



picolLM

Presented by: Aditeya Baral, Allen Ajith, Ayush Rajesh
Jhaveri, Kevin Mathew T, Tiasa Singha Roy

P A R T 01

Introduction

Premise

- Train and compare **three** types of Language Models:
 - LSTM
 - k-Gram MLP
 - Transformer
- Across three different **size variants**
 - Pico (3M)
 - Tiny (7M)
 - Small (30M)
 - Base (110M)

Experiment Setup - Dataset

- Dataset: **TinyStories**
 - Comprises 2M short sentences from children storybooks
- Dataset size chosen: **1M randomly sampled sentences**
 - Split into 3 sets
 - train (98%)
 - validation (1%)
 - test (1%)

Experiment Setup - Training

- All models were randomly initialized and trained using
 - **Epochs:** 5
 - **Optimizer:** AdamW
 - **Learning rate:** 3e-4
 - **LR Scheduler:** Cosine decay with linear warmup ratio = 0.1
 - **Batch size:** 64
- Experiment tracking and checkpointing
 - **Model checkpoints:** HuggingFace
 - **Logging:** WandB

Experiment Setup - Metrics

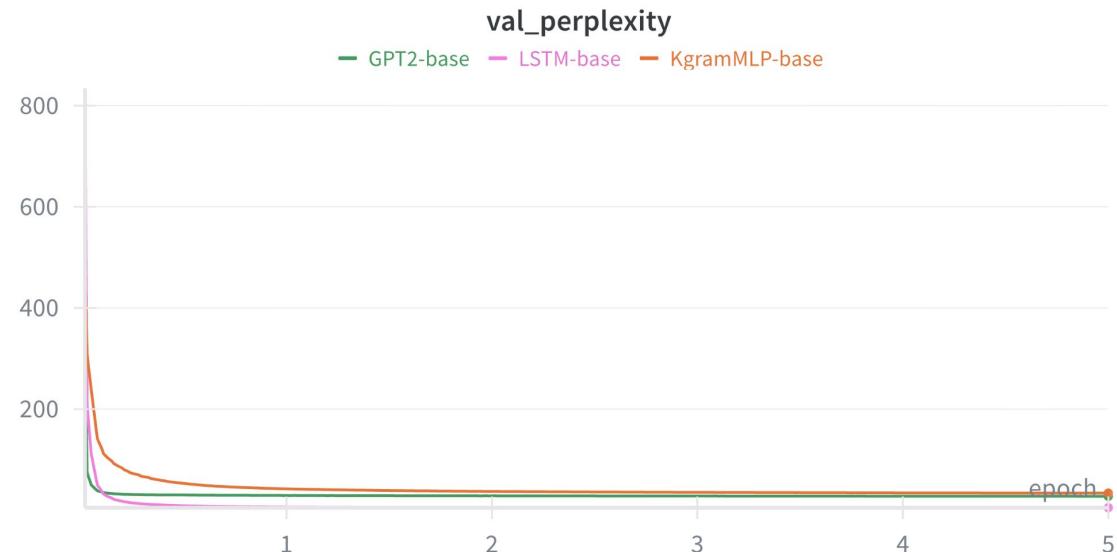
- **Training metrics**
 - Negative Log-Likelihood (NLL) Loss
 - Perplexity
 - Determines how well the model predicts the next sequence
- **Evaluation metrics**
 - NLL Loss
 - Perplexity
 - Distinct-N with $N = \{1, 2, 3, 4, 5\}$
 - Determines diversity in generated n-grams

P A R T 0 2

Models

Why scale down model size?

- Base (110M) models - validation loss flattens in <0.3 epochs
 - **model capacity far exceeds dataset complexity.**
- Downsize to smaller models
 - **hyperparameter changes produce meaningful differences in learning dynamics**



P A R T 2.1

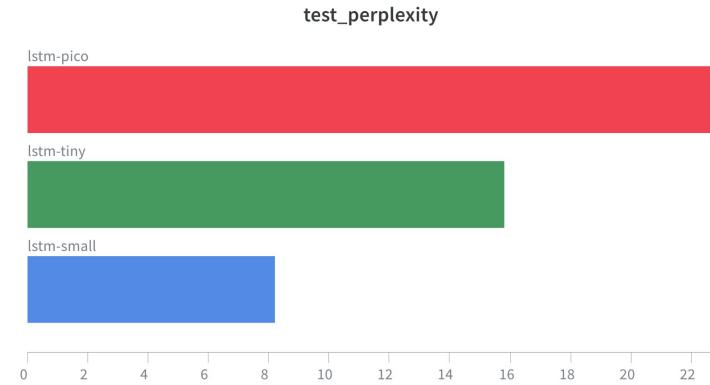
LSTM

LSTM - Model Parameters

- We train 3 LSTM model variants, each with different:
 - Embedding size
 - Hidden size

Model Name	Embedding Size	Hidden Size	#Parameters
LSTM-small	256	256	30M
LSTM-tiny	64	64	7M
LSTM-pico	32	32	3.2M

LSTM - Model Size Ablation



- We observe the trend of the validation loss
 - **larger LSTMs converge quicker**
- Similarly, the **larger LSTM yields lower test perplexity**

P A R T 2.2

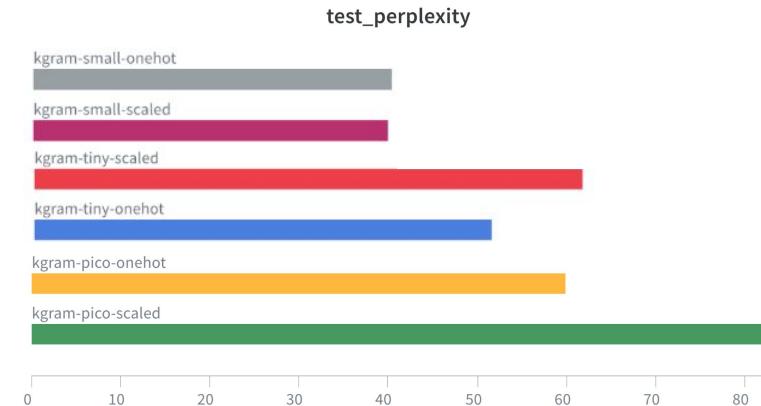
k-Gram MLP

k-Gram MLP - Model Parameters

- We train 3 k-Gram MLP model size variants, each with different:
 - Embedding size (32, 64, 256)
 - Number of inner layers (1, 3, 5)
 - Embedding Type (Onehot, scaled embeddings)

Model Name	Embedding Size	Inner Layers	Embedding Type	#Parameters
k-Gram-small	256	5	Onehot, Scaled	30M
k-Gram-tiny	64	3	Onehot, Scaled	7M
k-Gram-pico	32	1	Onehot, Scaled	3.2M

k-Gram MLP - Model Size Ablation



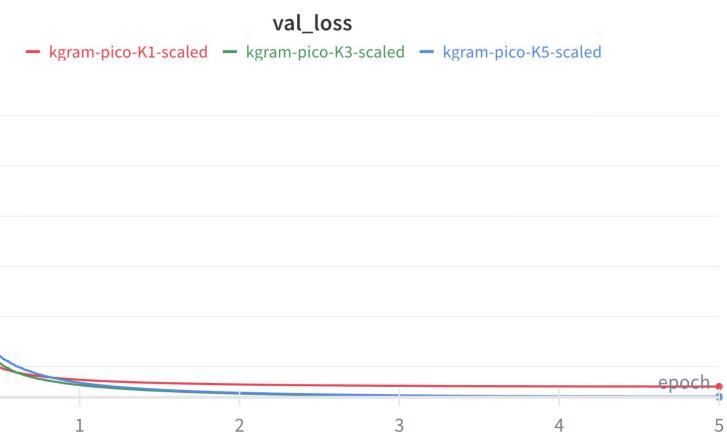
- We observe the trend of the validation loss
 - **Larger models converge quicker with lower test perplexity.**
- One-hot yields lower test perplexity than the scaled embeddings.
 - This **difference decreases with increase in model size.**

k-Gram MLP Pico - Model Parameters

- We train 6 k-Gram MLP pico model variants, each with different:
 - K (1, 3, 5)
 - Embedding Type (one-hot, scaled embeddings)

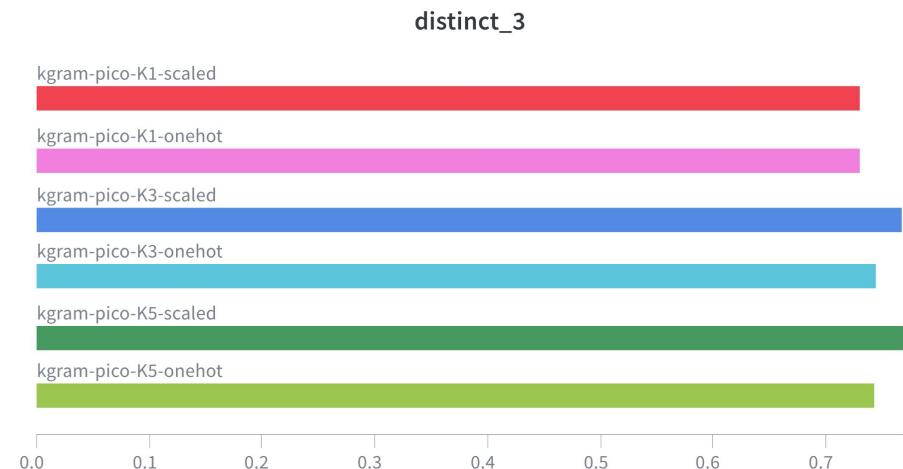
Model Name	K	Embedding Type
k-Gram-pico-k5	5	Onehot, Scaled
k-Gram-pico-k3	3	Onehot, Scaled
k-Gram-pico-k1	1	Onehot, Scaled

k-Gram MLP Pico - Varying K and embedding type



- Increase in K ($1 \rightarrow 3 \rightarrow 5$) converges at a lower validation loss.
- Loss curves are **smoother for scaled embeddings**, we notice a bend for the one-hot loss curves.

k-Gram MLP Pico - Varying K and embedding type



- **K=5 and one-hot embeddings yields the best performance**
 - lowest test perplexity, there was little variance in diversity.
- **Difference between the perplexity of scaled and one-hot increases with K.**

P A R T 2 . 3

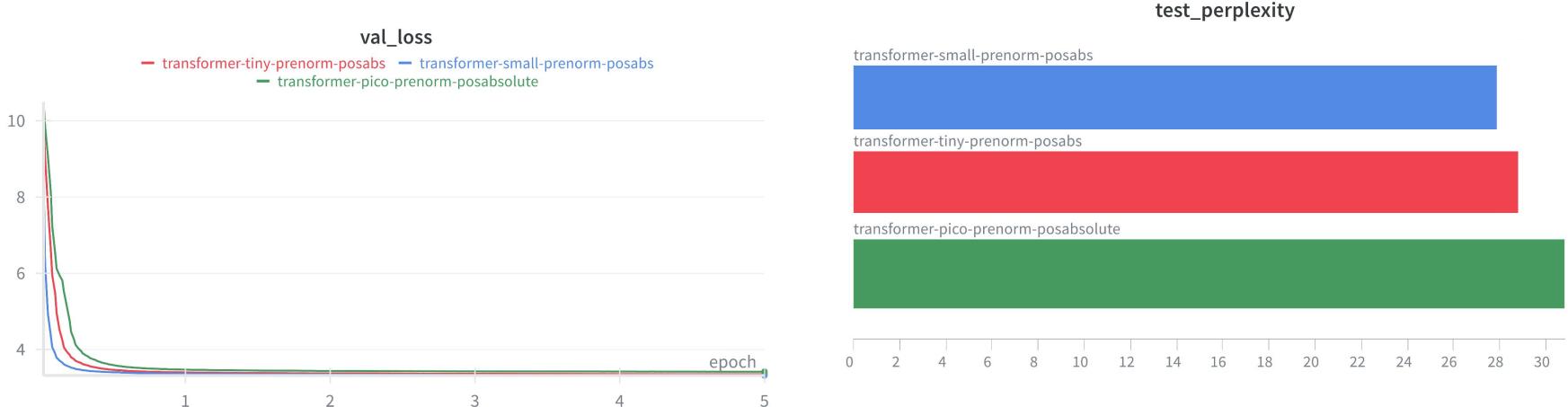
Transformer

Transformer – Model Parameters

- We train three GPT-2 based Transformer model variants, each with different:
 - Embedding size (64, 128, 384)
 - Number of Heads (1, 2, 6)
 - Number of Blocks (1, 2, 6)
- Training Setup:
 - Normalization: Pre-norm
 - Positional Embeddings: Absolute

Model Name	Embedding Size	Number of Heads	Number of Blocks	Parameters
transformer-small	384	6	6	30M
transformer-tiny	128	2	2	7M
transformer-piccolo	64	1	1	3.2M

Transformer – Model Size Ablation



- We observe the trend of the validation loss
 - **Larger models converge quicker with lower test perplexity.**

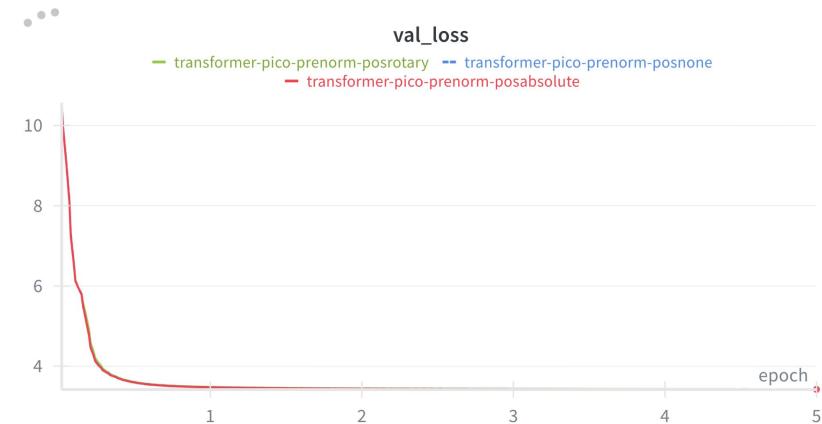
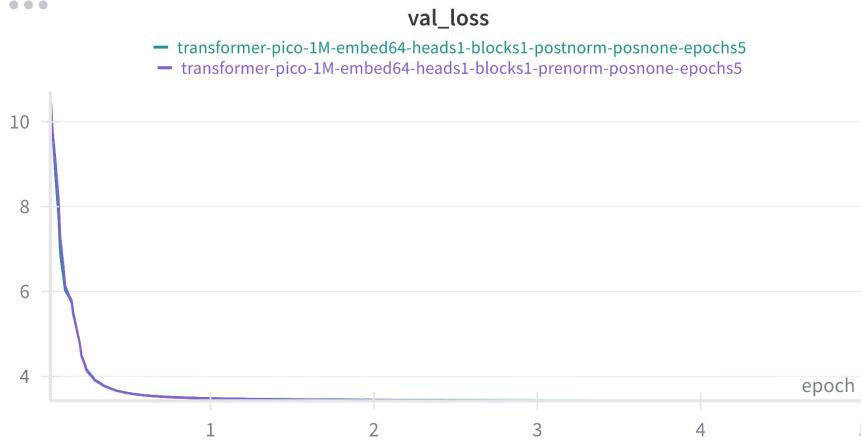
Transformer Pico – Model Parameters

- We train 3 Transformer pico model variants, each with different:
 - Normalization (prenorm, postnorm) – *Using RMSNorm*
 - Positional Embedding (absolute, RoPE, NoPE)

Model Name	Positional Embedding	Normalization
Transformer-pico-k5	absolute	prenorm, postnorm
Transformer-pico-k3	RoPE	prenorm, postnorm
Transformer-pico-k1	NoPE	prenorm, postnorm

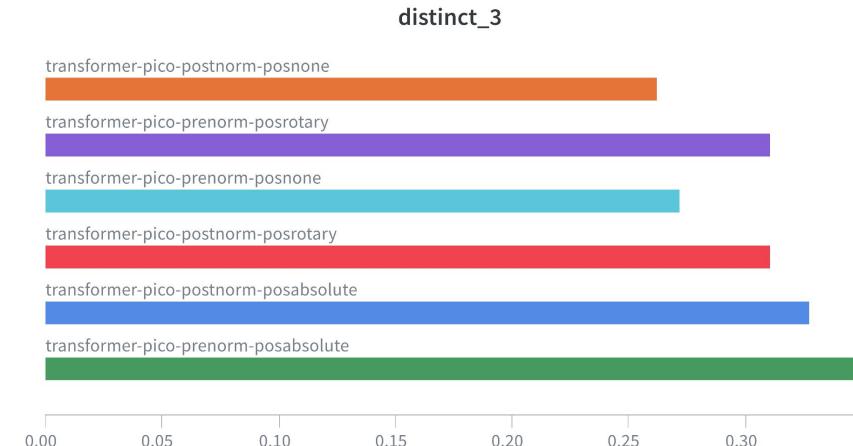
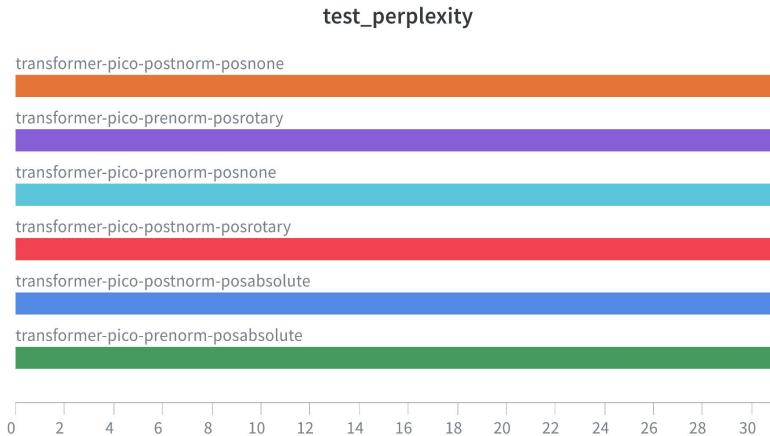
Transformer Pico - Varying PE / Norm

...



- We observe **minimal variance** in validation loss curves across choice of norm and positional encodings.

Transformer Pico - Varying PE / Norm

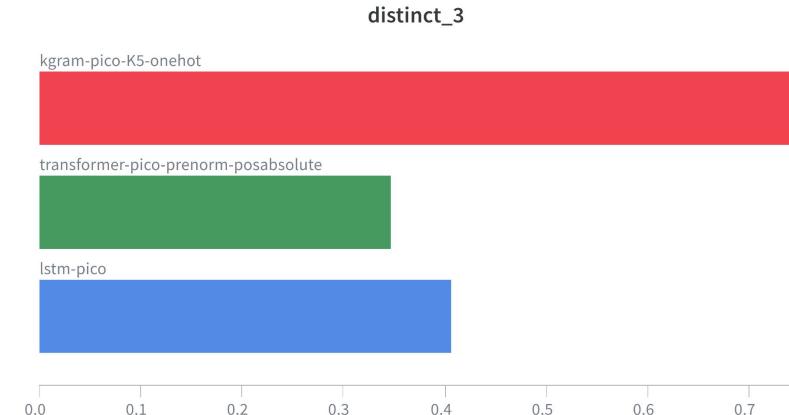
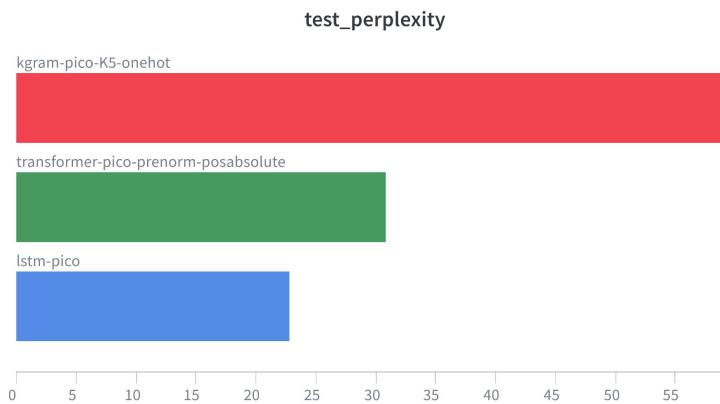


- Minimal variance in test-time perplexities on varying PE / Norm.
- Absolute PE yields highest diversity, while NoPE yields lowest.
- **PreNorm + Absolute PE** yields the best diversity, lowest perplexity.

PART 03

Model Comparison

Comparison of Best Pico Models



- LSTM gives lowest test perplexity, moderate diversity.
- Kgram MLP gives highest test perplexity, richest diversity.
- Transformers give moderate test perplexity and diversity.
- **Best model: LSTM** - more parameter efficient at smaller scale.
Transformers need more data.

P A R T 0 4

Decoding Strategies

Generated Samples - LSTM-pico

Prompt: Once upon a time, there was a very intelligent			
Decoding Strategy	Greedy	Top-p (0.9)	Top-p (1.0)
Output	Once upon a time, there was a very intelligent girl named Lily. She loved to play with her toys and play in the park. One day, she saw a big tree. She was very happy and she was very happy. She was so excited to see what was wrong	Once upon a time, there was a very intelligent girl named P hair. She was three years old and loved exploring. One day, Lily didn't want to play with his toys. Max smiled to her if he couldn't find him.	Once upon a time, there was a very intelligent with someone else. Timmy were driving through their friends.\n\nOne day, Jack woke up and stopped the ocean in the park. He looked around, and the door was so fast before.
Analysis	Repeats phrases, high coherence that fades later on.	Less repetition and more variety, but still somewhat disjoint.	Highly diverse but very incoherent and grammatically unstable.

Generated Samples - KGram-MLP-pico

Prompt: Once upon a time, there was a very intelligent			
Decoding Strategy	Greedy	Top-p (0.9)	Top-p (1.0)
Output	Once upon a time, there was a very intelligent girl. She loved to play with her toys. One day, she saw a big girl in her garden. She day to her a and, was was, happy saw she big...	Once upon a time, there was a very intelligento called to. he was made happy happy! Then found\n wildButa got like important about talk work fill lived., big morning colourful Jane scarf starting dad...	Once upon a time, there was a very intelligent man Every loved, adventures stones accidentally day the and. to wanted some make their and run out their the.. felt mom so...
Analysis	Starts with a coherent sentence, but loses meaning after.	Random generation but incoherent.	Varied with no grammatical or narrative structure.

Generated Samples - Transformer-pico

Prompt: Once upon a time, there was a very intelligent			
Decoding Strategy	Greedy	Top-p (0.9)	Top-p (1.0)
Output	Once upon a time, there was a very intelligent and said, \"I'm sorry,	Once upon a time, there was a very intelligent dog named Timmy replied, \"I think it.\n\nMolly saw a squirrel came to show his arms.\n\n\n\nHer mom and John.\n\n\n\nHe rolled around, he wanted to pick some time, so hungry,	Once upon a time, there was a very intelligent dog named Timmy, there was so she was always smiling. Tom was the soil and said \"Oh no. Its fur and the juice. He does not alert. Timmy saw a problem.\n\nThe boy named Ruby sitting on, their toys all both wanted to explore...
Analysis	Starts with coherence, then collapses into repetition of phrases.	Random generation but incoherent.	Creative and diverse but less coherent and more erratic.

P A R T 0 5

Conclusion

Implementing picoLLM

- **Robust data loaders**
 - No sampling of rows for training prevents losing training data
 - Parallel workers to speed up tokenization and batching
- **Enhanced PyTorch training pipeline**
 - Plug-and-play architecture implementations allowing fine-grained control while training
 - Customizable optimizers, schedulers, training args
 - Integrations with WandB and HuggingFace
 - Regular model performance logging and checkpointing
 - Easy model loading after training for inference



Thank you