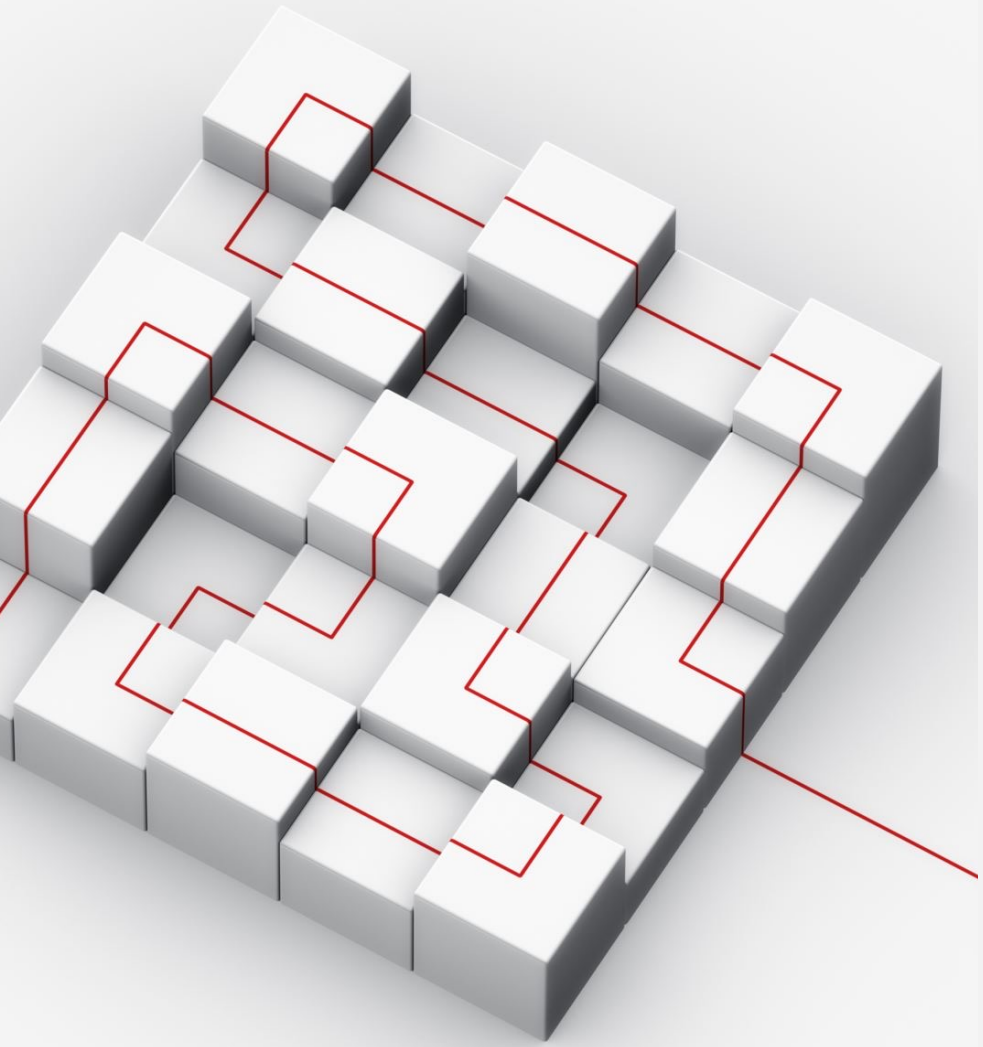
VALIDATING TOKEN-PRUNING METHODS IN TRANSFORMER MODELS

By: Edman Alicea & Cassandra Carlson

Date: November 28, 2023

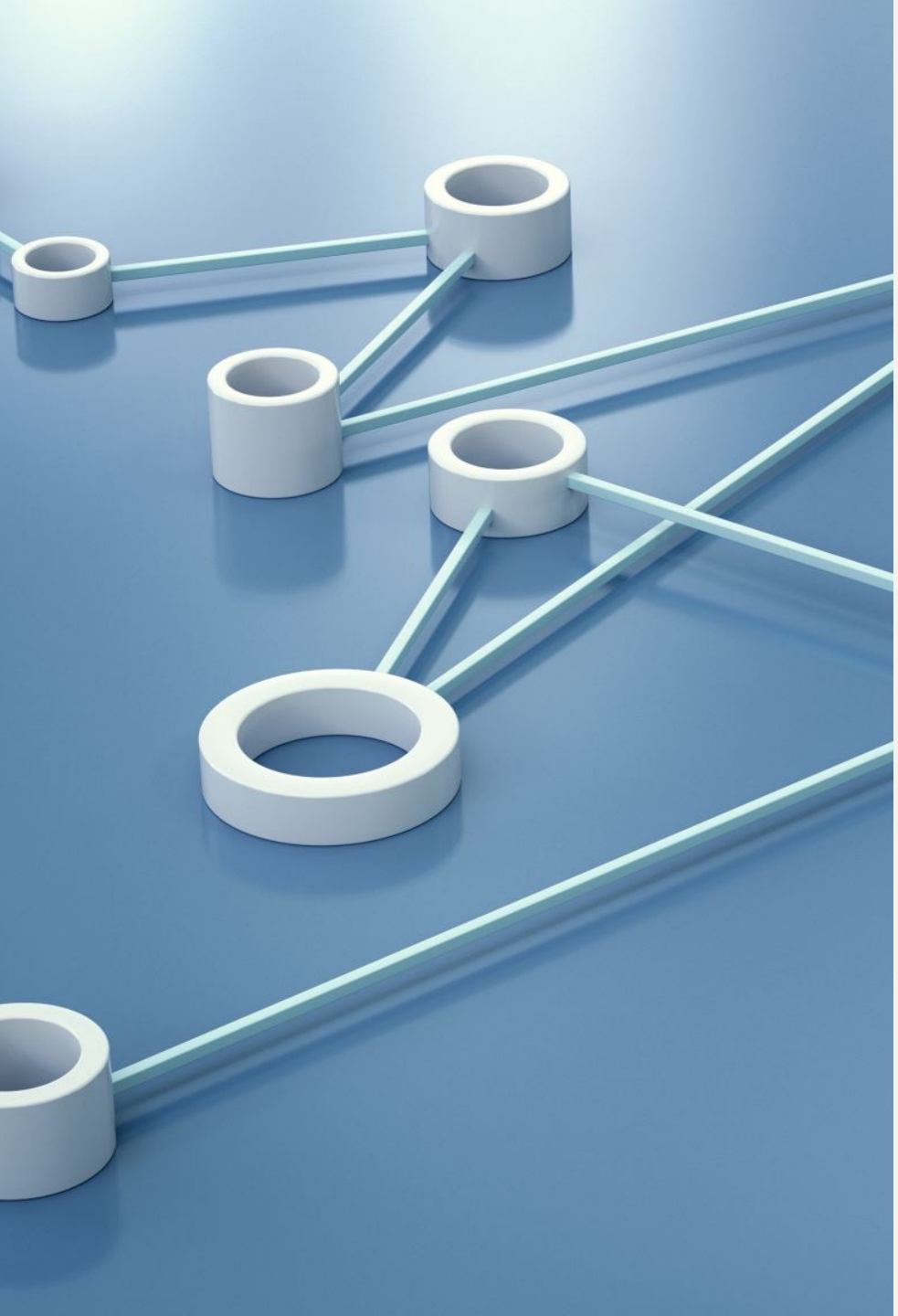
INTRODUCTION

- What are Transformers
  - A breakthrough in NLP for capturing long-range dependencies in text.
- Importance
  - Key to understanding context and nuances in language.
- Challenge
  - High inference cost
  - Increases greatly with larger models and longer input sequences.
- Relevance
  - Increasing need for efficient models in fast-paced, real-world applications.
  - Bringing transformer models to edge devices



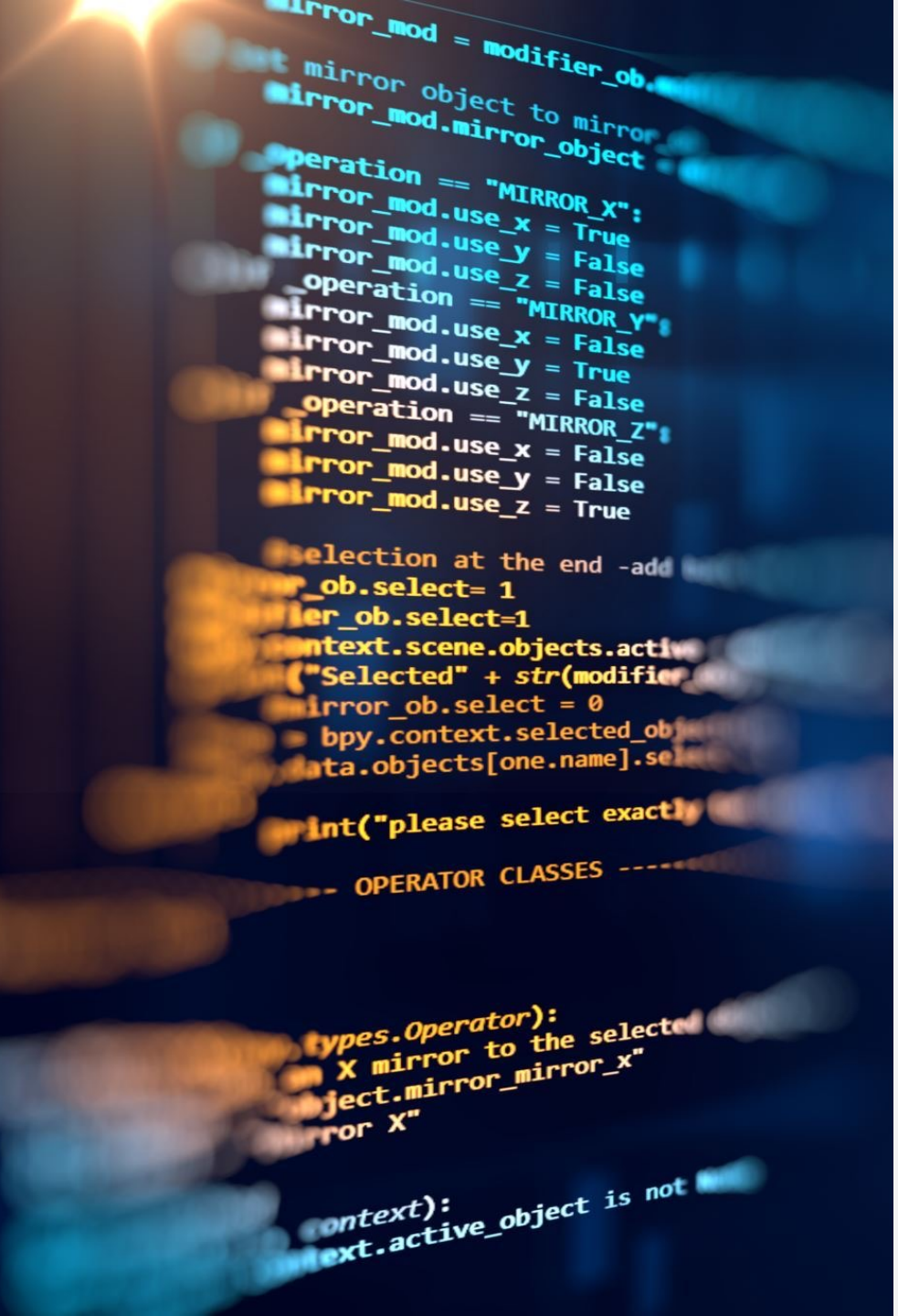
PROBLEM

- Core Issue
  - Transformer models' inference cost increases quadratically with input sequence length.
- Impact:
  - Delays in model deployment, especially in latency-sensitive environments.
- Goal:
  - Improve transformer performance for use in resource-limited scenarios.
  - Common techniques for increasing inference speed:
    - Token pruning
    - Knowledge distillation
    - Quantization



TOKEN-PRUNING TECHNIQUES

- Why Prune?
  - Essential for reducing size and improving latency without compromising accuracy.
  - Has distinct advantages over other model compression techniques (e.g., unlabeled data)
- Two Main Approaches
  - Attention-based scoring
    - Ranks tokens' importance using self-attention mechanism.
  - Prediction module
    - Utilizes an additional neural network for accurate importance scoring.



PREVIOUS WORKS

- Overview of other algorithms
  - PoWER-BERT
  - LTP
  - ToP
  - Transkimmer

COFI METHODOLOGY

- Approach
  - Task-specific structured pruning: CoFi (Coarse and Fine-grained Pruning).
- Key Insight
  - Joint pruning of coarse (e.g., layers) and fine-grained units (e.g., heads, hidden dimensions).
- Distillation Strategy
  - Layerwise distillation: Transfer knowledge from unpruned to pruned models during optimization.
- Advantages
  - Large speedups and competitive accuracy with less computation
  - Over 10× speedups and more than 90% accuracy preservation

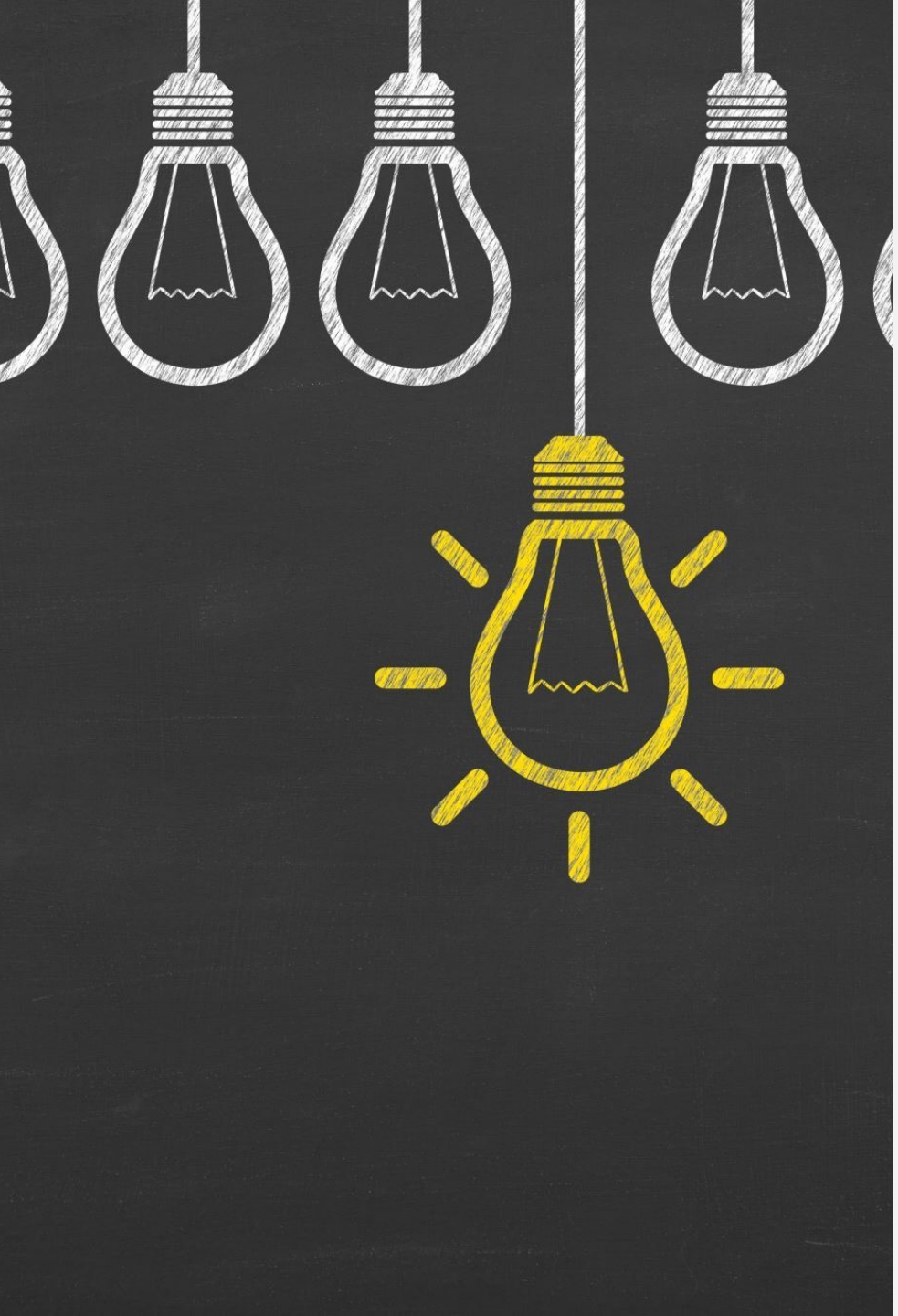
HOW IT WORKS

- We allow pruning MHA and FFN layers explicitly along with fine-grained units by introducing two additional masks zMHA and zFFN for each layer. Now the multi-head self attention and feed-forward layer become:

$$\mathrm{MHA}(X) = z_{\mathrm{MHA}} \cdot \sum_{i=1}^{N_h} (\mathbf{z}_{\mathrm{head}}^{(i)} \cdot \mathrm{Att}(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}, W_O^{(i)}, X)),$$

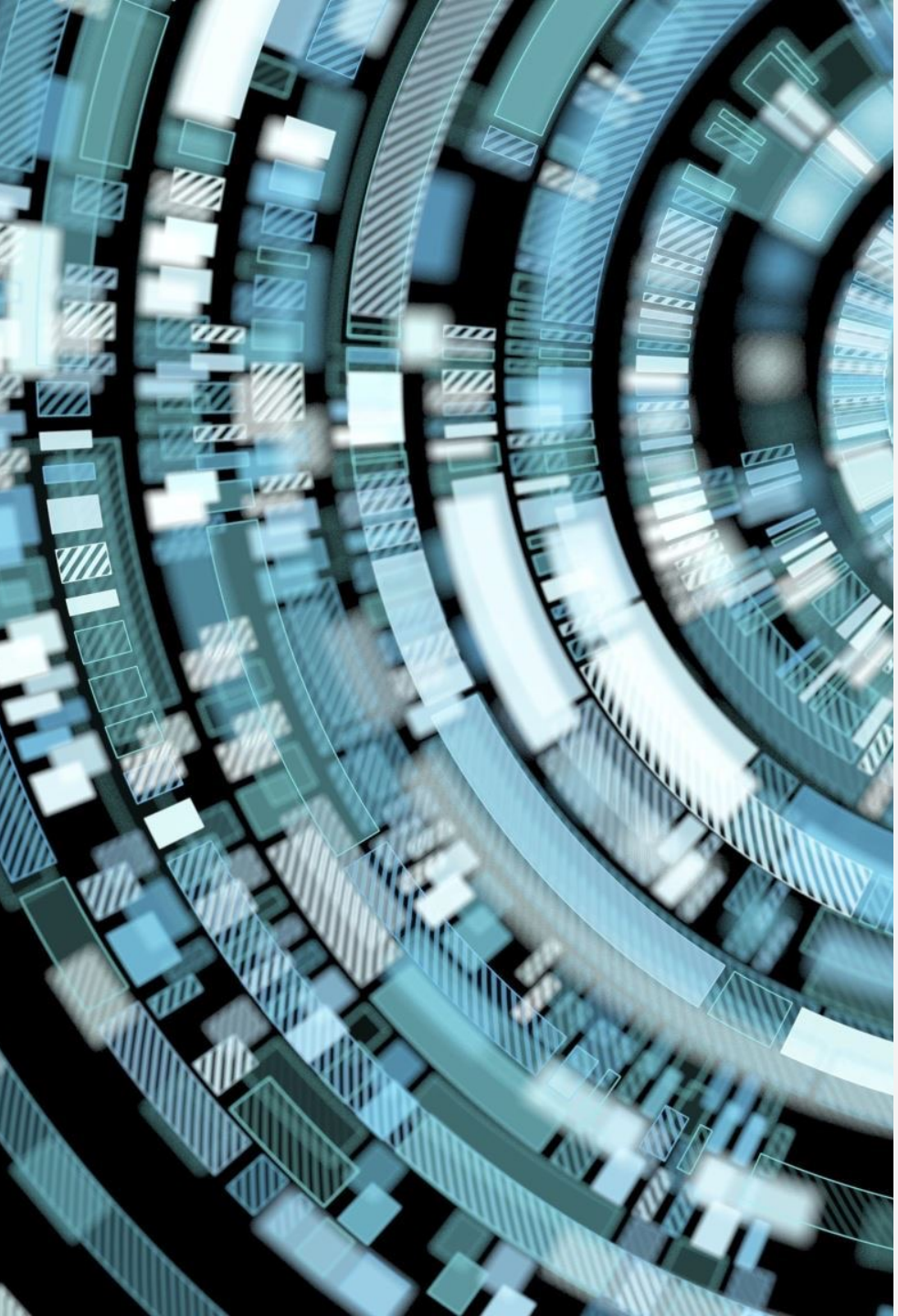
$$\mathrm{FFN}(X) = z_{\mathrm{FFN}} \cdot \mathrm{gelu}(XW_U) \cdot \mathrm{diag}(\mathbf{z}_{\mathrm{int}}) \cdot W_D.$$

- With these layer masks, we explicitly prune an entire layer, instead of pruning all the heads in one MHA layer
- (Not Shown) CoFi differs from previous pruning approaches in that multiple mask variables jointly control the pruning decision of one single parameter
- For example, a weight in an FFN layer is pruned when the entire FFN layer, or its corresponding intermediate dimension, or the hidden dimension is pruned
- (Not Shown) To learn the mask variables, we use l0 regularization modeled with hard concrete distributions



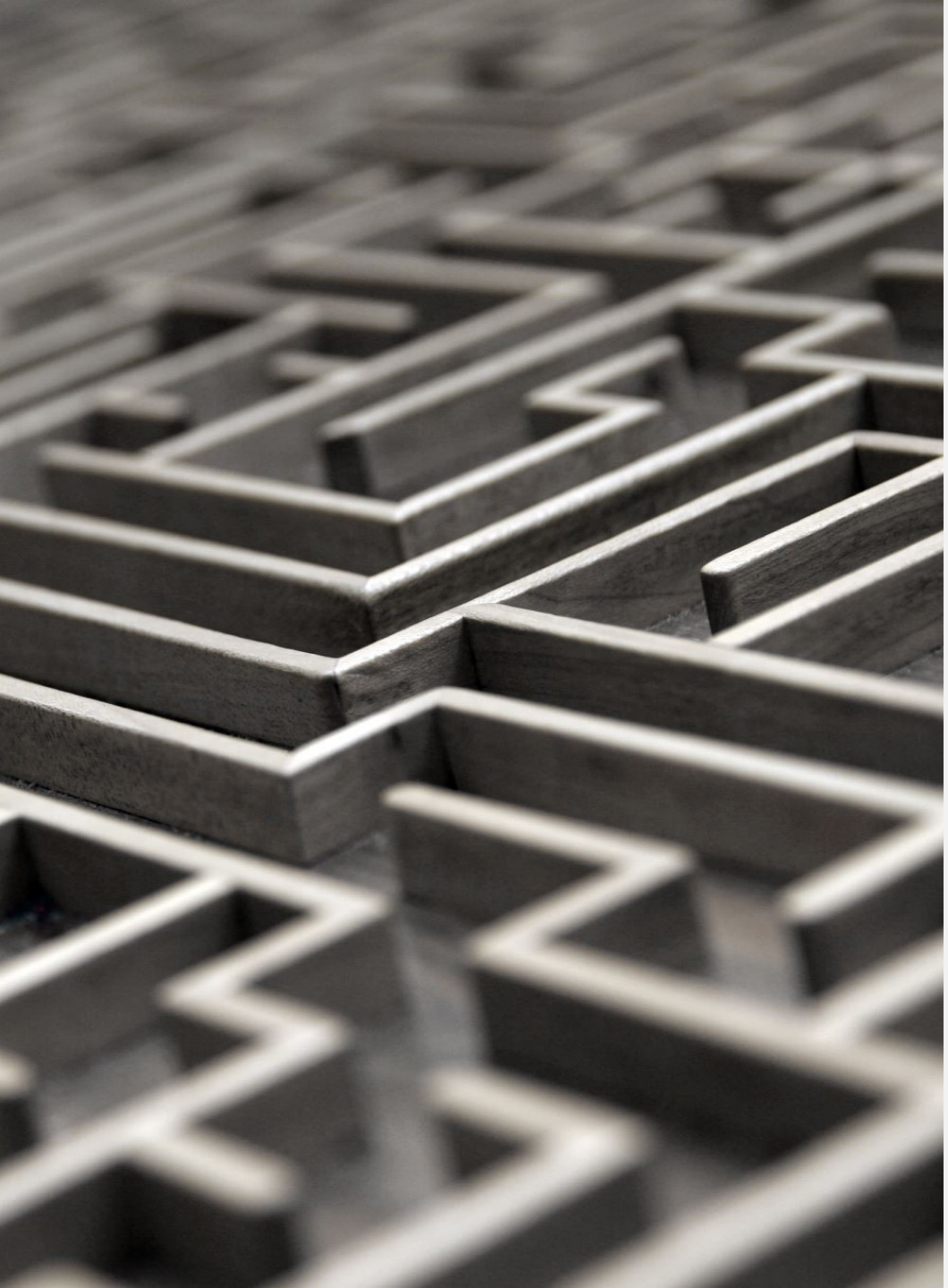
EXPERIMENTAL SETUP AND APPROACH

- Chosen Tasks
  - MNLI, CoLA, and SST-2 within GLUE tasks.
- Model Selection
  - We used the BERT model, specifically bert-base-uncased.
- Dataset Usage
  - Importance of GLUE benchmark in evaluating NLP models.
- Training Setup
  - Sparsity computed as pruned parameters divided by full model size. Distillation followed by pruning with linear sparsity increase



PROJECT CHALLENGES AND SOLUTIONS

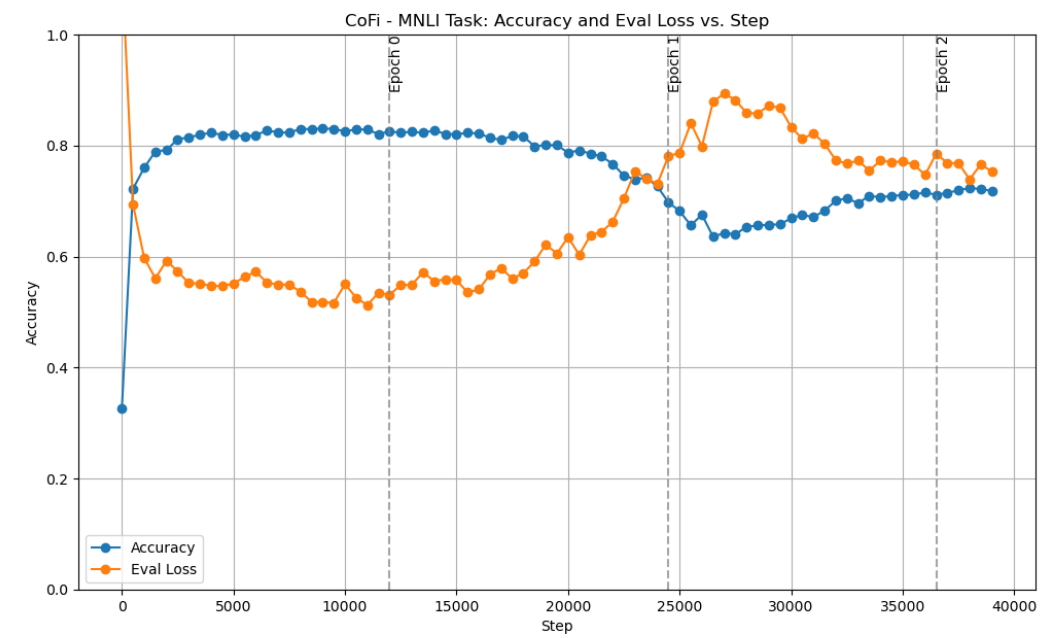
- Why use a Cloud VM?
  - Issues with teammates computers
  - Lack of CUDA Device
- Hardware/Software Limitations
  - Setting up the compute environment
- Struggles
  - Learning curve in setting up and managing cloud-based VMs.
  - Little experience with CUDA

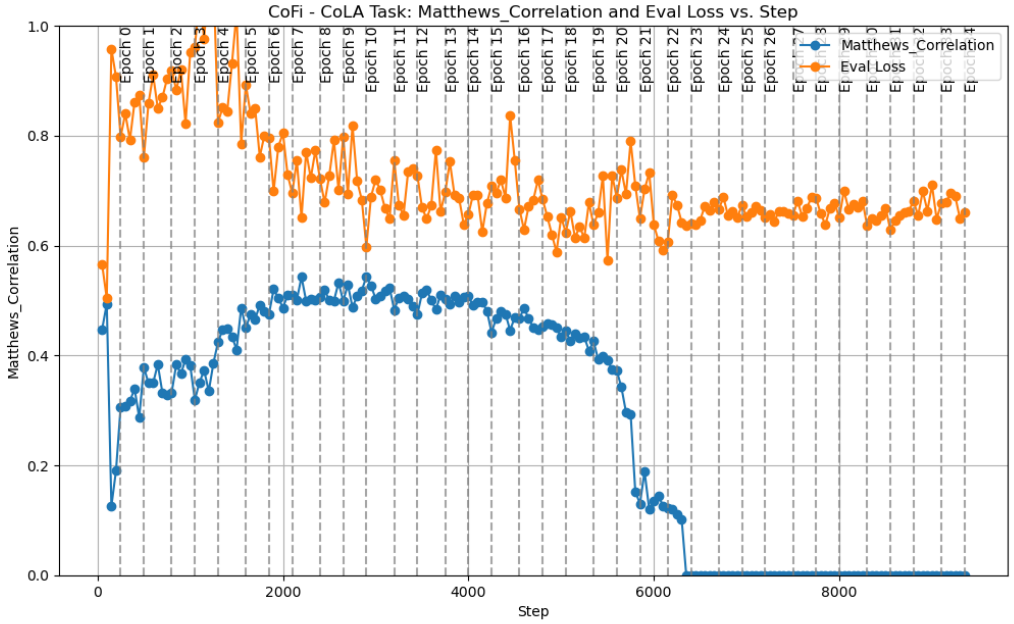


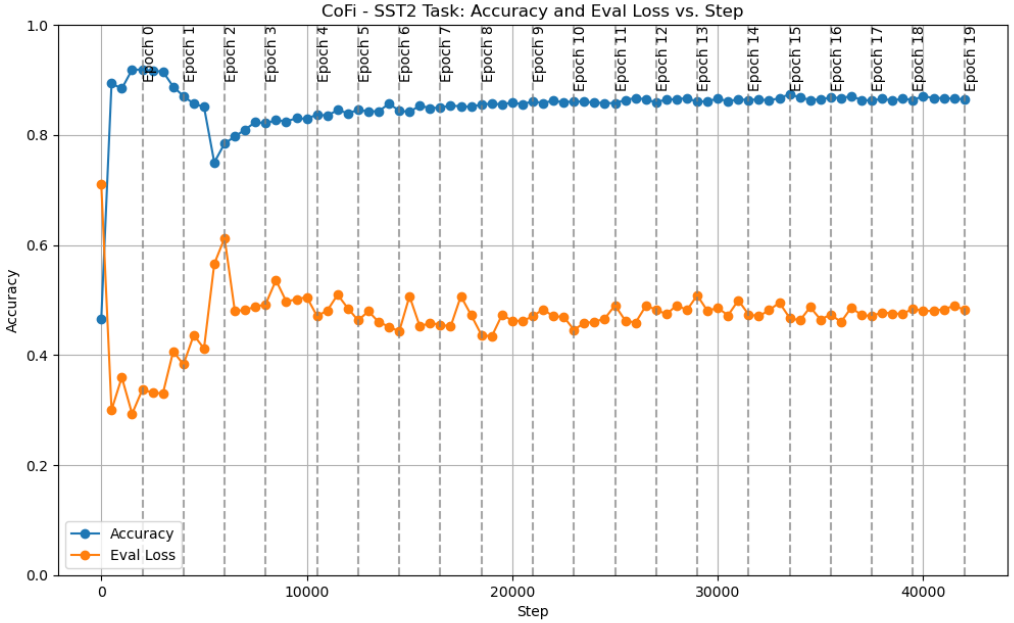
RESULTS: MODEL TRAINING

- We used trained 3 different models
  - All of them were based out of BERTbase-uncased
  - We used task specific pre-trained versions of the models from hugging face
- Plotted the accuracy and steps for each one
- Used the same CoFi pruning parameters for each
  - Sparsity: 0.95
  - Pruning Type: structured_heads+structured_mlp+hidden+layer

RESULTS


CoFi - MNLI Task: Accuracy and Eval Loss vs. Step
Epoch 0
Epoch 1
Epoch 2
Accuracy
Eval Loss
Step


CoFi - CoLA Task: Matthews_Correlation and Eval Loss vs. Step
Matthews_Correlation
Eval Loss
Step


CoFi - SST2 Task: Accuracy and Eval Loss vs. Step
Accuracy
Eval Loss
Step

ACCURACY AND SPEEDUP

- Models
  - Bert-base-uncased
    - the base bert model and the one that we use as reference
  - Princeton-nlp
    - Provided by cofi creators to verify their design
  - OurModel[TASK]
    - Task specific fine tuned version that we pruned
- Results
  - CoFi pruning results in significant improvements to tasks
  - Small accuracy penalty

MNLI Tasks		
Model Name	Accuracy	Speedup
bert-base-uncased	1	1
princeton-nlp/CoFi-MNLI-s95	0.8054	10.29575071
OurModelMNLI-epoch0	0.8182	12.36190476
SST2 Tasks	Accuracy	Speedup
bert-base-uncased	1	1
princeton-nlp/CoFi-SST2-s95	0.9037	10.4310344
OurModelSST2-epoch2	0.8658	8.9408867
CoLA Tasks	Matthew Correlation Multiplier	Speedup
bert-base-uncased	1	1
princeton-nlp/CoFi-CoLA-s95	1.573264781	9.593175853
OurModelCoLA-epoch10	1.597911227	9.092039801

CONCLUSIONS, FUTURE WORK AND LESSONS LEARNED

- Conclusion
  - Our goal was to verify the effectiveness of the CoFi pruning method
  - We confirmed the speedup and accuracy claims
- Future Work
  - Hyperparameter optimization
  - Model generalizability
- Lessons Learned
  - No longer having access to the models
  - Using Cloud Services
  - Setting up Billing

REFERENCES

- References
  - https://dl.acm.org/doi/pdf/10.1145/3580305.3599284
  - http://proceedings.mlr.press/v119/goyal20a/goyal20a.pdf
  - https://arxiv.org/pdf/2107.00910.pdf
  - https://aclanthology.org/2022.acl-long.502.pdf
  - https://github.com/princeton-nlp/CoFiPruning