Diego Toribio
Professor Sable
Project 3: Open-ended Deep Learning Project
ECE - 467 Natural Language Processing

1. Introduction

For this project, I chose to work on multi-label classification for emotion detection in text, using the GoEmotions dataset developed by Google Research. The dataset contains text samples labeled with one or more of 27 distinct emotions, making it a powerful resource for understanding nuanced emotional expressions. Each sample is annotated with multiple labels where applicable, capturing the complexity of emotions in natural language.

To address this task, I experimented with four widely used transformer architectures: BERT, RoBERTa, DistilBERT, and SqueezeBERT. BERT and RoBERTa provided robust baselines, leveraging their extensive pretraining on large text corpora. DistilBERT and SqueezeBERT offered lightweight alternatives, optimized for efficiency without significant compromise on performance. By exploring these models, I aimed to understand how variations in architecture and scale affect multi-label classification tasks, particularly for fine-grained emotion detection.

The project's development pipeline was carefully structured to ensure smooth experimentation and optimization. Initial testing began on Kaggle Notebooks, utilizing dual T4 GPUs for debugging and refining hyperparameters. After this, the workflow transitioned to Google Colab, where the code was tested with a single T4 GPU. Finally, I moved to Google Colab Pro, which provided access to the A100 GPU, enabling faster training and more computationally intensive experiments. This staged approach allowed for incremental improvements and ensured the codebase could adapt to varying computational resources.

Throughout the project, I used Weights & Biases (W&B) to track experiments and log performance metrics. W&B helped visualize training progress, monitor key metrics such as AUC scores, and conduct hyperparameter sweeps to explore configurations like learning rate and dropout rates. This integration ensures that every stage of the project is systematic and reproducible. The resulting pipeline was versatile and efficient, enabling seamless testing across different models and configurations.

2. Discussion

The GoEmotions dataset, developed by Google Research, serves as the foundation of this project. It contains over 58,000 text samples annotated with 28 emotion categories, including joy, anger, and sadness. Each sample can have multiple labels, reflecting the complexity of human emotional expression.

The dataset creation process for GoEmotions involved several carefully designed steps to ensure high-quality annotations and relevance to fine-grained emotion classification tasks. Reddit comments were selected based on specific criteria, including length (3-30 tokens), language (English), and subreddit

activity (minimum of 10,000 comments per subreddit). Comments containing offensive or harmful language were filtered out using predefined lists, while comments with mild profanity were retained because they were considered important for understanding negative emotions. Proper names and religion-related terms were masked using BERT-based Named Entity Tagger to minimize contextual biases during annotation.

Each comment was annotated by three to five native English-speaking raters who followed clear instructions. Raters were tasked with identifying emotions expressed in the text using a structured set of 27 emotions, along with a neutral category. They could select multiple labels only when confident, while the neutral label was choosent if no distinct emotion was evident. Examples deemed unclear or too challenging to label accounted for only 1.6% of the data and were excluded from further analysis. For illustrative purposes, **Table 1** highlights example text samples and their corresponding emotion labels, showcasing the diversity in expression captured by the dataset. To improve label reliability, disagreements were resolved by additional annotators, with 94% of examples achieving consensus from at least two raters on a single label.

Category	**Example Text**
Neutral	My favourite food is anything I didn't have to cook myself.
Joy	Happy to be able to help.
Realization	Maybe that’s what happened to the great white at Houston zoo.
Pride	I am just like this! Glad to know I’m not imagining it.
Excitement	It's crazy how far Photoshop has come. Underwater bridges?!! NEVER!!!

Table 1. Example Text Samples with Emotion Labels

A significant majority (83%) of the examples in the dataset are labeled with a single emotion, while fewer than 0.2% of examples contain more than four emotions labels. This distribution highlights the clarity of emotional expression in most comments and reflects the rigorous annotation process. As illustrated in **Figure 1**, certain emotions, such as joy and anger, are much more frequent than rarer ones like pride or grief, emphasizing the inherent class imbalance in the dataset. To address, sentiment balancing was applied to ensure a more even distribution across categories, preventing common emotions from overshadowing rarer ones, which could hinder a model’s ability to predict less frequent emotions accurately. **Table 2** further demonstrates the challenge of distinguishing between similar emotions by showing the most frequently co-occurring labels, such as admiration and gratitude or anger and annoyance.

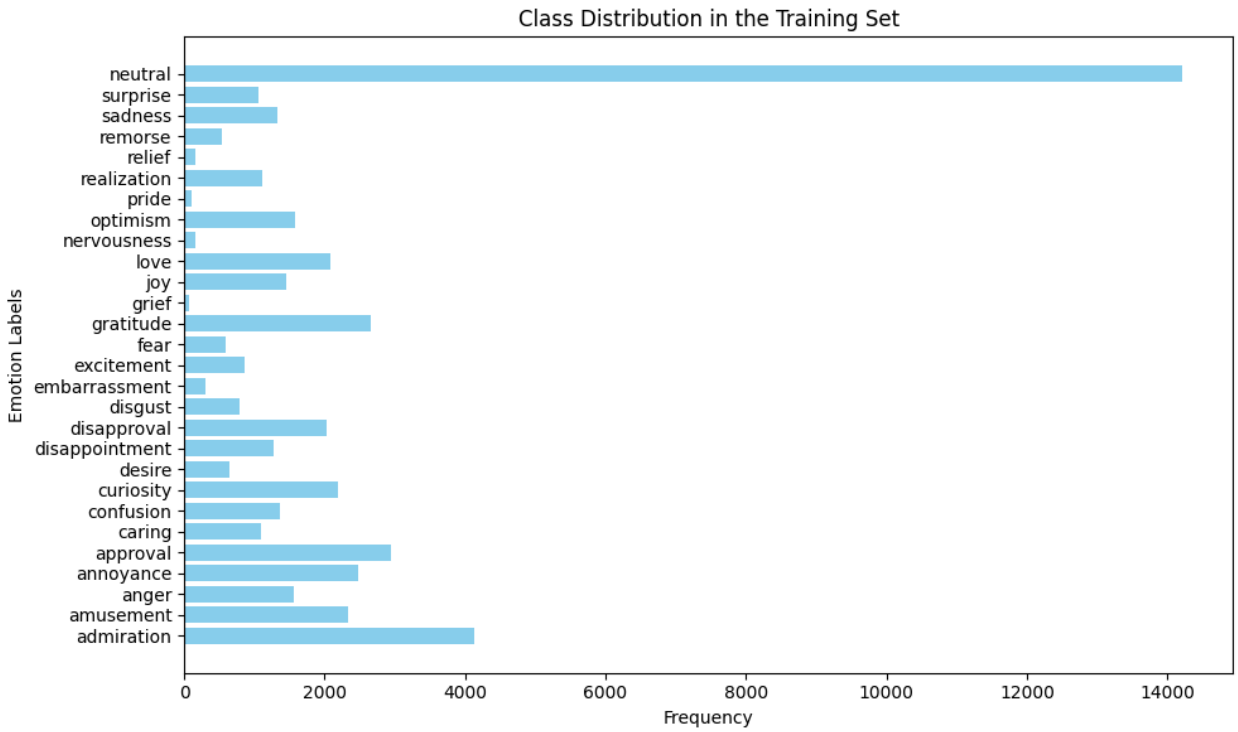


Figure 1. Class Distribution in the GoEmotions Dataset

Label 1	Label 2	Count
admiration	gratitude	279
anger	annoyance	269
admiration	approval	246
confusion	curisority	212
approval	neutral	202
admiration	love	192
annoyance	disapproval	178
disappointment	sadness	133
annoyance	neutral	132
admiration	joy	126

Table 2. Most frequently co-occurring labels showing the complexity of distinguishing between similar emotions.

Preprocessing the dataset involved several essential steps to prepare it for model training. Tokenization was carried out using the tokenizer associated with each model, with the text truncated to a maximum length of 64 tokens and shorter sequences padded to create consistent input dimensions. As shown in **Figure 2**, the token length distribution indicates that most comments range between 10 and 20 tokens, highlighting the need for padding and truncation to standardize input sizes across the dataset. Emotion labels were transformed into multi-hot encoding format, where each label set was converted into a binary vector representation. These preprocessing steps ensured that the dataset was efficiently structured for use in the transformer-based architectures.

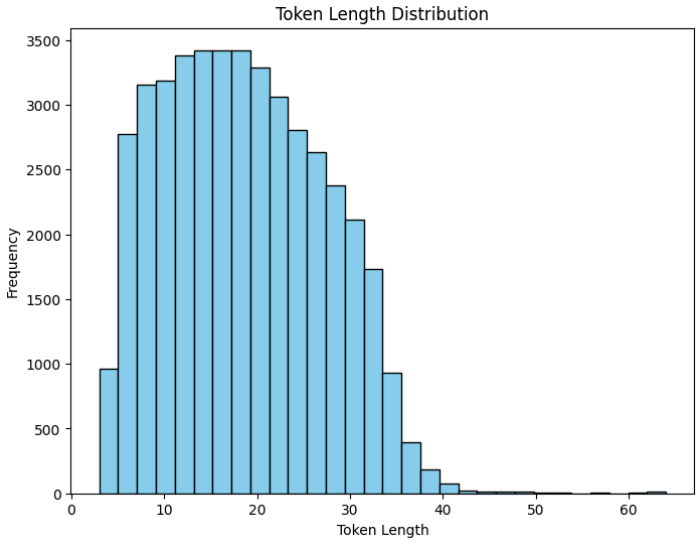


Figure 2. Histogram illustrating the distribution of tokenized text length.

3. Methods

Transformers excel at capturing complex relationships in text, leveraging self-attention mechanisms to process entire sequences. For this task, we fine-tuned pre-trained transformer models on the GoEmotions dataset to adapt their broad language understanding to the specific challenges of multi-label emotion classification. By updating the pre-trained weights, we aimed to optimize performance on the dataset while retaining the extensive prior knowledge embedded in these models.

We evaluated six transformer architectures: BERT, RoBERTa, their larger variants (BERT-large and RoBERTa-large), DistilBERT, and SqueezeBERT. BERT (Bidirectional Encoder Representations for Transformers) served as our foundational model, using its bidirectional architecture to analyze word context from surrounding tokens. In the original GoEmotions research, BERT was used as a baseline, achieving an average F1-score of 0.46 across the emotion taxonomy, making it an essential starting point for comparison. RoBERTa builds upon BERT with refinements to the pretraining, including dynamic masking and removal of the Next Sentence Prediction task, enhancing its ability to capture nuanced patterns in text.

To explore the effects of larger parameter counts, we incorporated BERT-large and RoBERTa-large in our experiments. These larger variants, with significantly higher parameter counts, enable more detailed representations. Comparing these models to their base versions allowed us to evaluate whether the additional capacity leads to improved performance and to observe whether the trends in the BERT and RoBERTa base comparisons persist at larger scales.

For lightweight alternatives, we included DistilBERT and SqueezeBERT, both designed for faster inference and reduced computational requirements. DistilBERT compresses BERT's architecture while retraining most of its accuracy, making it an efficient option for resource-constrained scenarios. SqueezeBERT takes this further by utilizing grouped convolutions, reducing memory usage while maintaining competitive performance. These models provided valuable insights into how smaller architectures perform on multi-label emotion classification tasks.

Hyperparameter tuning played a key role in optimizing model performance. We tested a range of configurations, including learning rates of 3e-5 and 5e-5, dropout rates from 0.3 to 0.5, and batch sizes of 32 and 64. Automated sweeps conducted using Weights & Biases streamlined this process, enabling efficient identification of optimal configurations for each model.

Model Name	Parameter Count (Millions)	Model Size (MB)
bert-base-uncased	125	418
roberta-base	340	478
bert-large-uncased	340	1311
roberta-large	355	1346
squeezebert-uncased	82	315
distilbert-base-uncased	66	256

Table 3. Parameter counts and model sizes of transformer architectures, illustrating efficiency trade-offs.

4. Results

Model	Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
BERT-base	3e-5	32	0.3	0.952	0.073	0.090
RoBERTa	3e-5	32	0.3	0.956	0.081	0.088
BERT Large	3e-5	32	0.3	0.953	0.060	0.097
RoBERTa Large	3e-5	64	0.3	**0.957**	0.078	0.089
DistilBERT	3e-5	64	0.4	0.948	0.052	0.096
SqueezeBERT	5e-5	32	0.3	0.944	0.092	0.093

Table 4. Best-performing results for each model with optimal hyperparameters and corresponding AUC scores.

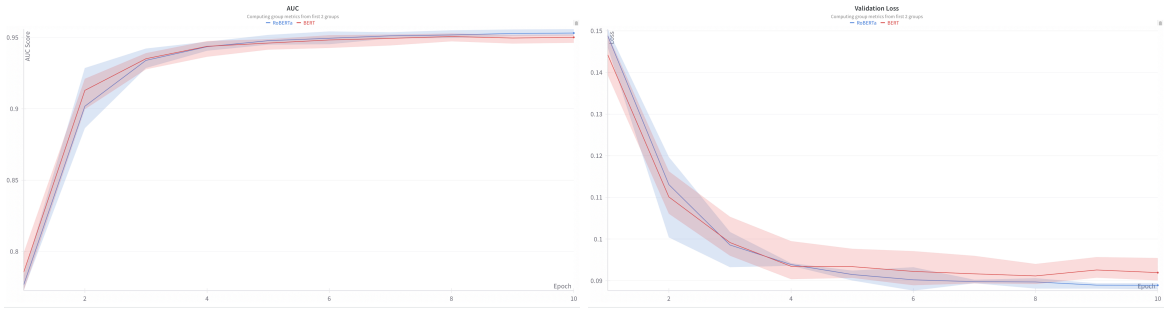


Figure 3. AUC Curve (left) and Validation Loss (right) for top-performing BERT and RoBERTa

As illustrated in **Figure 3**, RoBERTa consistently outperforms BERT, achieving a higher AUC score and stabilizing at a lower validation loss. This superior performance reflects RoBERTa's enhanced optimization and generalization capabilities. It converges more quickly and efficiently compared to BERT, which exhibits slower convergence and higher final loss values, indicating less effective training dynamics.

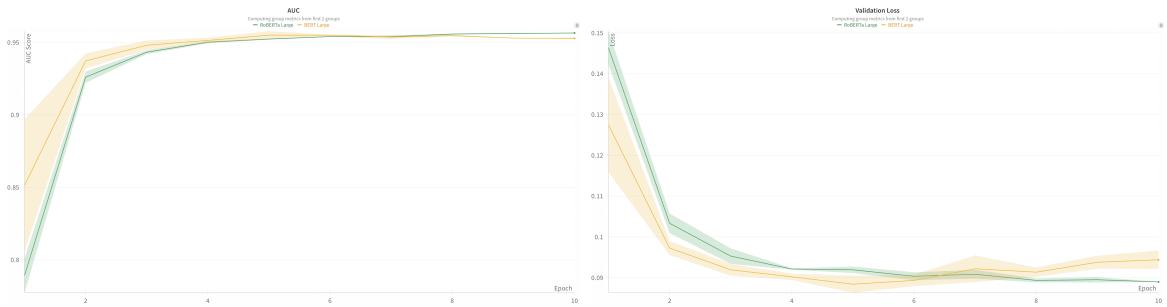


Figure 4. AUC Curve (left) and Validation Loss (right) for top-performing BERT-large and RoBERTa-large

Scaling up the large variants amplifies the performance gap between BERT and RoBERTa, as shown in **Figure 4**. RoBERTa-Large achieves faster convergence, lower validation loss, and slightly higher AUC scores compared to BERT-Large. Both models benefit from increased parameter counts, but RoBERTa-Large demonstrates greater gains in generalization and optimization.

The validation loss curves reveal that RoBERTa-Large stabilizes earlier and at lower values, while BERT-Large plateaus at higher loss. Similarly, RoBERTa-Large maintains stable AUC scores across epochs, showcasing its ability to leverage additional model capacity effectively.

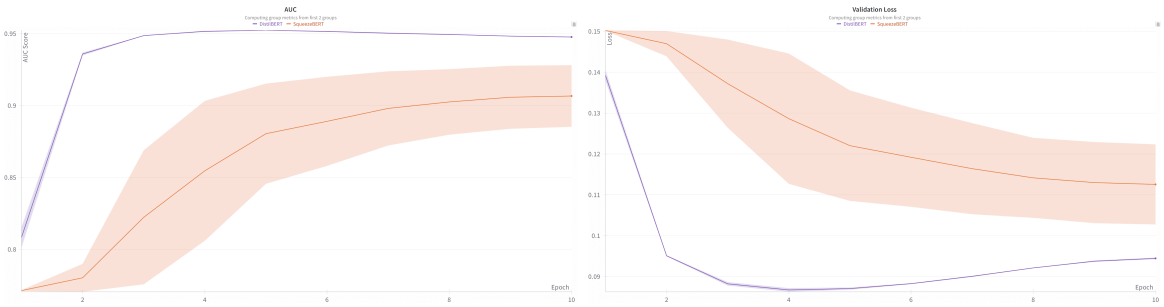


Figure 5. AUC Curve (left) and Validation Loss (right) for top-performing DistilBERT and SqueezeBERT

As shown in Figure 5, both models demonstrate quick convergence in early epochs due to their smaller parameter sizes. However, their final performance diverges, with DistilBERT consistently outperforming SqueezeBERT in both AUC and validation loss. DistilBERT stabilizes earlier and exhibits better generalization, achieving a strong balance between accuracy and efficiency.

In contrast, SqueezeBERT prioritizes computational speed and memory efficiency, resulting in slower convergence and higher final loss values. While it performs competitively in initial epochs, it ultimately falls short compared to DistilBERT and significantly lags behind larger models like RoBERTa-Large. Nevertheless, these lightweight architectures remain practical solutions for scenarios where efficiency takes precedence over peak performance.

5. Conclusion

This study evaluated six transformer architectures—BERT, RoBERTa, their larger variants (BERT-Large and RoBERTa-Large), DistilBERT, and SqueezeBERT—for multi-label emotion classification using the GoEmotions dataset. RoBERTa consistently outperformed BERT across all configurations, demonstrating faster convergence, lower validation loss, and higher AUC scores. Scaling to larger variants amplified these trends, with RoBERTa-Large showing superior optimization and generalization compared to BERT-Large. Among the lightweight models, DistilBERT struck a strong balance between accuracy and efficiency, outperforming SqueezeBERT in both AUC and validation loss.

While SqueezeBERT prioritized speed and memory efficiency, it struggled with slower convergence and higher final loss values.

Dropout emerged as the most critical hyperparameter across all models, significantly enhancing training stability and generalization. The importance of batch size and learning rate varied by architecture. For instance, BERT and RoBERTa showed the greatest benefit from optimized dropout, while DistilBERT and SqueezeBERT were more sensitive to batch size. Hyperparameter tuning using Weights & Biases streamlined the process, enabling efficient and reproducible exploration of optimal configurations.

Overall, RoBERTa and its large variant are best suited for tasks requiring high performance, while DistilBERT provides a practical option for resource-constrained environments. SqueezeBERT, despite its limitations, remains a viable choice for applications prioritizing speed and memory efficiency over peak accuracy. These findings highlight the importance of tailoring model selection and hyperparameter tuning to the specific requirements of a task to achieve optimal results.

6. Replicating results.

All experiments and results for this study were logged and tracked using Weights & Biases (W&B), a platform for managing machine learning projects. Metrics such as validation loss, training loss, AUC curves across various hyperparameter configurations, and system data like GPU utilization, memory usage, and runtime performance were carefully recorded. All relevant figures and results have been included in the Appendix section for review. The project began with an initial exploratory analysis of the dataset, conducted in Google Colab to better understand its structure and characteristics. The main experiments were then performed using an A100 GPU on Google Colab, leveraging its computational power for training large transformer models. For tasks that exceeded the A100's compute capacity, additional experiments were run on dual T4 GPUs in Kaggle Notebooks. Together, these resources and results provide a seamless and reproducible pipeline for both data exploration and model experimentation.

Appendix

A. Dataset

Label	Train (%)	Validation (%)	Test (%)
admiration	9.5	9	9.3
amusement	5.4	5.6	4.9
anger	3.6	3.6	3.6
annoyance	5.7	5.6	5.9
approval	6.8	7.3	6.5

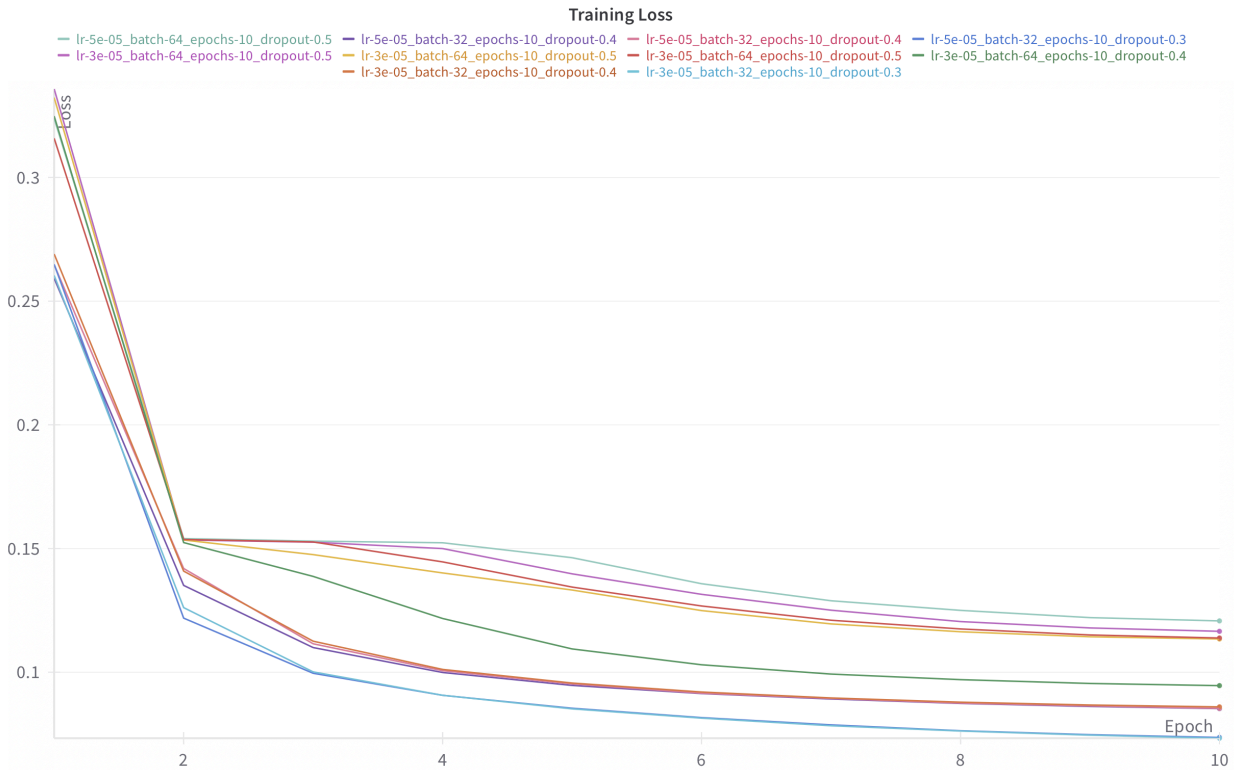
caring	2.5	2.8	2.5
confusion	3.2	2.8	2.5
curiosity	5	4.6	2.8
desire	1.5	1.4	1.5
disappointment	2.9	3	2.8
disapproval	4.7	5.4	4.9
disgust	1.8	1.8	2.3
embarrassment	0.7	0.6	0.7
excitement	2	1.8	1.9
fear	1.4	1.7	1.4
gratitude	6.1	6.6	6.5
grief	0.2	0.2	0.1
joy	3.3	3.2	3
love	4.8	4.6	4.4
nervousness	0.4	0.4	0.4
optimism	3.6	3.9	3.4
pride	0.3	0.3	0.3
realization	2.6	2.3	2.7
relief	0.4	0.3	0.2
remorse	2.9	1.3	1
sadness	3.1	2.6	2.9
surprise	2.4	2.4	2.6
neutral	32.8	32.5	32.9

B. All Results

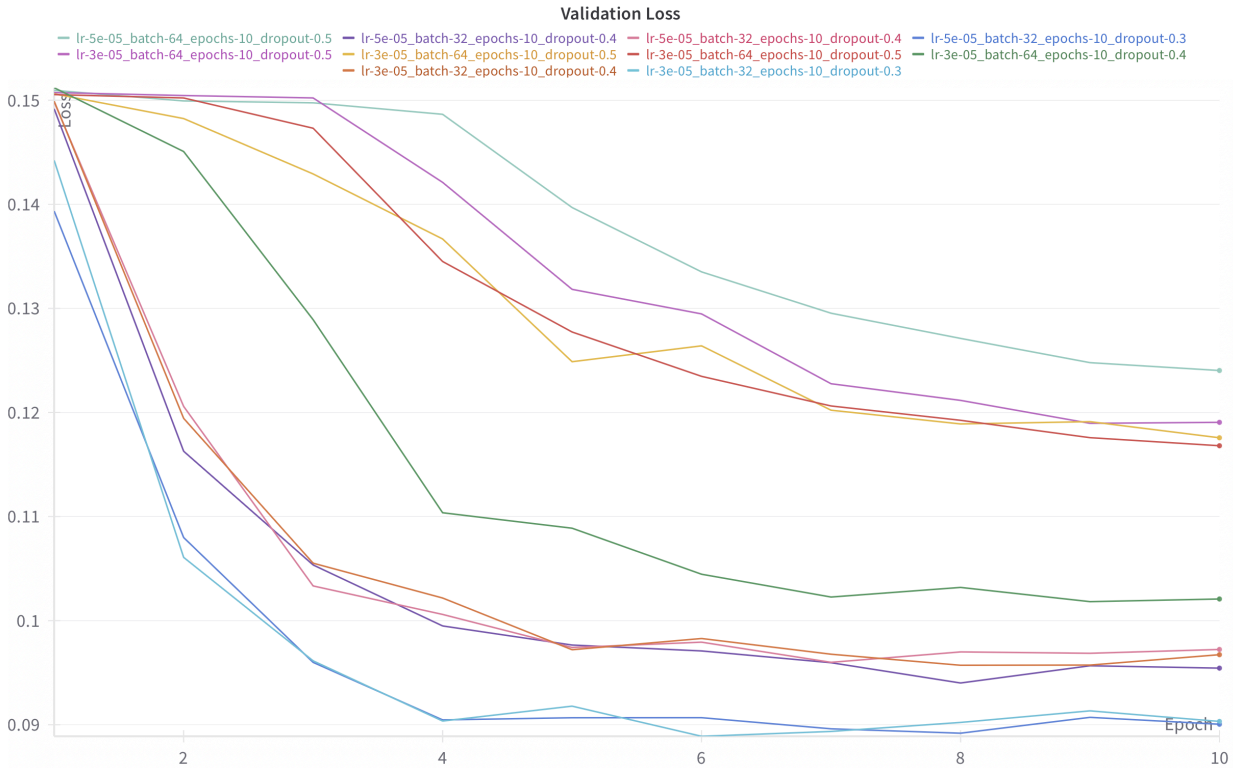
BERT Final Performance Metrics Table

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.3	**0.952**	0.073	0.090
		0.4	0.945	0.086	0.097
	64	0.4	0.935	0.095	0.102
		0.5	0.903	0.114	0.117
		0.5	0.901	0.113	0.118
		0.5	0.895	0.117	0.119
5e-05	32	0.3	**0.952**	0.074	0.090
		0.4	0.946	0.085	0.097
		0.4	**0.946**	0.085	0.095
	64	0.5	0.887	0.121	0.124

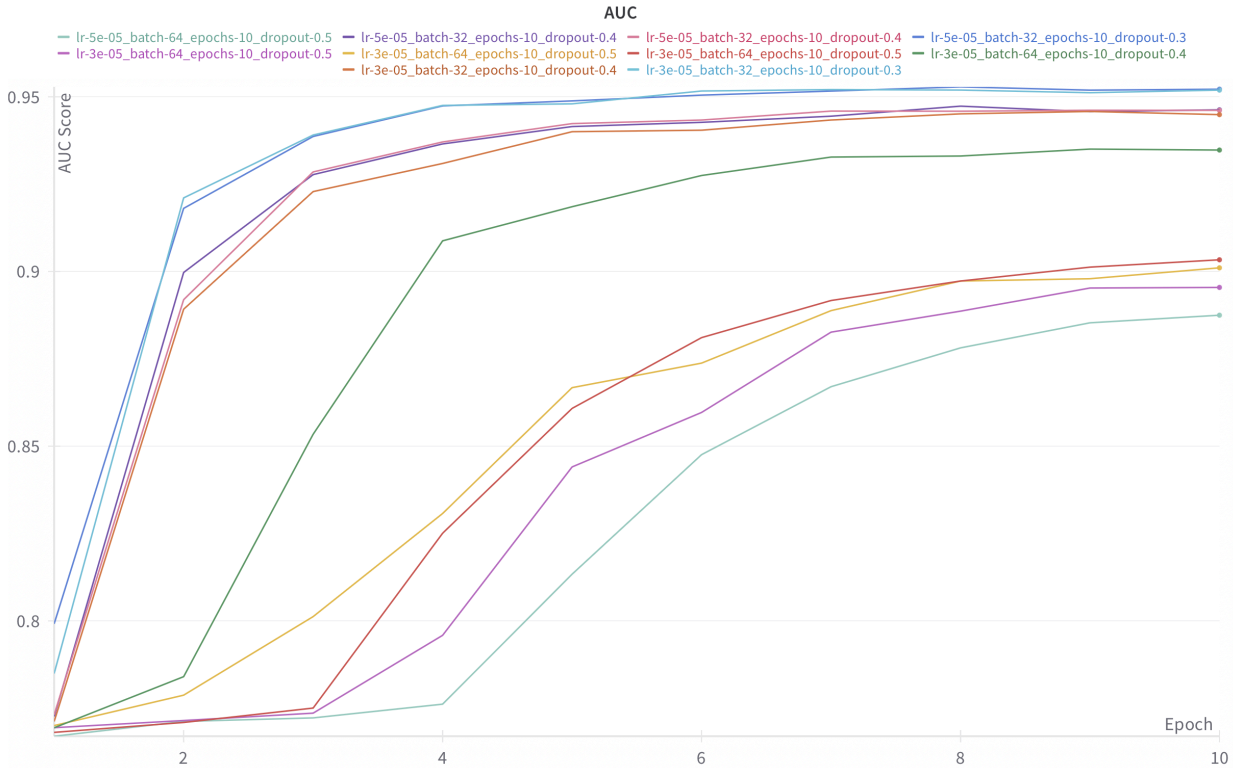
BERT Training Loss Curve



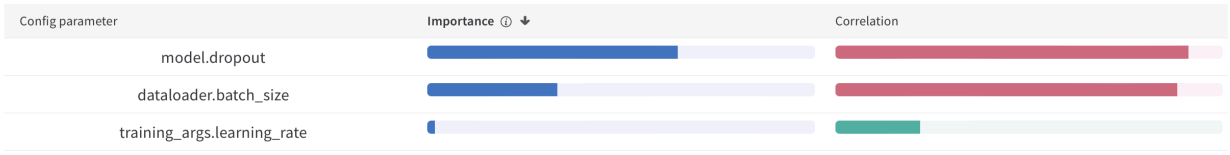
BERT Validation Loss Curve



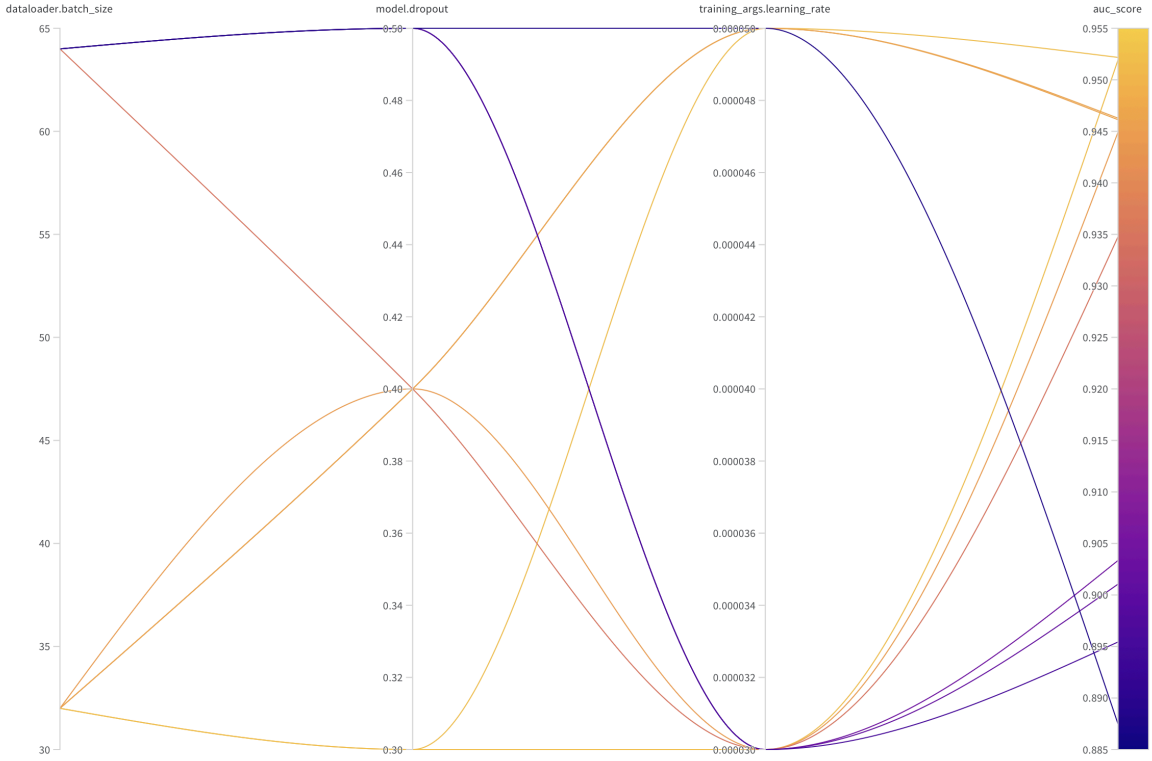
BERT AUC Score Curve



BERT Hyperparmeter Importance Plot



BERT Parallel Coordinates Plot

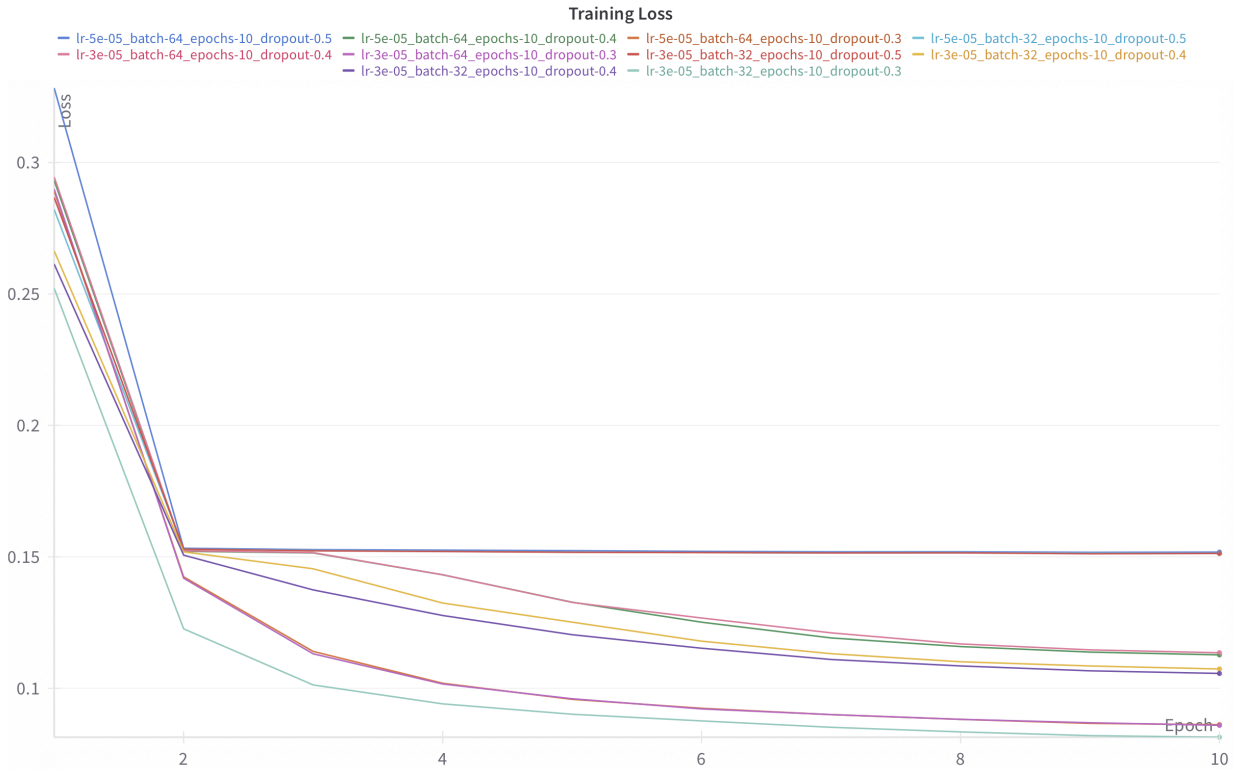


RoBERTa Final Performance Metrics Table

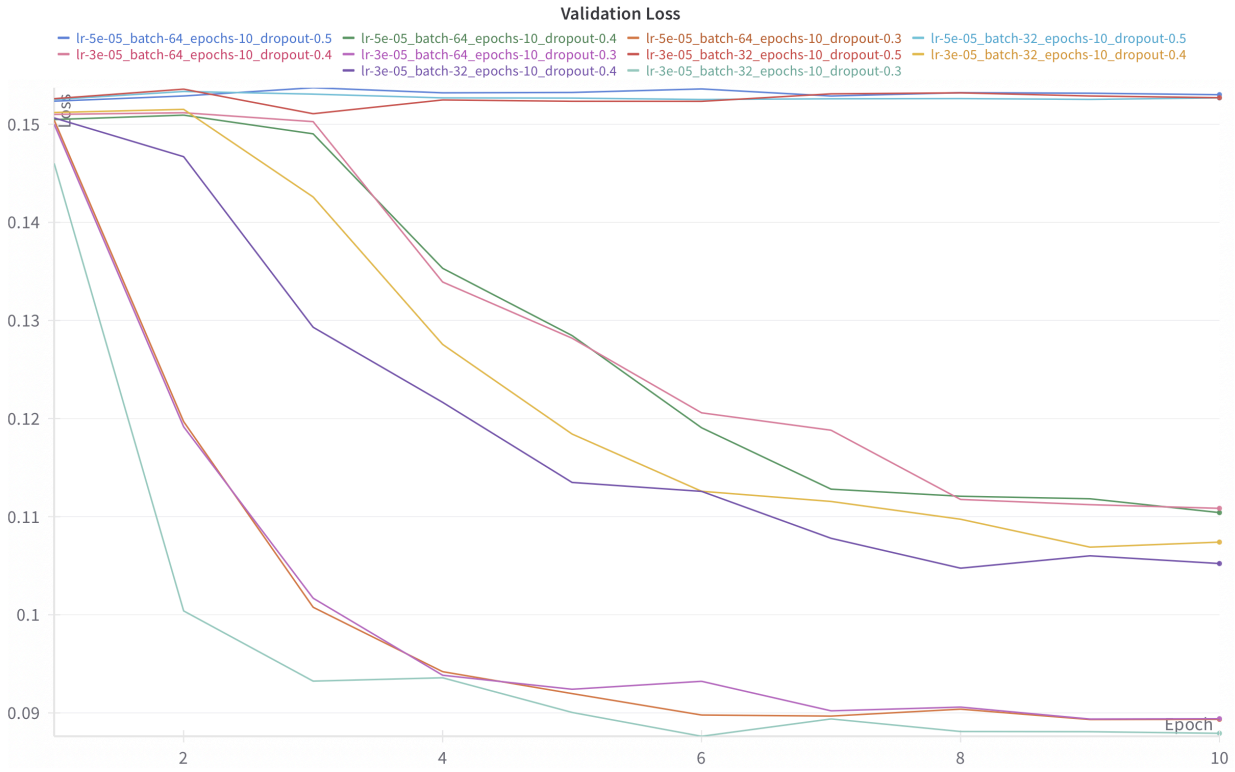
Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.3	**0.956**	0.081	0.088
		0.4	0.923	0.106	0.105
		0.4	0.919	0.107	0.108
		0.5	0.763	0.151	0.153
	64	0.3	**0.951**	0.086	0.089
		0.4	0.910	0.113	0.111
5e-05	32	0.5	0.765	0.151	0.153

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.3	**0.956**	0.081	0.088
		0.4	0.923	0.106	0.105
		0.4	0.919	0.107	0.108
		0.5	0.763	0.151	0.153
	64	0.3	**0.951**	0.086	0.089
		0.4	0.910	0.113	0.111
	64	0.3	**0.952**	0.086	0.089
		0.4	0.912	0.113	0.110
		0.5	0.767	0.152	0.153

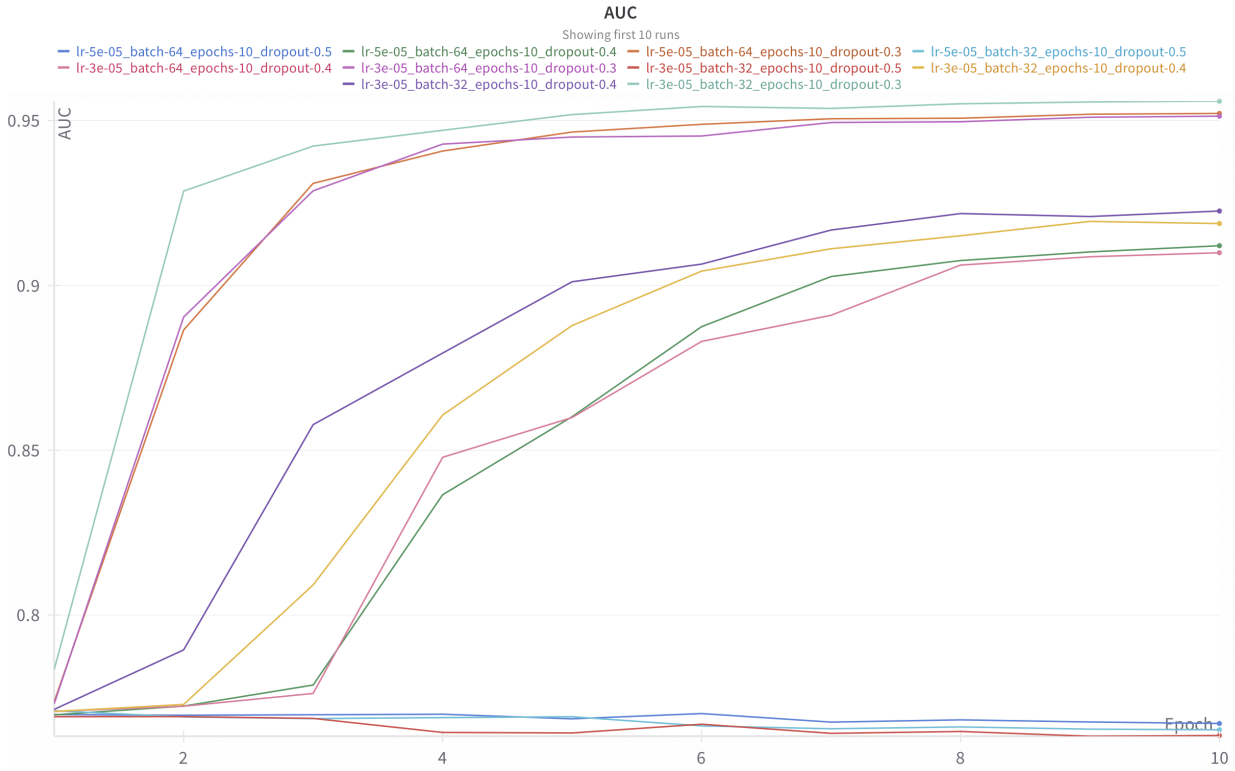
RoBERTa Training Loss Curve



RoBERTa Validation Loss Curve



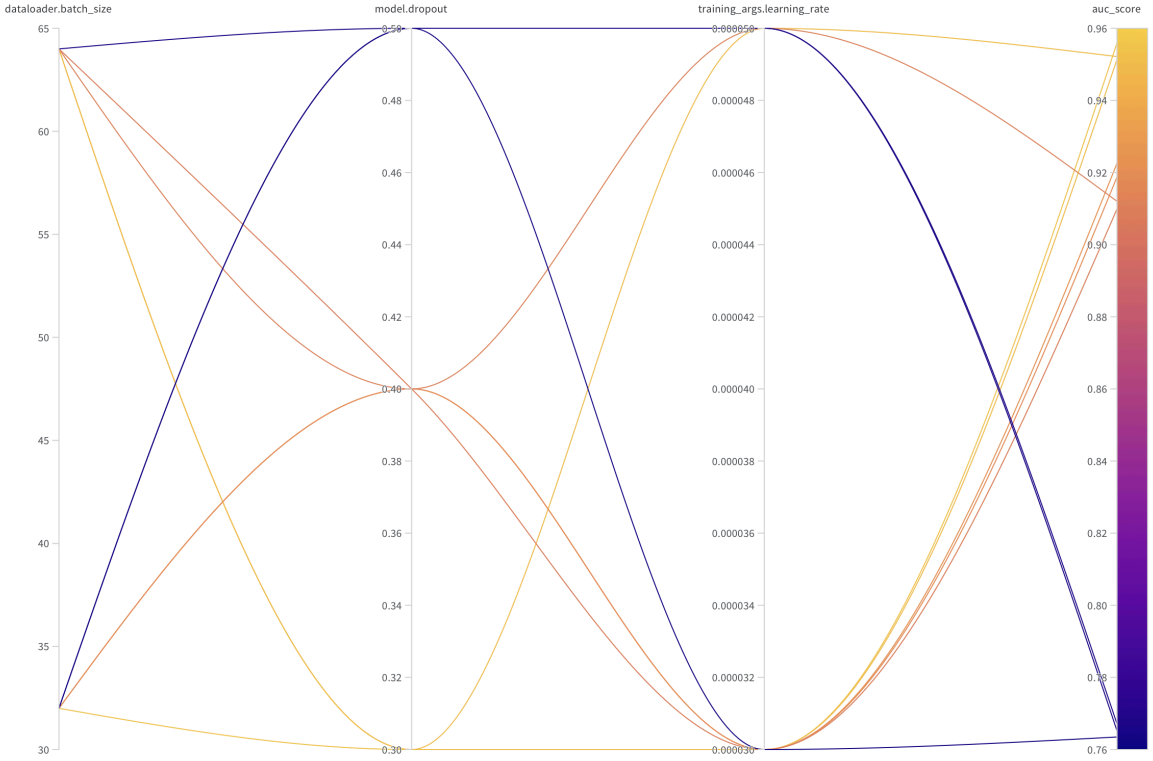
RoBERTa AUC Score Curve



RoBERTa Hyperparmeter Importance Plot

Config parameter	Importance	Correlation
model.dropout		
dataloader.batch_size		
training_args.learning_rate		

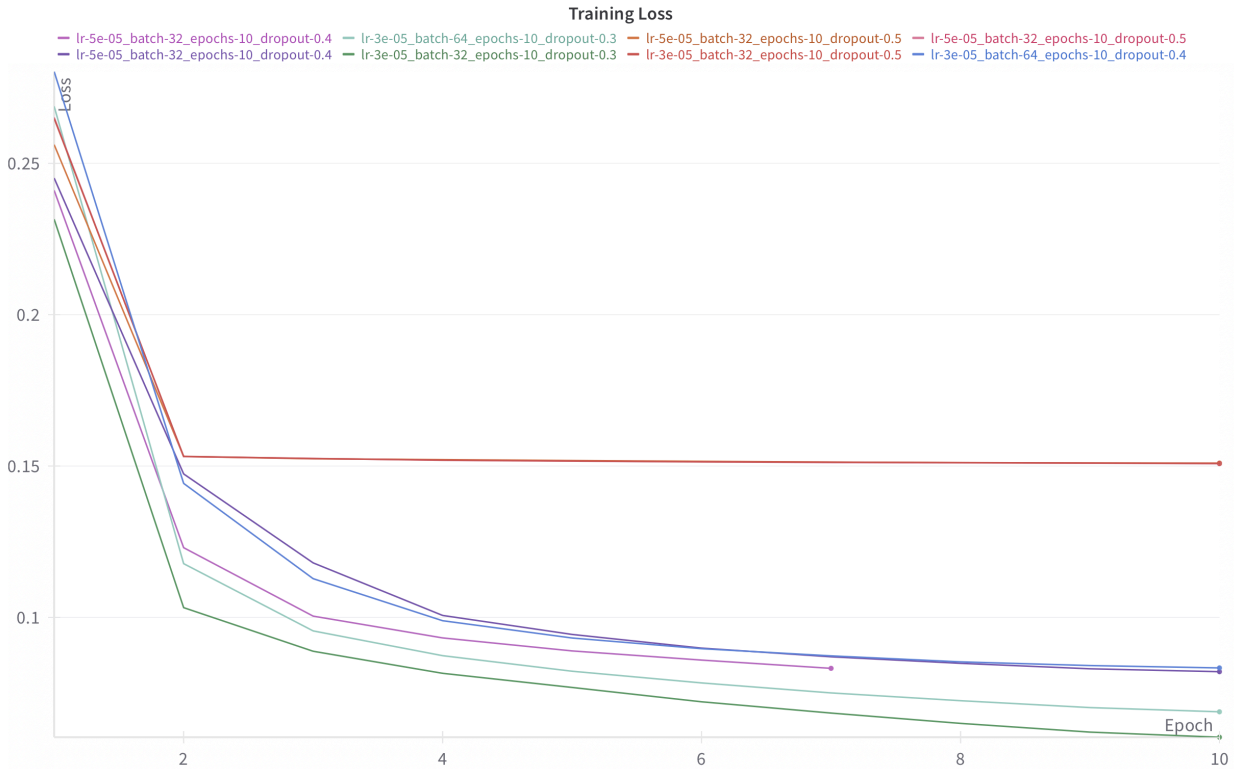
RoBERTa Parallel Coordinates Plot



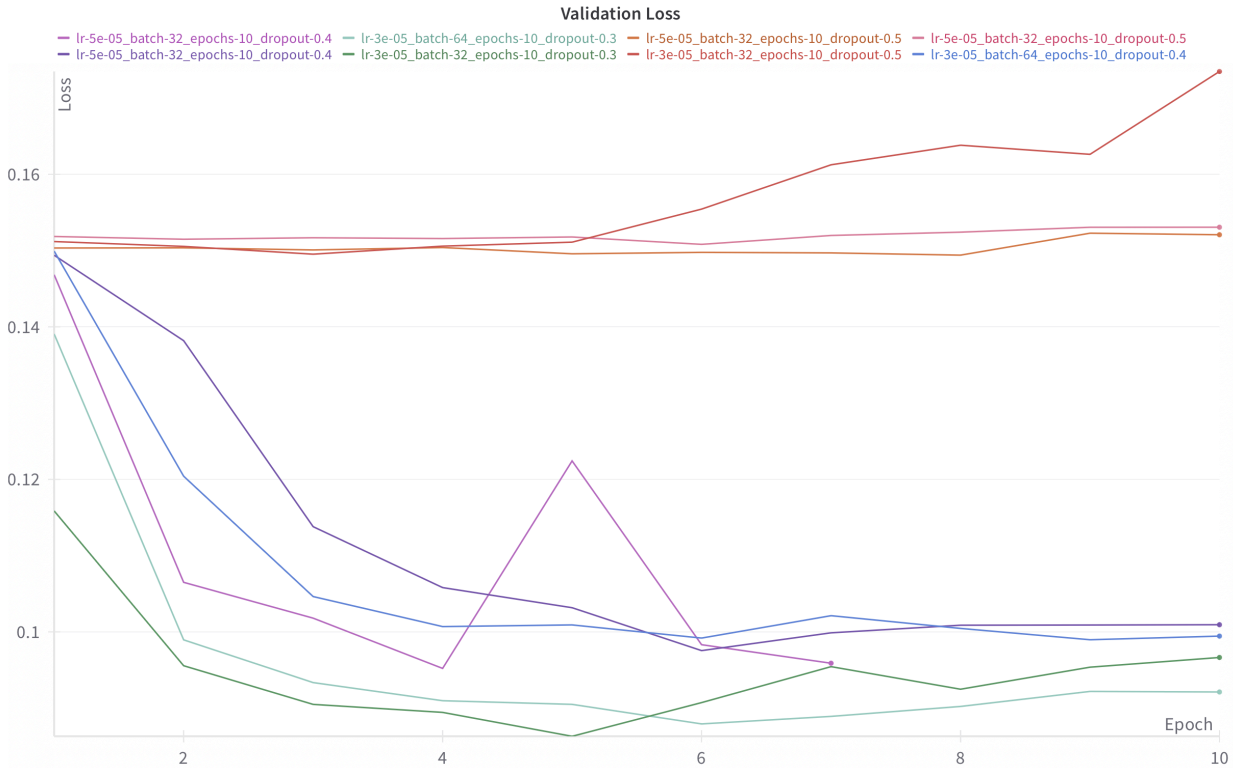
BERT Large Final Performance Metrics Table

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.3	**0.953**	0.060	0.097
		0.5	0.725	0.151	0.173
	64	0.3	**0.953**	0.069	0.092
		0.4	0.946	0.083	0.099
5e-05	32	0.4	0.947	0.082	0.101
		0.4	0.950	0.083	0.0959
		0.5	0.762	0.151	0.152
		0.5	0.743	0.151	0.153

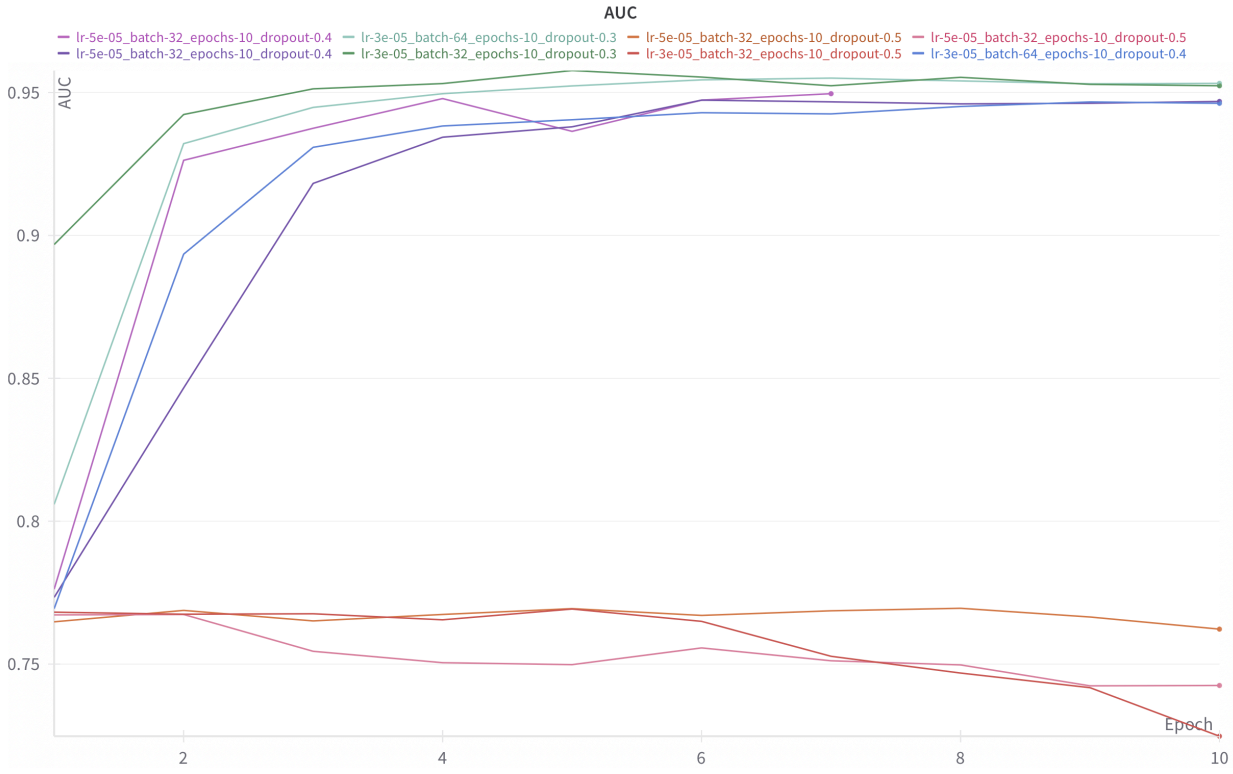
BERT Large Training Loss Curve



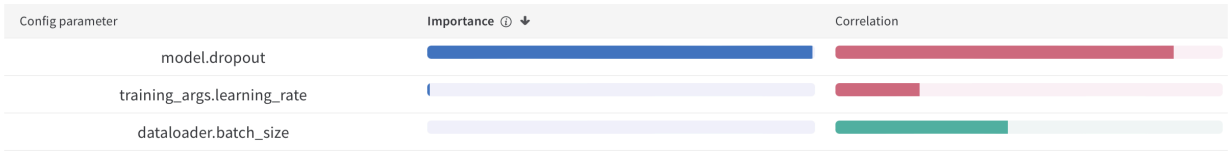
BERT Large Validation Loss Curve



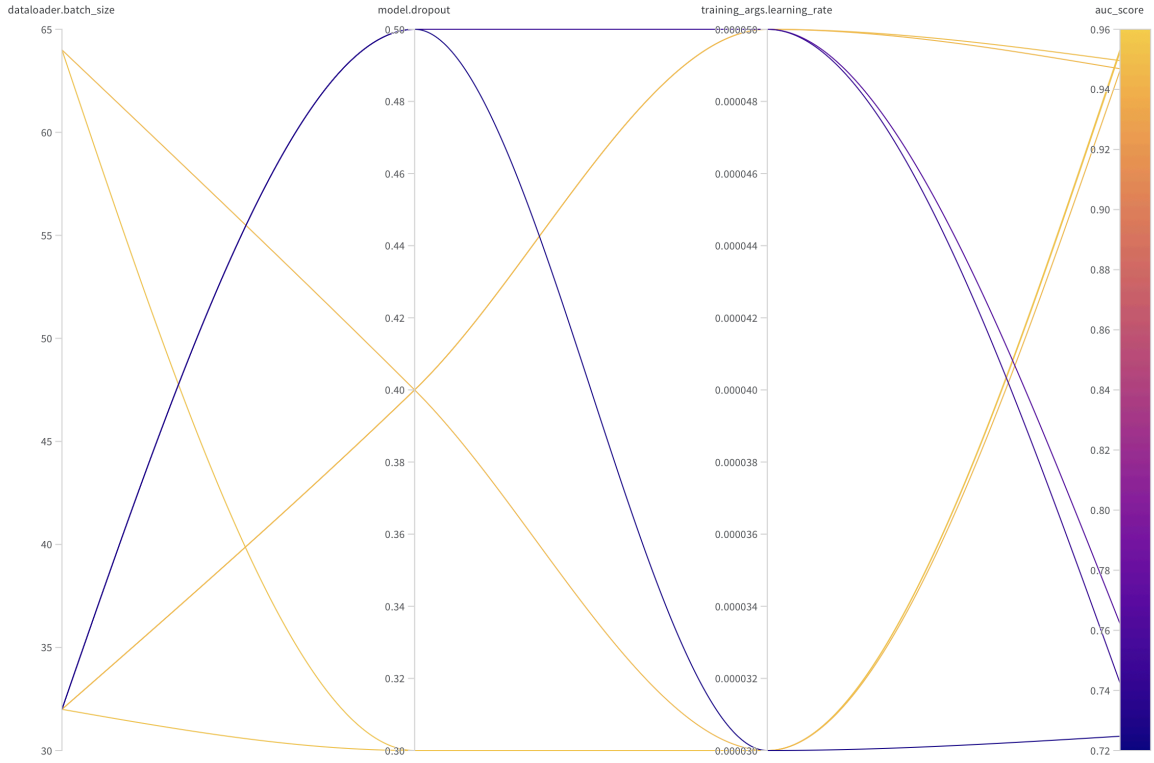
BERT Large AUC Score Curve



BERT Large Hyperparmeter Importance Plot



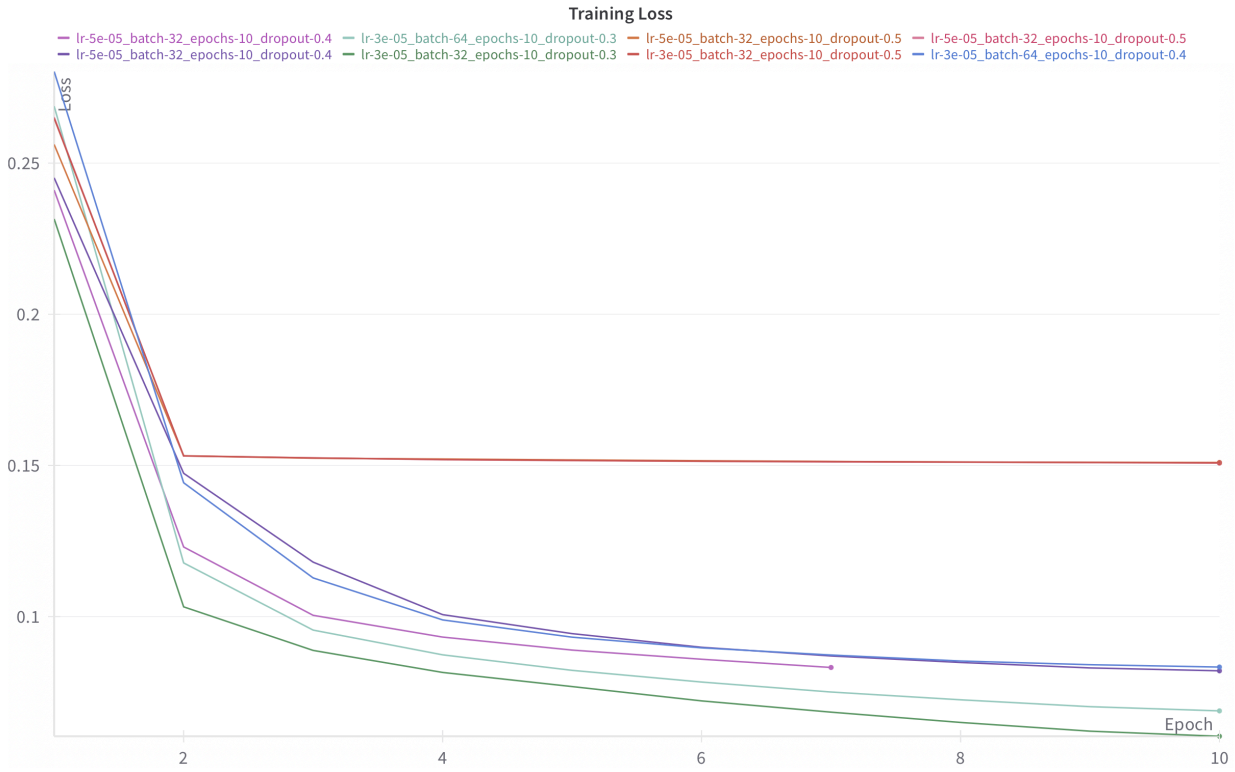
BERT Large Parallel Coordinates Plot



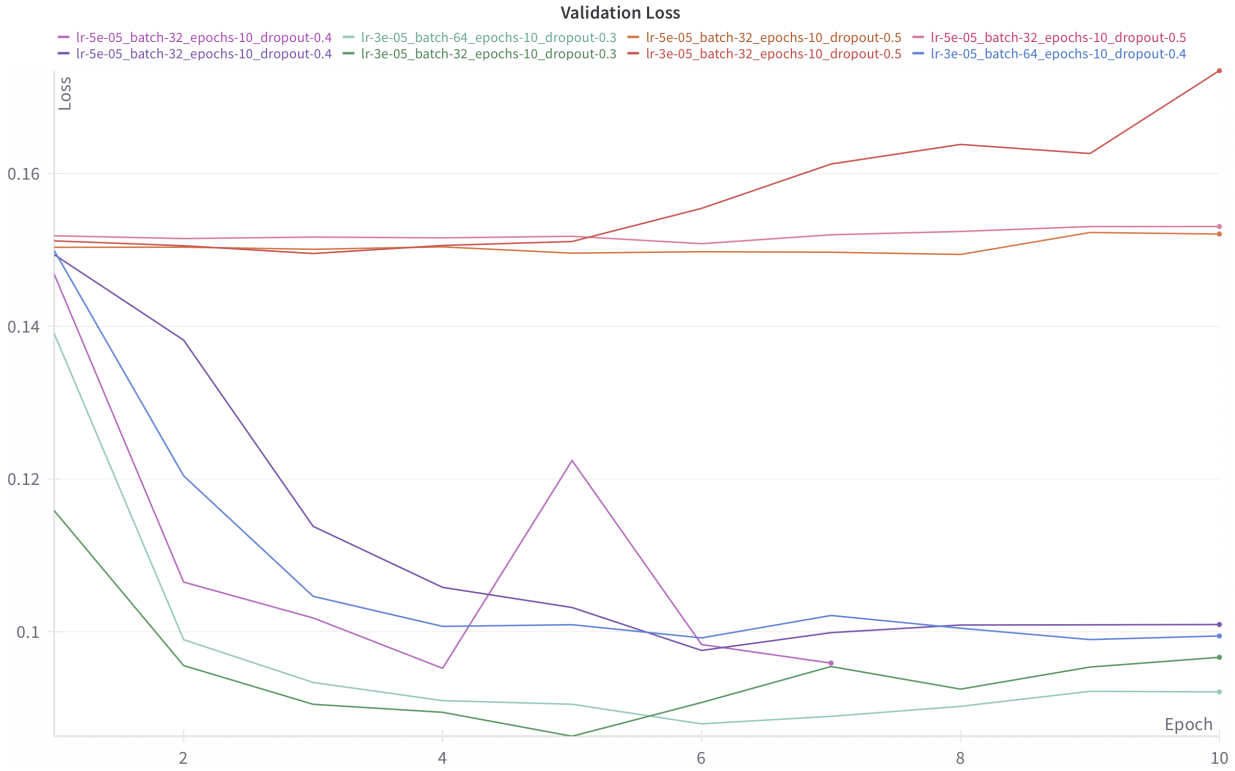
RoBERTa Large Final Performance Metrics Table

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.5	0.765	0.153	0.156
	64	0.3	**0.956**	0.0778	0.089
		0.4	0.764	0.152	0.153
5e-05	32	0.4	0.765	0.152	0.153
	64	0.3	**0.957**	0.078	0.089

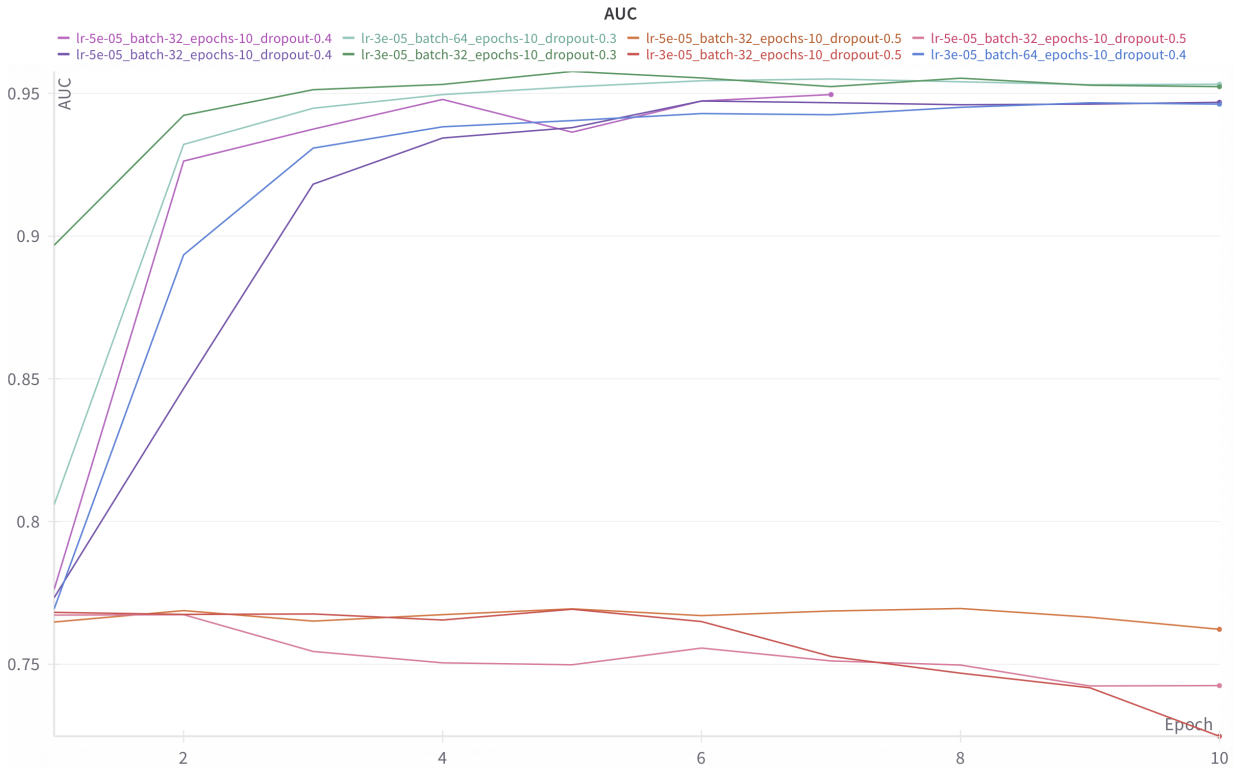
RoBERTa Large Training Loss Curve



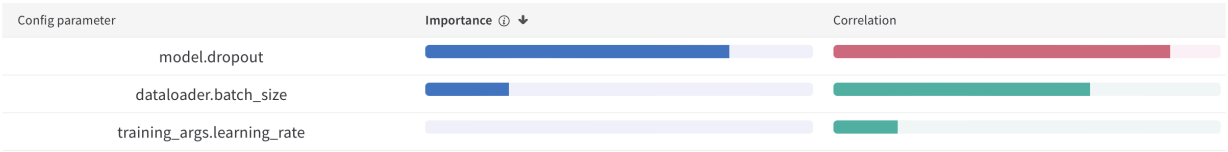
RoBERTa Large Validation Loss Curve



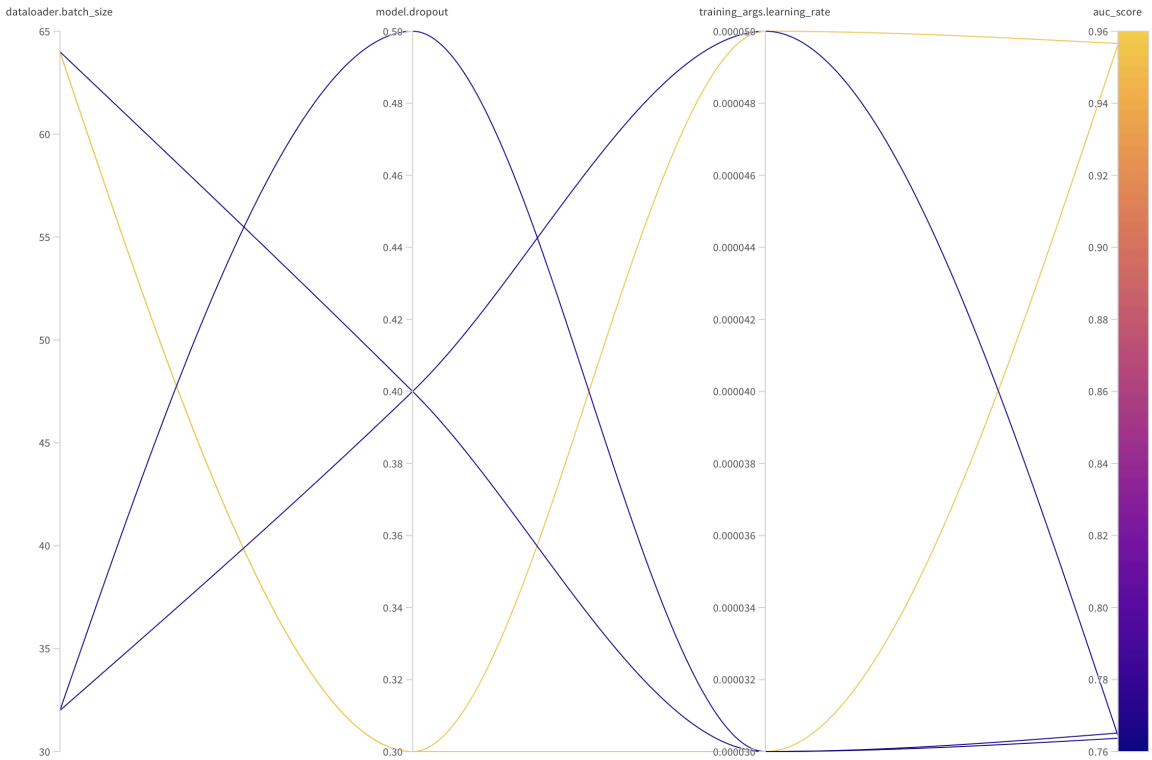
RoBERTa Large AUC Score Curve



RoBERTa Large Hyperparmeter Importance Plot



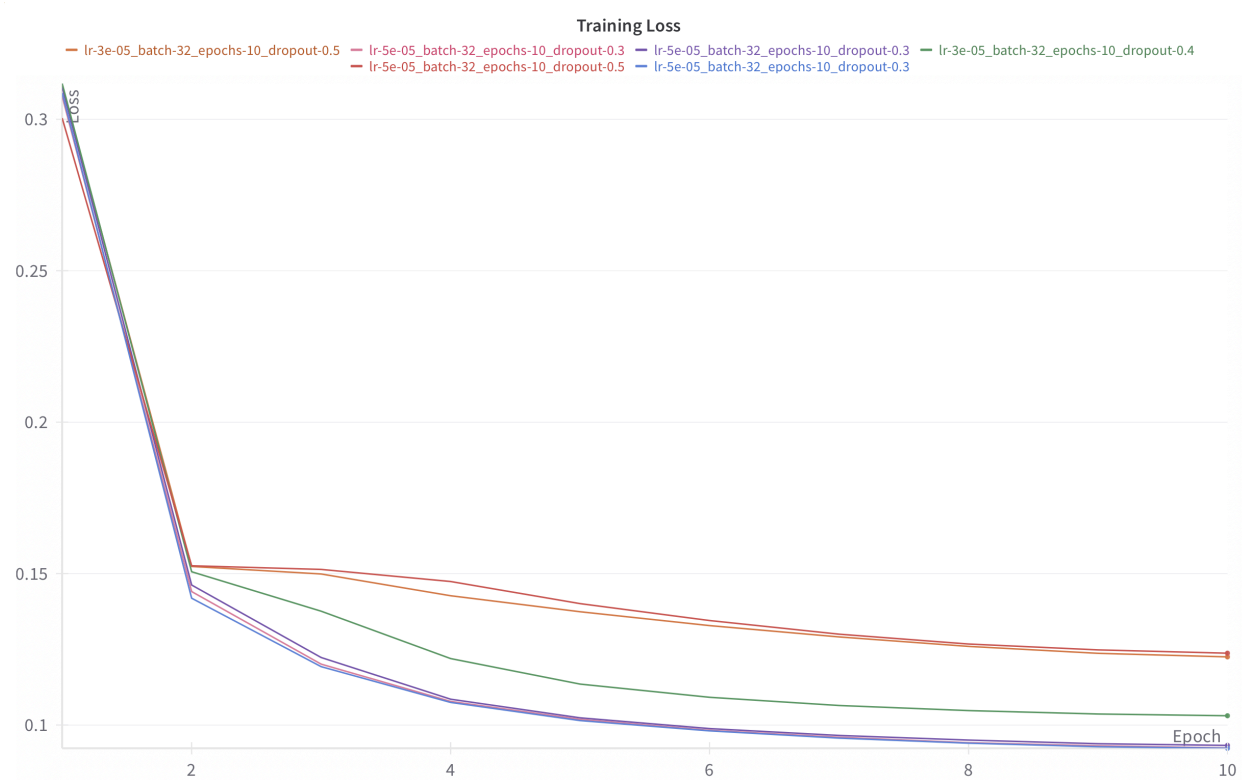
RoBERTa LargeParallel Coordinates Plot



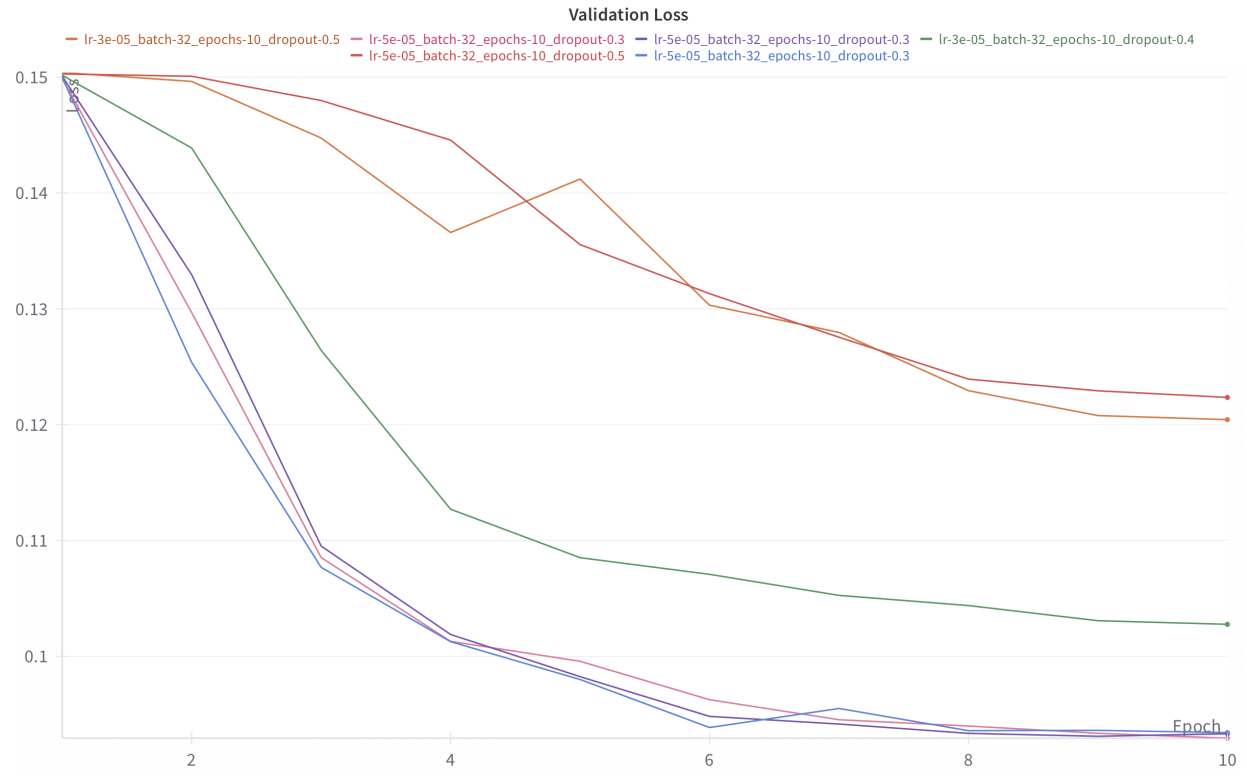
SqueezeBERT Final Performance Metrics Table

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.4	**0.928**	0.103	0.103
		0.5	0.887	0.123	0.120
5e-05	32	0.3	0.944	0.093	0.093
		0.3	0.943	0.092	0.093
		0.3	**0.943**	0.092	0.093
		0.5	0.807	0.146	0.144

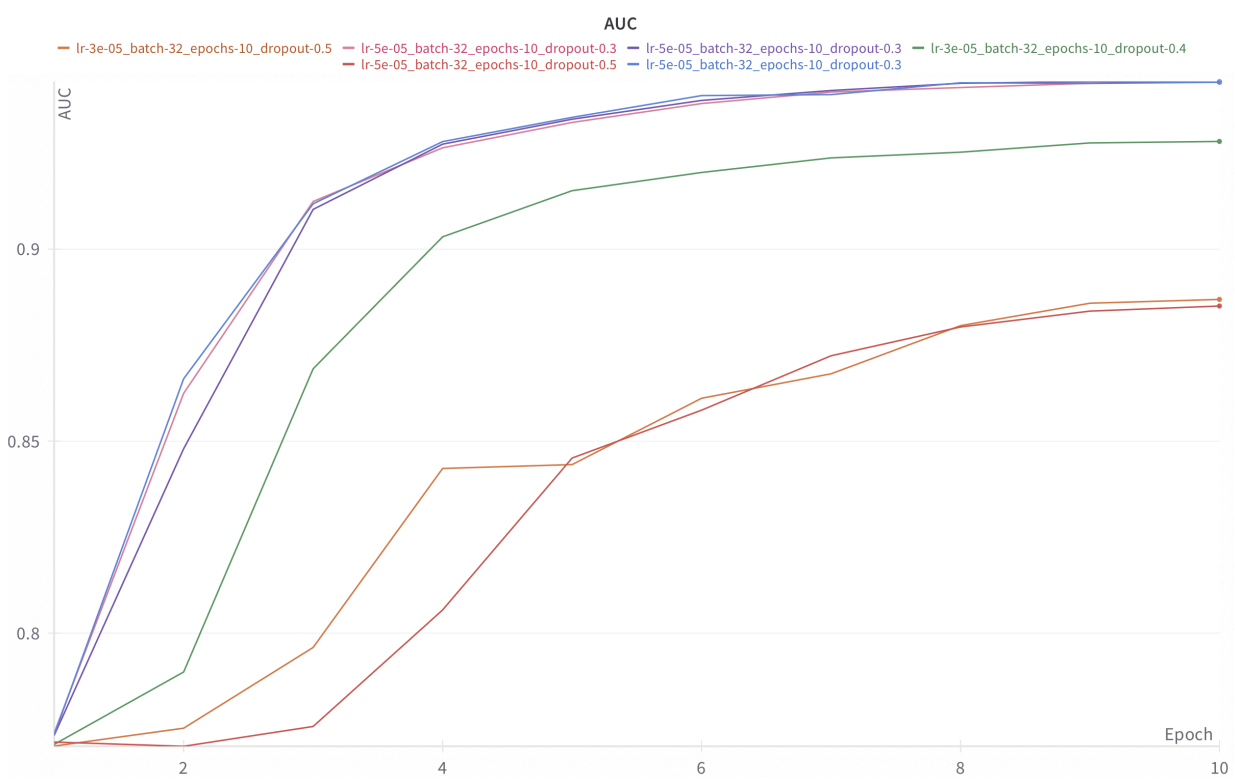
SqueezeBERT Training Loss Curve



SqueezeBERT Validation Loss Curve

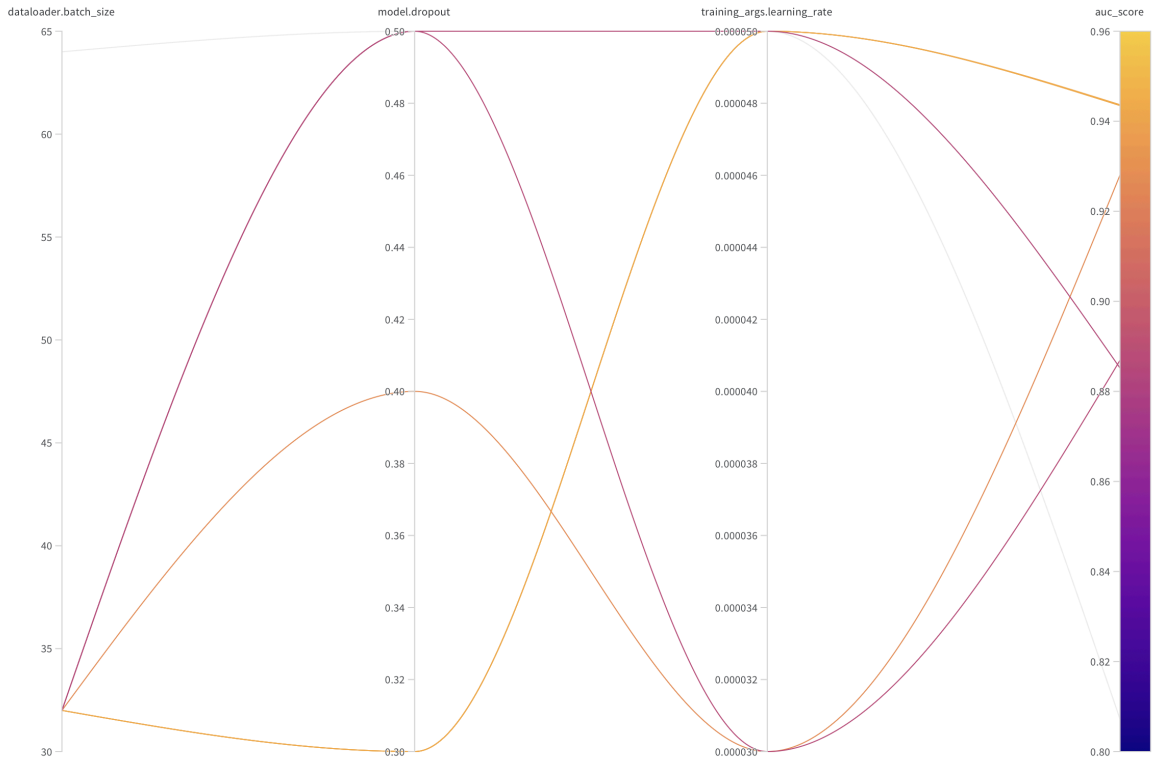


SqueezeBERT AUC Score Curve



SqueezeBERT Hyperparmeter Importance Plot

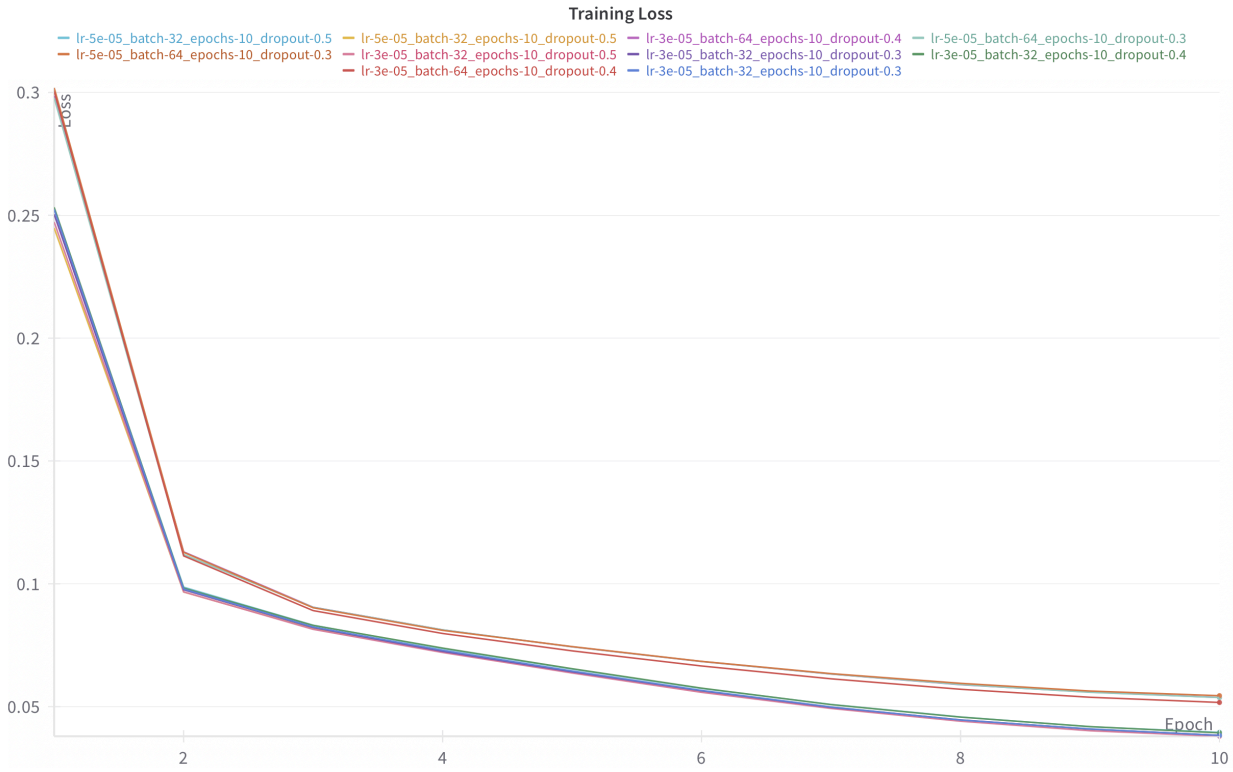
SqueezeBERT Parallel Coordinates



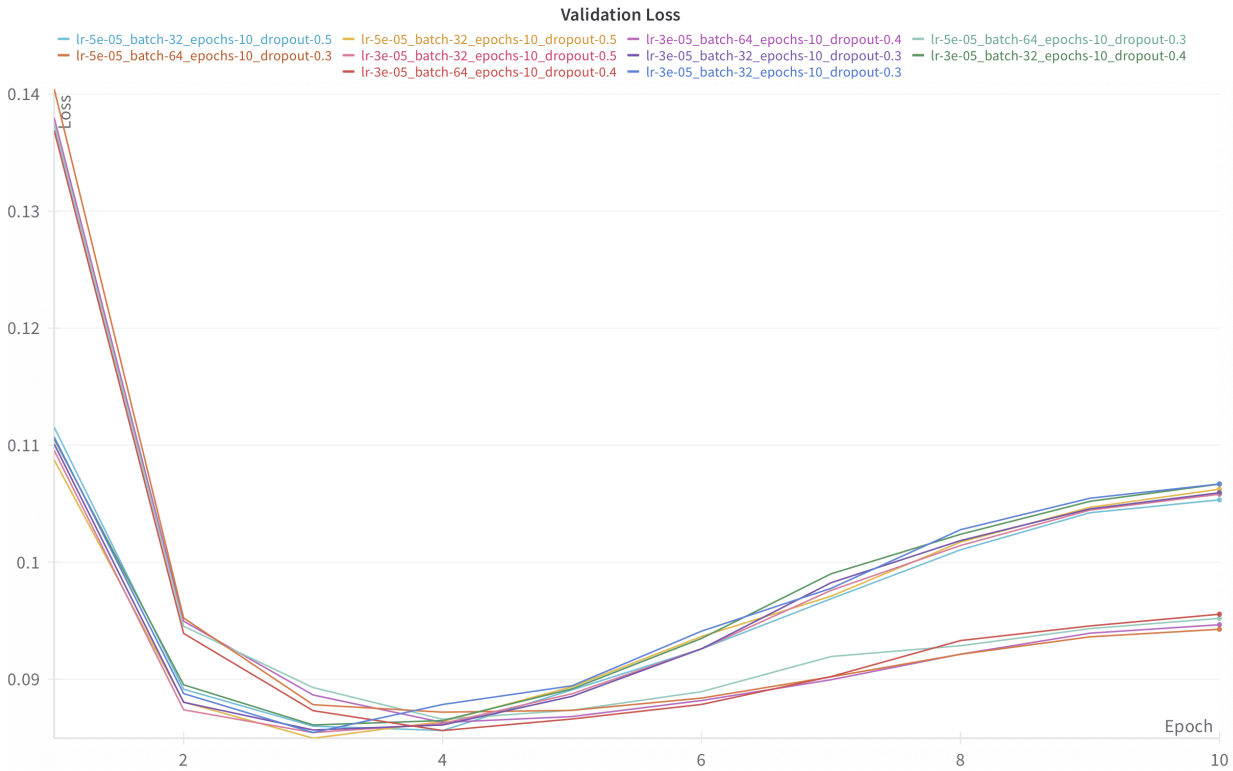
DistilBERT Final Performance Metrics Table

Learning Rate	Batch Size	Dropout	AUC Score	Training Loss	Validation Loss
3e-05	32	0.3	0.940	0.038	0.107
		0.3	0.943	0.038	0.106
		0.4	0.942	0.039	0.107
		0.5	0.942	0.038	0.196
	64	0.4	0.948	0.052	0.096
		0.4	**0.948**	0.054	0.095
5e-05	32	0.5	0.944	0.038	0.106
		0.5	0.943	0.040	0.105
	64	0.3	0.947	0.054	0.095
		0.3	**0.948**	0.054	0.094

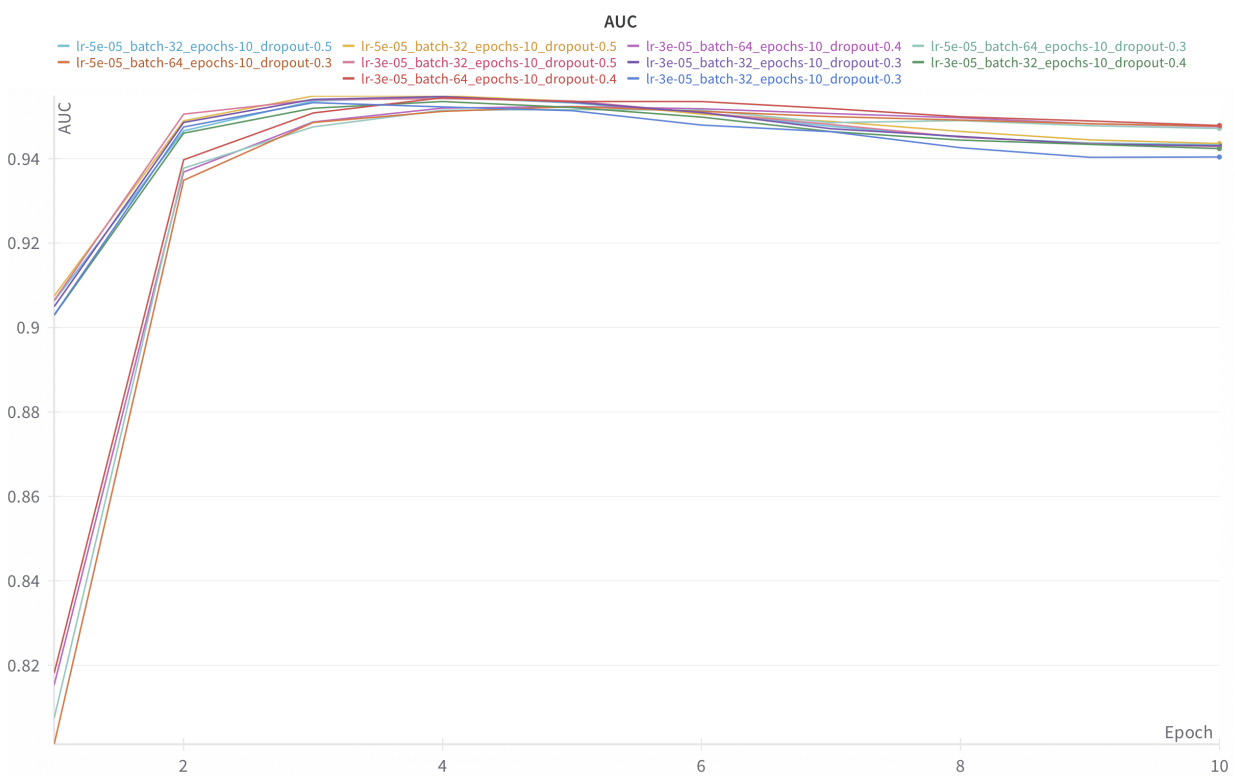
DistilBERT Training Loss Curve



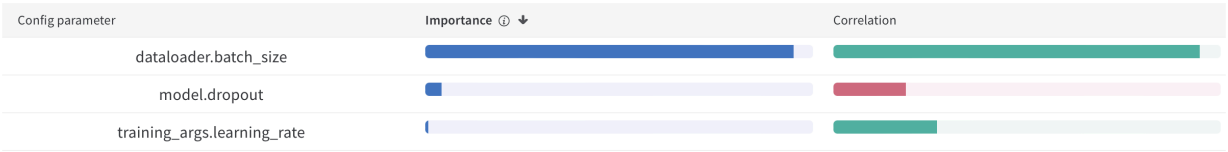
DistilBERT Validation Loss Curve



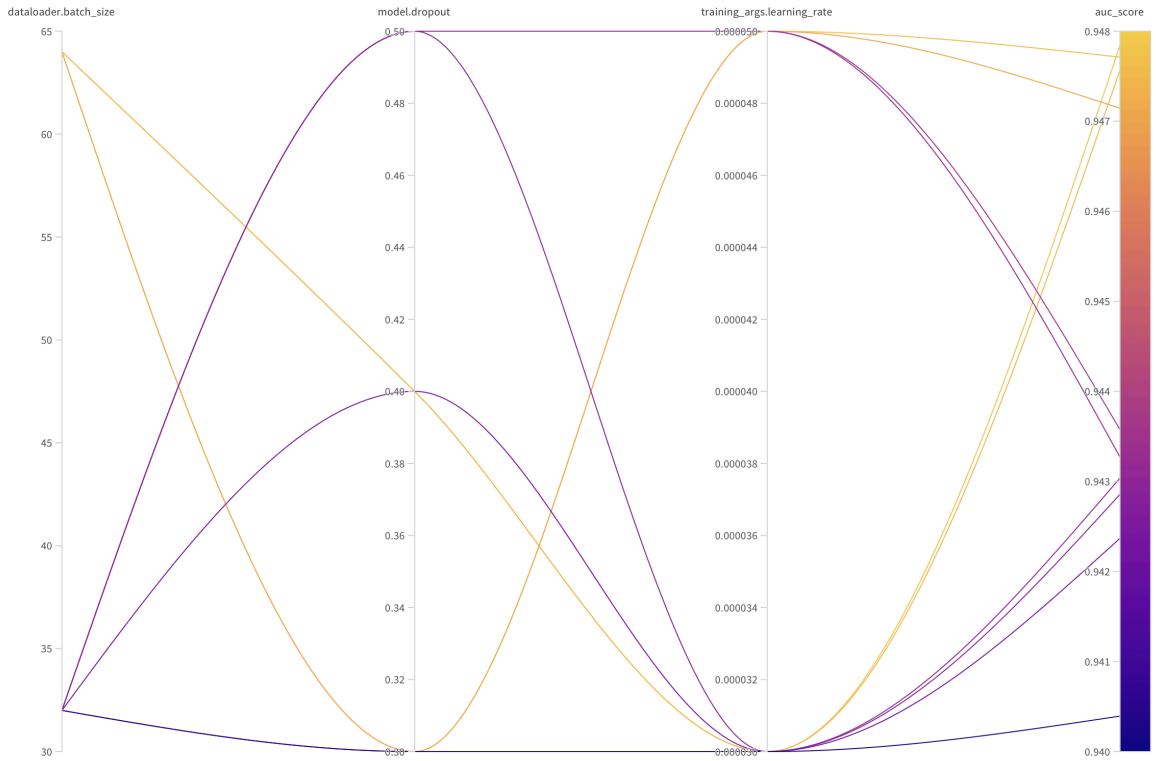
DistilBERT AUC Score Curve



DistilBERT Hyperparmeter Importance Plot



DistilBERT Parallel Coordinates Plot



C. Google Colab Pro

. Compute Units

Reporting alleged “compute units” per sweep.

- Two BERT runs:
  - Compute 99.80 → 92.87
  - Usage Rate: 8.74 per hour
- BERT
  - Compute Units: 92.87 → 71.69
  - Usage Rate: 8.47 per hour.
- RoBERTa
  - Compute Units: 71.69 → 50.52
  - Usage Rate: 8.79 per hour
- BERT-large-uncased
  - Compute Units: 50.52 → 0
  - Usage Rate: 8.72 per hour