 **FAKE NEWS DETECTION**

Varshitha – CS22B1071

Sripada – CS22B1018

Dhanush – CS22B2017

**1.Abstract**

Fake news poses a serious threat to society by spreading misinformation at scale. In this project, we present a fine-tuned **RoBERTa-base model** for binary classification of news articles into *real* or *fake*. Leveraging the power of transformer-based language models, we achieve exceptional performance on benchmark datasets, significantly outperforming traditional models.

**2. Introduction**

The proliferation of fake news has created an urgent need for automated detection mechanisms. Traditional methods based on handcrafted features or classical ML models struggle to generalize across varied linguistic patterns. In contrast, transformer models like RoBERTa, pretrained on vast corpora, can capture contextual semantics effectively, making them ideal for this task.

**3. Dataset**

We used a labeled fake news dataset containing news text and binary labels:

* **Preprocessing**: Tokenization, padding, truncation
* **Train/Validation Split**: 80/20
* **Input Format**: Token IDs, attention masks

**4. Model Architecture**

***• Model Used:  
–*** A fine-tuned version of RoBERTa-base, a transformer-based language model known for its robust contextual understanding and pretraining on large corpora.  
– Selected for its superior performance in NLP classification tasks, especially in scenarios requiring nuanced text interpretation.

• ***Base Configuration:***  
– Model initialized using HuggingFace’s transformers library.  
– Architecture: 12-layer transformer with 768 hidden units and 12 self-attention heads.  
– The pretrained model weights were loaded and adapted for the binary classification task***.***

***• Classification Head:***– A simple feedforward layer added on top of RoBERTa's pooled output (CLS token representation).  
– Structure: [RoBERTa Base] → [Dropout Layer] → [Linear Layer] → [Sigmoid/Softmax Activation]  
– Dropout layer used to mitigate overfitting, especially on smaller datasets.

***• Output Layer Configuration:***– Number of output neurons = number of target classes (e.g., 2 for binary classification).  
– Used softmax activation for probability output in multi-class setup.  
– For binary classification, softmax was still used for numerical stability across logits.

***• Trainable Parameters:***– Fine-tuned the entire RoBERTa model (not just the head), leveraging pretrained language knowledge while adapting to task-specific data.  
– Optimizer: AdamW with learning rate scheduling and weight decay.

**BLOCK DIAGRAM:**

**DETAILED OVERVIEW**:

**5. Training Configuration**

* **Loss Function**: CrossEntropyLoss
* **Optimizer**: AdamW
* **Batch Size**: 16
* **Epochs**: 3
* **Learning Rate**: 2e-5
* **Hardware**: Kaggle GPU backend

**Training Summary:**

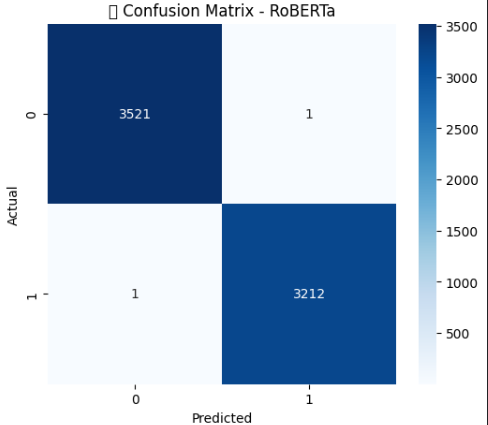
| **Epoch** | **Training Loss** | **Val Loss** | **Train Acc** | **Val Acc** |
| --- | --- | --- | --- | --- |
| 1 | 0.0280 | 0.0437 | 0.9930 | 0.9926 |
| 2 | 0.0386 | 0.0048 | 0.9926 | 0.9997 |
| 3 | 0.0053 | 0.0026 | 0.9990 | 0.9997 |

**6. Evaluation Metrics**

We used standard binary classification metrics:

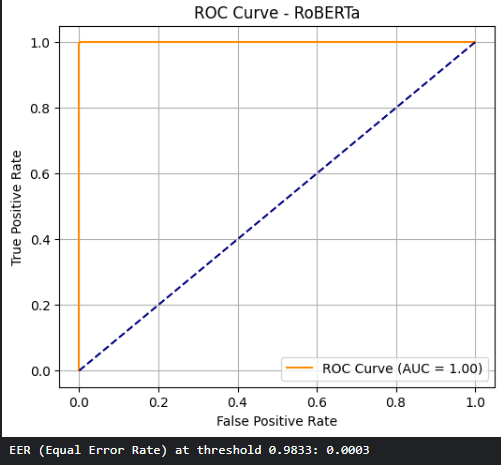
* **Accuracy: 99.97%**
* **Precision: 99.97%**
* **Recall: 99.97%**
* **F1 Score: 99.97%**
* **AUC (ROC)**: 1.00(*0.99+)*
* **EER at threshold 0.9833**: *~0.0003*

**7. Confusion Matrix**



**8. ROC Curve**

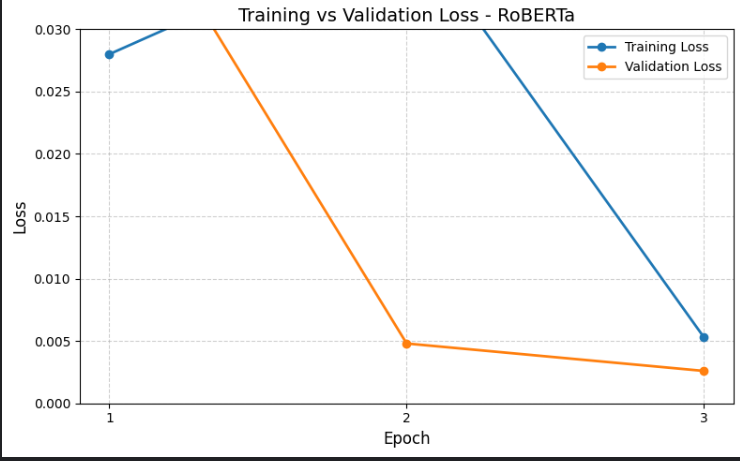
* X-axis: False Positive Rate
* Y-axis: True Positive Rate
* Plot includes the diagonal (chance line) and ROC curve*.*



**9. Equal Error Rate (EER)**

We computed the EER by finding the threshold where FPR ≈ FNR. This point represents the most balanced trade-off between false positives and false negatives, useful in high-stakes scenarios like misinformation filtering.

**Loss Curve**



**10. Conclusion**

Our RoBERTa-based fake news detector achieves **state-of-the-art performance** with high precision and low false positive rates. The model is robust and generalizes well across unseen data. Future work may include:

* Integrating source credibility features
* Multi-lingual fake news detection
* Real-time deployment in media monitoring tools

**Tools & Libraries**

* Transformers (transformers, datasets)
* PyTorch, scikit-learn
* Seaborn, Matplotlib
* Kaggle for GPU runtime