ANLP-A1 Report

Saketh Reddy Vemula
2022114014
CLD -UG3

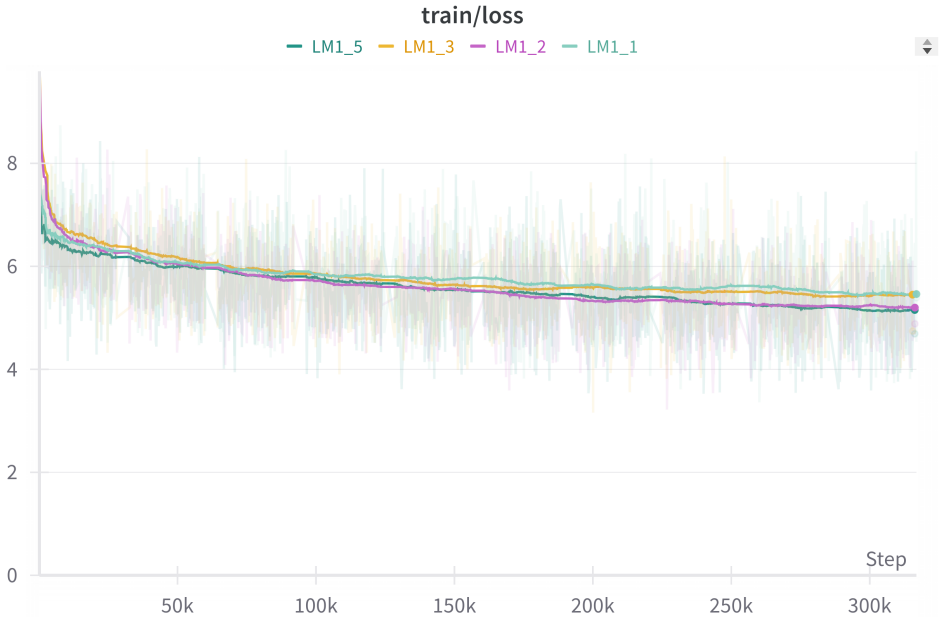
Neural-Network Language Model

Training Statistics: (except LM1_4)

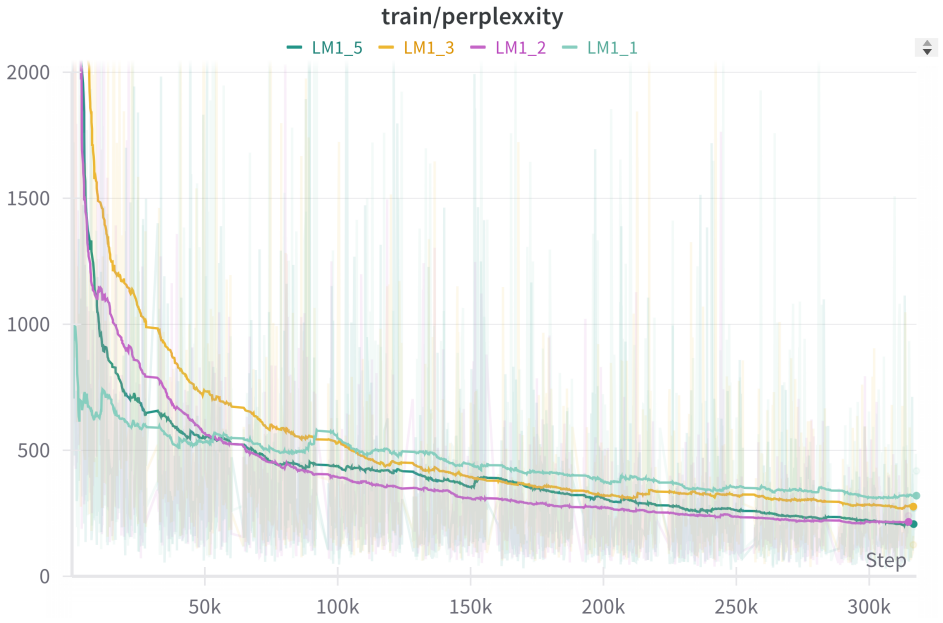
1. `train/avg_loss`:

2. `train/avg_perplexity`:

3. `train/loss`:

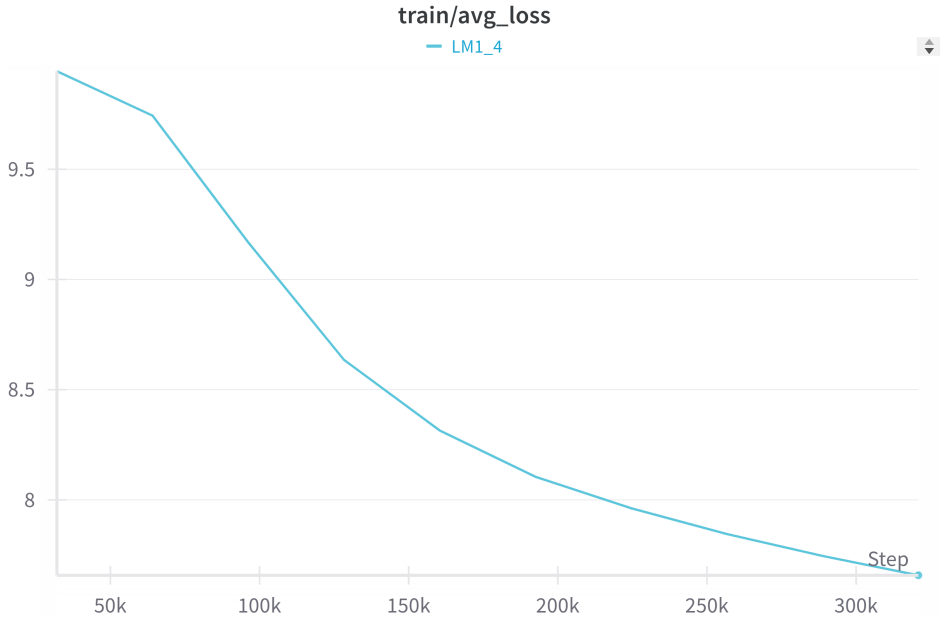


4. `train/perplexity`:



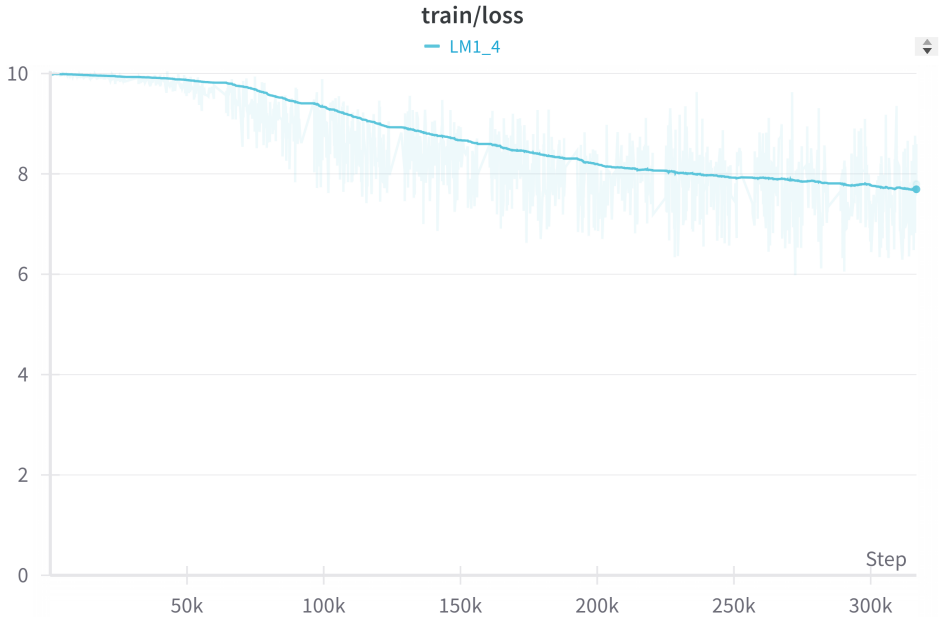
Training Statistics (for LM1_4)

1. `train/avg_loss`:

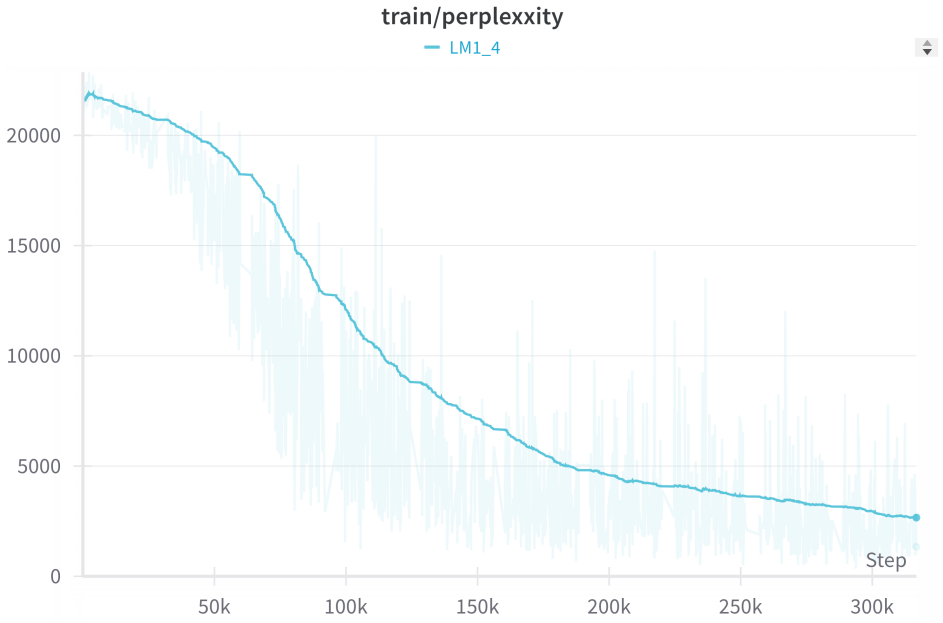


2. `train/avg_perplexity`:

3. `train/loss`:

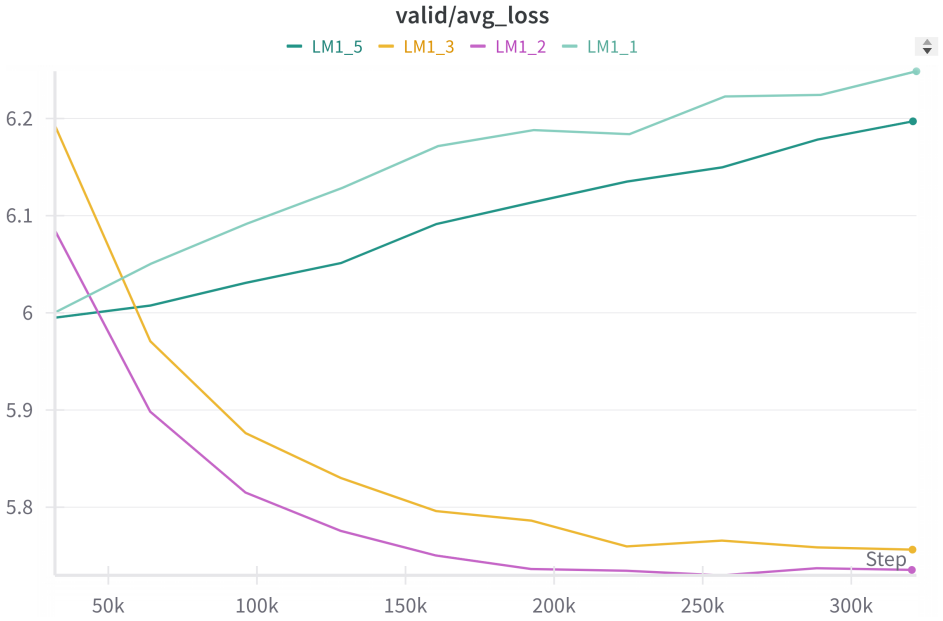


4. `train/perplexity`:

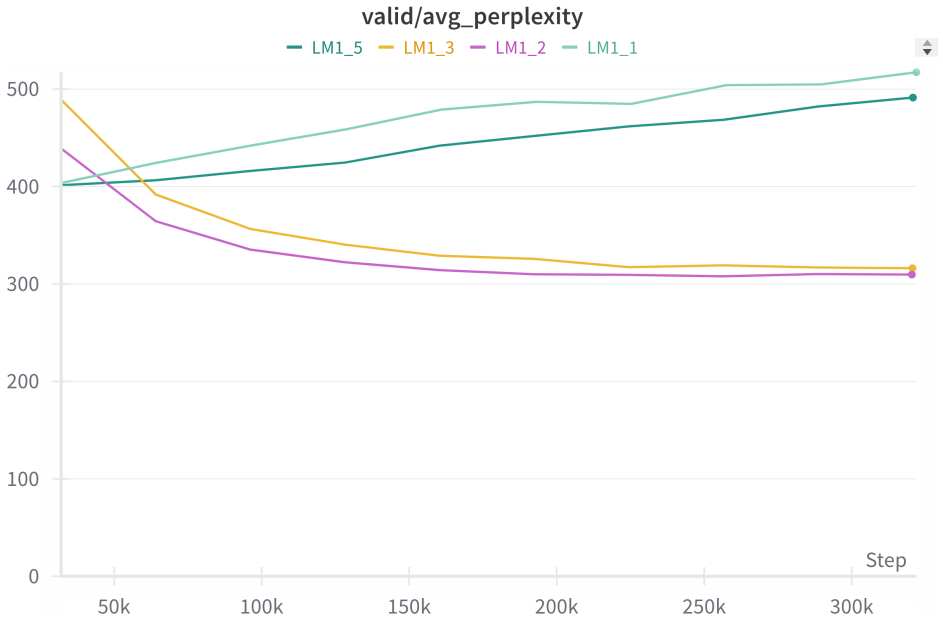


Validation Statistics

1. `valid/avg_loss`:

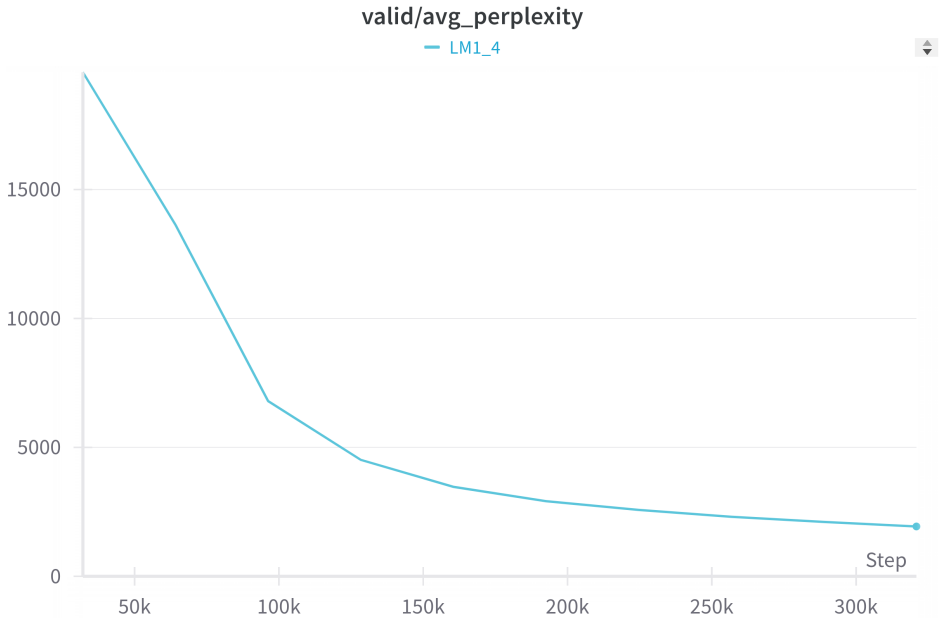


2. `valid/avg_perplexity`:



3. `valid_avg_loss` (LM1_4):

4. `valid_avg_perplexity` (LM1_4):



Hyperparameters:

LM vs Hyperparameters	batch_size	dropout_rate	embedding_dim	hidden_dim	learning_rate	num_epochs	optimizer	Schedu
LM1_1	32	0.2	300	128	5e-4	10	AdamW	-
LM1_2	32	0.3	300	128	5e-5	10	AdamW	-
LM1_3	32	0.3	300	64	5e-5	10	AdamW	-
LM1_4	32	0.3	300	128	5e-5	10	SGD	-
LM1_5	32	0.3	300	128	5e-4	10	AdamW	Expon

Name	train/avg_loss	train/avg_perplexity	valid/avg_loss	valid/avg_perplexity
LM1_1	5.415247284585	224.808166503906	6.24861893459051	517.297973632813

Name	train/avg_loss	train/avg_perplexity	valid/avg_loss	valid/avg_perplexity
LM1_2	5.1767177313546	177.100570678711	5.73543207796746	309.646759033203
LM1_3	5.36345302677497	213.460739135742	5.75630713374482	316.178558349609
LM1_4	7.65807013313926	2117.66650390625	7.56686441840309	1933.06958007813
LM1_5	5.02553572465951	152.25178527832	6.19708678387031	491.315673828125

Analysis:

Based on the results, and the corresponding hyperparameters, the performance can be analysed as:

1. LM1_1: Middle-range among LM1_x models.
2. LM1_2: Best among LM1_x modes
3. LM1_3: Second-best LM1_x models
4. LM1_4: Significantly worst among all models
5. LM1_5: Middle-range, slightly better than LM1_1

Key observations:

1. Dropout Rate:
   - LM1_1 uses 0.2, while others use 0.3
   - The higher dropout rate (0.3) generally performs better, suggesting it helps prevent overfitting in these feed-forward models for language modeling.
2. Learning Rate:
   - LM1_2 and LM1_3 with 0.00005 learning rate perform best
   - Higher learning rates (0.0005 in LM1_1 and LM1_5) lead to slightly worse performance
   - This suggests that for feed-forward networks in language modeling, a lower learning rate allows for more precise weight updates and better convergence.
3. Hidden Dimension:
   - LM1_3 uses a smaller hidden dim (64) compared to others (128)
   - Despite this, it performs second-best, indicating that for this task, 64 hidden units might be sufficient to capture the necessary language patterns. But to be safe we can use 128 dimensions if want to train for more complex data later.
4. Optimizer:
   - LM1_4 uses SGD, while others use AdamW
   - The dramatic underperformance of LM1_4 highlights the importance of adaptive optimizers like AdamW for language modeling tasks.
5. Learning Rate Scheduler:
   - LM1_5 uses ExponentialLR scheduler
   - Its performance is middling, suggesting that for these short training runs (10 epochs), a well-chosen fixed learning rate might be more effective than a scheduler.
6. Overall Performance Trend:
   - The best models (LM1_2 and LM1_3) achieve perplexities around 310-316
   - This is significantly higher than Transformer (LM3_1) models, indicating the limitations of feed-forward networks for language modeling

Possible reasons for the above trends:

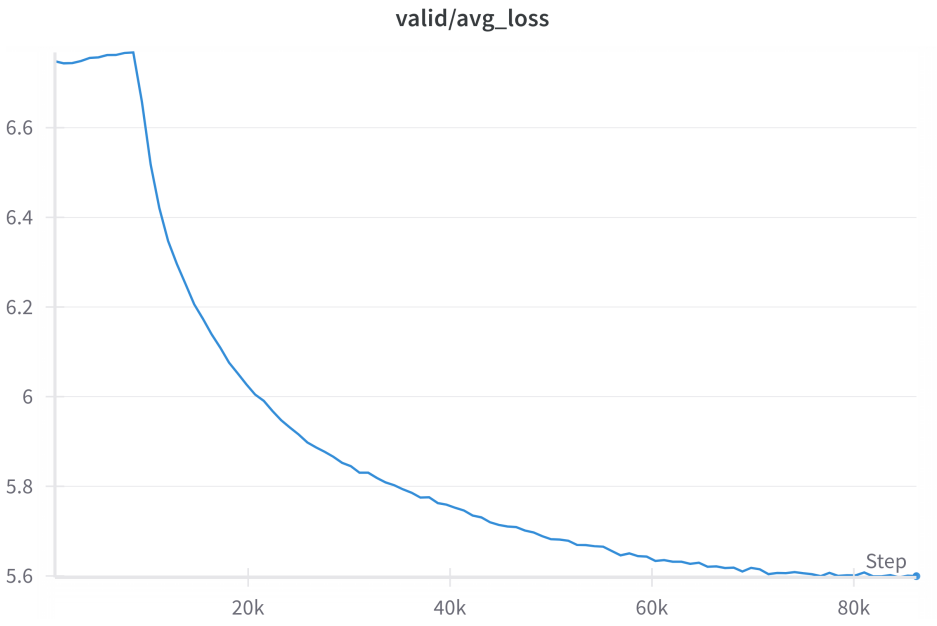
1. Limited Sequential Information:
   Feed-forward networks struggle with language modeling because they can't easily capture sequential dependencies. Each input is processed independently, making it difficult to model context in language.
2. Importance of Regularization:
   The better performance with higher dropout (0.3) suggests these models are prone to overfitting. Language has complex patterns, and without proper regularization, the model might memorize training sequences rather than learning generalizable patterns.
3. Sensitivity to Learning Rate:
   The superior performance of models with lower learning rates indicates that small, careful updates are crucial. Language modeling often involves a complex loss landscape, and larger learning rates might cause the optimizer to overshoot optimal solutions.
4. Optimizer Impact:
   The poor performance of SGD compared to AdamW highlights the need for adaptive learning rates in language modeling. AdamW can adjust learning rates for each parameter, which is beneficial given the varying importance of different words and patterns in language.

5. Model Capacity:
   The similar performance between 64 and 128 hidden units suggests that for these feed-forward models, increasing capacity beyond a certain point doesn't significantly improve language modeling capability. This could indicate that the model's architecture, rather than its size, is the limiting factor.

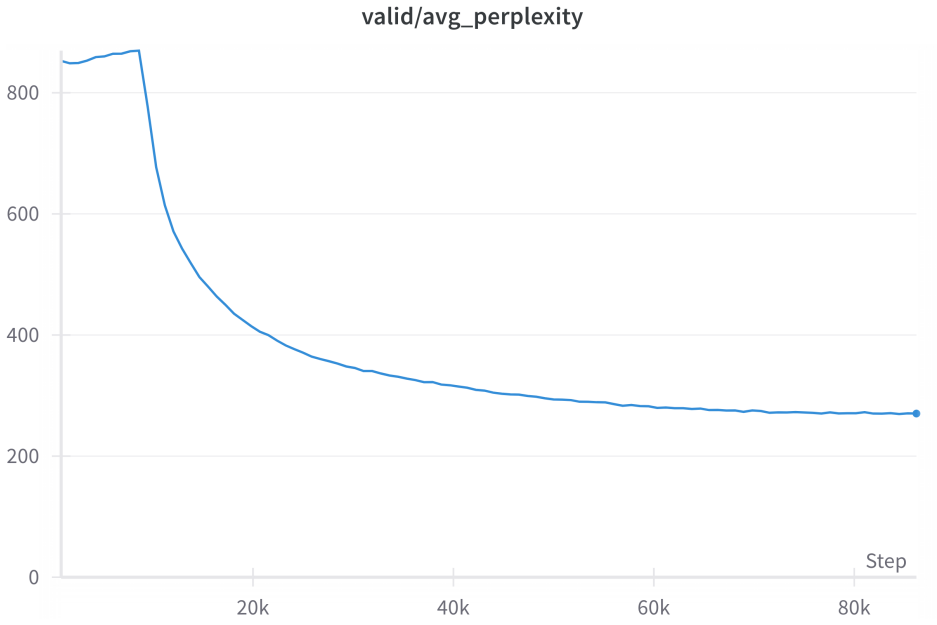
RNN-based Language Model

Training Statistics:

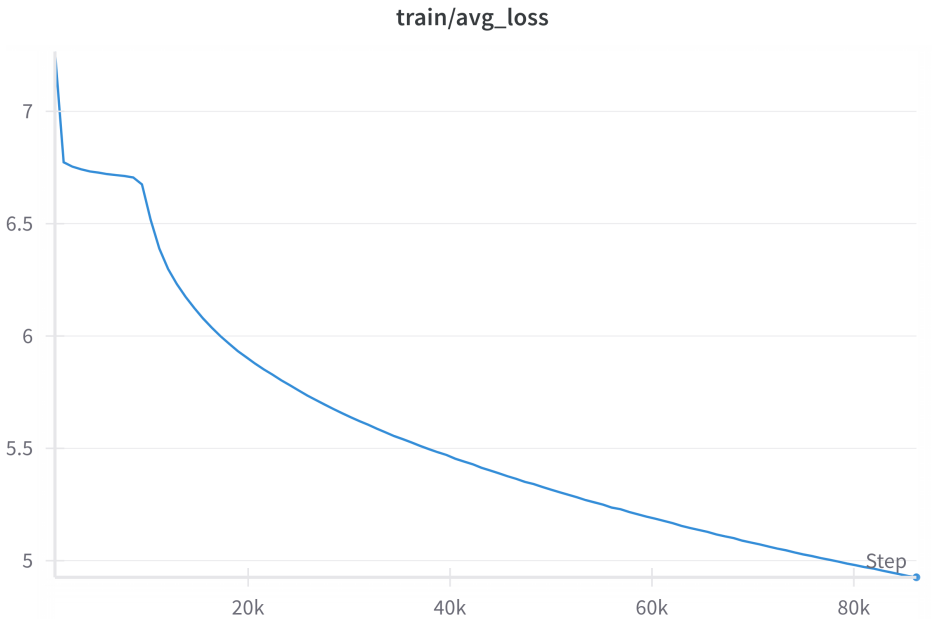
1. `valid/avg_loss`:



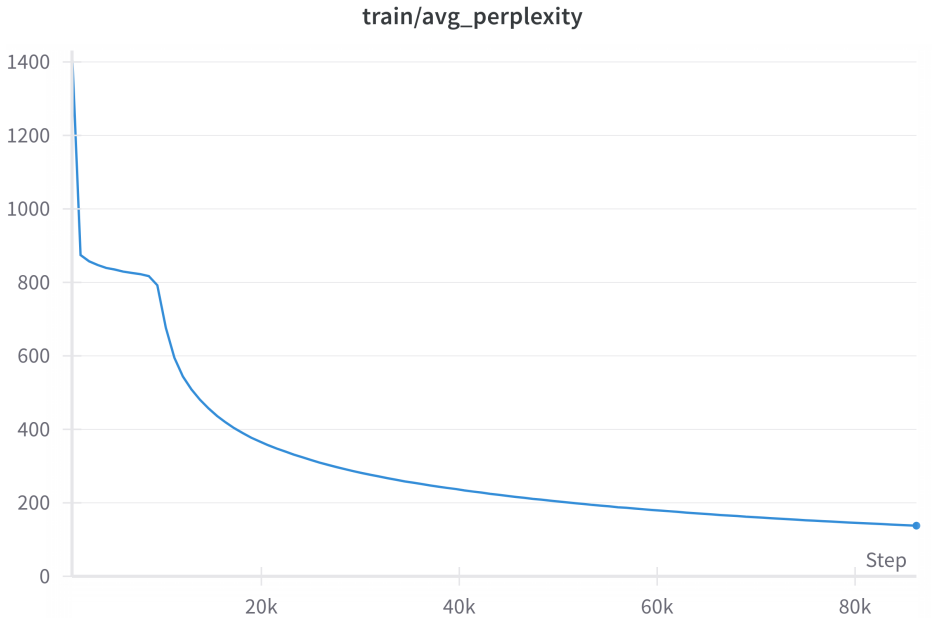
2. `valid/avg_perplexity`:



3. `train/avg_loss`:



4. `train/avg_perplexity`:



Hyperparameters:

LM vs Hyperparameters	batch_size	dropout_rate	embedding_dim	hidden_dim	learning_rate	num_epochs	optimizer
LM2_2	32	0.3	300	300	5e-5	100	AdamW

Name	train/avg_loss	train/avg_perplexity	valid/avg_loss	valid/avg_perplexity
LM2_2	4.926058325906648	137.8351287841797	5.599474633181536	270.2843322753906

Analysis:

1. Performance:
   - `LM2_2` (LSTM) achieved a validation perplexity of 270.28, which is better than most `LM1_x` models.
   - The best performing `LM1_x` model is `LM1_2`, with a validation perplexity of 309.65.

2. Training vs. Validation Gap:
    - `LM2_2` shows a larger gap between training (137.84) and validation (270.28) perplexity compared to most `LM1_x` models.
    - This suggests that the LSTM model might be overfitting more than the Feed Forward models.
3. Training Duration:
    - `LM2_2` was trained for 100 epochs, while `LM1_x` models were trained for only 10 epochs.
    - Despite the longer training, `LM2_2` still outperforms the `LM1_x` models, indicating better learning capacity. But in order to outperform `LM1_x` models, LSTM based models need to be trained for longer periods.
4. Learning Rate:
    - `LM2_2` uses a lower learning rate (0.00005) compared to some `LM1_x` models (e.g., `LM1_1` and `LM1_5` use 0.0005).
    - This lower learning rate is happened to be necessary to the LSTM model's stability and better performance.
5. Model Complexity:
    - `LM2_2` has 5 layers
    - The LSTM's recurrent nature allows it to capture longer-term dependencies compared to the 5-gram Feed Forward models.

Explanations for the observed trends:

1. Better Performance of LSTM:
    - LSTMs are designed to capture sequential information and long-term dependencies, which is crucial for language modeling. This explains why `LM2_2` outperforms the Feed Forward models, which are limited to a fixed context window (5-gram).
2. Overfitting in LSTM:
    - The larger gap between training and validation perplexity in `LM2_2` suggests that LSTMs, being more complex, are more prone to overfitting. This is especially true given the longer training duration (100 epochs vs. 10 epochs) in order to outperform Feed Forward Networks.
3. Impact of Training Duration:
    - The extended training of `LM2_2` allows it to learn more nuanced patterns in the data, contributing to its superior performance. However, this also increases the risk of overfitting.
4. Learning Rate Influence:
    - The lower learning rate used in `LM2_2` allows for more stable training, especially given the longer training duration. This helps in finding a better local minimum in the loss landscape.
5. Model Complexity and Generalization:
    - While `LM2_2` shows better performance, the larger gap between training and validation metrics suggests that simpler models like `LM1_2` might generalize better to unseen data, despite having slightly higher perplexity.

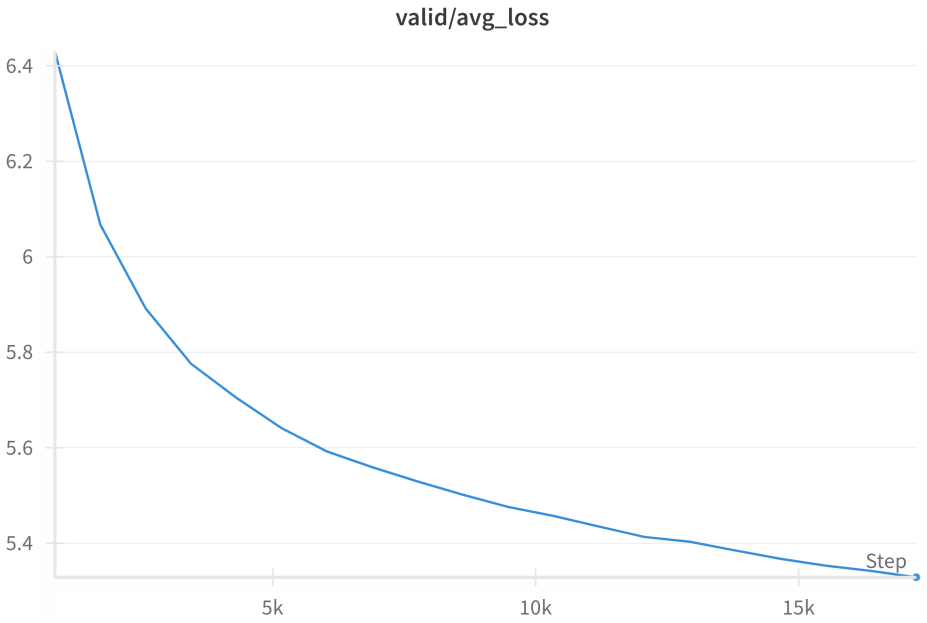
Transformer Decoder Language Model

LM vs Hyperparameters	batch_size	dropout_rate	embedding_dim	hidden_dim	learning_rate	num_epochs	optimizer
LM2_2	32	0.3	300	300	1e-4	40	AdamW

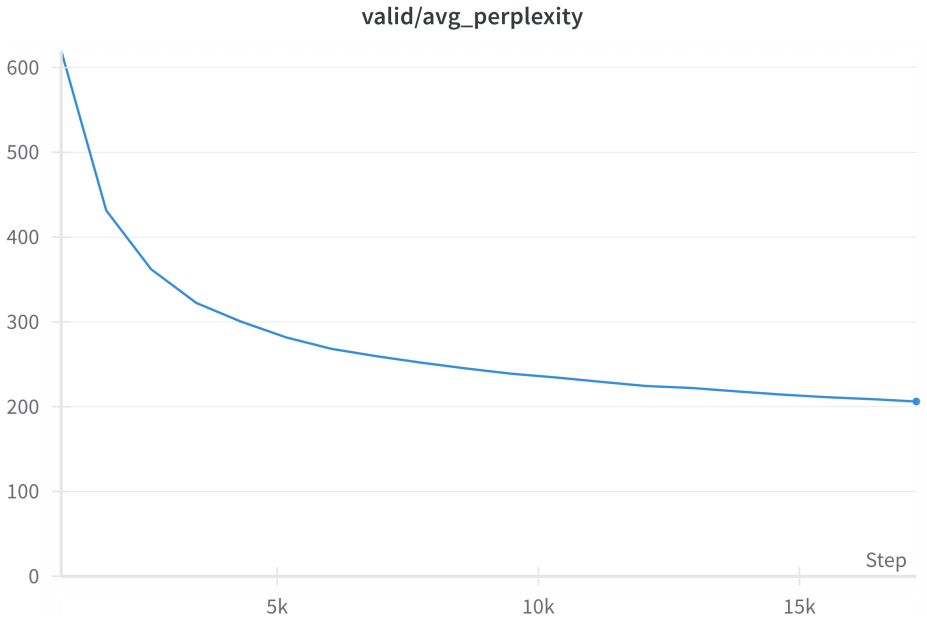
Name	train/avg_loss	train/avg_perplexity	valid/avg_loss	valid/avg_perplexity
LM3_1	5.03084700113899	153.062606811523	5.32861080876103	206.151412963867

Statistics:

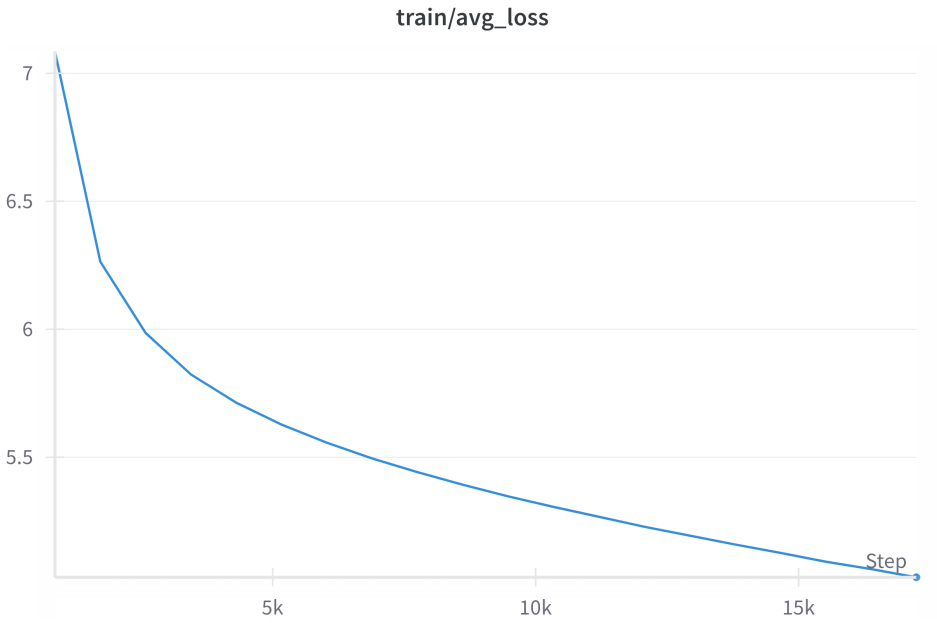
1. `valid/avg_loss`:



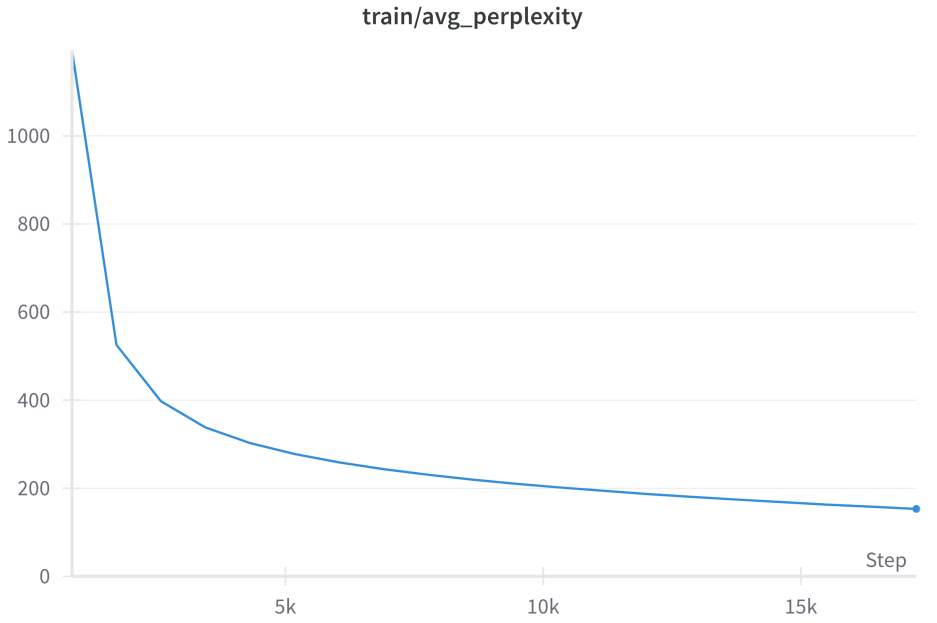
2. `valid/avg_perplexity`:



3. `train/avg_loss`:



4. `train/avg_perplexity`:



Analysis:

1. Performance:
   - `LM3_1` (Transformer) achieved the best validation perplexity of 206.15 among all models.
   - This is significantly better than `LM2_2` (LSTM) at 270.28 and the best `LM1_x` model (`LM1_2`) at 309.65.
2. Training vs. Validation Gap:
   - `LM3_1` shows a smaller gap between training (153.06) and validation (206.15) perplexity compared to `LM2_2`.
   - This suggests better generalization than the LSTM model.
3. Training Duration:
   - `LM3_1` was trained for 20 epochs, compared to 100 for `LM2_2` and 10 for `LM1_x` models.

- It achieved the best performance with fewer epochs than `LM2_2`, indicating faster convergence.

4. Model Architecture:
   - `LM3_1` uses a Transformer architecture with 6 decoder layers and 5 attention heads.
   - It has a larger dimension for the feed-forward network (1028) compared to its hidden dimension (128).
5. Sequence Length:
   - `LM3_1` uses a max sequence length of 64, while `LM2_2` uses 128.
   - Despite the shorter sequence length, it still outperforms other models.
6. Learning Rate:
   - `LM3_1` uses the same learning rate (0.00005) as `LM2_2`, which is lower than some `LM1_x` models.

Explanations for the observed trends:

1. Superior Performance of Transformer:
   - Transformers excel at capturing both long-range and local dependencies through self-attention mechanisms. This explains why `LM3_1` outperforms both LSTM and Feed Forward models.
   - The multi-head attention allows the model to focus on different aspects of the input simultaneously, leading to better language understanding.
2. Better Generalization:
   - The smaller gap between training and validation perplexity in `LM3_1` suggests that it generalizes better than the LSTM model. This could be due to the self-attention mechanism being less prone to overfitting compared to recurrent architectures.
3. Efficient Learning:
   - `LM3_1` achieves the best performance with fewer epochs than `LM2_2`, demonstrating the Transformer's ability to learn more efficiently. This is likely due to the parallel nature of self-attention, allowing for better gradient flow during training.
4. Impact of Architecture Choices:
   - The large dimension of the feed-forward network (1028) compared to the hidden dimension (128) allows for more complex transformations of the data, potentially contributing to its performance.
   - The use of multiple decoder layers (6) and attention heads (5) allows the model to capture intricate patterns in the language data.
5. Sequence Length Considerations:
   - Despite using a shorter max sequence length (64) compared to `LM2_2` (128), `LM3_1` still outperforms it. This suggests that the Transformer's self-attention mechanism is more effective at capturing relevant information within shorter sequences.

Conclusions:

1. Architecture Superiority:
   The Transformer-based model (
   `LM3_1`) demonstrates superior performance in language modeling compared to both LSTM (`LM2_2`) and Feed Forward (`LM1_x`) models. This aligns with the general trend in NLP where Transformers have become the go-to architecture for many tasks.
2. Efficiency:

   `LM3_1` achieves better results with fewer training epochs, indicating that Transformers can learn more efficiently than LSTMs or Feed Forward networks for this task.
3. Generalization:
   The Transformer model shows better generalization, with a smaller gap between training and validation perplexity. This suggests it might perform better on unseen data.
4. Scalability:
   The performance of
   `LM3_1` hints at the potential for further improvements by scaling up the model (e.g., increasing layers, heads, or dimensions), which is a known strength of Transformer architectures.
5. Trade-offs:
   While
   `LM3_1` performs best, it's important to consider the increased computational complexity of Transformers. For simpler tasks or resource-constrained environments, simpler models like `LM1_x` might still be preferable.ss

Overall Trend Analysis:

1. Model Performance Ranking:
   Transformer > LSTM > Feed Forward

Explanation: This aligns with the evolution of NLP architectures, showcasing the increasing ability to capture complex language patterns.

2. Generalization Capability:
   Transformer > Feed Forward > LSTM
   Explanation: Transformer's self-attention mechanism allows for better generalization, while LSTM shows signs of overfitting.
3. Training Efficiency:
   Transformer > LSTM > Feed Forward
   Explanation: Transformer achieves superior results in fewer epochs, demonstrating faster convergence. LSTM takes longer time to process, but overall still better than 5-gram Feed Forward Models.
4. Complexity vs Performance Trade-off:
   More complex models (Transformer, LSTM) outperform simpler ones (Feed Forward) but require more computational resources.
   Explanation: Advanced architectures capture more nuanced language patterns at the cost of increased complexity.
5. Hyperparameter Sensitivity:
   Lower learning rates generally led to better and more importantly stable performance across models.
   Explanation: Slower learning allows for more stable optimization, especially in complex architectures.
6. Sequence Length Impact:
   Transformer excels even with shorter sequences, while LSTM benefits from longer ones.
   Explanation: Self-attention mechanisms efficiently capture relationships regardless of sequence length.
7. Potential for Improvement:
   Transformer shows the most promise for further enhancements through scaling.
   Explanation: The architecture's flexibility allows for easy scaling of model size and complexity.