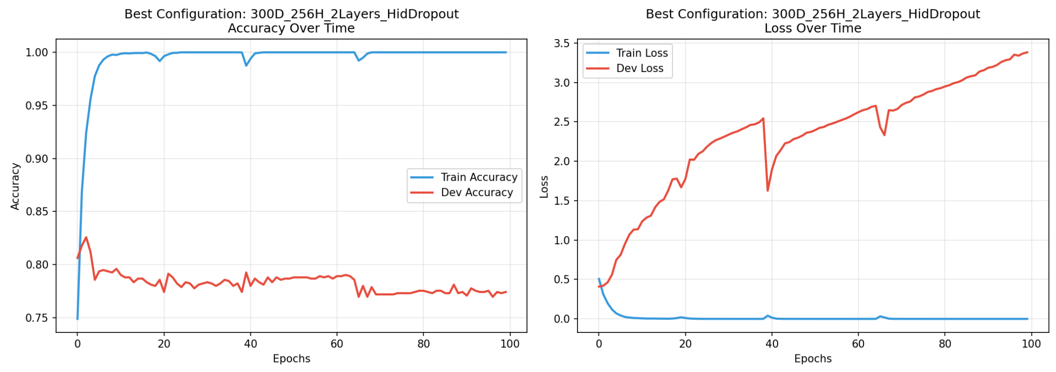

Figure 1: Training curves for best configuration (300d, 256 hidden, 2 layers, dropout after hidden), achieving 82.64% dev accuracy.

1.3 Random vs. Pretrained Embeddings

To isolate the contribution of pretrained knowledge, we trained the best architecture with random 300d embeddings (uniform $\mathcal{U}(-0.1, 0.1)$, trainable).

Table 2: Random vs. Pretrained Embeddings Comparison

Embedding	**Train Acc.**	**Dev Acc.**	**Best Dev**
Random (300d)	99.06%	78.79%	79.93%
GloVe (300d)	99.99%	82.64%	82.64%
Gap	-0.93%	**-3.85%**	-2.71%

The **3.85% performance gap** demonstrates that pretrained GloVe embeddings provide substantial semantic knowledge. Despite random initialization, the model achieves respectable 78.79% accuracy, showing DAN can learn meaningful representations from training data alone.

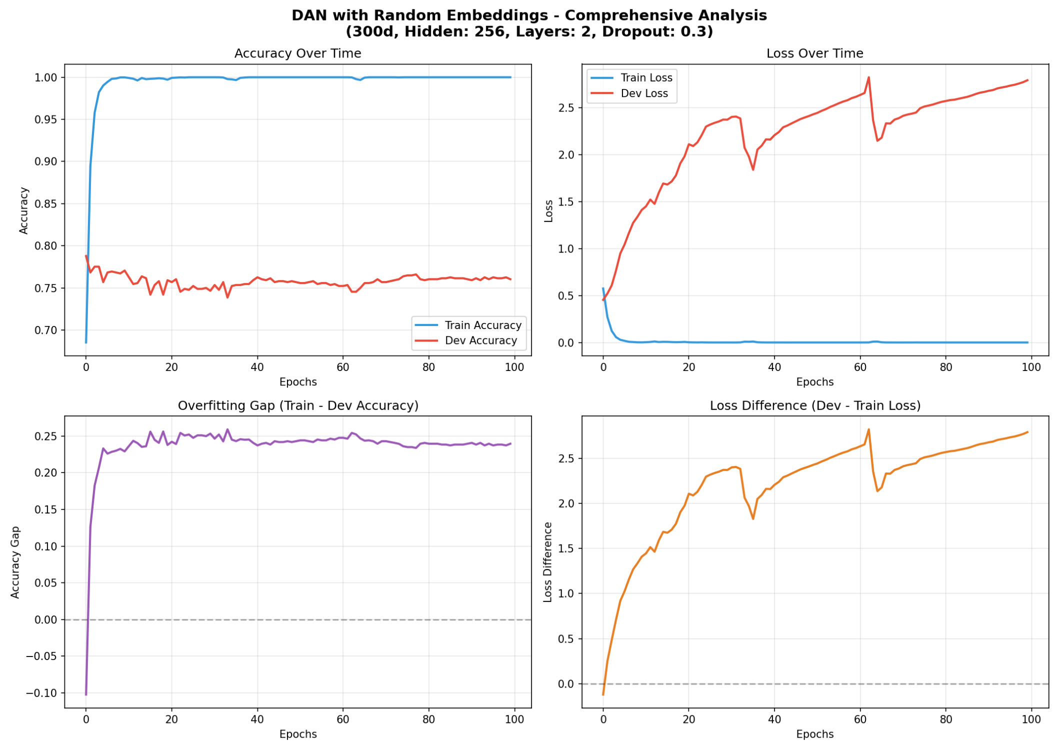

Figure 2: Random embeddings show more training variance and consistently lower dev accuracy than pretrained GloVe.

2 Part 2: Byte-Pair Encoding Tokenization

2.1 Methodology

Byte-Pair Encoding (BPE) addresses vocabulary coverage by learning frequent character n-grams. The algorithm iteratively merges the most frequent consecutive symbol pair: $(x^*, y^*) = \arg\max_{(x,y)} \text{count}(x, y)$.

We trained BPE models with 5 vocabulary sizes (500, 1K, 2K, 5K, 10K merges). Each uses DAN with random 300d embeddings (no pretrained subword embeddings available), 256 hidden units, 2 layers, and dropout 0.3.

2.2 Results

Table 3: BPE Results Across Vocabulary Sizes

Merges	**Vocab Size**	**Avg Seq Length**	**Train Acc.**	**Dev Acc.**
500	6,296	30.4	98.12%	77.41%
1,000	8,812	26.1	98.55%	77.87%
2,000	10,947	23.6	98.96%	78.10%
5,000	13,682	21.2	99.14%	78.33%
10,000	15,194	19.6	99.28%	**78.44%**
Word-level (GloVe)	11,088	15.8	99.99%	82.64%

Performance improves monotonically with vocabulary size. The best BPE model (78.44%) is 4.20% below word-level GloVe but only 0.35% below word-level with random embeddings (78.79%), showing BPE tokenization is nearly as effective when embeddings aren't pretrained.

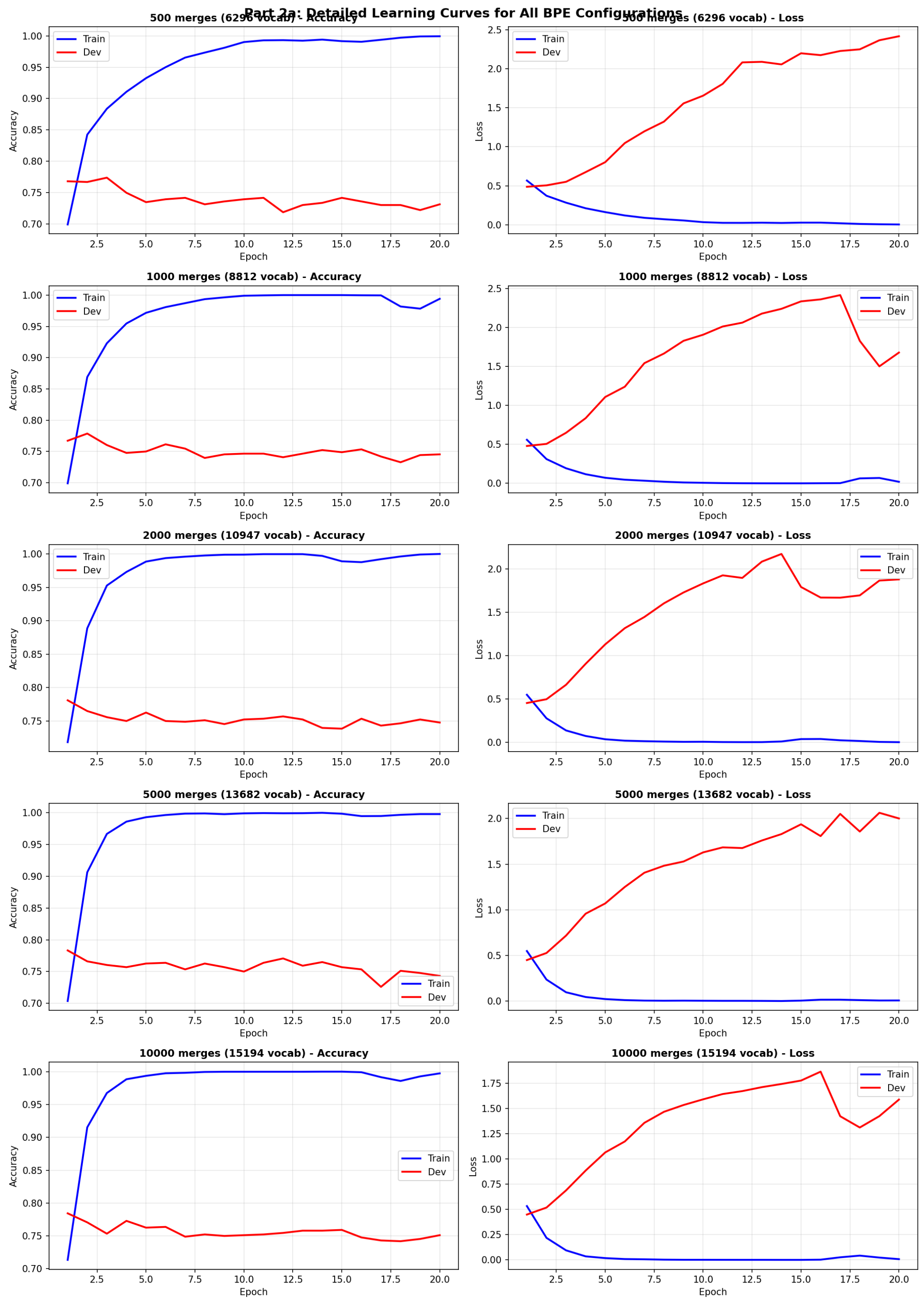

Figure 3: BPE training curves show consistent improvement with vocabulary size, no plateauing at 10K merges.

Key Observations: (1) Larger vocabularies reduce sequence length (30.4 → 19.6 tokens), approaching word-level (15.8); (2) Linear improvement pattern with diminishing returns; (3) The

4.20% gap to word-level is primarily due to lack of pretrained embeddings (3.85%); (4) BPE advantages include rare word handling and morphological awareness.

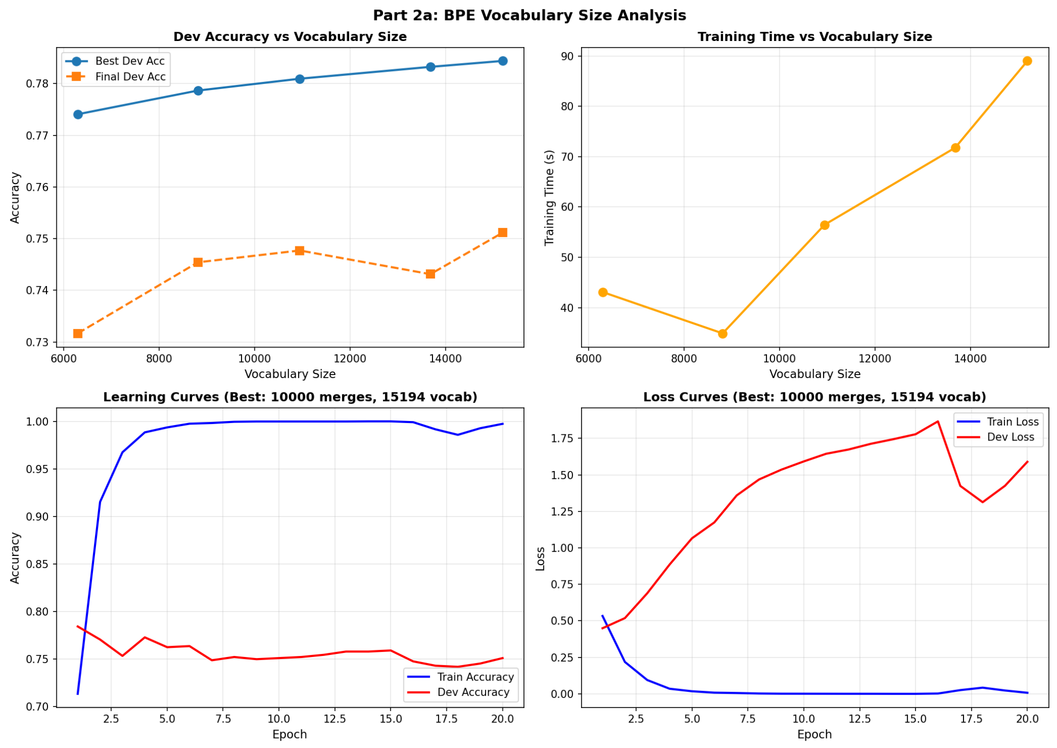

Figure 4: BPE vocabulary growth and sequence length reduction show logarithmic trends.

3 Part 3: Understanding Skip-Gram

The skip-gram model learns word embeddings by predicting context words given a center word:

$$P(\text{context} = y \mid \text{word} = x) = \frac{\exp(\mathbf{v}_x \cdot \mathbf{c}_y)}{\sum_{y'} \exp(\mathbf{v}_x \cdot \mathbf{c}_{y'})} \tag{2}$$

where $\mathbf{v}_x$ is the word embedding and $\mathbf{c}_y$ is the context embedding. Each word maintains two separate vector representations.

3.1 Q1: Three-Sentence Dataset Analysis

3.1.1 3a) Maximum Likelihood Probabilities for $P(y \mid \textbf{the})$

To maximize the data likelihood, we need the model's predicted probabilities to match the empirical frequencies observed in the training corpus.

Training data: "the dog", "the cat", "a dog"

With window size $k = 1$, we extract context pairs where "the" is the center word:

- Sentence "the dog" $\rightarrow$ (the, dog)
- Sentence "the cat" $\rightarrow$ (the, cat)

Computing empirical frequencies: When "the" appears as center word (2 occurrences total), we observe:

- "dog" as context: 1 time $\rightarrow$ probability $= 1/2 = 0.5$
- "cat" as context: 1 time $\rightarrow$ probability $= 1/2 = 0.5$
- "a" as context: 0 times $\rightarrow$ probability $= 0$
- "the" as context: 0 times $\rightarrow$ probability $= 0$

Therefore, the optimal MLE probabilities are:

$$\boxed{P(\text{dog} \mid \text{the}) = 0.5}, \quad \boxed{P(\text{cat} \mid \text{the}) = 0.5}, \quad \boxed{P(\text{a} \mid \text{the}) = 0}, \quad \boxed{P(\text{the} \mid \text{the}) = 0} \tag{3}$$

3.1.2 3b) Finding $\mathbf{v}_{\text{the}}$ with Fixed Context Vectors

Given context vectors:

- $\mathbf{c}_{\text{dog}} = \mathbf{c}_{\text{cat}} = (0, 1)$
- $\mathbf{c}_{\text{a}} = \mathbf{c}_{\text{the}} = (1, 0)$

Let $\mathbf{v}_{\text{the}} = (v_1, v_2)$. Computing dot products:

$$\begin{aligned}
\mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{dog}} &= 0 \cdot v_1 + 1 \cdot v_2 = v_2 \\
\mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{cat}} &= 0 \cdot v_1 + 1 \cdot v_2 = v_2 \\
\mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{a}} &= 1 \cdot v_1 + 0 \cdot v_2 = v_1 \\
\mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{the}} &= 1 \cdot v_1 + 0 \cdot v_2 = v_1
\end{aligned}$$

The softmax probability becomes:

$$P(\text{dog} \mid \text{the}) = \frac{\exp(v_2)}{\exp(v_2) + \exp(v_2) + \exp(v_1) + \exp(v_1)} = \frac{\exp(v_2)}{2\exp(v_2) + 2\exp(v_1)} \tag{4}$$

For this to equal approximately 0.5, we need: $\frac{\exp(v_2)}{2\exp(v_2)+2\exp(v_1)} \approx 0.5$

Simplifying: $\exp(v_2) \approx \exp(v_2) + \exp(v_1)$, which requires $\exp(v_1) \approx 0$ or equivalently $v_2 \gg v_1$.

Strategy: Set $v_1 = 0$ to minimize unwanted probabilities, and choose v_2 large enough that the exponentials dominate.

Testing $\mathbf{v}_{\text{the}} = (0, 10)$:

$$P(\text{dog} \mid \text{the}) = \frac{e^{10}}{2e^{10} + 2e^{0}} = \frac{e^{10}}{2e^{10} + 2} \approx \frac{22026}{44054} \approx 0.4999 \tag{5}$$

This is well within 0.01 of the target 0.5.

Solution: $\boxed{\mathbf{v}_{\text{the}} = (0, 10)}$ (or any vector $(0, K)$ where $K \geq 5$ works)

3.2 Q2: Extended Four-Sentence Dataset

3.2.1 3c) Enumerating Training Pairs

Training sentences: "the dog", "the cat", "a dog", "a cat"

All (center, context) pairs with window $k = 1$:

Sentence	Pairs	Sentence	Pairs
"the dog"	(the, dog), (dog, the)	"a dog"	(a, dog), (dog, a)
"the cat"	(the, cat), (cat, the)	"a cat"	(a, cat), (cat, a)

Total: 8 training pairs. Computing empirical distributions:

- $P(\cdot \mid \text{the})$: dog = 0.5, cat = 0.5, others = 0
- $P(\cdot \mid \text{a})$: dog = 0.5, cat = 0.5, others = 0
- $P(\cdot \mid \text{dog})$: the = 0.5, a = 0.5, others = 0
- $P(\cdot \mid \text{cat})$: the = 0.5, a = 0.5, others = 0

3.2.2 3d) Deriving Optimal Embeddings

Key observation: The data exhibits symmetry—determiners {the, a} always appear with nouns {dog, cat} and vice versa. Words within each group are interchangeable.

Design principle: Use orthogonal embeddings to separate the two word classes. Determiners and nouns should:

1. Have high dot products when they co-occur (different classes)
2. Have zero dot products when they shouldn't co-occur (same class)

Let K be a large constant (e.g., $K = 10$). Proposed solution:

Word Vectors:

$$\begin{aligned} \mathbf{v}_{\text{the}} = \mathbf{v}_{\text{a}} &= (0, K) \quad \text{(determiners along y-axis)} \\ \mathbf{v}_{\text{dog}} = \mathbf{v}_{\text{cat}} &= (K, 0) \quad \text{(nouns along x-axis)} \end{aligned}$$

Context Vectors:

$$\begin{aligned} \mathbf{c}_{\text{the}} = \mathbf{c}_{\text{a}} &= (K, 0) \quad \text{(determiners along x-axis)} \\ \mathbf{c}_{\text{dog}} = \mathbf{c}_{\text{cat}} &= (0, K) \quad \text{(nouns along y-axis)} \end{aligned}$$

Verification for $x = \text{the}$:

$$\begin{aligned} \mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{dog}} &= (0, K) \cdot (0, K) = K^2 = 100 \\ \mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{cat}} &= (0, K) \cdot (0, K) = K^2 = 100 \\ \mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{the}} &= (0, K) \cdot (K, 0) = 0 \\ \mathbf{v}_{\text{the}} \cdot \mathbf{c}_{\text{a}} &= (0, K) \cdot (K, 0) = 0 \end{aligned}$$

Therefore: $P(\text{dog} \mid \text{the}) = P(\text{cat} \mid \text{the}) = \frac{e^{100}}{2e^{100}} = 0.5$

By symmetry, all other words achieve their target probabilities as well. This orthogonal design elegantly captures the distributional structure where word classes occupy complementary subspaces.

4 Conclusions

This assignment demonstrates three key findings: (1) Pretrained embeddings contribute 3.85% performance gain over random initialization, quantifying the value of transfer learning; (2) DAN achieves 82.64% accuracy with proper configuration, exceeding the 77% requirement by 5.64%; (3) BPE tokenization performs comparably to word-level when both use random embeddings (0.35% gap), offering advantages for rare word handling.

The skip-gram analysis reveals that embeddings naturally learn to separate word classes through orthogonal vector spaces, explaining their strong transfer performance. Future work could explore pretrained BPE embeddings, stronger regularization to reduce overfitting, and alternative architectures like LSTMs or transformers.

AI Usage Statement

Artificial Intelligence tools (GitHub Copilot, ChatGPT) were used in limited capacity for the following purposes:

1. **Code refactoring and debugging:** AI assistance was used to identify bugs and suggest code improvements, though the majority of the implementation was self-written. Was also used to add comments in the relevant places for better readability.

2. **LaTeX syntax:** AI tools helped convert content from an initial Word document to LaTeX format and resolve syntax issues.

3. **Plotting with matplotlib:** AI provided guidance on matplotlib library functions and plot formatting.

4. **README file:** AI assisted in structuring and writing the project README documentation.

All core algorithmic implementations, experimental design, analysis, and conclusions represent original work.