**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 seems to 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 containsapproximately200,000 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 count while retaining most core functionality. Lowering it to 160 would be a good compromise, as it will 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. Reducing it to 3 layers will cut down on parameters by about half, making the model shallower and faster without fully sacrificing depth.

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 parameter count without drastically limiting the attention mechanism's expressiveness, which is sufficient for a model aimed at sub-1M parameters.

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 and speed up training, which is particularly useful if the input\_childSpeech\_trainingSet.txt dataset contains shorter sequences.

Batch Size (batch\_size): Although not directly influencing model parameters, reducing it from 64 to 32 will speed up training and lower memory consumption. This is practical for limited compute environments.

Dropout (dropout): While it doesn’t impact parameter count, reducing it slightly (from 0.2 to 0.1) 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

**Validation Loss**: 0.3552 by the final step, indicating relatively stable performance across epochs.

**Overfitting**: The training and validation losses converge, suggesting minimal overfitting for this configuration.

**Output Quality**: Given the slightly larger embedding and layer sizes, this configuration would likely yield coherent outputs and retain some complexity in patterns, though it's relatively close to the 1M parameter limit.

**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

**Validation Loss**: 0.3612, slightly higher than Configuration 1, which suggests a small trade-off in performance due to the reduction in embedding dimension and block size.

**Overfitting**: The similar trends in training and validation losses indicate minimal overfitting, with the model still generalizing fairly well.

**Output Quality**: With fewer parameters, this configuration might produce simpler, more repetitive output, but it could still capture basic structural patterns in the data.

**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**: 0.3787, higher than the previous configurations but still within an acceptable range.

**Overfitting**: The overfitting seems low, with training and validation losses following similar trends than the above two.

**Output Quality**: With only 2 layers and a small block size, the model may produce simpler and potentially more repetitive output but still captures basic patterns.

**Question (i) d**

In transformer models, the inclusion of bias terms in self-attention layers can have subtle but important impacts on both the model’s learning dynamics and its final performance. Here’s an exploration of how bias terms affect transformer performance, particularly in the context of smaller models:

1. Impact on Learning and Convergence

* Bias in Linear Transformations: In self-attention, key, query, and value transformations rely on linear layers to project input data into different vector spaces. Including a bias term in these projections allows the model to learn a shift in distribution within each of these spaces. This can accelerate convergence by giving the model flexibility to adjust for features that may not align precisely with the origin in the projected space.
* Training Stability: Models with bias terms in attention layers can adapt more flexibly to different data distributions. This adaptability tends to enhance training stability, particularly in deeper transformer models, where layer-wise normalization may interact favorably with biases to stabilize gradients and reduce vanishing or exploding gradients.

2. Influence on Expressive Capacity

* Increased Expressiveness: Bias terms in self-attention can make the model more expressive, allowing it to capture complex relationships that might otherwise be missed. This is particularly useful in capturing low-frequency or subtle features within the input data, as the model can adjust for patterns that don’t strictly correlate with the mean values.
* Reduced Underfitting: By accommodating these shifts, bias terms help prevent underfitting. Models without bias may struggle to approximate data distributions accurately, particularly in applications involving rich, structured text, audio, or visual data.

3. Qualitative Impact on Generated Output

* Fine-Grained Pattern Recognition: In tasks requiring sequential or linguistic coherence, bias terms allow the model to distinguish subtle syntactic and semantic relationships. Without these biases, the output may appear more rigid or mechanical, potentially failing to capture nuanced relationships.
* Bias in Smaller Models: For configurations with limited parameters (e.g., smaller embedding sizes or fewer layers), bias terms can partially compensate for the reduced model complexity, making smaller transformers more robust and yielding higher-quality output.

Experimental Consideration

In downsized transformer models, including bias terms in the key, query, and value linear layers typically provides a noticeable improvement in validation loss and generalization. Therefore, bias terms are generally recommended, especially in small-scale transformers, to support both convergence and expressiveness without significantly increasing the parameter count.

**Question (i) e**

1. Impact on Training Stability and Gradient Flow

* Gradient Flow Improvement: Skip connections help mitigate the vanishing and exploding gradient problems, which are common in deep networks. By providing a direct path for gradients to flow backward during training, skip connections prevent gradients from diminishing or magnifying excessively across layers, making it easier to train deep architectures.
* Stabilized Convergence: With skip connections, the model typically converges faster and more reliably. These connections allow the model to retain input information at each layer, minimizing disruptions from randomly initialized weights and resulting in smoother loss descent, especially in the early stages of training.

2. Enhanced Representation Learning

* Preservation of Input Information: Skip connections allow each layer to access both the input and the transformed representation. This enables each layer to learn representations that build upon previous layers while retaining the original information. As a result, the model can maintain a more balanced representation that combines high-level and low-level information, contributing to a richer and more nuanced output.
* Reduced Information Loss: Without skip connections, each layer’s output would rely solely on transformations applied by that layer, risking the loss of important information over time. Skip connections address this by keeping a residual of the input information, allowing deeper layers to learn refinements rather than starting from scratch with each layer.

3. Effect on Model Depth and Complexity

* Facilitates Increased Depth: Skip connections make it feasible to add more layers without facing severe degradation in model performance, as is common in very deep architectures. They provide a “shortcut” for information to bypass unnecessary transformations, enabling transformers to effectively scale in depth while retaining generalization.
* Balanced Complexity: The residual connections make each layer operate more effectively, even with reduced dimensionality. This is particularly valuable in smaller, downsized models, where skip connections allow the model to be deep enough to capture complex relationships but without the excessive parameters required for each layer to handle the entire data structure independently.

4. Qualitative Impact on Generated Output

* Smoother Outputs: In natural language processing and other sequential tasks, skip connections contribute to more coherent and fluent generated text. They help avoid issues where the output might degrade over long sequences, as each layer retains access to the input sequence's structure.
* Enhanced Pattern Recognition: Skip connections reinforce the model’s ability to recognize and synthesize patterns across long contexts, improving its performance on tasks with dependencies across various positions. This results in output that is more contextually aware and coherent.

5. Experimental Impact on Loss and Overfitting

* Lower Validation Loss: By improving gradient flow and representation retention, skip connections often lead to a lower validation loss, as the model is better able to generalize to unseen data. This helps reduce overfitting, as each layer is less likely to over-specialize, focusing instead on building progressively refined representations.
* Reduced Overfitting in Small Models: In configurations with limited parameters, skip connections are especially helpful for maintaining model accuracy without relying on additional parameters. They allow smaller models to approximate deeper model behaviors, improving generalization without significantly increasing parameter count or complexity.