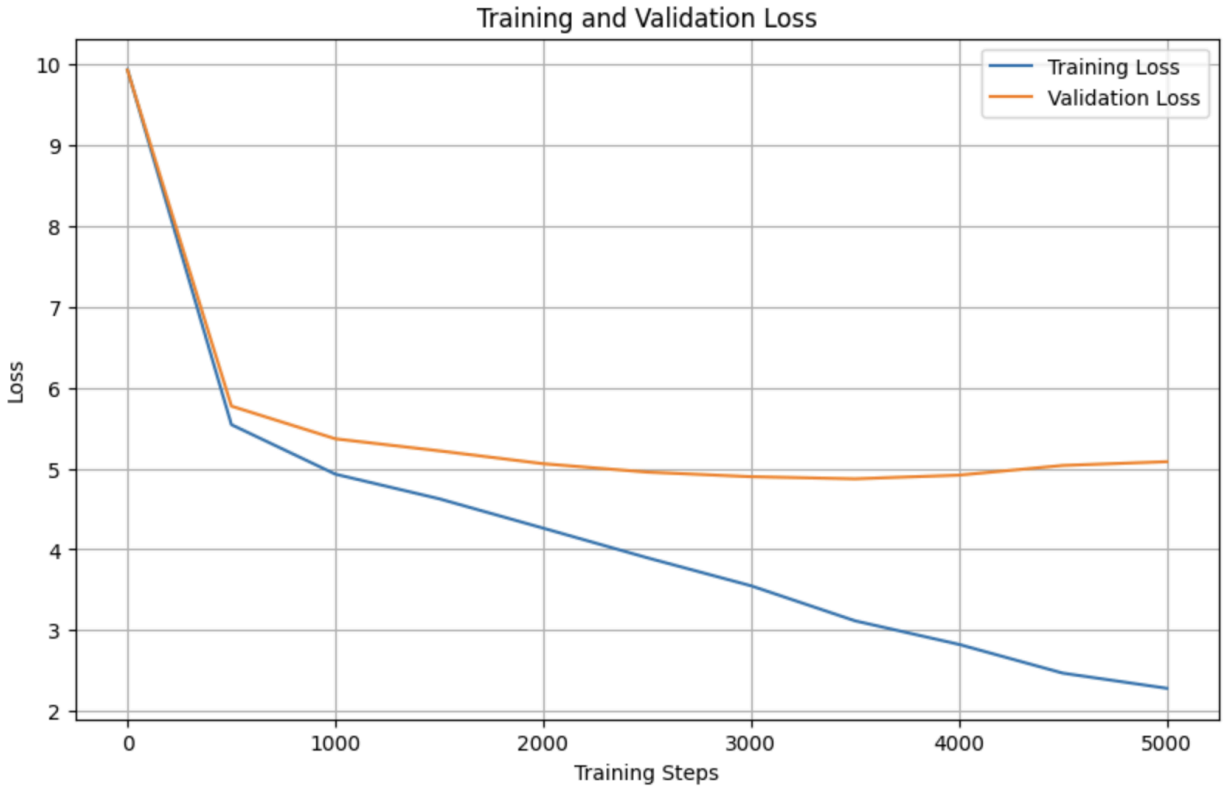
DS5983 PA3

Yunyu Guo

```
Using device: cuda
Loading dataset from local parquet files...
Train texts: 23767
Validation texts: 2461
Test texts: 2891
Training BPE tokenizer...
Tokenizer trained with vocab size: 18239
Preparing training data...
Tokenizing texts...
100%|██████████| 23767/23767 [00:04<00:00, 5150.50it/s]
Tokenizing texts...
100%|██████████| 2461/2461 [00:00<00:00, 6330.44it/s]
Tokenizing texts...
100%|██████████| 2891/2891 [00:00<00:00, 6910.16it/s]
Train sequences: 20433
Val sequences: 2122
Test sequences: 2427
Model parameters: 99.16M
Starting training...
step 0: train loss 9.9312, val loss 9.9247, lr 0.00e+00
step 500: train loss 5.5423, val loss 5.7712, lr 6.00e-04
step 1000: train loss 4.9302, val loss 5.3672, lr 5.84e-04
step 1500: train loss 4.6256, val loss 5.2183, lr 5.37e-04
step 2000: train loss 4.2627, val loss 5.0593, lr 4.65e-04
step 2500: train loss 3.8971, val loss 4.9553, lr 3.77e-04
step 3000: train loss 3.5502, val loss 4.8989, lr 2.83e-04
step 3500: train loss 3.1174, val loss 4.8724, lr 1.95e-04
step 4000: train loss 2.8262, val loss 4.9180, lr 1.23e-04
step 4500: train loss 2.4706, val loss 5.0378, lr 7.63e-05
step 4999: train loss 2.2834, val loss 5.0843, lr 6.00e-05

Final Evaluation:
Test Perplexity: 176.31
```




```
Training completed!
Best validation loss: 4.8724
Test perplexity: 176.31
```

Training and Validation Loss

Loss values (e.g., train loss 2.28, val loss 5.08) are decreasing which means the model is learning.

Validation loss higher than training loss is normal, but a large gap may indicate overfitting.

The validation loss starts increasing at step 4000 while training loss keeps decreasing might be due to model overfitting to the training data.

Overfitting: The model is learning the training data very well (train loss drops), but it is not generalizing to unseen data (validation loss rises).

Learning rate decay: As the learning rate gets smaller, the model may "settle" into a solution that fits the training data too closely.

Insufficient regularization: Dropout or weight decay may not be strong enough to prevent overfitting.

Training too long: The model starts to memorize the training set rather than learning general patterns.

How to address this?

Early stopping: Stop training when validation loss starts to rise.

Increase regularization: Use higher dropout or weight decay.

Use more data: More training data helps prevent overfitting.

Tune model size: A smaller model may generalize better if data is limited.

2. Test Perplexity

test perplexity is 176.31, which is quite high (likely due to limited data, small model, or short training).

```
Generating text samples...

Greedy: The quick brown

Top-k (k=50): The quick brown M iam i i ' s long , was soft in a bit of a fourth inches in relation to the general public with M iam i ' s head , while M iam i allowed th

Nucleus (p=0.9): The quick brown G iac a N i ina e B an ai does not B an ai ( Ne j ah ) — 5 ; The te ( k ' an i B an ai — 4 : 38 ) — 12 ;

Analyzing sampling strategies...

GREEDY — Prompt: 'The quick brown'
Diversity: 0.20 (1/5 unique)
  Sample 1: The quick brown
  Sample 2: The quick brown
  Sample 3: The quick brown

TOP_K — Prompt: 'The quick brown'
Diversity: 1.00 (5/5 unique)
  Sample 1: The quick brown M iam i ' s tail ed jack r ab bit ( L ep us cre am fer i ) The ch ips ( L ud ac
  Sample 2: The quick brown The underside of a star , cre am y ne ( s ax i ) . The underside of the forest is mostly mostly on the ground
  Sample 3: The quick brown S imp son h ur r ican e would possibly pers ist for the entire storm ; they were last described on January 1 . G rah am struck

NUCLEUS — Prompt: 'The quick brown'
Diversity: 1.00 (5/5 unique)
  Sample 1: The quick brown The black @-@ tail ed jack r ab bit is the black @-@ tail ed and white in black . It has black @-@ tail ed jack r ab
  Sample 2: The quick brown The black @-@ tail ed jack r ab bit , the black @-@ tail ed jack r ab bit ( Su gar ) , or white @-@ tail ed
  Sample 3: The quick brown The loud shr ink s down the ground , or those of all races , in the form of a long hair on the ground , a dull est

GREEDY — Prompt: 'In the future'
Diversity: 0.20 (1/5 unique)
  Sample 1: In the future
  Sample 2: In the future
  Sample 3: In the future

TOP_K — Prompt: 'In the future'
Diversity: 1.00 (5/5 unique)
  Sample 1: In the future = = = Critical reception = = On the morning of April 10 , 2008 , the episode was the final episode of the first two episodes
  Sample 2: In the future = = = Adaptations = = = The first , Guitar H ero and H ero on May 2 were released in September 2008 . The series
  Sample 3: In the future The series was nominated for the 2014 Game of the Year . Among the songs , the first , and the second only soundtrack in a single print edition

NUCLEUS — Prompt: 'In the future'
Diversity: 1.00 (5/5 unique)
  Sample 1: In the future In 2000 , Fey depicted in the comedy series Flash back , called Fey " the funniest characters of Fey ' s comedy , comedy @-@ drama Weekend Updat
  Sample 2: In the future = = = Career = = = = = top @-@ scorer and financial statistics = = = = = = March = = =
  Sample 3: In the future = = = = = General election = = = = In the same year , the House of Common s had a long time coming to


GREEDY — Prompt: 'Once upon a time'
Diversity: 0.20 (1/5 unique)
  Sample 1: Once upon a time
  Sample 2: Once upon a time
  Sample 3: Once upon a time

TOP_K — Prompt: 'Once upon a time'
Diversity: 1.00 (5/5 unique)
  Sample 1: Once upon a time And so , they are the greatest thing ever came to sit . But I could like , I felt that I wanted to do it . That was
  Sample 2: Once upon a time The French and British forces in the early morning , the British left flank of the Royal Navy was led by Commodore Eric Black stone . However
  Sample 3: Once upon a time The S ister W ives of the S K S were character ized in The Original Series episode " The A . C . S . " The "

NUCLEUS — Prompt: 'Once upon a time'
Diversity: 1.00 (5/5 unique)
  Sample 1: Once upon a time Meanwhile , Ell en attempted to qu it his shirt to the hospital instead of driving by a depart ment . In this up the hospital , he reveals
  Sample 2: Once upon a time = = Production = = = = = Critical reception = = = C reat ive Director , Anderson has said that he was poorly
  Sample 3: Once upon a time Char ge died in The In stit ute of Man it ob a ' s documentary , Wal sh and ido Po h l earned a further w ology
```

3. Text Generation Quality

Greedy decoding: Always picks the most likely next token.

Top-k and Nucleus sampling: Introduce randomness, allowing more diverse and creative outputs.

Observation:
For all prompts, greedy decoding just repeats the prompt and stops (e.g., "The quick brown").

Reason:
The model likely predicts the end-of-sequence token (<|eos|>) or a padding token as the most probable next token right after the prompt. Greedy decoding always picks this, so generation stops immediately.

Why does this happen?

The model is undertrained or not yet good at language modeling.

The dataset or tokenizer may be causing the model to see <|eos|> too often.

Greedy decoding is very conservative and will always pick the "safest" option.

Why Do Top-k and Nucleus Sampling Work Better?

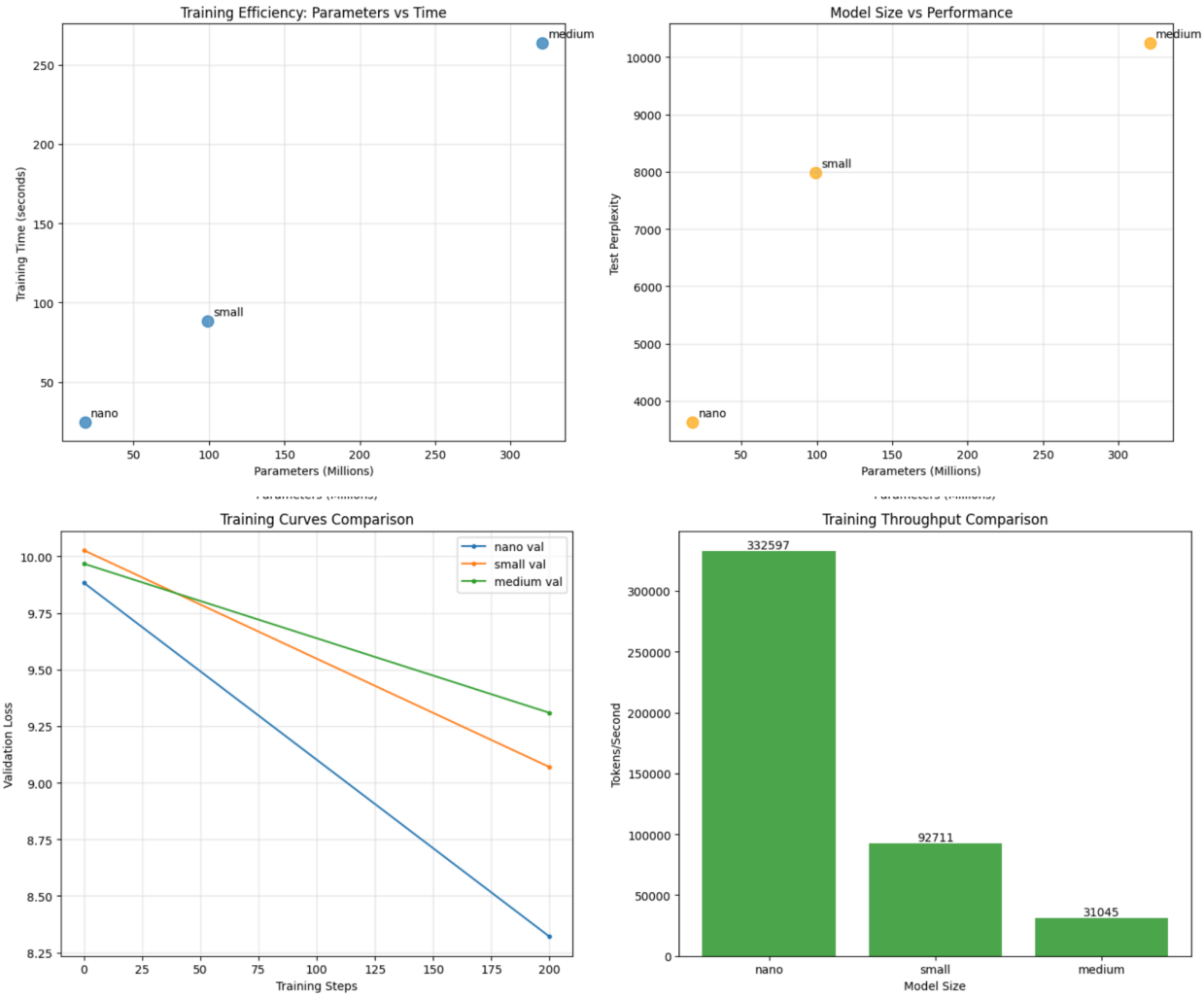
They introduce randomness and allow the model to pick less probable tokens, so generation continues even if <|eos|> is likely.

This leads to more diverse and longer outputs, even if the model is not perfect.

```
============================================================
STARTING MODEL SIZE COMPARISON
============================================================
Starting model size comparison experiment...
Loading dataset from local parquet files...
Train texts: 23767
Validation texts: 2461
Test texts: 2891
Training BPE tokenizer...
Tokenizer trained with vocab size: 18239

============================================================
Training GPT2_NANO
============================================================
Tokenizing texts...
100%|██████████| 500/500 [00:00<00:00, 5935.74it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 5281.04it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 5775.37it/s]
Model parameters: 17.70M
Starting training...
step 0: train loss 9.8852, val loss 9.8834, time 0.9s
step 199: train loss 3.2441, val loss 8.3204, time 24.5s

Evaluating gpt2_nano...

GPT2_NANO RESULTS:
Parameters: 17.70M
Training time: 24.6s
Time per iteration: 0.123s
Tokens/second: 332597
Final train loss: 3.2441
Final val loss: 8.3204
Test perplexity: 3629.37
Sample (greedy): The quick brown...
```

```
============================================================
Training GPT2_SMALL
============================================================
Tokenizing texts...
100%|██████████| 500/500 [00:00<00:00, 7706.77it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 6285.39it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 7518.69it/s]
Model parameters: 99.16M
Starting training...
step 0: train loss 10.0128, val loss 10.0271, time 3.2s
step 199: train loss 0.6128, val loss 9.0696, time 88.0s

Evaluating gpt2_small...

GPT2_SMALL RESULTS:
Parameters: 99.16M
Training time: 88.4s
Time per iteration: 0.442s
Tokens/second: 92711
Final train loss: 0.6128
Final val loss: 9.0696
Test perplexity: 7978.87
Sample (greedy): The quick brown position...
```

```
============================================================
Training GPT2_MEDIUM
============================================================
Tokenizing texts...
100%|██████████| 500/500 [00:00<00:00, 7979.09it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 5847.51it/s]
Tokenizing texts...
100%|██████████| 100/100 [00:00<00:00, 6933.42it/s]
Model parameters: 321.12M
Starting training...
step 0: train loss 9.9631, val loss 9.9679, time 10.3s
step 199: train loss 0.2421, val loss 9.3092, time 262.8s

Evaluating gpt2_medium...

GPT2_MEDIUM RESULTS:
Parameters: 321.12M
Training time: 263.9s
Time per iteration: 1.319s
Tokens/second: 31045
Final train loss: 0.2421
Final val loss: 9.3092
Test perplexity: 10252.42
Sample (greedy): The quick brown...
```




```
================================================================================
MODEL SIZE COMPARISON SUMMARY
================================================================================
Model        Params(M)  Time(s)    Perplexity   Val Loss   Tokens/s
--------------------------------------------------------------------------------
nano         17.7       24.6       3629.37      8.3204     332597
small        99.2       88.4       7978.87      9.0696     92711
medium       321.1      263.9      10252.42     9.3092     31045

================================================================================
KEY INSIGHTS:
• Best Perplexity: gpt2_nano (3629.37)
• Fastest Training: gpt2_nano (24.6s)
• Highest Throughput: gpt2_nano (332597 tokens/s)
```

4. Different Model sizes comparisons (GPT-nano, GPT-small, GPT-medium)

a. Training Efficiency

Training Time: As expected, larger models take much longer to train.

nano: 24.6s

small: 88.4s

medium: 263.9s

Tokens/Second: This shows how fast each model processes data. Smaller models are much more efficient.

nano: 332k tokens/sec (fastest)

medium: 31k tokens/sec (slowest)

Insight: You pay a significant computational cost for larger models.

b. Model Performance

Training Loss: Larger models achieve a much lower training loss.

nano: 3.24

small: 0.61

medium: 0.24 (lowest)

The larger models are better at memorizing the training data.

Validation Loss & Perplexity: Performance gets worse as the model gets bigger.

nano: Val Loss 8.32, Perplexity 3629 (Best)

small: Val Loss 9.06, Perplexity 7978 (Worse)

medium: Val Loss 9.30, Perplexity 10252 (Worst)

The larger models are failing to generalize to new, unseen data. They have overfit the training set.

Why is this Happening?

The compare_model_sizes() function uses a very small amount of data (train_texts[:500]) and a short training time (max_iters=200) for a quick comparison.

For gpt2-small and gpt2-medium, this is not nearly enough data or training. With so much capacity (99M and 321M parameters), they instantly memorize the tiny dataset and fail to learn any general language patterns.

The gpt2-nano model, being smaller, is less prone to extreme overfitting on this small dataset, which is why it performs the best in this specific test.

Summary Table

Model Size	Training Fit	Generalization (Perplexity)	Efficiency (Time)	Overall Result
Nano	Good	Best (3629)	Fastest	Best performing model for this short test.
Small	Very Good	Worse (7978)	Slower	Overfitting the small dataset.
Medium	Excellent	Worst (10252)	Slowest	Severely overfitting the small dataset.

Conclusion

Bigger is not always better, especially with limited data or training time.