# SMALL MODEL IS ALL YOU NEED

**Toh Jing Hua**[♠] **Toh Jing Qiang**[♠] **Lenson Lim**[♠]

[♠]Nanyang Technological University

{*tohj0037, tohj0038, llim071*}*@e.ntu.edu.sg*

*Code*: https://github.com/ztjhz/miniLM *Runs*: https://wandb.ai/sc4001

## ABSTRACT

The rise of large language models (LLMs) such as GPT-3 and its successors has heralded a new era in natural language processing (NLP), offering unprecedented capabilities across a wide spectrum of tasks. However, the universal application of these computational giants for all NLP tasks has been questioned, particularly when more straightforward tasks may not necessitate the heft and complexity of LLMs. This report investigates the performance of large and small language models on sentiment analysis tasks, aiming to ascertain whether such tasks genuinely benefit from the scale of LLMs. Through a novel slicing technique applied to the Llama model, we reveal that not all layers contribute equally, with middle layers often outperforming others. We extend our analysis to RoBERTa, contrasting the effectiveness of pretraining against fine-tuning on smaller datasets such as IMDb. Additionally, we fine-tune smaller models like RoBERTa, GPT-2, and T5, which demonstrate results comparable with LLMs across several benchmarks, including IMDb, SST-2, and Yelp datasets. We also did a further analysis on slicing RoBERTa and discovered that RoBERTa's capabilities is best realized through fine-tuning across all layers due to its less capable zero-shot abilities. Our findings advocate for a more nuanced selection of language models, demonstrating that smaller models can be equally adept as LLMs for certain applications, thus providing a more cost-effective and resource-efficient alternative. This project aims to guide companies and startups in choosing the right model for the right task, highlighting that while LLMs hold substantial value, they are not a one-size-fits-all solution in the realm of AI-driven language tasks.


Figure 1: Efficiency in Simplicity: A charming Mini Model robot sorts simple tasks into neat compartments, while its larger counterpart, Llama, stands by, its presence unneeded. This image was created with the assistance of DALL·E 3.

CONTENTS

# 1 INTRODUCTION

In recent advancements within the realm of Natural Language Processing (NLP), state-of-the-art Large Language Models (LLMs) such as GPT-4, Llama (Touvron et al., 2023b), and Bard have emerged as significant disruptors. These models exhibit extraordinary proficiency across a spectrum of applications — from sophisticated question-answering systems to comprehensive summarization tools and nuanced sentiment analysis.

The rapid adoption of LLMs by companies and startups is indicative of an industry trend; organizations are increasingly leveraging these powerful models for all kind of tasks. Nevertheless, this ubiquity prompts a critical inquiry: Are LLMs an optimal solution for every problem, irrespective of its complexity?

Considering the substantial computational and financial resources required to operate LLMs, it is essential to evaluate their necessity in scenarios where the tasks are relatively straightforward. This drives our investigation into the potential efficacy of Small Language Models (SLMs). Our objective is to determine whether SLMs can perform comparably well on less complex tasks, offering a more cost-efficient alternative without a significant trade-off in performance. To investigate this, we have decided on the task of sentiment analysis.

# 2 RELATED WORKS

Sentiment analysis as a field has matured, encompassing methods ranging from lexicon-based approaches to machine and deep learning techniques. Here, we outline the evolution of these methodologies, setting a foundation for our subsequent exploration into the capabilities of SLMs.

## 2.1 LEXICON-BASED APPROACHES

Lexicon-based methods have been the initial go-to for sentiment analysis, relying on dictionaries of words each tagged with sentiment scores. These methods operate under two primary frameworks: the Dictionary-Based and the Corpus-Based approaches. The former aggregates sentiment scores of words based on pre-defined dictionaries, while the latter adapts sentiment values based on word co-occurrence in large text corpora (Gupta & Agrawal, 2020).

These approaches, however, face limitations. They struggle with language's subtleties—sarcasm, idioms, and varying contexts pose significant challenges (Bonta et al., 2019). When the sentiment analysis task introduces more nuanced categories beyond simple positive or negative orientations, their effectiveness diminishes markedly (Sadia et al., 2018).

## 2.2 MACHINE LEARNING

The progression to machine learning algorithms like Naive Bayes, Support Vector Machines (Rana & Singh, 2016), and Decision Trees (Zharmagambetov & Pak, 2015) marked a significant step forward in sentiment analysis. These techniques learn from labeled data, often using a bag-of-words model that, while disregarding word order. These models focus on the frequency of words and n-grams for sentiment prediction, disregarding the order of words. However, as highlighted in Bhavitha et al. (2017), despite the numerous machine learning methods applied, there remains a considerable demand for more automated solutions. These solutions should be capable of tackling the full spectrum of challenges in sentiment analysis. This includes the effective classification of nuanced elements like indirect opinions, comparative statements, and sarcasm, areas which continue to demand further innovation and the development of more efficient methodologies.

## 2.3 DEEP LEARNING

Deep learning models have significantly advanced sentiment analysis, each presenting unique benefits while also encountering notable limitations. This discussion centers on the more prominent models.

**Convolutional Neural Networks (CNNs)** CNNs are efficient in training and effective in feature extraction but face challenges with long-term text dependencies (Dos Santos & Gatti de Bayser, 2014; Lee et al., 2018).

**Recursive Neural Networks (RecNNs)** RecNNs are adept at handling grammatical structures and negations, yet their performance diminishes with non-grammatical data, such as tweets (Socher et al., 2013).

**Recurrent Neural Networks (RNNs)** RNNs and their variants, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), excel in processing sequential data. Nevertheless, RNNs encounter issues with gradient problems (Hassan & Mahmood, 2018), and while LSTMs offer improved sequential information capture, they are computationally intensive (Rao et al., 2018). GRUs present a simpler yet less effective option for longer texts (Verma et al., 2017).

**Deep Belief Networks (DBNs)** DBNs are adept at handling large-scale vocabulary, suffer from inefficient memory functions and high computational demands (Jin et al., 2016).

In summary, while these deep learning models enhance sentiment analysis capabilities, they also face challenges in managing long-term text dependencies, computational efficiency, and memory management.

### 2.3.1 TRANSFORMERS

Following the introduction of the original transformer model by Vaswani et al. (2017), which surpassed traditional deep learning models in performance, a multitude of models and their variations have been developed, extending their application beyond just natural language processing tasks. The Transformer architecture consists of two primary components: the encoder, which interprets and processes the input, and the decoder, which generates output based on this processed information. These components can either operate independently or in unison, contingent on the specific task at hand:

- *Encoder*: Encoder-only models such as BERT (Vaswani et al., 2017) are designed to understand the context and meaning of text, making it highly effective for classification tasks. It processes the entire input sequence as a whole, allowing it to capture the context better and is often used for tasks like sentiment analysis.
- *Decoder*: Decoder-only models such as GPT-2 (Radford et al., 2019) are built for generating text. The decoder model predicts the next word in a sequence, making it adept at tasks such as text generation. For sentiment analysis, it can be adapted by using the generated tokens as a proxy for the input's sentiment.
- *Encoder-Decoder*: Encoder-Decoder models such as T5 (Raffel et al., 2020) combines the strengths of both encoders and decoders. It can understand context (like BERT) and generate text (like GPT-2), which makes it versatile for a wide range of tasks including translation and summarization.

### 2.3.2 LARGE LANGUAGE MODELS

Complementing the evolution of transformer models, Large Language Models (LLMs) have significantly enhanced transformer capabilities. LLMs such as GPT-3 (Brown et al., 2020), with its colossal 175 billion parameters, are trained on expansive text. This extensive training endows them with an exceptional ability to generate coherent and fluent text. Their efficacy in a wide array of natural language processing tasks, ranging from language translation to text summarization and conversational interfaces, is attributed to their thorough pre-training on large text corpora and subsequent fine-tuning for specific applications.

The introduction of Llama (Touvron et al., 2023a) marks a further evolution in the capabilities of LLMs, even surpassing GPT-3 in some aspects. Llama, with parameter sizes ranging from 7B to 65B, emphasizes efficiency in training and inference, resource utilization, and scalability, enabling it to outperform GPT-3 despite having fewer parameters.

# 3 METHODOLOGY

## 3.1 MODEL IMPLEMENTATION

### 3.1.1 SLICING LARGE LANGUAGE MODELS

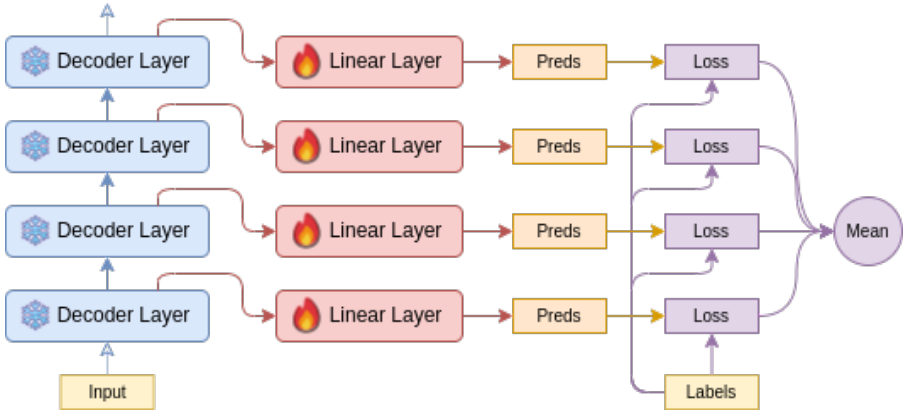



Figure 2: Sliced model architecture. Pre-trained weights are frozen and trainable linear classification layers are added to the hidden output of every latent decoder layer. With the loss calculated separately for each layer.

To investigate whether all the layers of the LLMs are needed to perform a relatively simple task of sentiment analysis, we sliced the model into different layers (Figure 2). Our choice of LLM was Llama 2 (Touvron et al., 2023b) due to its open-source accessibility. To slice the model, we froze the weights of the entire Llama model. Subsequently, we added a trainable classification layer to the output of each hidden decoder layer of the Llama model. This classification layer takes in the last token of each sequence as its input and outputs `n_labels` number of logits. This was done to evaluate the contribution of each individual layer to the overall task.

Instead of naively training 32 different models with different number of layers, we leveraged the fact the the back-propagation of the losses of the different classification layers will be independent given the the model weights of the original model is frozen, ensuring that updates to the model weights of each classification layer will be independent. The proof is provided in Section 3.1.2 and the experiment details are found in Section 4.1.

### 3.1.2 PROOF THAT WEIGHTS UPDATE IS INDEPENDENT

This proof demonstrates that the update of weights in the classification layers are independent of each other.

Let

- $W_i$ be the weight matrix of the $i^{\text{th}}$ linear classification layer,
- $x_i$ be the input vector to the $i^{\text{th}}$ linear classification layer,
- $y$ be the ground truth labels,
- $C$ be the number of classes.

The output of each linear classification layer is given by $\hat{y}_i = W_i x_i$:

$$< \hat{y}_0, \hat{y}_1, ..., \hat{y}_n > = < W_0 x_0, W_1 x_1, ..., W_n x_n > \tag{1}$$

The loss for the $i^{\text{th}}$ linear layer is cross entropy loss, where

$$Loss(y, \hat{y}_i) = -\sum_{c=1}^{C} y_c \log(\hat{y}_{i,c}) \tag{2}$$

The combined mean loss is as follows:

$$L = \frac{1}{n}(Loss(\hat{y}_0, y) + Loss(\hat{y}_1, y) + ... + Loss(\hat{y}_n, y)) \tag{3}$$

During back-propagation, partial derivatives of the loss are calculated with respect to the individual weights and is given by Equation 4. Expanding the function, we get Equation 6. Given that $W_i$ is independent of $W_j$ for $i \neq j$, we can simplify the equation to get Equation 7.

$$\frac{\partial L}{\partial W_i} = \frac{\partial}{\partial W_i}\frac{1}{n}(Loss(\hat{y}_0, y) + Loss(\hat{y}_1, y) + ... + Loss(\hat{y}_n, y)) \tag{4}$$

$$= \frac{\partial}{\partial W_i}\frac{1}{n}(-\sum_{c=1}^{C} y_c \log(\hat{y}_{0,c}) - \sum_{c=1}^{C} y_c \log(\hat{y}_{1,c}) - ... - \sum_{c=1}^{C} y_c \log(\hat{y}_{n,c})) \tag{5}$$

$$= \frac{\partial}{\partial W_i}\frac{1}{n}(-\sum_{c=1}^{C} y_c \log(W_0 x_{0c}) - \sum_{c=1}^{C} y_c \log(W_1 x_{1c}) - ... - \sum_{c=1}^{C} y_c \log(W_n x_{nc})) \tag{6}$$

$$= \frac{\partial}{\partial W_i}\frac{1}{n}(-\sum_{c=1}^{C} y_c \log(W_i x_{ic})) \quad \text{(since } W_j \text{ is independent of } W_i \text{ for } j \neq i) \tag{7}$$

Subsequently, the update rule for the weights $W_i$ of the $i^{\text{th}}$ linear layer is determined by the gradient of the loss $L$ with respect to $W_i$:

$$W_i \leftarrow W_i - \eta \frac{\partial L}{\partial W_i} \tag{8}$$

Since $\frac{\partial L}{\partial W_i}$ only depends on $W_i$ and $x_i$ as shown in Equation 7 and $x_i$ is independent of $x_j$ for $i \neq j$ as shown in Figure 2, the update of weights in Equation 8 is independent. Hence, we conclude that $W_i$ is updated independently of $W_j$ for all $j \neq i$. We have also provided an alternative proof in Appendix G.

#### 3.1.3 Small Language Models

To assess the efficacy of SLMs in comparison with LLMs, we conducted a comparative analysis using three distinct SLMs, each with a unique transformers model architecture. The models selected for this evaluation were RoBERTa (Liu et al., 2019b), an encoder-only model; GPT-2 (Radford et al., 2019), a decoder-only model; and T5 (Raffel et al., 2020), an encoder-decoder model. These models were chosen to represent a broad spectrum of language processing capabilities within the domain of small transformers model architectures. For the implementation and initialization of our SLMs, we utilized the Hugging Face Transformers library (Wolf et al., 2020) which is an open-source library with model implementations and pre-trained weights.

### 3.2 Dataset

For our study, we utilized the IMDb dataset (Maas et al., 2011), the Yelp Review Full dataset (Zhang et al., 2015), and the Stanford Sentiment Treebank (SST-2) (Socher et al., 2013). This combination was selected to challenge the model's adaptability across varying domains within the sentiment analysis landscape.

**IMDb Dataset** The IMDb dataset contains movie reviews tagged as either positive or negative. This was used for binary classification.

**Yelp Review Dataset** The Yelp dataset contains business reviews tagged with multiple rating classes (1-5). This was used for non-binary classification.

**Subset of Yelp Dataset** Confronted with time constraints posed by the voluminous nature of the Yelp Review Full dataset, which contains over 650,000 training entries.

**SST-2 Dataset** The SST-2 dataset consists of movie reviews from Rotten Tomatoes with sentences labeled as positive or negative. It has fine-grained sentiment annotations at the sentence level, allowing us to assess model performance in a more granular context.

### 3.3 Training Configuration

**Loss Function** For our training objectives, we employed cross-entropy loss which measures the difference between the predicted probability distribution and the actual distribution, with a lower value indicating better performance. Cross-entropy loss is particularly suited for our binary classification task on the IMDb dataset and the multi-class classification on the Yelp dataset, as it penalizes incorrect classifications in a manner that scales with the confidence of the prediction (Mao et al., 2023).

**Optimizer** We opted for the AdamW optimizer (Loshchilov & Hutter, 2019) with its default parameters. AdamW is an extension of the original Adam optimizer (Kingma & Ba, 2017) that incorporates a decoupled weight decay regularization, which helps in preventing overfitting. This choice is motivated by the optimizer's ability to adapt the learning rate for each parameter individually, leading to more efficient training and convergence, especially in complex models with large datasets.

**Evaluation Metrics** To comprehensively evaluate the performance of our models, we tracked a suite of metrics:

1. **Accuracy:** The proportion of correct predictions over the total number of cases evaluated. This gives us a straightforward measure of the model's overall performance.
2. **Precision:** The ratio of true positive predictions to the total positive predictions. High precision indicates a low rate of false positives, which is crucial in applications where the cost of a false positive is high.
3. **Recall:** This measures the proportion of actual positives that were correctly identified. It is particularly important in scenarios where missing a positive instance is costly.
4. **F1 Score:** The harmonic mean of precision and recall, this metric provides a balance between them and is useful when we seek a model that maintains a balance between precision and recall.
5. **AUC (Area Under the ROC Curve):** This metric measures the ability of the model to distinguish between classes and is robust to class imbalance.

### 3.4 Training Environment

**Google Colab** For preliminary experiments, we utilized Google Colab for two initial runs. Google Colab offered a no-cost platform that allows us to quickly set up and test our models. This environment was particularly suitable for our initial runs as it allowed us to validate our setup and models without incurring infrastructure costs.

**Advanced GPU Setup** The bulk of our training, totaling 17 runs, was conducted on a more robust setup consisting of 4x A100 80GB GPUs. These GPUs are currently among the most powerful available for machine learning tasks and provided the necessary computational efficiency and capacity to handle the extensive training demands of our LLMs and SLMs.

### 3.5 Training Optimisation

To enhance the efficiency and scalability of our training process, we integrated DeepSpeed (Rasley et al., 2020), a cutting-edge deep learning optimization library designed to accelerate deep learning. DeepSpeed offers a suite of features such as model parallelism, pipeline parallelism, and kernel optimizations, which are pivotal in optimizing training time and resource utilization. DeepSpeed's ZeRO optimizer (Rajbhandari et al., 2020) played a crucial role in our setup. By partitioning model states across available GPUs, it dramatically reduced memory usage, allowing us to train LLMs such as Llama.

### 3.6 Results Logging

**Integration with Weights & Biases** For logging and tracking the results of our experiments, we integrated Weights & Biases (W&B) (Biewald, 2020) which provides a comprehensive suite for machine learning experiment tracking such as real-time logging of metrics, interactive dashboards, real time collaboration and sharing, among others. Our W&B runs can be found at `https://wandb.ai/sc4001/text-sentiment-analysis`.

**Result Reproducibility** As W&B logs the exact state of our models, including the architecture, hyperparameters, and the optimizer state, it ensures the reproducibility of our results.

## 4 Experiments

### 4.1 Slicing Large Language Models (Llama)

LLMs are renowned for their versatility, capable of executing a vast array of tasks. However, when considering the straightforward task of binary sentiment analysis, one might question the necessity of utilizing the model's full depth. To investigate this, we conducted experiments to find out whether all layers are indeed required.

As outlined in section 3.1.1, we began by freezing the pre-trained weights of Llama 2 so that the foundational language understanding of the model is preserved. To each hidden decoder layer, we attached a trainable linear classification layer. This was done to evaluate the contribution of each individual decoder layer to the overall task of sentiment analysis.

We then fine-tuned the model on the IMDb dataset and the Yelp Review Full subset dataset with a batch size of 8. Detailed outcomes, along with the corresponding visualizations, are provided in Appendix A for the IMDb dataset and in Appendix B for the Yelp subset dataset.

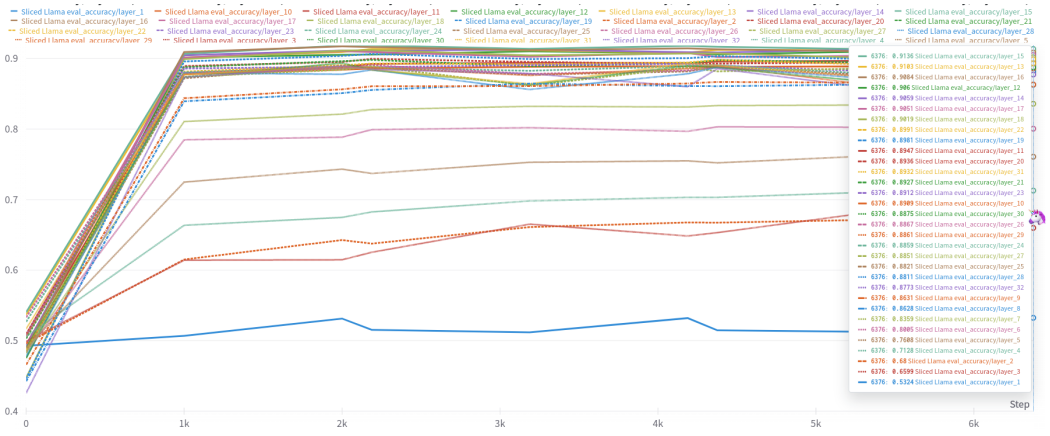



Figure 3: Comparison of the evaluation accuracy of the different layers of the Sliced Llama model on the Imdb dataset.

Interestingly, Figure 3 shows an interesting relationship between the model's layer count and its evaluation accuracy. The graph depicts an initial uptrend in accuracy as the number of layers rises

from 0. Notably, the accuracy peaks at around the layer-15 mark, beyond which we observe a decline. This indicates a diminishing return on model performance with the addition of extra layers beyond this optimal point, implying that increasing the complexity of the language model does not necessarily equate to better performance for the comparatively simpler task of binary sentiment analysis. In constrast, having more layers may harm the model performance instead.

This phenomenon highlights the importance of selecting the correct model architecture to align with the complexity of the task at hand in order to avoid falling into the pitfalls of unnecessary computational overhead.

### 4.2 Efficiency of Fine-Tuning Over Training from Scratch on Limited Data

Before analysing whether SLMs can performance just as well as LLMs, there is a need to decide whether to pre-train or fine-tune the model. The dichotomy between pre-training and fine-tuning in language model training can be especially pronounced when dealing with smaller datasets, such as the IMDb dataset. To ascertain the most effective approach for this dataset, we conducted an experiment comparing the two methodologies using RoBERTa, a robustly optimized BERT architecture, that is an encoder-only model making it suitable for classification task

**Pre-training** First, we pre-trained a RoBERTa model from scratch on the IMDb dataset with a batch size of 32. The model weights were initialized randomly weights without any prior knowledge of language, relying entirely on learning from the sentiment-focused reviews of IMDb. Given the relatively modest size of this dataset, the model's ability to learn and generalize was inherently limited.

**Fine-tuning** Next, we also fine-tuned a pre-trained RoBERTa with a batch size of 32. The RoBERTa had been pre-trained on a large and diverse text corpus in the original paper (Liu et al., 2019a). The pretrained RoBERTa model has a broad understanding of natural language and the fine-tuning process tailored this pre-acquired knowledge to the specific domain of movie reviews and their associated sentiments.


Figure 4: Comparison of the evaluation loss and metrics for finetuning RoBERTa and pretraining RoBERTa. Refer to Appendix C for all the graphs.

The differences in model performance are succinctly captured in Figure 4. The fine-tuned model consistently outperforms the one trained from scratch, achieving higher accuracy except in instances of data overfitting. The trained-from-scratch model, on the other hand, plateaus quickly and fails to show significant improvement, underscoring the limitations of learning from a small dataset without

the foundational knowledge provided by pre-training. This comparison highlights the advantages of fine-tuning a pre-trained model, particularly in scenarios where the available data is limited.

**Analysis** The superiority of fine-tuning a pre-trained model can be attributed to several factors:

- *Limited Data*: The IMDb dataset's relatively small size is insufficient for a model to learn the nuances of language from scratch to the same extent as it could from a larger corpus. Pre-trained models have been exposed to a diverse range of language use cases, which imbues them with a generalized understanding that is beneficial when adapting to specific tasks.
- *Transfer Learning*: By fine-tuning a pre-trained model, we are capitalizing on the transfer learning effect. The model has already learned a rich set of language features during pre-training, which can be effectively adapted to the sentiment analysis task with minimal additional data.
- *Efficiency and Time*: Although pre-training the model might achieve similar results when trained long enough, training a model from scratch requires significant computational resources and time. Fine-tuning, conversely, is more resource-efficient and can achieve better results in a shorter time frame, which is particularly advantageous when dealing with smaller datasets.

These findings highlight the efficacy of fine-tuning when working with small, domain-specific datasets. It demonstrates that even with limited data, leveraging a pre-trained model can lead to significant gains in performance, providing a compelling case for the fine-tuning approach in similar scenarios.

### 4.3 Architecture Choices for Small Language Models

Can SLMs perform just as well as LLMs? In the realm of SLMs, architecture plays a critical role in the model's ability to learn and perform on tasks such as sentiment analysis. To evaluate the efficacy of different model architecture, we tested three different architectures, namely an encoder-based model (RoBERTa), a decoder-based model (GPT-2), and an encoder-decoder-based model (T5). For each model, we conducted experiments across four datasets with a batch size of 32: IMDb, SST-2, Yelp, and a subset of Yelp. Our results can be found in Appendix D (Figure D.1 for Imdb, Figure D.2 for sst2, Figure D.3 for Yelp, and Figure D.4 for Yelp subset).


Figure 5: Comparison of the evaluation loss and metrics of finetuning RoBERTa, GPT-2, and T5 on the SST2 dataset.

Across all datasets (Figure 5 and Appendix D), the encoder-decoder model (T5) consistently outperformed the encoder-only (RoBERTa) and decoder-only (GPT-2) models in terms of accuracy. The encoder-decoder architecture of T5, which benefits from understanding context and the capacity to generate text, proves to be the most versatile and effective for sentiment analysis tasks.

It is particularly noteworthy that despite their smaller size compared to larger models like LLMs, these architectures were able to achieve similar results. This highlights the potential of small language models to deliver performance that rivals or comes close to their larger counterparts for less complex task.

### 4.4 Slicing Small Language Models (RoBERTa)

The concept of slicing language models to assess the capabilities of individual layers within the model has been demonstrated effectively in LLMs such as Llama 2. Hence, we wanted to find out whether a similar approach can be applied to SLMs, specifically RoBERTa.

We followed the same slicing methodology highlighted in Section 4.1 with the only difference being that the linear classification layer is added to the hidden encoder layer with the first token of each sequence being passed into the classification layer. We fine-tuned on the Imdb dataset and the Yelp subset dataset with a batch size of 8 and our findings can be found in Appendix E and Appendix F respectively.

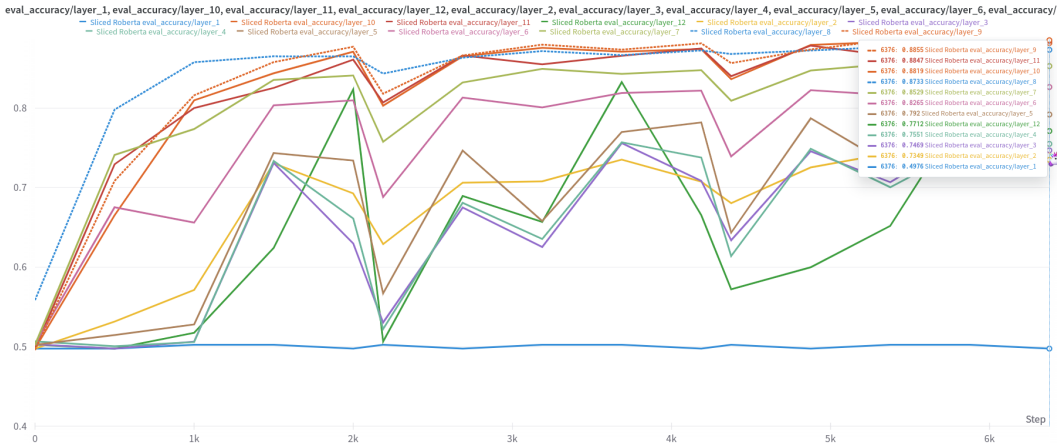



Figure 6: Comparison of the evaluation accuracy of the different layers of the Sliced RoBERTa model on the Imdb dataset.

Our findings as shown in Figure 6 revealed 3 key insights:

- The last few layers of RoBERTa demonstrated the highest accuracy, indicating that these layers contain the most refined representations for the sentiment analysis task.
- The accuracy was lower when using sliced RoBERTa models compared to both sliced Llama and fully fine-tuned RoBERTa models. This points to the necessity of fine-tuning the entire model to achieve optimal performance.
- The need to fine-tune the entire model rather than just the classification layers may be attributed to RoBERTa's less effective zero-shot capabilities compared to LLMs like Llama. RoBERTa, being a smaller model, relies more heavily on fine-tuning across all its layers to adapt to specific tasks.

## 5 Limitations and Future Work

### 5.1 Hyperparameter Tuning

Hyperparameter tuning is critical for optimizing model performance. However, it can be a time and resource intensive process, and our study faced constraints that limited the extent of hyperparameter

exploration. Given more time, more work would be done on tuning batch size and learning rates and exploring other optimizers such as Distributed Shampoo (Anil et al., 2021) and LAMB (You et al., 2020).

### 5.2 Micro Models

In our study, we focused on models like RoBERTa, GPT-2, and T5, which are considered small relative to larger models like Llama. However, there are even smaller models that may provide benefits in terms of efficiency and cost for certain applications, especially in resource-constrained environment. Given more time, more work would be done on exploring the impact of scaling down model size on the balance between performance, efficiency, and cost. More work can also be done on developing new metrics to evaluate model efficiency by considering both performance and resource consumption so as to aid in selecting the optimal model size for specific use cases.

### 5.3 Diverse Data

Our datasets were limited to sentiment analysis in English. Including a diverse array of datasets, across different languages and domain, would test our theory more rigourously. The ability of these models to adapt to different domains (e.g., medical, legal) was not covered and remains an area for further exploration.

## 6 Conclusion

Our investigation into the performance of LLMs and SLMs reveals a nuanced landscape of model applicability. Our findings highlight that the efficacy of a model is highly dependent on the complexity of the task at hand. For straightforward applications such as binary sentiment analysis, a monolithic approach using LLMs is not always warranted. The middle layers of sliced LLaMA models exhibited sufficient proficiency, challenging the assumption that larger models are universally superior.

Moreover, SLMs like RoBERTa, when fine-tuned, demonstrated that they could hold their own against the might of LLMs. This observation suggests that a tailored approach, where model selection is customized to the task's requirements, is crucial for efficient and effective problem-solving.

In light of these insights, businesses, particularly startups with limited resources, should consider the trade-offs carefully. The allure of LLMs, driven by their impressive zero-shot and few-shot capabilities, should be weighed against the nature of the problem to be solved. For tasks that do not require the full breadth of knowledge and capabilities of an LLM, smaller models can be more appropriate, offering savings in computational and financial resources.

Ultimately, this report advocates for a strategic choice in model deployment. The objective should not be to utilize the largest model available, but to employ the most suitable one. By aligning the model's capabilities with the task's demands, organizations can ensure a balance between performance, cost, and resource utilization.

## REFERENCES


Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Towards practical second order optimization for deep learning, 2021. URL https://openreview.net/forum?id=Sc8cY4Jpi3s.

B. K. Bhavitha, Anisha P. Rodrigues, and Niranjan N. Chiplunkar. Comparative study of machine learning techniques in sentimental analysis. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 216–221, 2017. doi: 10.1109/ICICCT.2017.7975191.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Venkateswarlu Bonta, Nandhini Kumaresh, and N Janardhan. A comprehensive study on lexicon based approaches for sentiment analysis. *Asian Journal of Computer Science and Technology*, 8 (S2):1–6, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Cicero Dos Santos and Maira Gatti de Bayser. Deep convolutional neural networks for sentiment analysis of short texts. 08 2014.

Neha Gupta and Rashmi Agrawal. Chapter 1 - application and techniques of opinion mining. In Siddhartha Bhattacharyya, Václav Snášel, Deepak Gupta, and Ashish Khanna (eds.), *Hybrid Computational Intelligence*, Hybrid Computational Intelligence for Pattern Analysis and Understanding, pp. 1–23. Academic Press, 2020. ISBN 978-0-12-818699-2. doi: https://doi.org/10.1016/B978-0-12-818699-2.00001-9. URL https://www.sciencedirect.com/science/article/pii/B9780128186992000019.

A Hassan and A Mahmood. Convolutional recurrent deep learning model for sentence classification. ieee access 6: 13949–13957, 2018.

Yong Jin, Harry Zhang, and Donglei Du. Improving deep belief networks via delta rule for sentiment classification. *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 410–414, 2016. URL https://api.semanticscholar.org/CorpusID:7618183.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Gichang Lee, Jaeyun Jeong, Seungwan Seo, CzangYeob Kim, and Pilsung Kang. Sentiment classification with word localization based on weakly supervised learning with a convolutional neural network. *Knowledge-Based Systems*, 152:70–82, 2018.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019a. URL http://arxiv.org/abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

Shweta Rana and Archana Singh. Comparative analysis of sentiment orientation using svm and naive bayes techniques. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pp. 106–111. IEEE, 2016.

Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

Azeema Sadia, Fariha Khan, and Fatima Bashir. An overview of lexicon-based approach for sentiment analysis. In *2018 3rd International Electrical Engineering Conference (IEEC 2018)*, pp. 1–6, 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Sharad Verma, Mayank Saini, and Aditi Sharan. Deep sequential model for review rating prediction. pp. 1–6, 08 2017. doi: 10.1109/IC3.2017.8284318.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Arman S Zharmagambetov and Alexandr A Pak. Sentiment analysis of a document using deep learning approach and decision trees. In *2015 Twelve international conference on electronics computer and computation (ICECCO)*, pp. 1–4. IEEE, 2015.

# A GRAPHS FOR SLICED LLAMA (IMDB)

For the original graphs, please visit https://api.wandb.ai/links/sc4001/vz711n0m.

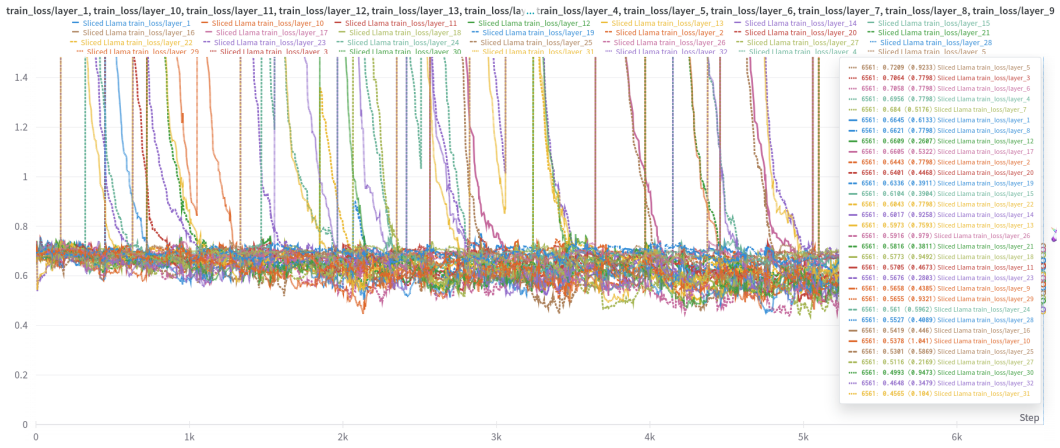



Figure A.1: Train loss for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

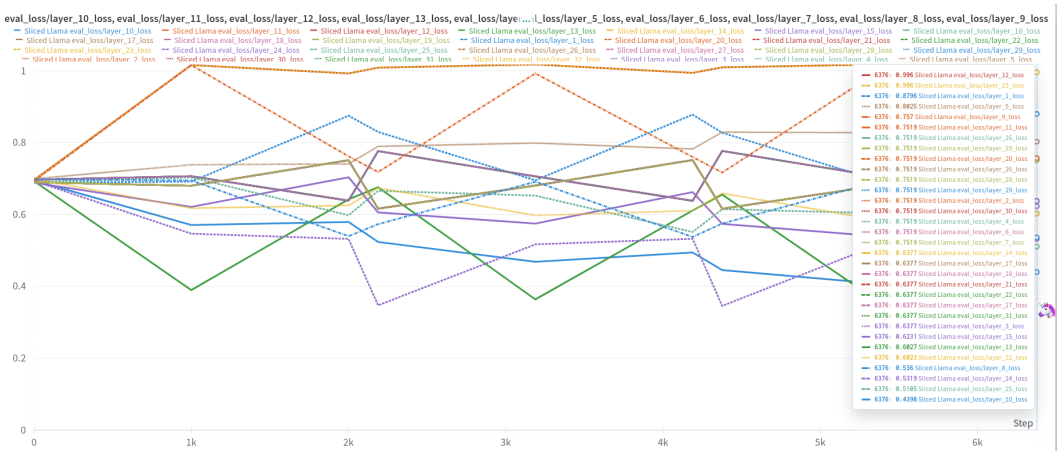



Figure A.2: Evaluation loss for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

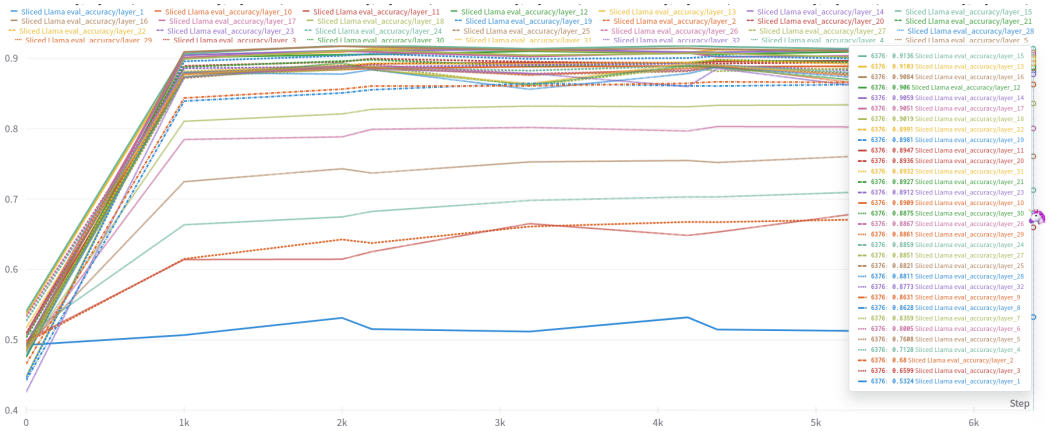


Figure A.3: Evaluation accuracy for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

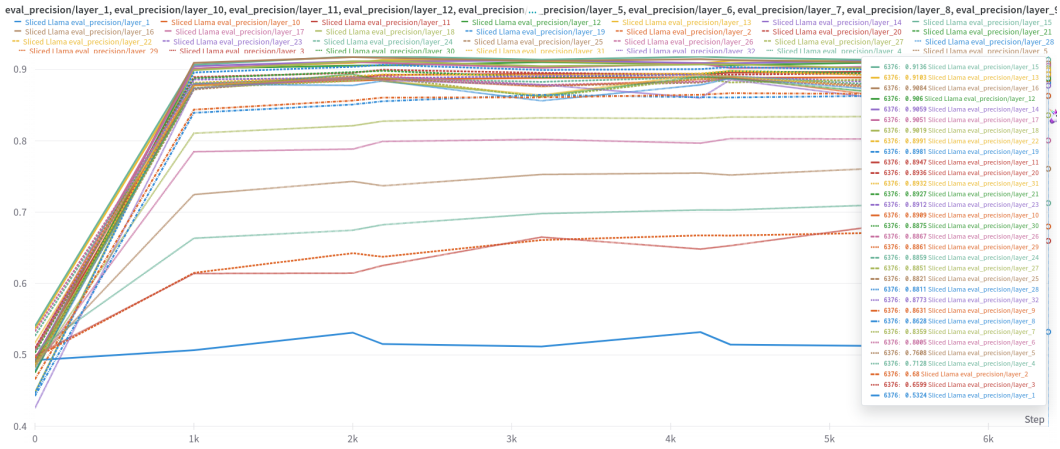


Figure A.4: Evaluation precision for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

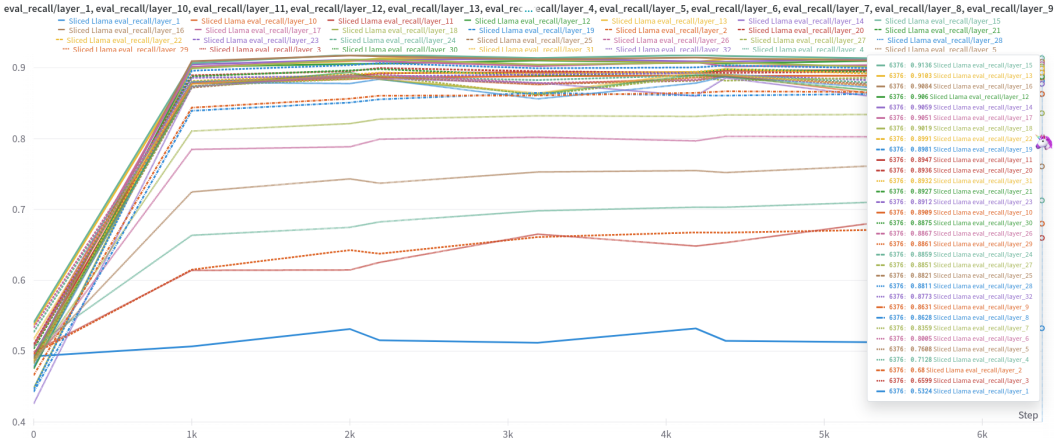


Figure A.5: Evaluation recall for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

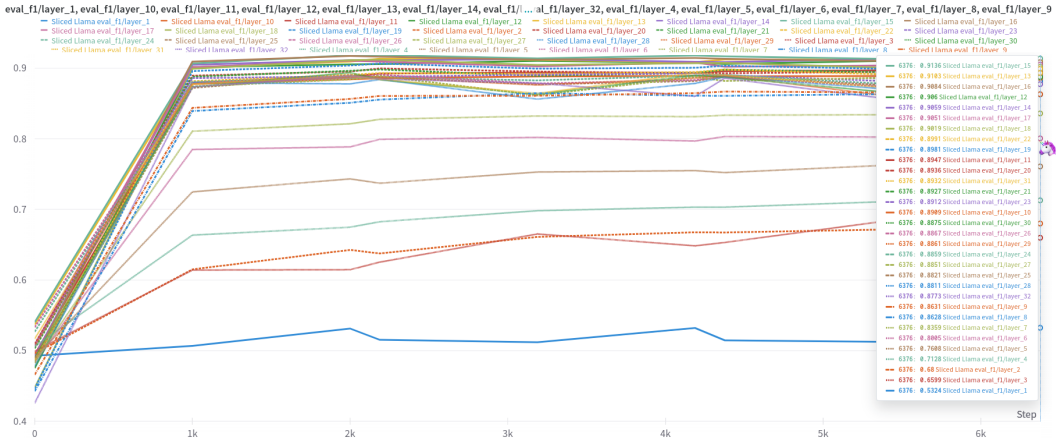



Figure A.6: Evaluation F1 score for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

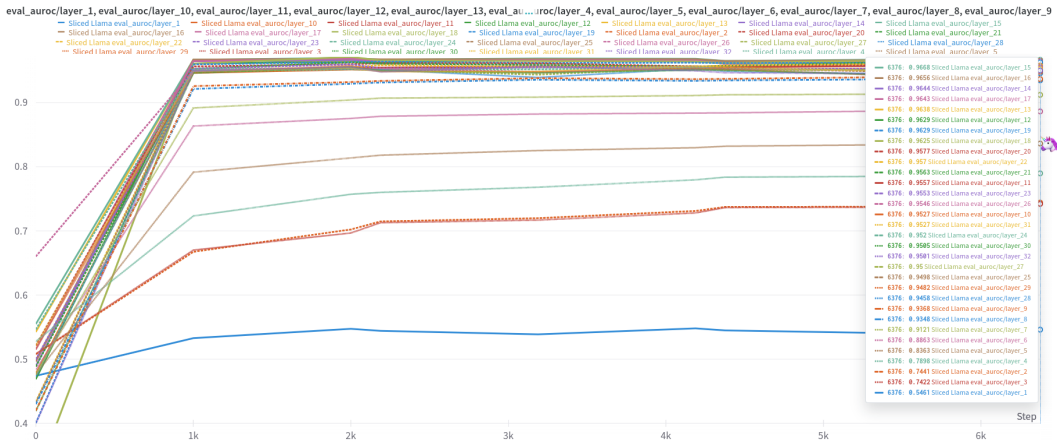



Figure A.7: Evaluation AUROC for the 32 classification layers of the Sliced Llama model for the Imdb Dataset.

## B GRAPHS FOR SLICED LLAMA (YELP SUBSET)

For the original graphs, please visit https://api.wandb.ai/links/sc4001/vz711n0m.

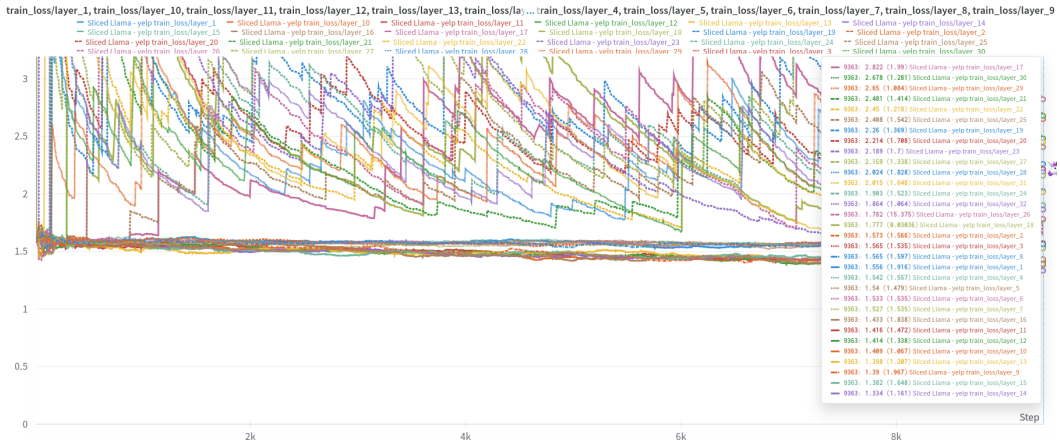



Figure B.1: Train loss for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

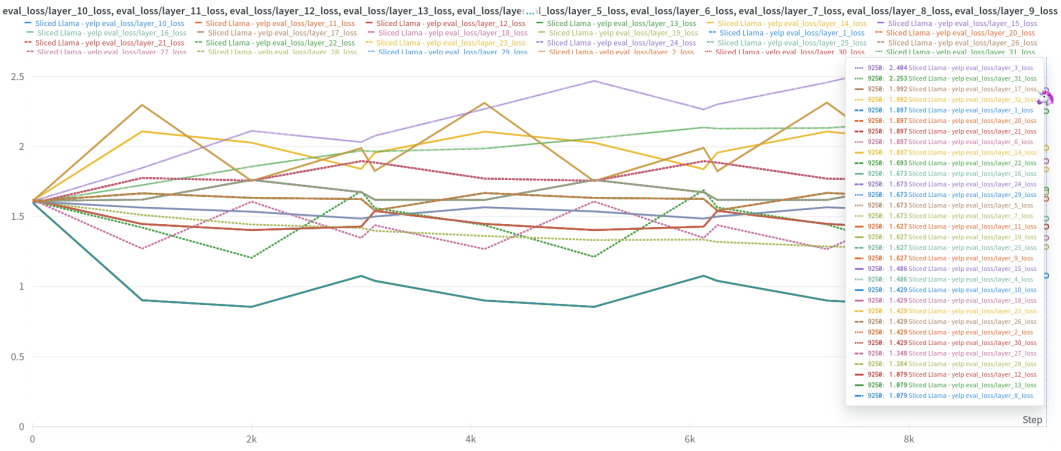



Figure B.2: Evaluation loss for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

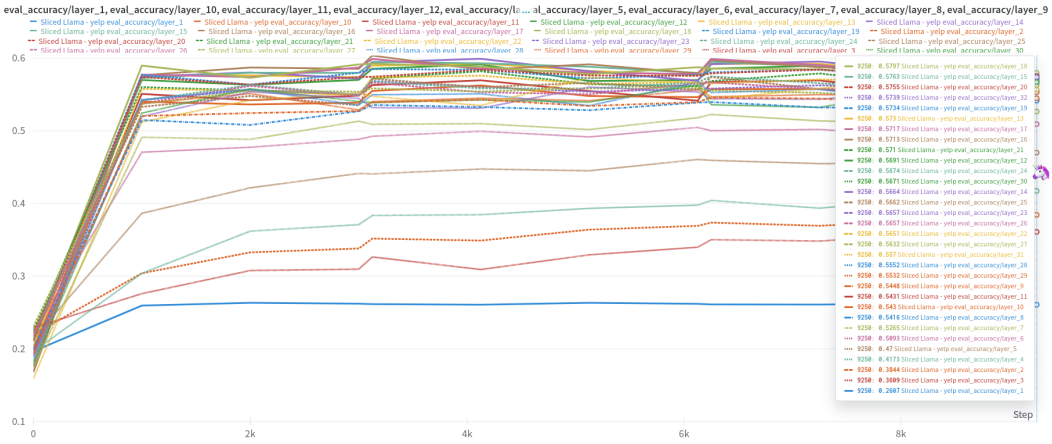



Figure B.3: Evaluation accuracy for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

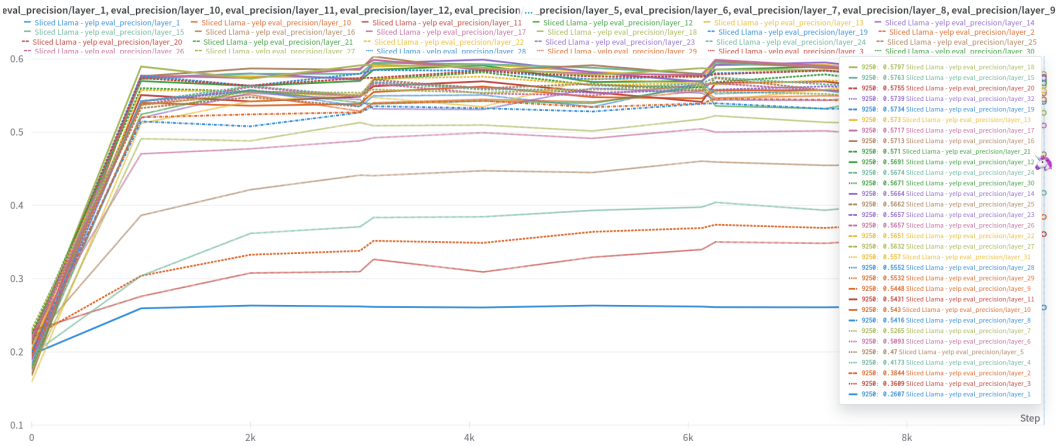



Figure B.4: Evaluation precision for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

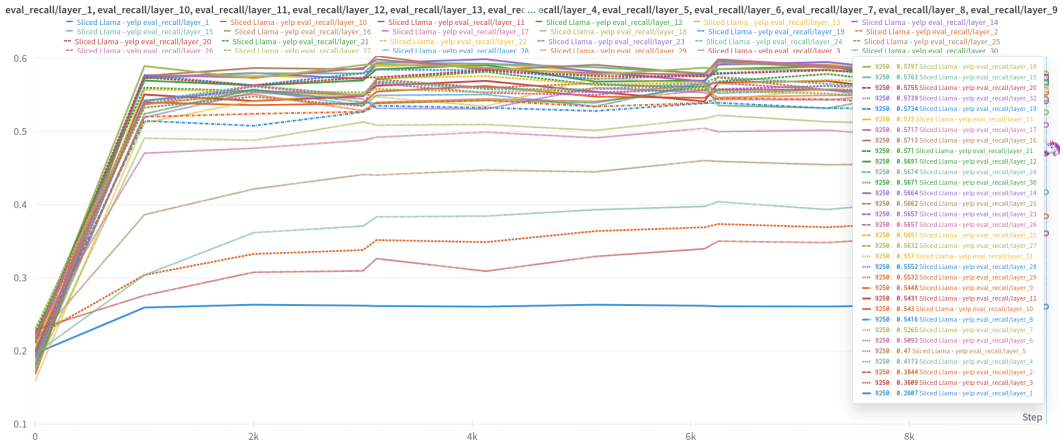



Figure B.5: Evaluation recall for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

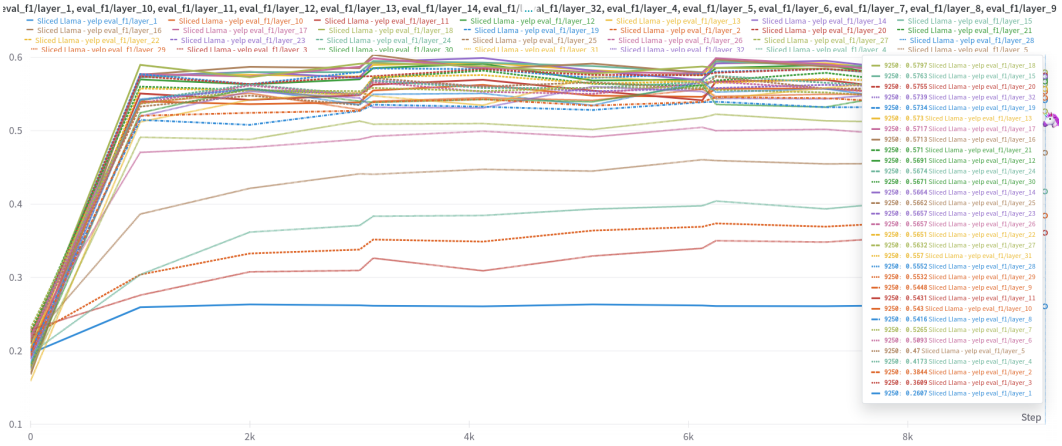



Figure B.6: Evaluation F1 score for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

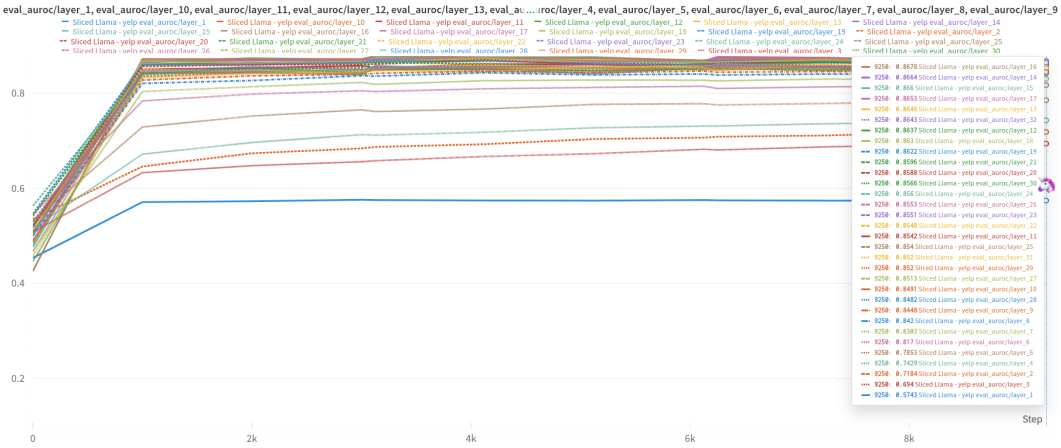
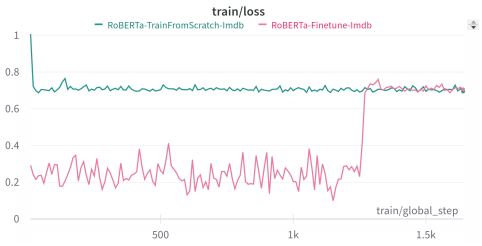


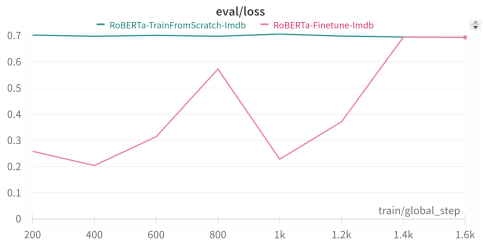

Figure B.7: Evaluation AUROC for the 32 classification layers of the Sliced Llama model for the Yelp Subset Dataset.

## C GRAPHS FOR PRE-TRAINING VS FINE-TUNING

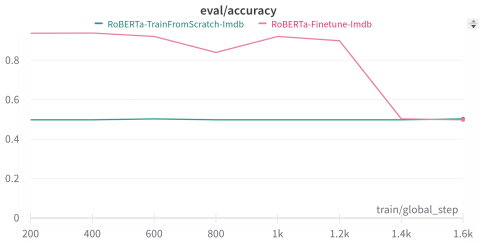
For the original graphs, please visit https://api.wandb.ai/links/sc4001/vz711n0m.



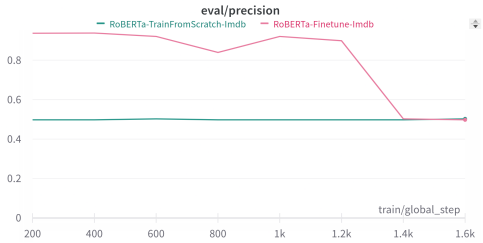

(a) Train loss.



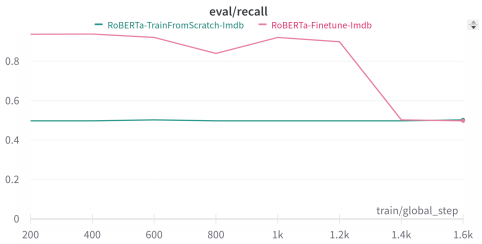

(b) Evaluation loss.



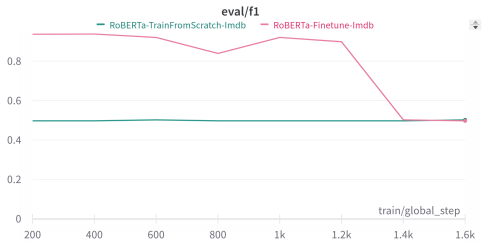

(c) Evaluation accuracy.



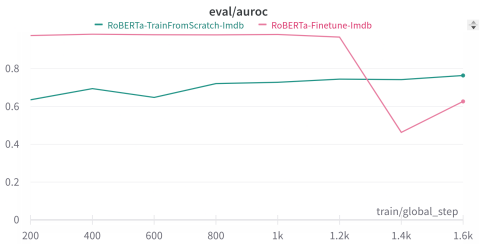

(d) Evaluation precision.




(e) Evaluation recall.



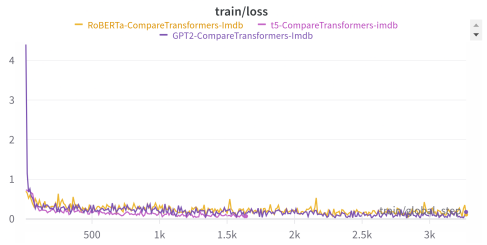

(f) Evaluation F1 score.



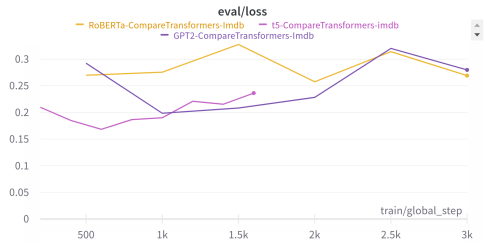

(g) Evaluation AUROC.

Figure C.1: Train loss, evaluation loss and evaluation metrics of pre-training and fine-tuning RoBERTa.

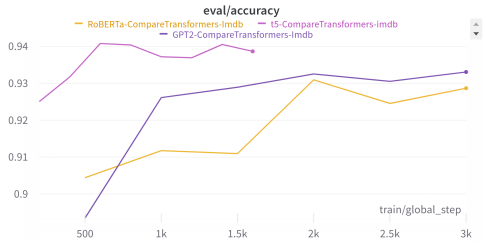
## D GRAPHS FOR COMPARING TRANSFORMERS ARCHITECTURE



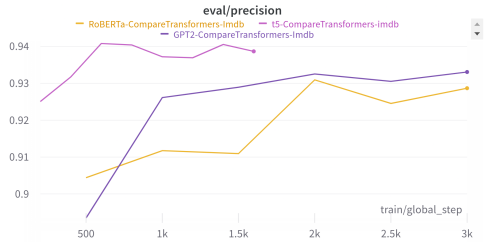

(a) Train loss.



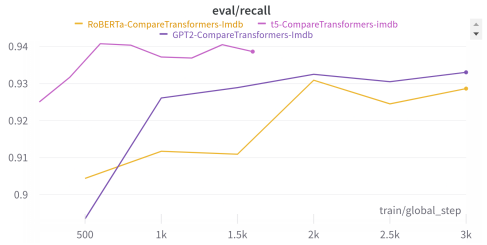

(b) Evaluation loss.



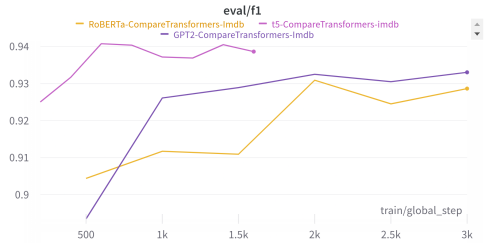

(c) Evaluation accuracy.



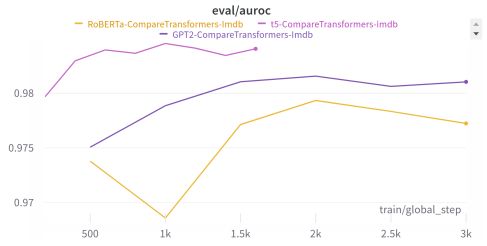

(d) Evaluation precision.




(e) Evaluation recall.

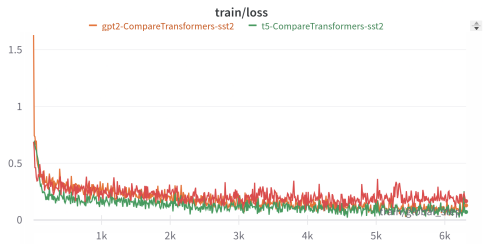



(f) Evaluation F1 score.

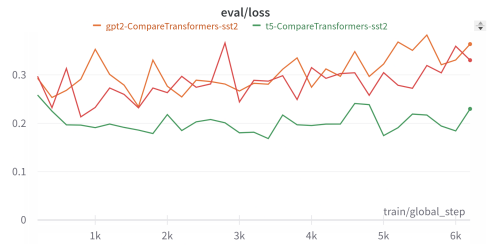



(g) Evaluation AUROC.

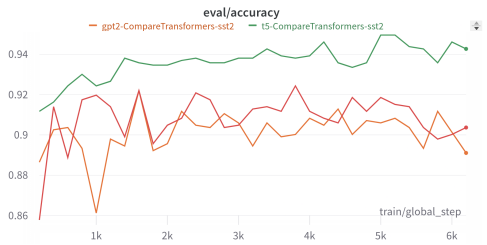
Figure D.1: Train loss, evaluation loss and evaluation metrics of RoBERTa, GPT-2 and T5 on the Imdb dataset.



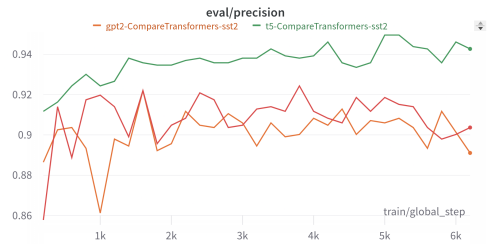

(a) Train loss.



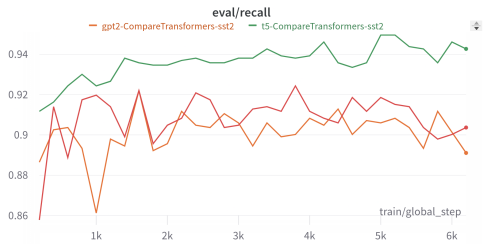

(b) Evaluation loss.



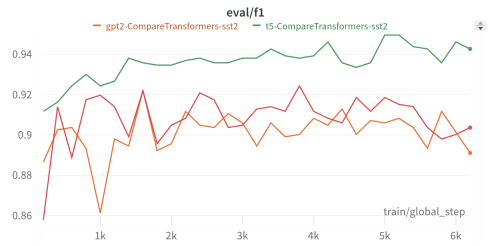

(c) Evaluation accuracy.



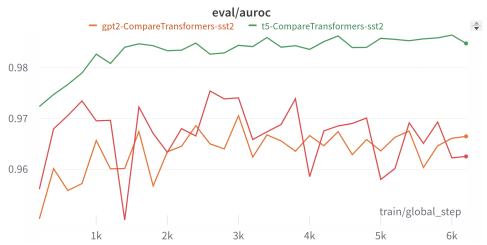

(d) Evaluation precision.




(e) Evaluation recall.

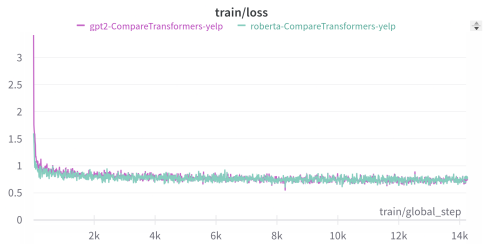



(f) Evaluation F1 score.

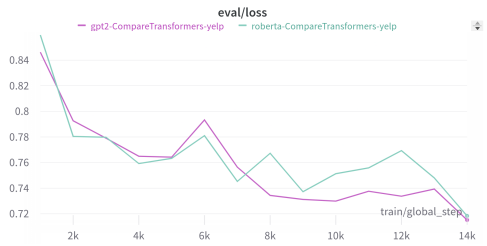



(g) Evaluation AUROC.

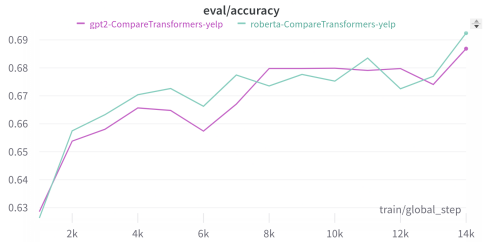
Figure D.2: Train loss, evaluation loss and evaluation metrics of RoBERTa, GPT-2 and T5 on the sst2 dataset.



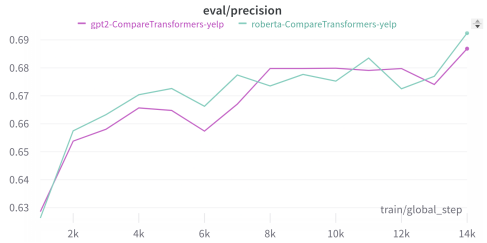

(a) Train loss.



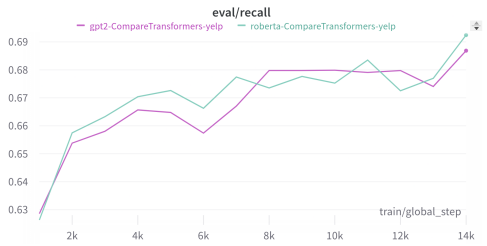

(b) Evaluation loss.



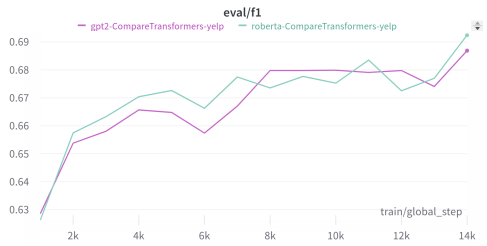

(c) Evaluation accuracy.



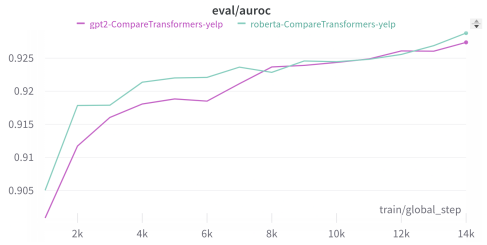

(d) Evaluation precision.




(e) Evaluation recall.

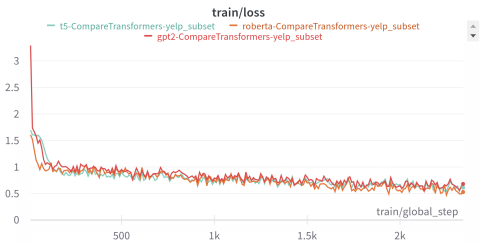



(f) Evaluation F1 score.

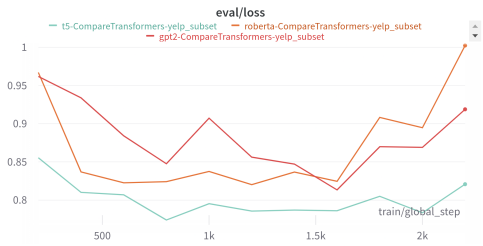



(g) Evaluation AUROC.

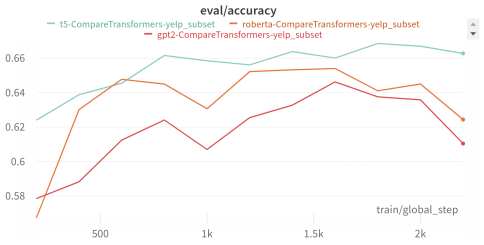
Figure D.3: Train loss, evaluation loss and evaluation metrics of RoBERTa and GPT-2 on the Yelp review full dataset.



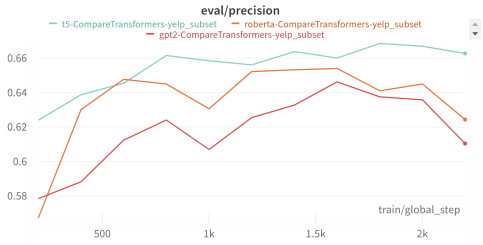

(a) Train loss.



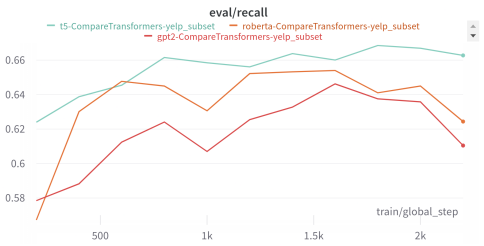

(b) Evaluation loss.



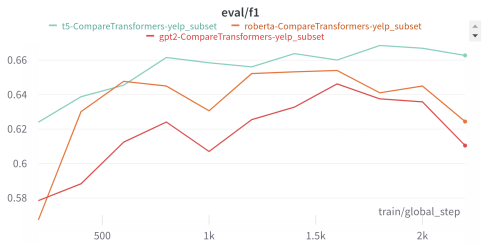

(c) Evaluation accuracy.



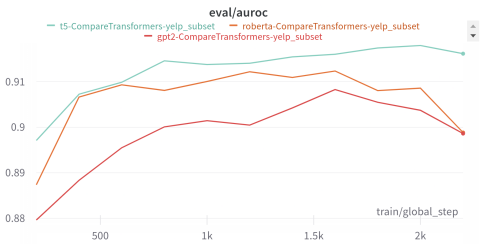

(d) Evaluation precision.




(e) Evaluation recall.




(f) Evaluation F1 score.




(g) Evaluation AUROC.

Figure D.4: Train loss, evaluation loss and evaluation metrics of RoBERTa, GPT-2 and T5 on the subset of the Yelp review subset dataset.

## E GRAPHS FOR SLICED ROBERTA (IMDB)

For the original graphs, please visit https://api.wandb.ai/links/sc4001/vz711n0m.

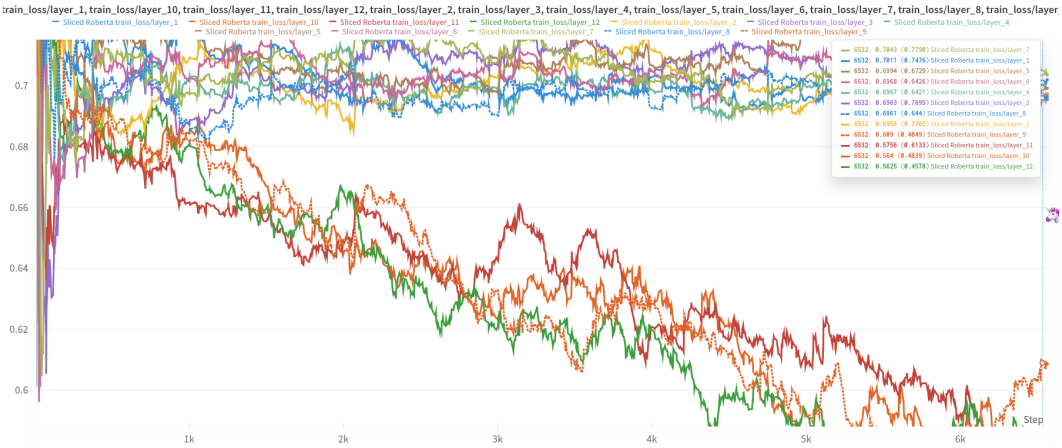



Figure E.1: Train loss for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

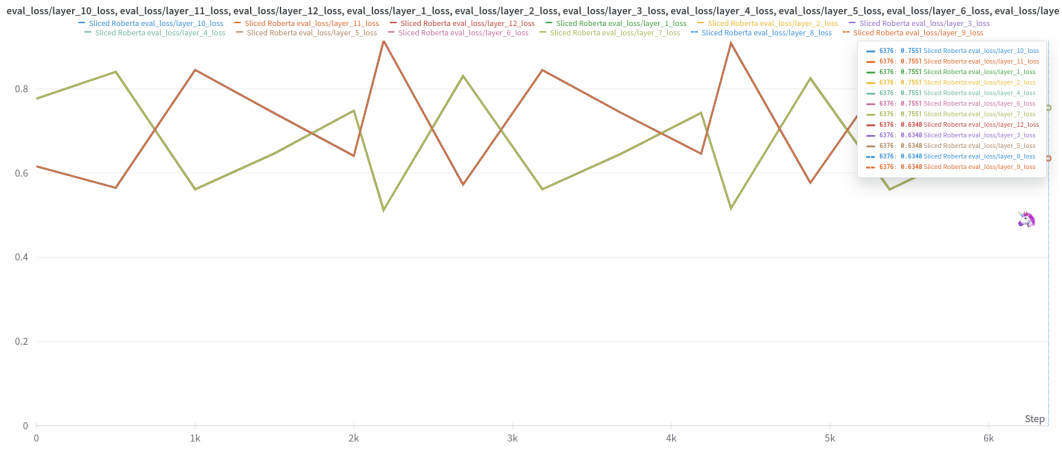



Figure E.2: Evaluation loss for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

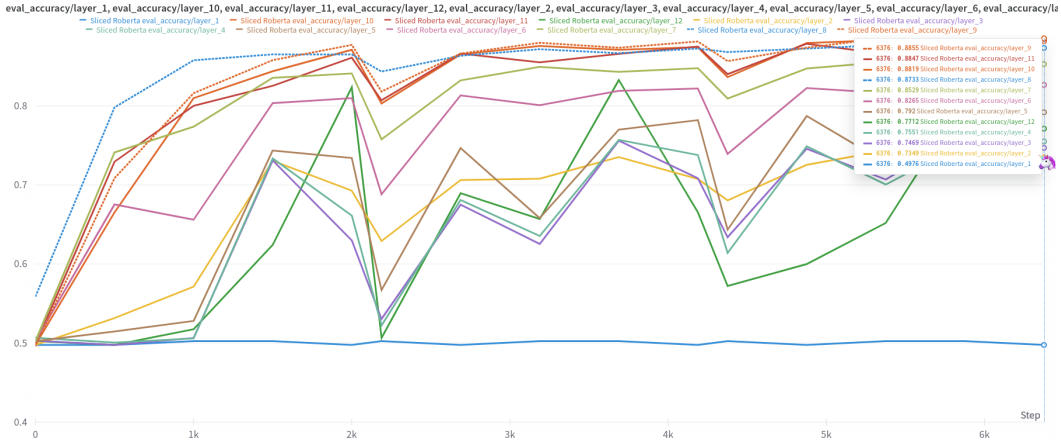



Figure E.3: Evaluation accuracy for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

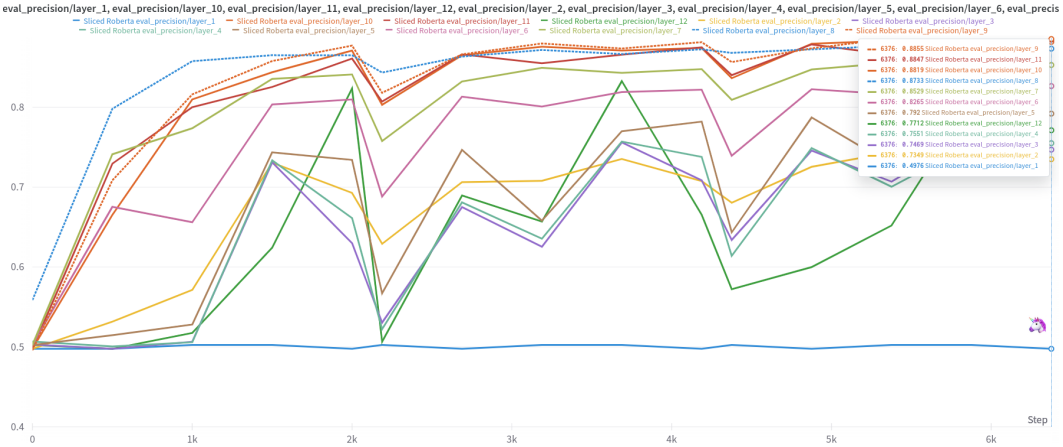



Figure E.4: Evaluation precision for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

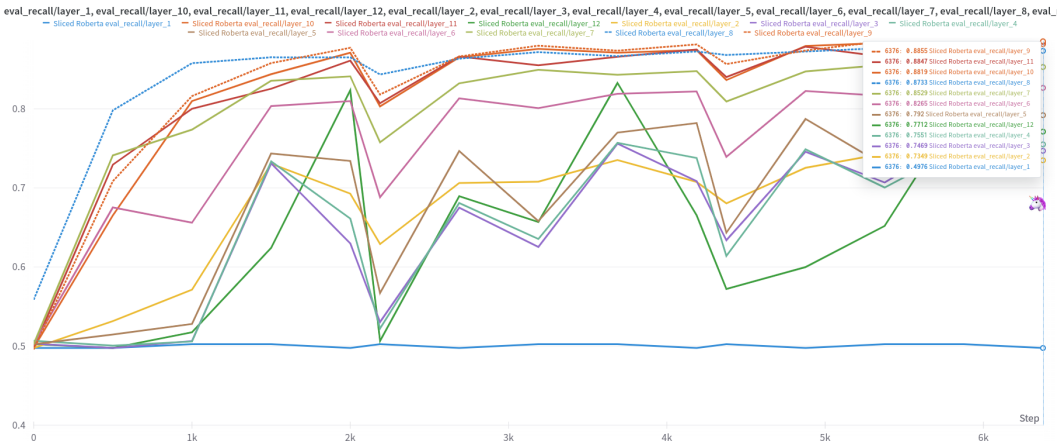



Figure E.5: Evaluation recall for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

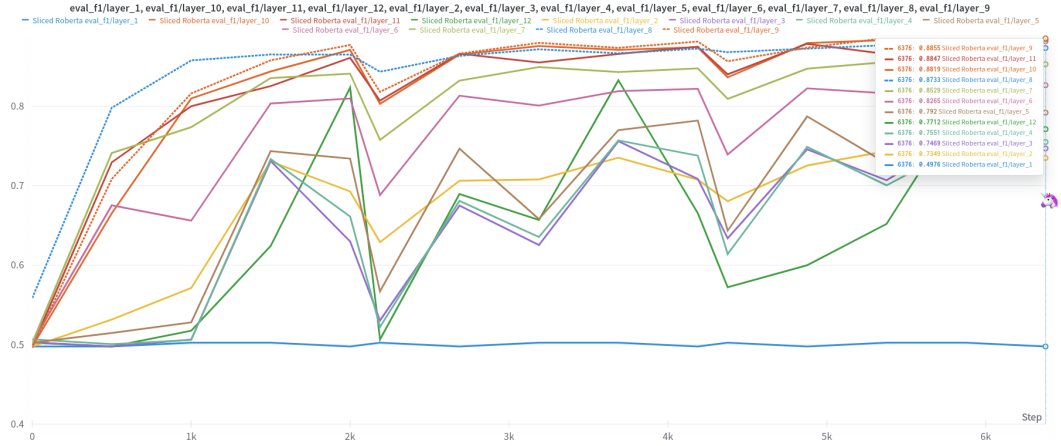



Figure E.6: Evaluation F1 score for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

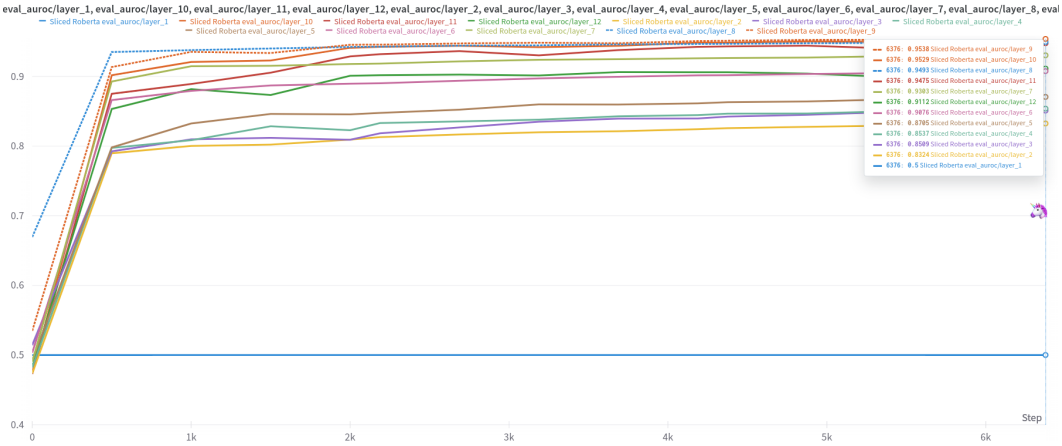



Figure E.7: Evaluation AUROC for the 16 classification layers of the Sliced RoBERTa model for the Imdb Dataset.

## F GRAPHS FOR SLICED ROBERTA (YELP SUBSET)

For the original graphs, please visit https://api.wandb.ai/links/sc4001/vz711n0m.

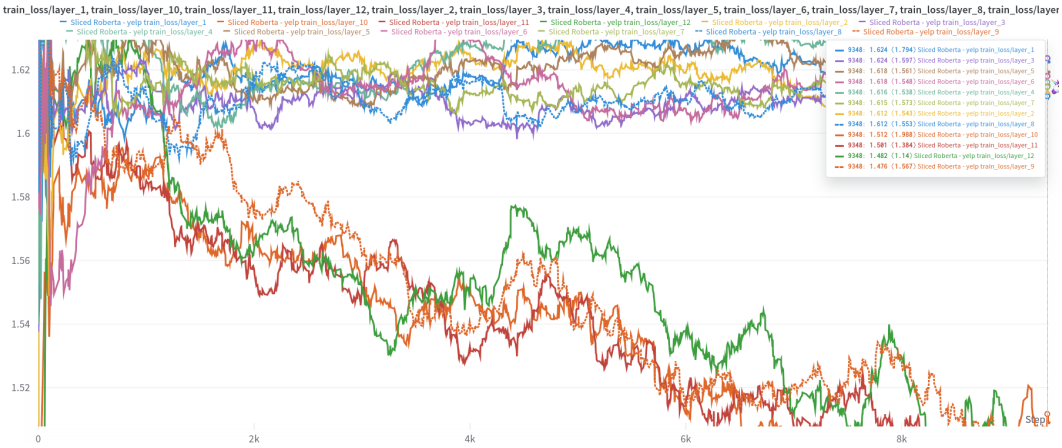



Figure F.1: Train loss for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

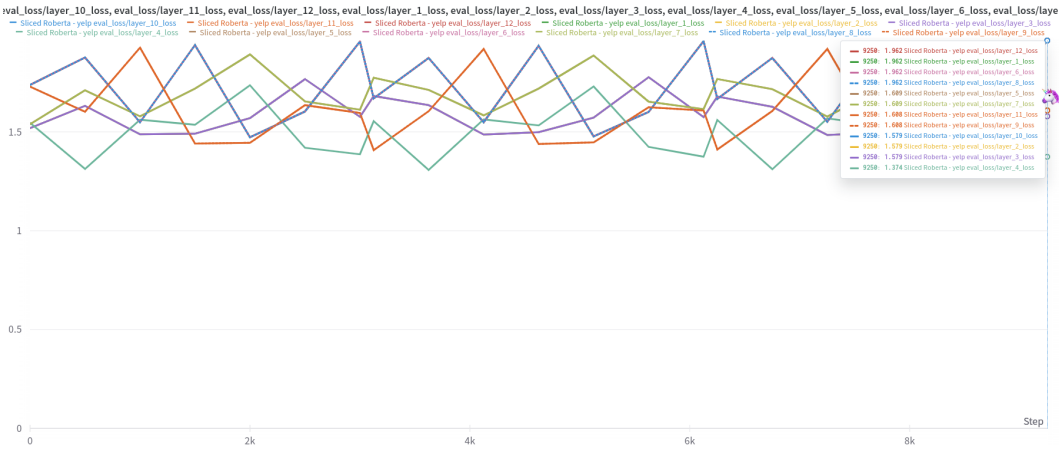



Figure F.2: Evaluation loss for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

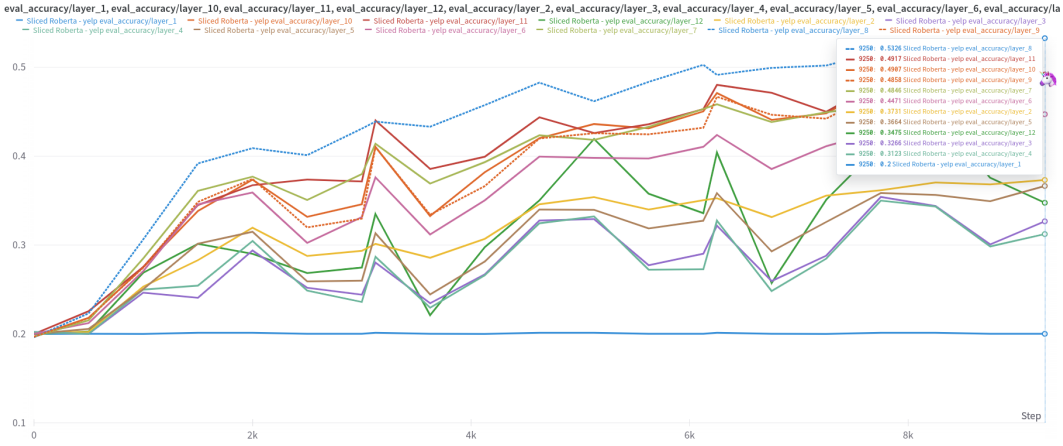



Figure F.3: Evaluation accuracy for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

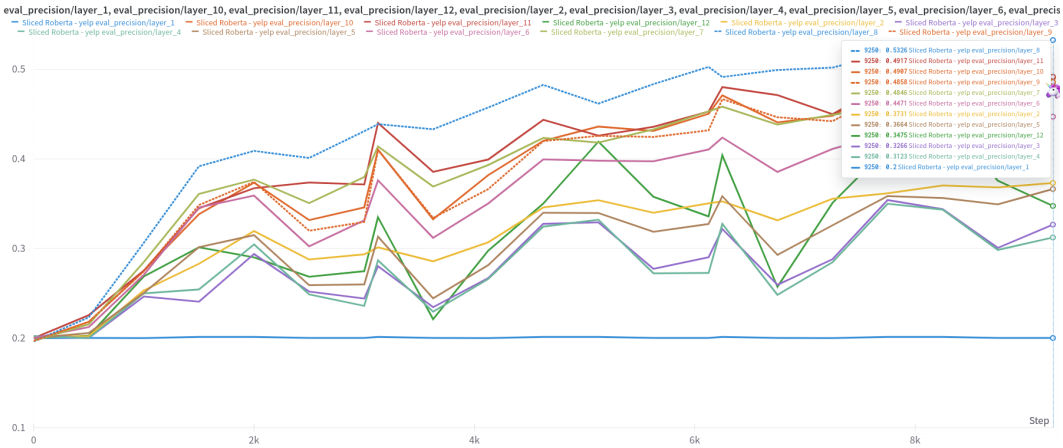



Figure F.4: Evaluation precision for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

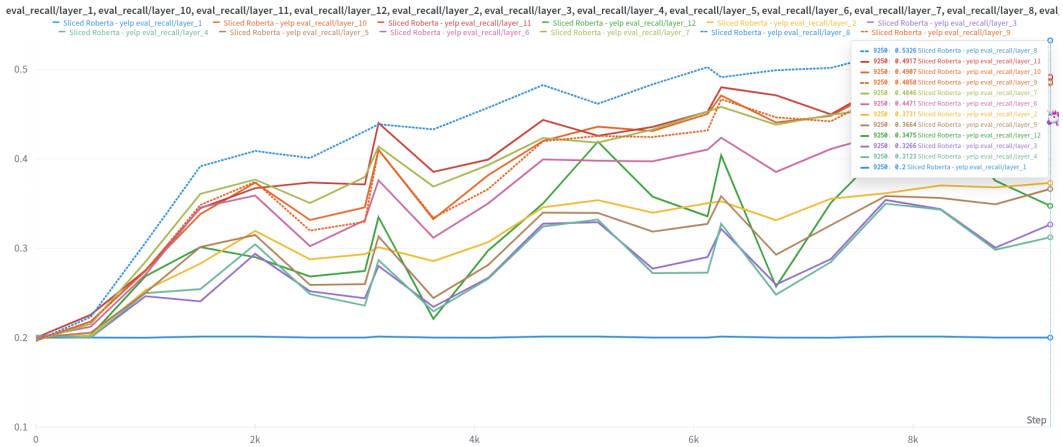



Figure F.5: Evaluation recall for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

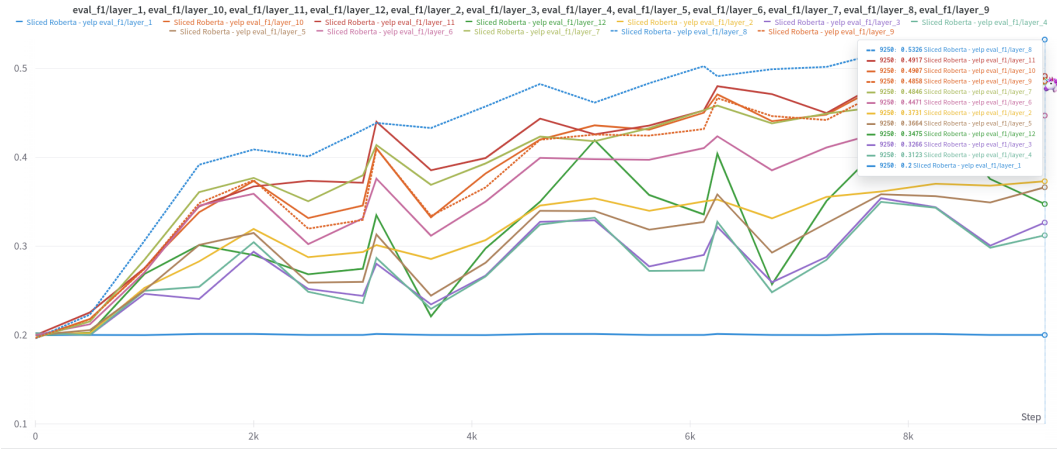



Figure F.6: Evaluation F1 score for the 16 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

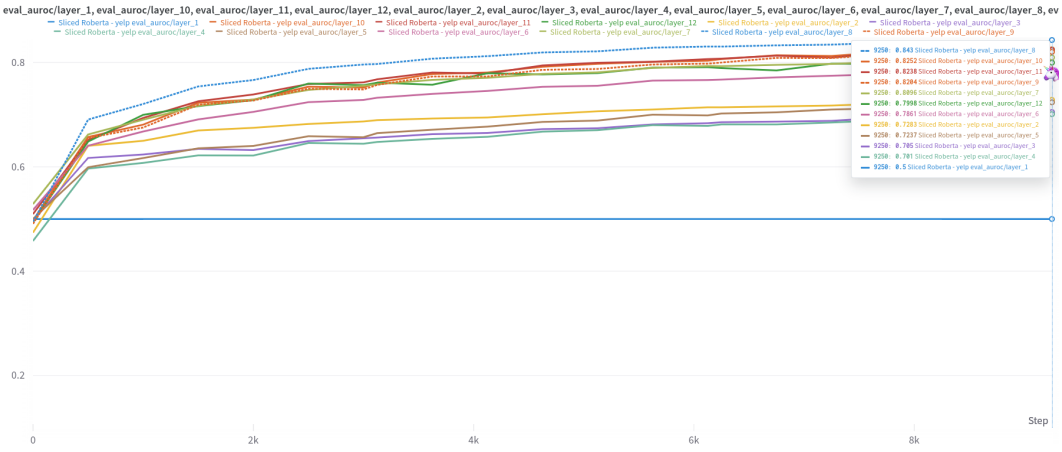



Figure F.7: Evaluation AUROC for the 32 classification layers of the Sliced RoBERTa model for the Yelp Subset Dataset.

## G Alternative Proof for Independent Weight Updates

This is an alternative proof demonstrates that the update of weights in the classification layers are independent of each other in Sliced Models.

Let

- Let $W_i$ be the weights of the i-th linear classification layer.
- Let $x_i$ be the input to the linear classification layer.
- Let $y$ be the labels.
- Let $\hat{y}_i = W_i x_i$ be the prediction of the $i$-th linear layer.
- Let $\eta$ be the learning rate.

The loss for the $i^{\text{th}}$ linear layer is cross entropy loss, which we'll denote as $L_i$, where

$$L_i = Loss(y, \hat{y}_i) = -\sum_{c=1}^{C} y_c \log(\hat{y}_{i,c}) \tag{9}$$

Since each $\hat{y}_i$ in Equation 9 is computed independently (as $\hat{y}_i = W_i x$), the losses $L_i$ and $L_j$ are independent for $i \neq j$:

$$L_i \cap L_j = \varnothing \quad \forall i \neq j \tag{10}$$

Given that weights are independent as shown in Figure 2, $W_i$ is independent of $W_j$ for $i \neq j$:

$$W_i \cap W_j = \varnothing \quad \forall i \neq j \tag{11}$$

Since $W_i$ is independent of $W_j$ (Equation 11) and $L_i$ is independent of $L_j$ (Equation 10), the gradients $\frac{\partial L_i}{\partial W_i}$ and $\frac{\partial L_j}{\partial W_j}$ would be independent for $i \neq j$:

$$\frac{\partial L_i}{\partial W_i} \cap \frac{\partial L_j}{\partial W_j} = \varnothing \quad \forall i \neq j \tag{12}$$

Subsequently, during back-propagation, the update rule for the weights $W_i$ of the $i$-th linear layer is determined by the gradient of the loss $L_i$ with respect to $W_i$:

$$W_i \leftarrow W_i - \eta \frac{\partial L_i}{\partial W_i} \tag{13}$$

Since the gradients are independent, the updates are also independent, and we can conclude that $W_i$ is updated independently of $W_j$ for all $j \neq i$.