# Evaluating MinGRU on Complex Tasks

**Indira Vats, Kyung Jae Lee, Luc Baier-Reinio**
University of Toronto
{indiravats, kjlee, lucbr}@cs.toronto.edu

## Abstract

In this project, we implement the minGRU RNN proposed by Feng et. al [1] in two distinct settings: First, as a replacement for attention in a standard encoder-only transformer, and second, as an aggregation layer of an encoder-only Transformer for long sequences. We design synthetic datasets to assess each model's viability for long-context reasoning. Then, we evaluate various configurations of the minGRU-based encoder-only architecture on SQuAD, a challenging question-answering dataset. We find that while minGRU is capable of handling long sequences, its performance is questionable on a more complex NLP task. [1]

## 1 Introduction and related works

Recurrent Neural Networks (RNNs) [2, 3] model dynamic data sequences by selectively retaining previous information. However, standard RNNs struggle with long-term dependencies due to vanishing or exploding gradients. Long Short-Term Memory (LSTM) networks [4] address these issues by introducing memory cells to preserve gradients over time, facilitating learning across long sequences. Gated Recurrent Units (GRUs) [5] further simplify RNN architectures with reset and update gates, enabling efficient temporal modeling with fewer parameters than LSTMs. A major drawback of traditional RNNs is having to backpropagate through time, which prohibits efficient training.

Transformers [6] overcome the limitations of RNNs by using attention to parallelize over the time dimension and better handle long-range dependencies. Transformers have been responsible for many of the recent breakthroughs in deep learning. However, the attention mechanism has a quadratic computational and memory complexity with respect to sequence length, which limits their scalability for tasks with larger context requirements. This constraint has spurred interest in more efficient transformer variants [7, 8] as well as alternative architectures to better handle longer sequence lengths [9].

[1] revisits LSTMs and GRUs and addresses inefficiencies arising from BPTT to propose minLSTMs and minGRUs. These architectures simplify their respective counterparts by removing dependencies on previous hidden states in their gates, enabling parallel training and reducing parameters. The paper evaluates the models on relatively simple tasks such as selective copying or character-level language modelling. While it was found that the minGRU architecture can achieve comparable performance to the Transformer on these tasks, in this project, we hope to evaluate the model on more complex tasks (long-context reasoning and question answering).

## 2 Model architectures

The minGRU architecture introduces a key departure from the traditional GRU by decoupling the computation of hidden states and gate values from the previous hidden state, relying solely on the input at the current timestep. This design enables efficient parallel computation of hidden states across the time dimension using the parallel scan algorithm [10]. However, as a result, the hidden states of a single layer of minGRU lack temporal dependency on inputs from prior timesteps. To address this limitation, multiple minGRU layers are stacked,

[1] Code available on GitHub: https://github.com/lbaierreinio/min-gru-transformers

where the hidden states from one layer are passed as inputs to the subsequent layer. Through this hierarchical stacking, the final layer produces hidden states that effectively capture dependencies across preceding timesteps.

We replace self-attention in a standard transformer encoder block to construct a minGRU block. In particular, each minGRU block comprises multiple layers of minGRU layers followed by a feed-forward network and employs pre-layer normalization and residual connections. The configurable hyperparameters are the number of layers to stack, the hidden dimension size, and bidirectionality of the minGRU. This modified block serves as the basis for our encoder models to tackle the synthetic long-context task and question answering on SQuAD. Detailed task descriptions and the corresponding adaptation of the model are provided in the respective sections.

The Long Context Transformer model architecture is defined using sequence chunking and a unidirectional minGRU aggregation head. After token embedding, each sequence is partitioned into fixed-sized chunks. Each chunk is processed independently by N encoder-only Transformer layers. Then, the aggregation head processes the [CLS] tokens for each chunk and returns the final hidden state. This model is designed to handle long sequence lengths. Since the attention mechanism size is fixed by chunking, the time and memory complexity of the Transformer layers grow linearly with the sequence length.

## 3 Synthetic classification dataset experiments

We identified the following elements that make long-context reasoning difficult and built them into our synthetic dataset:[2]: (1) Long-range dependencies, (2) irrelevant/redundant information, (3) needle-in-haystack tasks, and (4) summarization [11, 12, 13]. Each example consists of two subsequences, both generated from one of two grammars, G1 or G2. Two indicator tokens, both either X or Y, are inserted randomly into the example. Examples are labeled by the ordering of grammars and equality of indicator tokens.

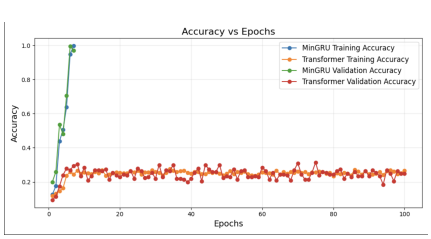
| **Indicator Tokens Equal** | | | | | | **Indicator Tokens Not Equal** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Order** | **Ind.** | **Label** | **Order** | **Ind.** | **Label** | **Order** | **Ind.** | **Label** | **Order** | **Ind.** | **Label** |
| G1, G2 | EQ | 0 | G2, G2 | EQ | 2 | G1, G2 | NEQ | 4 | G2, G2 | NEQ | 6 |
| G2, G1 | EQ | 1 | G1, G1 | EQ | 3 | G2, G1 | NEQ | 5 | G1, G1 | NEQ | 7 |

To solve the task, the model must (1) learn the dependency between indicator tokens, (2) process repetitive subsequences, (3) find each indicator token, and (4) summarize the grammars' ordering. For our experiments, we train the following models: a minGRU model with 3 bi-directional layers, 384 embedding dimension, and 15,275,144 parameters; and a Long Context Transformer with 6 layers, 8 heads, 256 embedding dimension, 1024 feed-forward hidden dimension, chunk size of 512, and 12,685,832 parameters. We attach a classification head to the output of each model and use scores for each label for prediction. Models are trained using cross-entropy loss. Findings from synthetic experiments follow.

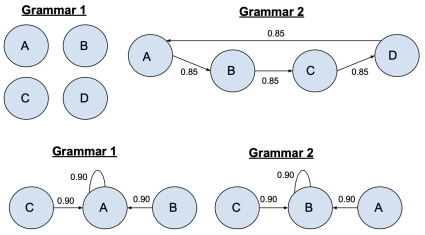
First, the Transformer model cannot differentiate between similar grammars. In Difficult Grammars, characters appear with the same frequency in both grammars. The two grammars can only be distinguished by character ordering. The minGRU captures the nuances of ordering as tokens are processed in order. The Transformer relies on positional encoding for ordering and is unable to differentiate between the two. Both models learn to distinguish between the two grammars in Easy Grammars, as their character frequencies differ.

Second, we observe that training and validation accuracy hover around 50% before rapidly approaching 100%. This phenomenon is known as 'grokking' [14], where a model suddenly solves a synthetic or algorithmic task after doing no better than random for many epochs. Both models learn the grammar summarization component of the task (hence 50% accuracy) before grokking the indicator token dependency.

[2]We wrote a Python script that generates instances of the dataset given parameters that control the difficulty of the tasks, which helped us in obtaining preliminary results.

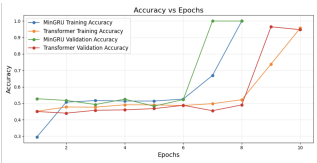



(a) Transformer cannot learn Difficult Grammars

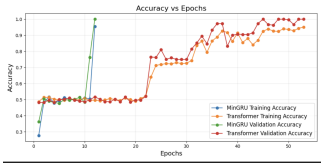



(b) Difficult Grammars (top); Easy Grammars (bottom)

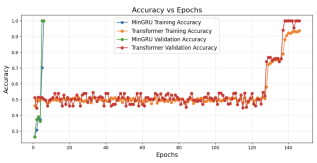
Figure 1: Transformer vs. minGRU on Difficult Grammars and grammars definitions




(a) 512 sequence length




(b) 1024 sequence length




(c) 2048 sequence length

Figure 2: Transformer vs. minGRU on Easy Grammars (4000 examples)

Both models solve the 512 sequence length task within 10 epochs, indicating that they are comparably effective at learning the synthetic task with Easy Grammars. However, as the sequence length grows, the training trends diverge. The deep, bi-directional minGRU model presents a simpler solution. The single hidden state per layer only needs to track the minGRU's understanding of which grammar(s) and indicator token(s) it has processed. MinGRU's computational complexity grows sub-quadratically as the sequence length increases, allowing the model to handle longer sequences without chunking or truncating. The minGRU continues to grok the indicator token dependency with increasing sequence length.

The Transformer model uses chunking when the sequence length exceeds 512. Since chunks are processed independently, the Transformer layers must handle the cases when 0, 1, or 2 indicator tokens lie in the same chunk and convey these cases to the aggregation head. This added complexity is reflected in the Transformer's accuracy graphs. The Transformer no longer groks the indicator token dependency but learns these distinct cases separately.

Ultimately, both models have some viability for long context reasoning. However, the minGRU model scales more gracefully due to its simplicity, efficiency, and ability to process long sequences in their entirety. The Long Context Transformer model relies on chunking and aggregation to overcome attention's quadratic computational complexity, and slowly converges to solve the synthetic task.

## 4 SQuAD experiments

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset comprising 100,000 question-answer pairs derived from Wikipedia articles, with each answer being a specific segment of text from the corresponding reading passage [15]. We chose this dataset to evaluate the minGRU-based encoder architecture on a practical NLP task, as it is a challenging enough dataset that has been explored with non-pretrained models.

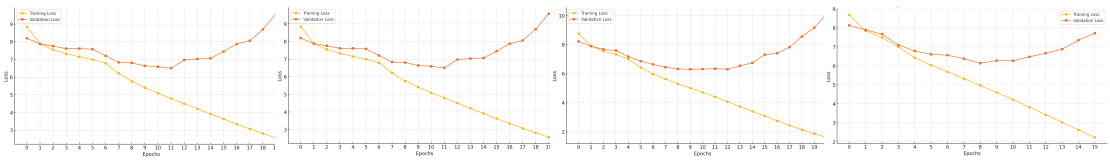
Each training example in SQuAD consists of a question, context, and an answer that is a substring of context. We transform each example into the format `Question [SEP] Context` and formulate the task as predicting the start and end indices of the answer in the context using a 2-layer MLP as a classification head on top of the minGRU encoder output. We tokenize using the `bert-base-cased` tokenizer. We selected four target model sizes: 16M, 24M, 32M, and 40M; and constructed various configurations by adjusting the number of layers and the hidden dimensions (kept equal to the number of classification dimensions)

within a tolerance range of ±5%. In addition to assessing the effectiveness of the minGRU-based architecture on a practical NLP task, we also explore the parameter efficiency of key hyperparameters that influence the model's size. Specifically, we analyze the effects of increasing the number of layers, the hidden dimension size, as well as bidirectional and unidirectional configurations. The resulting configurations are outlined in Table 1.

We trained each model for 30 epochs with a batch size of 64 and a cosine learning rate scheduler. The learning rate ranged from $6e-5\times3$ to $6e-4\times3$ with 5000 warmup steps. We report on the EM/F1 values at the epoch at which the highest F1 value was observed.

Table 1: SQuAD experiment results (best configuration per paramater size highlighted)

| **Size** | **Layers** | **Dim** | **Bi** | **EM** | **F1** | **Size** | **Layers** | **Dim** | **Bi** | **EM** | **F1** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16M | 4 | 384 | Y | 4.9007 | 11.3754 | 32M | 4 | 640 | Y | 11.1258 | 19.8844 |
| 16M | 8 | 384 | N | 3.6140 | 11.6855 | 32M | 8 | 512 | Y | 13.1315 | 22.6535 |
| **16M** | **10** | **302** | **Y** | **11.7219** | **20.2815** | 32M | 8 | 640 | N | 6.7360 | 16.7491 |
| 24M | 4 | 608 | N | 2.8004 | 10.5282 | 32M | 10 | 460 | Y | 12.2611 | 20.0046 |
| 24M | 4 | 512 | Y | 10.1892 | 18.6474 | 32M | 16 | 400 | Y | 15.3926 | 25.6889 |
| 24M | 8 | 512 | N | 3.4248 | 11.0791 | **32M** | **30** | **302** | **Y** | **16.4049** | **26.6710** |
| 24M | 10 | 400 | Y | 5.9886 | 11.7654 | 40M | 10 | 552 | Y | 2.6868 | 9.7336 |
| 24M | 16 | 340 | Y | 14.7588 | 25.0282 | 40M | 16 | 460 | Y | 15.4659 | 25.5984 |
| **24M** | **22** | **302** | **Y** | **15.7143** | **26.0953** | **40M** | **22** | **400** | **Y** | **15.9130** | **26.1369** |
| | | | | | | 40M | 30 | 354 | Y | 12.7152 | 22.6534 |



(a) 16M model (b) 24M model (c) 32M model (d) 40M model

Figure 3: Loss plots for the best model per target capacity

In all configurations, we observed severe overfitting well before completing the 30 epochs. While the model nearly perfectly memorized the training data, the validation loss diverged significantly. A positive correlation was observed between model capacity and the best EM/F1 scores, but this trend plateaued beyond a model size of 32M parameters, with no significant performance improvements. Consequently, we conclude that the architecture's generalization ability is insufficient to effectively solve the SQuAD task. For reference, competitive non-pretrained models are able to achieve EM and F1 scores exceeding 60 [16].

Regarding the impact of of tuning the available hyperparameters, performance gains plateau or even sometimes decrease with respect to model capacity as noted above, highlighting the importance of architectural optimization. Models with bidirectionality consistently outperform similarly sized unidirectional models, demonstrating the effectiveness of incorporating bidirectionality for tasks where context plays a crucial role. Furthermore, the results indicate that increasing the number of layers is generally more parameter-efficient than increasing the hidden dimension size, possibly because the additional layers provide the model with more opportunities to condition on past sequence items.

## 5 Conclusion

In this paper, we implemented and analyzed the minGRU architecture. We showcased the model's ability to scale with longer sequences but revealed weaknesses to generalize to more complex problems. We also conducted hyperparameter analysis on the SQuAD dataset to find more parameter-efficient configurations. Future works should secure the compute necessary to pre-train a minGRU-based model and unlock its full potential in NLP tasks.

# References


[1] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadegh. Were rnns all we needed? *arXiv preprint arXiv:2410.01201*, 2024.

[2] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014. doi: 10.3115/v1/D14-1179. URL `https://aclanthology.org/D14-1179`.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017. URL `https://arxiv.org/abs/1706.03762`.

[7] Daniel Y. Fu, Simran Arora, Jessica Grogan, Isys Johnson, Sabri Eyuboglu, Armin W. Thomas, Benjamin Spector, Michael Poli, Atri Rudra, and Christopher Ré. Monarch mixer: A simple sub-quadratic gemm-based architecture, 2023. URL `https://arxiv.org/abs/2310.12109`.

[8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL `https://arxiv.org/abs/2205.14135`.

[9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL `https://arxiv.org/abs/2312.00752`.

[10] Franz A Heinsen. Parallelization of an ubiquitous sequential computation. *arXiv preprint arXiv:2311.06281*, 2023.

[11] Hengyi Wang, Haizhou Shi, Shiwei Tan, Weiyi Qin, Wenyuan Wang, Tunyu Zhang, Akshay Nambi, Tanuja Ganu, and Hao Wang. Multimodal needle in a haystack: Benchmarking long-context capability of multimodal large language models. *arXiv preprint arXiv:2406.11230*, 2024.

[12] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[13] Jon Saad-Falcon, Daniel Y Fu, Simran Arora, Neel Guha, and Christopher Ré. Benchmarking and building long-context retrieval models with loco and m2-bert. *arXiv preprint arXiv:2402.07440*, 2024.

[14] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[15] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[16] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL `http://arxiv.org/abs/1611.01603`.

# 6 Appendix

## 6.1 Contributions

- Luc: I was largely responsible for implementing the minGRU and bi-minGRU classes, which are building blocks for the minGRU models (`rnn`). I was responsible for implementing the TransformerEncoder and TransformerEncoderBlocks (`transformer`), which are building blocks for the Long Context Transformer model. I implemented the MinGRUSynthetic and TransformerSynthetic classes, which are the models used in the synthetic dataset. I implemented the logic to generate the synthetic datasets (`datasets/synthetic`) and the logic to train, evaluate, and profile the minGRU and Transformer models on the synthetic datasets (`train`). I implemented, ran, and analyzed the experiments for the synthetic dataset (`experiments`).
- Indira: I implemented the `Positional Encoding` class for the Transformer and added `memory profiling` for the synthetic dataset experiments. I was responsible for running the SQuAD experiments wherein I generated various MinGRU `configurations` for our target model sizes and handled training and `evaluation`, which included retrieving top-performing models using log analysis and `checkpointing`, and further analyzing performance by question and title type and comparative answer analysis. Additionally, I investigated techniques for counting FLOPs in synthetic experiments, testing tools such as the torch profiler, fvcore, and DeepSpeed profiler, which proved unsuitable for our custom models due to unsupported operations. I also explored Flopper, an fvcore wrapper with support for custom operations, but did not integrate it due to challenges in justifying FLOP assignments.
- Kyung Jae: I was mainly responsible identifying a suitable real-world NLP task to evaluate minGRU on, as well as writing the relevant pipeline (dataset loading, pre-processing, training, and evaluation code). Initially, I had proposed to evaluate the model on LoCoV1 directly to test long-context reasoning abilities. I wrote the pipleline for LoCoV1 (`load_locov1.py`), but soon realized that it would be difficult to see meaningful results on the dataset without the use of pre-trained models. I decided to switch to SQuAD due to its comparative simplicity, as well as the availability of numerous benchmarks without the use of pre-trained models for comparison. I then wrote the pipeline for SQuAD (`train_squad.py`) along with the `MinGRUSquadQA` class, which is the model used to tackle the task. I was also heavily involved with debugging and reviewing minGRU architecture-related changes.

## 6.2 Relation of final project to project proposal

In our project proposal, we indicated that we would like to assess the minGRU architecture on LoCoV1, a natural language benchmark designed for long-context reasoning. We believed the minGRU architecture had the potential to perform well on long-context reasoning tasks, and noted that Bengio's experiments on the architecture had been fairly simple. Accordingly, we hoped to perform the first analysis of the minGRU architecture on a complex natural language task for long-context reasoning.

However, following our own research and internal discussions, as well as conversations with Professor Raffel, we realized that natural language pre-training was necessary but prohibitively expensive to achieve meaningful results on LoCoV1. Upon this realization, Professor Raffel suggested that we explore the minGRU architecture in the context of a simpler, possibly synthetic task, and a real-world NLP task, hence the conception of our synthetic dataset and our analysis on the SQuAD Dataset.

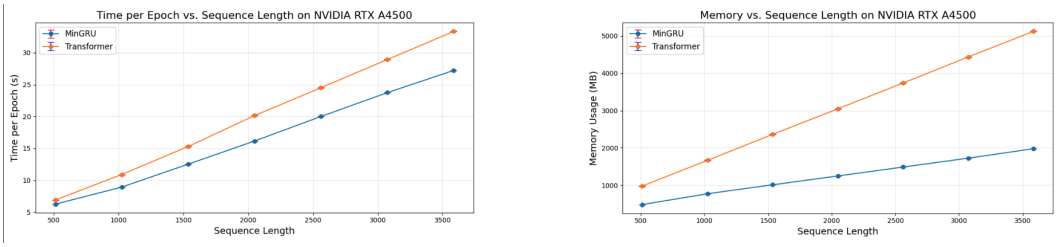
## 6.3 Grammar definitions

The following is a more detailed explanation of sequence generation from the Grammars. Recall the Grammar Definitions from Page 3. Consider arbitrary letters A and B. An edge from A to B with an associated probability p indicates that if A is the current token, B will be selected as the next token with probability p. All nodes that A does not have an outgoing edge to will be selected as the next token with equal probability if B is not selected. For example, in sequences generated from Grammar 1 of Difficult Grammars, the next token will

always be selected with equal probability between A, B, C, and D, since there are no edges. In sequences generated from Grammar 2 of Difficult Grammars, A is likely to be followed by B (and if it is not, the token will be selected randomly from [A, C, D]), B is likely to be followed by C (and if it is not, the token will be selected randomly from [A, B, D]), and so on. In Easy Grammars, sequences generated from Grammar 1 will tend to have mostly A's, and sequences generated from Grammar 2 will tend to have mostly B's.

### 6.4 Memory and runtime overheads

The following graphs compare the memory and runtime for an epoch of training on a dataset with 1000 examples as the sequence length varies from 512 to 4096. While both appear to grow linearly, the Transformer is more expensive. The Transformer model can grow linearly as the attention window is fixed at 512 tokens. The minGRU model uses the parallel scan algorithm, which is subquadratic, to process sequences. All memory and runtime experiments were run on the same NVIDIA RTX A4500 GPU.




(a) Runtime

(b) Memory overhead

Figure 4: Computational comparison of minGRU and Transformer across different sequence lengths

### 6.5 Further analysis on SQuAD

We initially evaluated all 19 configurations as seen in Table 1 using a batch size of 64 to identify the top-performing configurations for all model sizes. We then re-tested these best configurations with a batch size of 32 and with early stopping at the first epoch validation loss starts to increase (we had to lower the batch size due to difficulties in sourcing GPUs with higher RAM toward the end of the submission deadline).

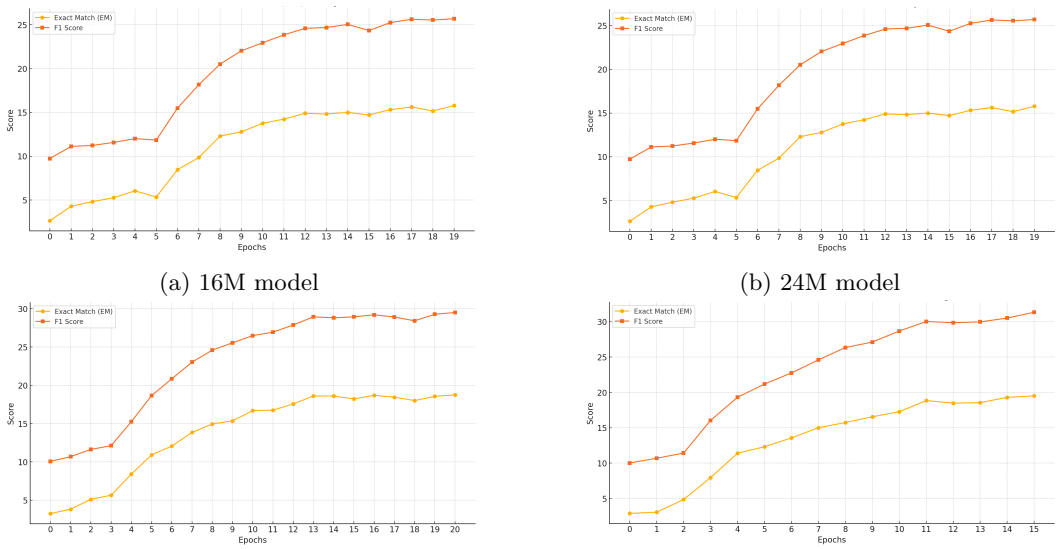
We then conducted a comprehensive analysis of our best-performing models (as seen in Table 2) across various sizes, evaluating their performance on the validation set. This analysis focused on different question types (e.g., "what," "when," "where," "who," "which," "why," "how," or "other") as well as on the diverse categories represented in the title column of the SQuAD dataset. This was done to assess the strengths and limitations of our models, aiming to identify areas where they excel and where there is potential for improvement. While specific claims are challenging to make due to the uneven distribution of classes in the validation set, we have included our results in the appendix to provide insights into the minGRU's performance.

#### 6.5.1 Re-training with early stopping

Table 2: Best performing models *(B=32)*

| Model Size | n_layer | hidden_dim | Bidirectional | EM Score | F1 Score |
|---|---|---|---|---|---|
| 16M | 10 | 302 | True | 15.7805 | 25.6927 |
| 24M | 22 | 302 | True | 18.2781 | 29.167 |
| 32M | 30 | 302 | True | 18.7512 | 29.4995 |
| 40M | 22 | 400 | True | 19.5175 | 31.3402 |

### 6.5.2 Accuracy vs Epochs




(a) 16M model

(b) 24M model

(c) 32M model

(d) 40M model

Figure 5: Accuracy (F1/EM) curves for models with varying sizes across epochs

### 6.5.3 Performance by question type

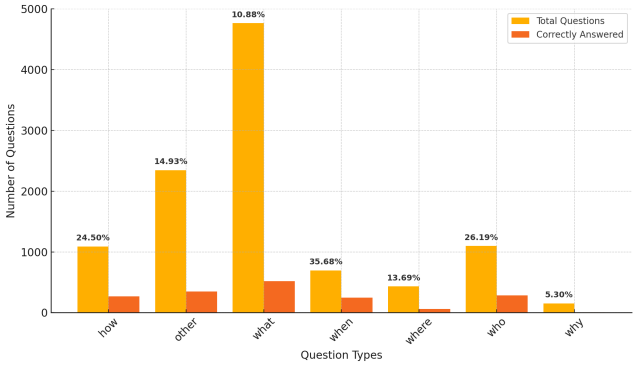



Figure 6: Accuracy of 16M model by question type

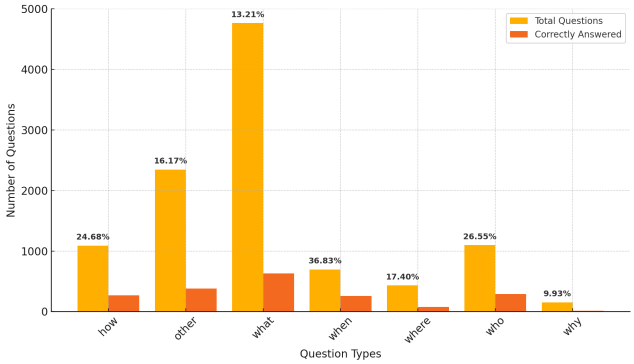



Figure 7: Accuracy of 24M model by question type

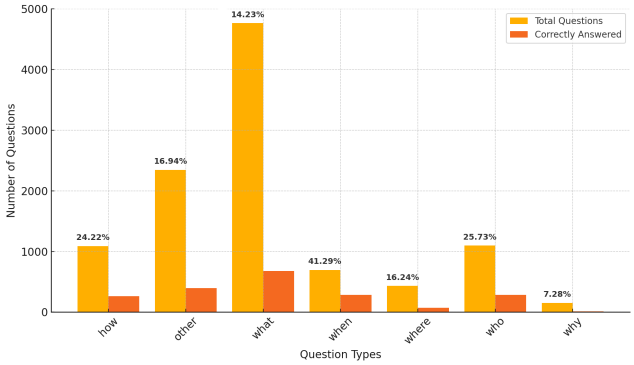



Figure 8: Accuracy of 32M model by question type

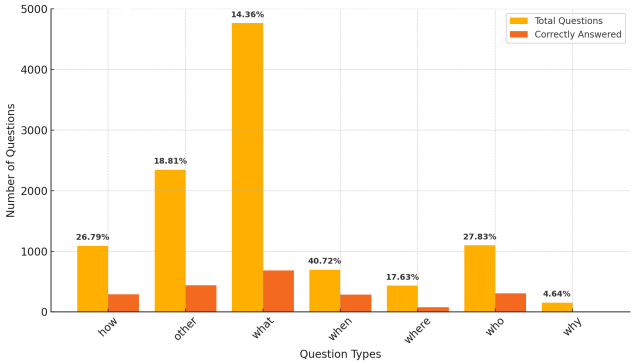



Figure 9: Accuracy of 40M model by question type

### 6.5.4 Performance by question type

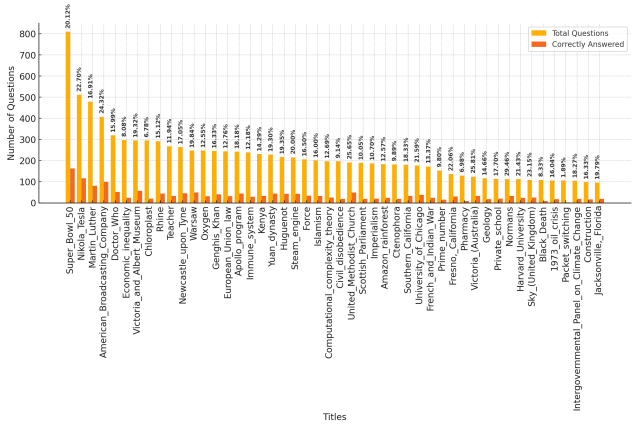



Figure 10: Accuracy of 16M model by title type

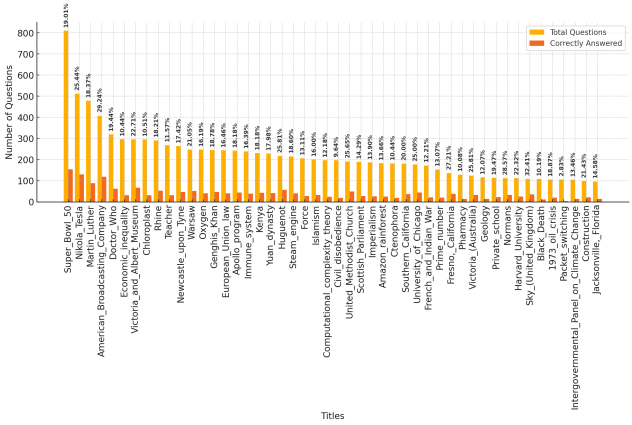



Figure 11: Accuracy of 24M model by title type

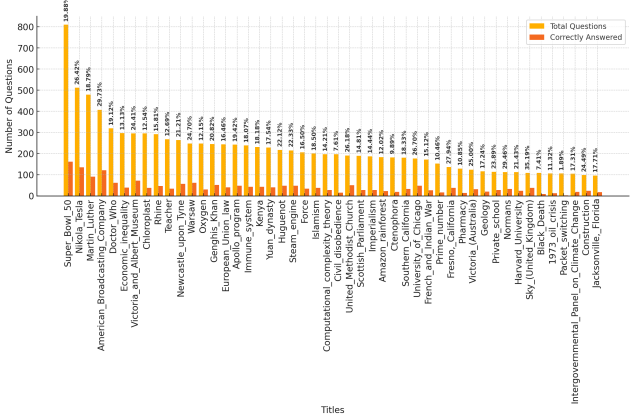



Figure 12: Accuracy of 32M model by title type

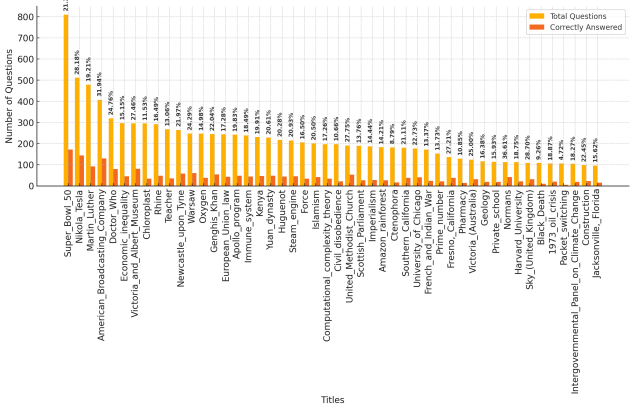



Figure 13: Accuracy of 40M model by title type

## 6.6 Answer analysis

To better understand the performance of the models, we compare the correctness of answers given by the different models (16M, 24M, 32M and 40M) on the SQuAD dataset. Cells are highlighted in green for correct matches and red for incorrect responses. We randomly

sampled 16 examples from our validation set for the same. These examples provide insight into the types of questions each model handles well and where they struggle.

**Question:** When was the Edict of Worms presented?
**Correct Answer:** 25 May 1521

| Model | Prediction |
|---|---|
| 16M: | 25 May 1521 |
| 24M: | 25 May 1521 |
| 32M: | 25 May 1521 |
| 40M: | 25 May 1521 |

**Question:** What is the process of constructing a building or infrastructure?
**Correct Answer:** Construction

| Model | Prediction |
|---|---|
| 16M: | Construction |
| 24M: | Construction |
| 32M: | Construction |
| 40M: | Construction |

**Question:** Who was the NFL Commissioner in early 2012?
**Correct Answer:** Roger Goodell

| Model | Prediction |
|---|---|
| 16M: | Roger Goodell |
| 24M: | Commissioner Roger Goodell |
| 32M: | Roger Goodell |
| 40M: | Roger Goodell |

**Question:** Who is the NFL's vice president of brand and creative?
**Correct Answer:** Jaime Weston

| Model | Prediction |
|---|---|
| 16M: | Jaime Weston |
| 24M: | Jaime Weston |
| 32M: | Jaime Weston |
| 40M: | Jaime Weston |

**Question:** What color were the Bronco's uniforms in Super Bowl 50?
**Correct Answer:** white

| Model | Prediction |
|---|---|
| 16M: | Denver |
| 24M: | Denver |
| 32M: | Denver |
| 40M: | Atlanta Falcons 34...they defeated the Seattle Seahawks |

**Question:** In what city is SAP Center located?
**Correct Answer:** San Jose

| Model | Prediction |
|---|---|
| 16M: | SAP Center |
| 24M: | SAP Center |
| 32M: | San Jose |
| 40M: | SAP Center |

**Question:** How many will the host committee dedicate to local charities?
**Correct Answer:** 25 percent

| Model | Prediction |
|---|---|
| 16M: | 25 percent |
| 24M: | 50 |
| 32M: | *No answer provided.* |
| 40M: | 25 percent |

**Question:** What type of image was Tesla thought to have accidentally taken a week prior to the announcement of its discovery?
**Correct Answer:** X - ray image

| Model | Prediction |
|---|---|
| 16M: | *No answer provided.* |
| 24M: | metal locking screw |
| 32M: | inadvertently |
| 40M: | X - ray image |

**Question:** For what invention was U.S. Patent 1,655,114 granted?
**Correct Answer:** VTOL aircraft

| Model | Prediction |
|---|---|
| 16M: | VTOL aircraft |
| 24M: | biplane |
| 32M: | biplane |
| 40M: | a biplane |

**Question:** How many other important people sent letters?
**Correct Answer:** more than 70

| Model | Prediction |
|---|---|
| 16M: | 70 |
| 24M: | 70 pioneers |
| 32M: | more than 70 |
| 40M: | 70 pioneers |

**Question:** What is the prize offered for finding a solution to P=NP?
**Correct Answer:** $1,000,000

| Model | Prediction |
|---|---|
| 16M: | The P versus NP problem |
| 24M: | P equals NP |
| 32M: | whether P equals NP |
| 40M: | US $ 1,000,000 |

**Question:** In contrast how were Catholic saints portrayed?
**Correct Answer:** frail Catholic saints

| Model | Prediction |
|---|---|
| 16M: | printed images...Protestantism |
| 24M: | spread of Protestantism |
| 32M: | *No answer provided.* |
| 40M: | were crucial |

**Question:** What types of engines are steam engines?
**Correct Answer:** external combustion

| Model | Prediction |
|---|---|
| 16M: | solar power, nuclear power or geothermal energy |
| 24M: | reduced-pressure steam |
| 32M: | solar power, nuclear power or geothermal energy |
| 40M: | external combustion engines |

**Question:** For what purpose is oxygen used by animal life?
**Correct Answer:** cellular respiration

| Model | Prediction |
|---|---|
| 16M: | Due to its energy content |
| 24M: | its energy content |
| 32M: | animals |
| 40M: | *No answer provided.* |

**Question:** What objects in organisms absorb singlet oxygen to prevent harm?
**Correct Answer:** Carotenoids

| Model | Prediction |
|---|---|
| 16M: | photolysis of ozone |
| 24M: | Carotenoids |
| 32M: | Carotenoids in photosynthetic organisms |
| 40M: | organic molecules |

**Question:** By which year did the American cars mpg start to improve?
**Correct Answer:** 1985

| Model | Prediction |
|---|---|
| 16M: | 1974 to 1979 |
| 24M: | 1985 |
| 32M: | *No answer provided.* |
| 40M: | *No answer provided.* |