Week9 Report

Question (i) a

input_childSpeech_trainingSet.txt

Contents: The dataset contains short, declarative sentences, often starting with "I" or simple commands like "Go Park" and "Look moon". The phrases are mostly two- to five-word sentences without complex grammar. Common phrases like "More more," "I want cookie," and "All gone" appear multiple times, which suggests frequent repetition within the text, which is a characteristic often seen in datasets aimed at language acquisition or early reading.
Vocabulary Size: The vocabulary size should be around 50 to 100. Words are simple nouns, verbs, and descriptors, indicating a vocabulary focused on familiar objects, actions, and desires, which are typical in child-directed language.
Dataset Length: The dataset is 10,000 rows long and contains approximately 55,000 words

input_childSpeech_testSet.txt

Contents: Similar to the training set, belongs to child-directed language
Vocabulary Size: Similar to the training set, the vocabulary should be around 50 to 100
Dataset Length: The dataset is 1,000 rows long and contains approximately 5,000 words. Which means the test set contents is 1/10 of the training set

input_shakespeare.txt

Contents: The dataset appears to contain dialogue structure from a play or dramatic script, with characters speaking in Old or Early Modern English. It features a conversational structure with lines attributed to various characters, including "First Citizen," "Second Citizen," "All," "MENENIUS," and so on. The content focus on themes of conflict, social issues, and power dynamics, characteristic of classical literature by Shakespeare.
Vocabulary Size: Given the length and the complexity of Shakespearean language, the dataset likely contains several thousand unique words, possibly between 2,000 and 4,000.
Dataset Length: The dataset is 40,000 rows long and contains approximately 200,000 words in total

Question (i) b

1. Embedding Dimension (n_embd): Currently set to 384, this parameter affects the dimensionality of each token's vector representation. Reducing it has a significant impact on parameter because it directly scales the embedding table size (vocab_size * n_embd), projection matrices and feedforward layers. Lowering it to 160 would make it still capture sufficient feature richness but with fewer

parameters.

2. Number of Layers (n_layer): Currently set to 6, dictating the depth of the transformer model. Reducing it to 3 layers will cut down on transformer blocks and hence parameters, making the model shallower and faster without fully sacrificing depth. (might reduce model ability to generalize over long dependencies)

3. Number of Attention Heads (n_head): Currently set to 6, each head requires its own set of attention weights. Reducing to 4 heads minimizes the size of projection matrices, but still keeps the attention mechanism's expressiveness. (might reduce performance)

4. Block Size (block_size): Currently set to 256, this determines the maximum context length for each sequence. Reducing it to 96 will reduce memory usage, computational cost and hence speed up training. This is particularly useful for datasets that contain short sequences.

5. Batch Size (batch_size): This parameter doesn't directly influence model parameters, reducing it will speed up training and lower memory consumption.

6. Dropout (dropout): This parameter doesn't impact parameter count, but reducing it slightly can improve performance on smaller datasets and models by retaining more information flow through each layer.

Model parameters: 0.95492 M parameters

Question (i) c

Configuration 1:

```
n_embd: 144
n_layer: 4
n_head: 4
block_size: 128
```

```
1.031368 M parameters
step 0: train loss 3.7122, val loss 3.7140
step 100: train loss 1.9694, val loss 1.9770
step 200: train loss 0.7857, val loss 0.7923
step 300: train loss 0.4126, val loss 0.4153
step 400: train loss 0.3759, val loss 0.3806
step 500: train loss 0.3669, val loss 0.3681
step 600: train loss 0.3607, val loss 0.3636
step 700: train loss 0.3596, val loss 0.3628
step 800: train loss 0.3546, val loss 0.3582
step 900: train loss 0.3496, val loss 0.3524
step 999: train loss 0.3510, val loss 0.3552
```

Configuration 2:

```
n_embd: 160
n_layer: 3
n_head: 4
block_size: 96

0.95492 M parameters
step 0: train loss 3.7532, val loss 3.7512
step 100: train loss 1.8554, val loss 1.8655
step 200: train loss 0.5476, val loss 0.5527
step 300: train loss 0.3993, val loss 0.4046
step 400: train loss 0.3782, val loss 0.3791
step 500: train loss 0.3729, val loss 0.3752
step 600: train loss 0.3671, val loss 0.3687
step 700: train loss 0.3663, val loss 0.3668
step 800: train loss 0.3613, val loss 0.3629
step 900: train loss 0.3607, val loss 0.3636
step 999: train loss 0.3591, val loss 0.3612
```

Configuration 3:

```
n_embd: 192
n_layer: 2
n_head: 4
block_size: 64

0.916648 M parameters
step 0: train loss 3.7265, val loss 3.7274
step 100: train loss 1.6023, val loss 1.6120
step 200: train loss 0.4597, val loss 0.4642
step 300: train loss 0.4068, val loss 0.4111
step 400: train loss 0.3969, val loss 0.3995
step 500: train loss 0.3876, val loss 0.3912
step 600: train loss 0.3836, val loss 0.3868
step 700: train loss 0.3822, val loss 0.3837
step 800: train loss 0.3788, val loss 0.3812
step 900: train loss 0.3757, val loss 0.3786
step 999: train loss 0.3750, val loss 0.3787
```

Validation Loss: Validation loss serves as a key indicator of the model’s generalization capabilities. The following are the final validation losses:

Configuration 1: 0.3552

Configuration 2: 0.3612

Configuration 3: 0.3787

1. Configuration 1 achieves the lowest validation loss, suggesting the best generalization to unseen data. This is because it has the highest parameter count and thus the most representational capacity.
2. Configuration 2 performs nearly as well, with only a slightly higher validation loss. This configuration strikes a good balance between model complexity and size.
3. Configuration 3 has the highest validation loss, reflecting that downsizing to fewer layers and embedding dimensions reduced its capacity to capture the training distribution effectively.

Overfitting: Overfitting can be assessed by comparing the gap between training and validation losses. Small gaps suggest good generalization, while large gaps mean overfitting.

Configuration 1: Last train loss: 0.3510; val loss: 0.3552 → Gap: **0.0042**

Configuration 2: Last train loss: 0.3591; val loss: 0.3612 → Gap: **0.0021**

Configuration 3: FLast train loss: 0.3750; val loss: 0.3787 → Gap: **0.0037**

1. All configurations demonstrate minimal overfitting
2. Configuration 2 shows the smallest gap, suggesting it achieves the best balance between model size and regularization.

Qualitative Assessment of Generated Output:

Configuration 1:

I hungry More more
Big truck All gone
Yummn apple I found it
More more More bubbles
I want more juice please
No mine Uhoh spill
My teddy I want more juice please
Night night I did
I tired
I draw Bath time
No nap I love yo
No nap No might n
I singred My teddy
I climb
No like it All gone
All done Saw big fluffy dogggy at park it run fast
Where kittty go Big hug

```
I d ide it I draw
I hunde Dadddyay play
I did ide I love you
I make mess Big hug
Read book All done
Help me please All gone
Dadddy read me
```

Fluency: The text maintains reasonable fluency with coherent short phrases. Many sequences closely resemble natural child speech, as seen in training data

Relevance: The output includes frequent phrases from the training set, such as "More more," "All gone," and "I found it." It also integrates varied topics like bedtime, play, and food, which relates to the themes in the training data

Adherence: The model adheres well to the repetitive, fragmented style characteristic of child speech, but with some minor deviations such as "Yummn apple" rather than "Yummy apple", which suggest slight overfitting.

Configuraton 2:

```
Daddy read me dinosaur book before bed
I jump I rump high
I seee ball No touch
I jump Mored more
I jump se I jump
No stop Big hug
No mine Daddy read me dinosaur book before bed
More bubbbles Look airplane
Daddy play Help me please Uh oh spill
No touch All gone
Look moon No nap
All done My teddy
I hide What is that
Alll gone All gone
More more No stop
No nap Come here
All done More more
I jump want play with big fluffy doggy at park it run fast
I tireasine Saw big fluffy doggy at park it
I run fa
```

Fluency: The sentences remain concise and plausible, similar to configuration 1

Relevance: Phrases like "Daddy read me dinosaur book before bed" indicate good retention of thematic elements from the training set. However, some artifacts like "I rump high" suggest reduced model capacity to generalize.

Adherence: The output captures the dataset's tone and structure but occasionally repeats simpler patterns more than Configuration 1, suggest reduced model capacity

Configuration 3:

```
No mine Look airplane
I jump Big hug
What is that I hungry
I hide No like it
I love you I found it
No stop All done
Read book I want that
No mine No like it
Come here Daddy read me dinosaur boook before bed Yummy apple
I found it Yucmmy apple
Go park I jump
Daddy play I hide
Where ball I jump All
I wash hands Go park
I found it All done
I see doggy All gone
Read book Whands pill No mine
Look airplane I see bird
I climb I jump high
Big truck Big hug
Go park All gone
I jump
Come here I see doggy
N
```

Fluency: The output maintains basic coherence but lacks richness in vocabulary and sentence variety compared to Configurations 1 and 2

Relevance: Phrases like "doggy" and "Yummy apple" reflect the training data but tend to repeat more frequently, which means the model become weaker to generate diverse context.

Adherence: The simple and repetitive structure reflects child speech but demonstrates a tendency to lose coherence in longer sequences.

Question (i) d

In the transformer architecture, self-attention layers compute attention scores to model relationships between tokens in a sequence. These computations rely on linear transformations of input embeddings to derive query, key, and value vectors, which are then used to calculate attention weights. The inclusion or exclusion of bias terms in these linear transformations can influence the model's behavior and performance.

1. Impact on including bias terms
    - Advantages
        (1) **Enhanced Expressive Power**: The bias allows the self-attention layer

to better adapt to diverse input distributions. In low-resource languages, the model can shift focus without significant input signal variation.

(2) **Improved Gradient Flow**: Bias terms can mitigate issues of vanishing gradients by providing a non-zero baseline. This is helpful for the early stages of training when weight values are small.

- Disadvantages

  (1) **Increased Parameter Count**: Including biases in the linear transformations adds additional learnable parameters, which may increase the model size

  (2) **Overfitting**: The added flexibility can lead to overfitting if not regularized properly

2. Impact on performance:
   - With Bias:

     (1) **Attention Distribution**: Bias terms allow the model to adjust attention scores even for uniform or poorly separated inputs, leading to better modeling of token relationships

     (2) **Generalization**: Models with biases often generalize better in tasks with diverse inputs

     (3) **Adaptability:** More robust to noisy or sparse inputs, improving downstream task performance
   - Without Bias:

     (1) **Attention Distribution**: The model relies solely on the input signal, which can be limiting when the signal is weak or ambiguous

     (2) **Efficiency:** Slightly fewer parameters, hence smaller memory usage and computation, might be useful in extremely constrained environments

     (3) **Simpler Training:** Models without bias may be not easy to become overfitting in resource-limited tasks

3. Conclusion: The inclusion of bias terms in self-attention layers enhances the transformer's ability to model diverse input distributions, improves generalization, and leads to more robust attention mechanisms, but it also marginally increases parameter count and risks overfitting on small datasets.

Question (i) e

Skip connections were introduced in the transformer architecture to facilitate effective training of deep networks. These connections allow input from one layer to "skip over" the subsequent layers and directly contribute to the output of the block. Skip connections are typically applied before layer normalization to ensure effective gradient flow

1. Impact of skip connections
   - Advantages

     (1) **Improved Gradient flow:** Skip connections create shortcuts that enable gradients to backpropagate more directly to earlier layers. This mitigates the vanishing gradient problem, hence converge faster during training

     (2) **Enhanced Representation Learning:** Skip connections allow each layer

to modify the input representation incrementally rather than transforming it entirely. This enables the model to retain and leverage features learned in earlier layers.

(3) **Regularization Effect:** Skip connections blend the input directly with the layer output, which effectively regularize the network and reduces the risk of overfitting, especially for small datasets

(4) **Training Stability:** Skip connections make it easier to optimize deep models by ensuring non-zero gradients reach the earlier layers

(5) **Model Performance:** Models with skip connections generally achieve lower training and validation loss compared to those without

- Disadvantages

  (1) **Increased Computational Cost:** Skip connections require additional element-wise operations and memory to store intermediate results

2. Impact on performance:
   - With Skip Connections:

     (1) **Training Dynamics:** Faster and more stable convergence

     (2) **Output Quality:** More coherent and contextually rich outputs

     (3) **Overfitting:** Regularization effect improves generalization
   - Without Skip Connections:

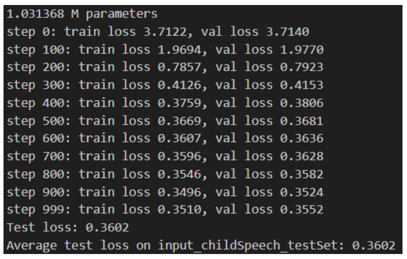
     (1) **Training Dynamics:** Slower convergence and potential instability

     (2) **Output Quality:** Less coherent outputs due to degraded feature refinement

     (3) **Overfitting:** Higher risk of overfitting individual layers' features

3. Conclusion: Skip connections are a critical component of transformer architectures, enabling deep stacking of layers without degradation in performance. They improve training stability, enhance representation learning, and contribute to the generalization and robustness of the model.

Question (ii) a



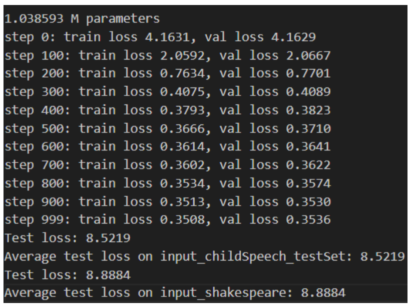
```
1.031368 M parameters
step 0: train loss 3.7122, val loss 3.7140
step 100: train loss 1.9694, val loss 1.9770
step 200: train loss 0.7857, val loss 0.7923
step 300: train loss 0.4126, val loss 0.4153
step 400: train loss 0.3759, val loss 0.3806
step 500: train loss 0.3669, val loss 0.3681
step 600: train loss 0.3607, val loss 0.3636
step 700: train loss 0.3596, val loss 0.3628
step 800: train loss 0.3546, val loss 0.3582
step 900: train loss 0.3496, val loss 0.3524
step 999: train loss 0.3510, val loss 0.3552
Test loss: 0.3602
Average test loss on input_childSpeech_testSet: 0.3602
```

1. Reported Test Loss
   - Test Loss for input_childSpeech_testSet: 0.3602
2. Evaluation of Test Loss
   - The reported test loss is the average **cross-entropy loss** for predicting the next character in the input_childSpeech_testSet. The test loss of 0.3602 is relatively low for a language model predicting character-level sequences. This suggests the model has successfully learned the simplicity and repetitive nature patterns in the training data and generalizes well to similar test data.

- Lower test loss indicates better performance, where the model predictions closely match the actual next characters in the test set

3. Possible Drawbacks
   - The dataset's simplicity and repetition might make it easier for the model to memorize patterns rather than truly generalize, hence make user overestimate the capabilities
4. Comparing to baseline model (Dummy - Random Prediction)
   - A dummy character-level model generates predictions by assigning equal probabilities to all characters in the vocabulary
   - For a vocabulary size of 100: Baseline Loss = -log (1/40) = log (40) = 1.602
   - The baseline loss for random predictions is **significantly higher** than 0.3602, showing that the GPT model outperforms random guessing

Question (ii) b



```
1.038593 M parameters
step 0: train loss 4.1631, val loss 4.1629
step 100: train loss 2.0592, val loss 2.0667
step 200: train loss 0.7634, val loss 0.7701
step 300: train loss 0.4075, val loss 0.4089
step 400: train loss 0.3793, val loss 0.3823
step 500: train loss 0.3666, val loss 0.3710
step 600: train loss 0.3614, val loss 0.3641
step 700: train loss 0.3602, val loss 0.3622
step 800: train loss 0.3534, val loss 0.3574
step 900: train loss 0.3513, val loss 0.3530
step 999: train loss 0.3508, val loss 0.3536
Test loss: 8.5219
Average test loss on input_childSpeech_testSet: 8.5219
Test loss: 8.8884
Average test loss on input_shakespeare: 8.8884
```

1. Reported Test Loss
   - Test Loss for input_shakespeare: 8.8884
2. Evaluation of Test loss
   - A test loss of **8.8884** is very high, signaling that the model struggles to predict the next character in the Shakespearean text (* input test set struggles as well because of vocabulary size increasing, if we calculate input test set together with input shakespeare. But that's not the correct loss for input test set)
   - This outcome is expected because the model was **not trained on Shakespearean text**. It was only trained on child-directed language, which means the model's limited vocabulary cannot represent the complexity and diversity of Shakespearean language
   - The training set (child-directed language) consists of short, repetitive phrases with simple vocabulary and grammar, while the Shakespearean text involves complex sentence structures and early modern English vocabulary
   - **Implications: This highlights the importance of training on a representative dataset when evaluating performance on out-of-domain tasks**
3. Comparing to baseline model (Dummy - Random Prediction)
   - For a vocabulary size of 65: Baseline Loss = -log (1/65) = log (65) = 1.812
   - The test loss of **8.8884** is worse than a random baseline. This suggests

the model's predictions are effectively random because it cannot leverage the patterns learned from the training data

4. Comparing between two test losses
   - Child Speech Test Loss (0.3602); Shakespearean Test Loss (8.8884)
   - The drastic difference between the two test losses highlights the **domain-specificity** of the trained model. It performs well only within the scope of its training data and faces difficulty with unfamiliar datasets
5. Interpretation of the Results
   - **The model’s performance is highly domain-dependent**
   - **Training a model on a narrow dataset makes it unsuitable for broader tasks**
6. How could this pipeline be used in practice
   - Domain Specific Applications: The pipeline is effective for tasks where the domain consistent with the training data
   - Transfer Learning: To handle diverse tasks like Shakespearean text generation, the pipeline could be extended with transfer learning: Pre-train the model on a large, diverse corpus to capture a broader understanding of language
   - Multi Models: Separate models could be trained for specific domains, and a classifier could be used to direct input data to the appropriate model