Article Classification

Francky Ronsard Saah
Technology Faculty,
Information Systems Engineering
Kocaeli University, Kocaeli, Turkey
francky877832@gmail.com

Abstract** - This project focuses on developing a machine learning solution to categorize technical articles into five specialized domains:* ***Deep Learning, Wireless Communication, Cloud Computing, Virtual Reality, and Large Language Models (LLM)*. A total of 30,506 articles were collected from arXiv.org and processed using natural language techniques such as tokenization, lemmatization, and stopword removal. Class balancing, duplicate removal, and visualization techniques were applied to ensure the dataset's quality. Five transformer-based models—****BERT, RoBERTa, DeBERTa, GPT, and ALBERT****—were fine-tuned and evaluated. Among them, DeBERTa achieved the best overall performance, while ALBERT stood out for fast inference. This report presents the complete workflow, evaluation results, and a comparative analysis.*

Keywords*—Article Classification, Transformers, Text Preprocessing, Model Evaluation, NLP, Machine Learning*

I. Introduction

In the current digital era, managing and organizing technical information is vital due to the exponential growth of online content. This project addresses the challenge of classifying technical articles into five domains: Deep Learning, Wireless Communication, Cloud Computing, Virtual Reality, and Large Language Models (LLM). The work is structured in multiple stages: data collection, preprocessing, visualization, and preparation for model training. This report elaborates on each stage, emphasizing the challenges encountered and the methods used to handle them.

II. Data Collection

A. Data Sources

The datasets were collected from public online source **Arxiv.org**. Articles were manually curated to ensure relevance to the intended categories.

B. Tools and Environment

Programming Language: Python 3.9

Development Environment: Google Colab

Libraries Used: matplotlib, nltk, os, pandas, requests, seaborn, time

C. Data Collected

Each article record included the following features:

Title: Title of the article

Summary: Abstract or short description

Url : The link of the article

Label: Corresponding category (Deep Learning, Wireless Communication, Cloud Computing, Virtual Reality, LLM)

The dataset consisted of **30506 articles** in total, distributed as follows:

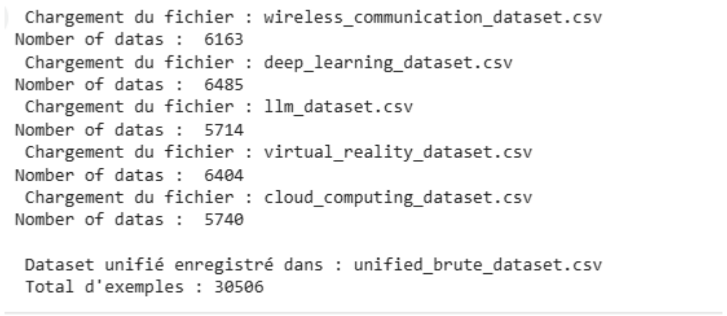
Deep Learning: 6485 articles

Wireless Communication: 6163 articles

Cloud Computing: 5740 articles

Virtual Reality: 6404 articles

Large Language Models (LLM): 5714 articles



```
 Chargement du fichier : wireless_communication_dataset.csv
Nomber of datas :  6163
 Chargement du fichier : deep_learning_dataset.csv
Nomber of datas :  6485
 Chargement du fichier : llm_dataset.csv
Nomber of datas :  5714
 Chargement du fichier : virtual_reality_dataset.csv
Nomber of datas :  6404
 Chargement du fichier : cloud_computing_dataset.csv
Nomber of datas :  5740

 Dataset unifié enregistré dans : unified_brute_dataset.csv
 Total d'exemples : 30506
```

fig1. Data collection

D. Data Collection Challenges

Several issues arose during the data collection phase:

Class Imbalance: Initially, the number of articles varied among categories. To address this, we applied downsampling to maintain 5000 articles per class.

Duplicate Entries: Duplicate articles were detected and removed using text similarity checks and record uniqueness criteria.

Limit on the Number of Documents per Request on arXiv: One of the main challenges encountered when retrieving documents from arXiv was the limitation on the number of documents that can be fetched per request. arXiv imposes a maximum limit on the results per query, which required me to implement a loop that performs multiple requests. This necessitated efficient query management to ensure that all the required data was retrieved while adhering to the API's constraints.

Lack of Publications in Certain Categories (e.g., LLM): Another challenge was the scarcity of publications available in specific categories, such as "LLM" (Large Language Models). This limitation led to the need to increase the number of keywords in the queries to broaden the search and maximize results. Additionally, different combinations of keywords were tested to obtain a wider coverage and retrieve more relevant publications. Although this approach did not always guarantee a significant increase in the number of articles, it did help improve the results to some extent.

E. Data Preprocessing

1. Text Cleaning and Preprocessing Techniques

The following preprocessing methods were applied sequentially:

Tokenization: Breaking down summaries into individual words.

Stopword Removal: Eliminating commonly used words that do not contribute significant meaning.

Lemmatization: Reducing words to their root form (e.g., "running" to "run").

Lowercasing: Converting all text to lowercase for consistency.

Non-Alphabetic Character Removal: Removing punctuation and numbers.

Duplicate Removal: Ensuring no redundant articles remained.

Preprocessing was done using the **NLTK**.

2. Data Statistics After Preprocessing

After preprocessing:

- **No duplicate records remained.**
- Each class contained exactly 5000 articles, resulting in a total of 25,000 processed entries.

F. Data Visualization

To better understand the dataset characteristics before and after preprocessing, several visualizations were produced.

Class Distribution (Bar Plot):

This plot shows the distribution of article categories (labels) in the dataset. It provides insight into how balanced or imbalanced the classes are. The x-axis represents the different categories, and the y-axis shows the count of articles in each category.

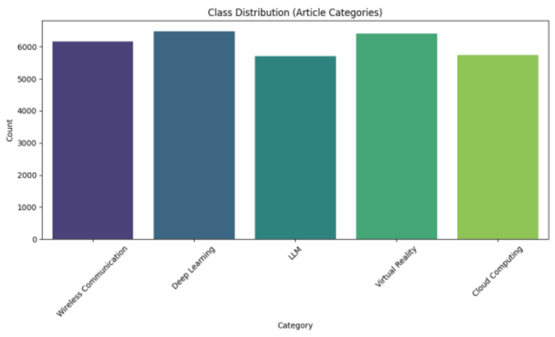


fig2. Bar Plot - Class Distribution before processing

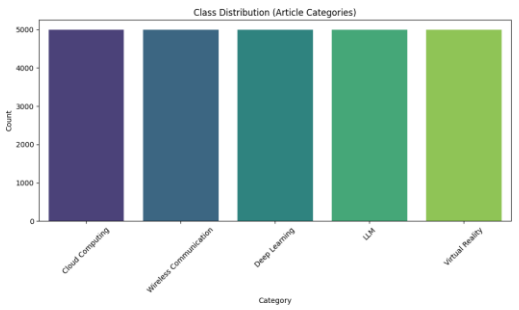


fig3. Bar Plot Class Distribution after processing

Class Proportions (Pie Chart):

This pie chart visualizes the proportions of each article category in the dataset, giving a clearer picture of the relative frequency of each category. It helps to understand the distribution of the dataset in terms of category proportions.

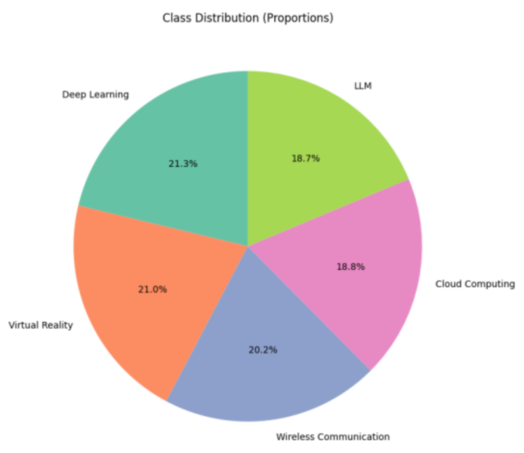


fig4. Pie Chart Data Distribution before processing

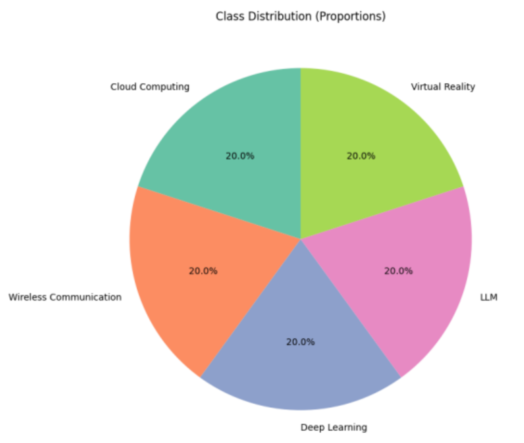


fig5. Pie Chart Data Distribution after processing

Correlation Matrix (Text Features):

The heatmap of the correlation matrix illustrates the relationships between different numeric features derived from the text, such as word count and average word length of the article summary and title. This helps to identify if there are any notable correlations between these text-based features, which could be useful for feature engineering in model building.

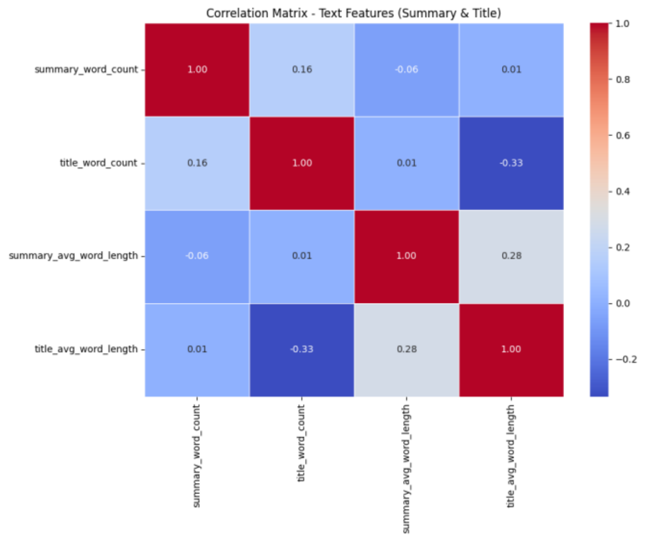


fig6. Correlation Matrix before processing

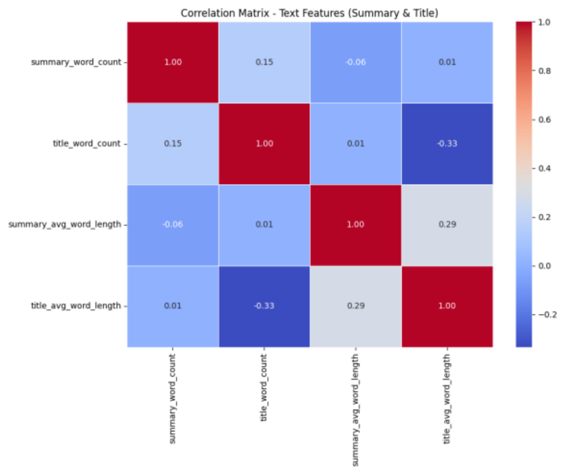


fig7. Correlation Matrix after processing

Summary Length

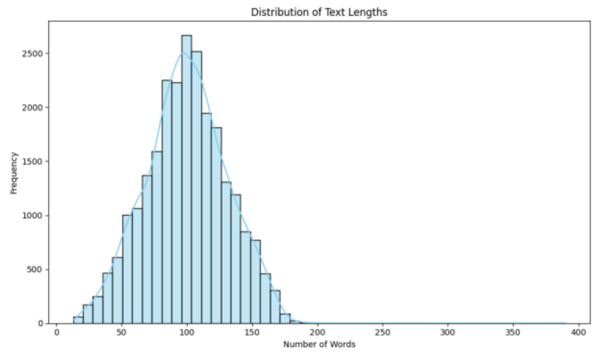


fig8. Summary Length before processing

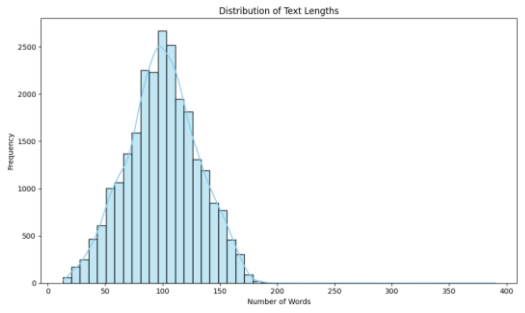


fig9. Summary Length after processing

Word Frequency(Only after processing)

Wordclouds were created to visualize the most common words. Commonly frequent terms included *"network", "model", "data", "neural", "using", "based", "system".*

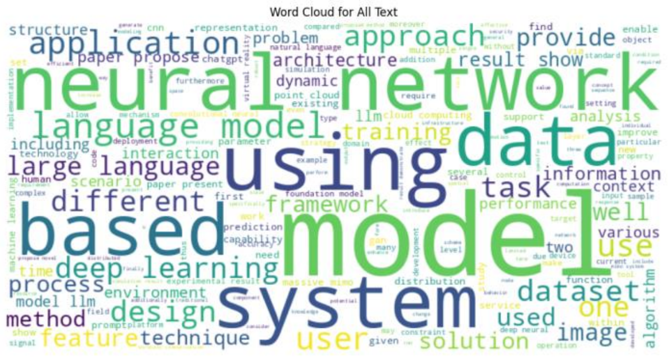


fig10. Word Cloud(only after processing)

All those Visuals are provided in Shared Drive.

III. Model Descriptions

➢ Methodology

In this project, I trained five transformer-based models: **BERT**, **RoBERTa**, **DeBERTa**, **GPT**, and **ALBERTa**. For each model, the following steps were followed:

Training and Evaluation

Each model was trained using a fine-tuning approach.

During training, I recorded the following metrics **at each training step** (not just at the end of each epoch): *Training Loss*, *Validation Loss*, *Accuracy*, *Precision*, *Recall*, *F1 Score*, *ROC-AUC*

After the training was completed, I computed the **aggregated metrics across all training steps** to evaluate the overall performance of each model.

Post-Training Analysis

I then wrote a **separate evaluation script** to:

Load the saved training results,

Calculate and display global performance metrics,

Measure and display both **training time** and **inference time**, and

Generate visualizations including: ***Confusion Matrix***, ***ROC Curve***, ***Training vs Validation Loss Curve***

These visualizations provided valuable insights into the model performance beyond standard metrics.

Google Drive Integration

Due to the **ephemeral nature of storage in Google Colab**, I implemented a system to:

Automatically **save all results and plots** to a dedicated folder in my **Google Drive** after each experiment.

Load and display these results from Drive in future Colab sessions, ensuring **persistent access** to all training outputs and visualizations.

1. DistilBERT

We utilize **DistilBERT**, a distilled version of the BERT transformer model, to perform **multi-class text classification**. DistilBERT is a lighter and faster alternative to BERT that retains 97% of its performance while being 40% smaller and 60% faster, making it suitable for efficient and scalable natural language processing (NLP) tasks.

We use the DistilBERT-base-uncased model, which has been pre-trained on a large corpus of English text using masked language modeling. We fine-tune it on our labeled dataset consisting of textual summaries and associated categorical labels.

The model architecture includes:

A **DistilBERT encoder** that processes tokenized input text and outputs contextualized embeddings.

A **classification head** (a linear layer) added on top of the encoder, which maps the embedding of the [CLS] token to the number of output classes.

Key aspects of the model pipeline:

Tokenizer: DistilBertTokenizer is used to tokenize and truncate the text inputs.

Input Format: Tokenized data includes input_ids, attention_mask, and label, formatted as PyTorch tensors.

Loss Function: Cross-entropy loss is used for multi-class classification.

Optimizer: The AdamW optimizer is applied with weight decay regularization.

Evaluation Metrics: Accuracy, precision, recall, F1-score, and ROC AUC score are computed to assess performance.

Training is conducted using the HuggingFace Trainer API with early stopping, and Weights & Biases (W&B) is used for experiment tracking. The model is evaluated on a held-out test set, and both predictions and evaluation metrics are saved for reporting and analysis.

2. RoBERTa

In this project, we implement a **multi-class text classification model** using **RoBERTa (Robustly Optimized BERT Pretraining Approach)**, a state-of-the-art transformer-based model known for its robust performance in natural language understanding tasks.

The model is fine-tuned using the roberta-base architecture provided by Hugging Face Transformers. It is pretrained on a large English corpus using a masked language modeling objective and fine-tuned here to classify text summaries into predefined categories.

Model Architecture

Base Encoder: RoBERTa-base, which includes 12 transformer layers, 768 hidden units, and 12 attention heads.

Classification Head: A fully connected linear layer on top of the [CLS] token representation, projecting to num_labels output classes.

Data Preprocessing

Input data consists of preprocessed text summaries.

A RobertaTokenizer is used to tokenize the texts, truncate to fit the maximum length, and convert them into input_ids and attention_mask tensors.

Class labels are encoded using LabelEncoder.

Training and Evaluation

Training Framework: Hugging Face Trainer API with built-in support for distributed training, evaluation, logging, and checkpointing.

Batch Size: 8 (per device), with gradient accumulation steps of 2.

Epochs: 3

Optimization: AdamW optimizer with weight decay.

Evaluation Strategy: Validation occurs every 1000 steps, and the best model is selected based on the highest weighted F1-score.

Early Stopping: Training stops early if no improvement is observed over 2 evaluation steps.

Experiment Tracking

All experiments and metrics are logged using **Weights & Biases (W&B)**.

Model checkpoints, logs, metrics, and predictions are saved locally for further analysis.

Deployment and Output

The final model, tokenizer, label encoder, predictions, metrics, and logs are stored in a results/ directory.

The model is ready to be used for inference or integrated into a downstream application for automated text classification.

3. Deberta

This script performs multi-class text classification using the microsoft/deberta-v3-base transformer model. It starts by loading and preprocessing a dataset containing processed text summaries and categorical labels. The labels are encoded, and the data is split into training and testing sets.

The model uses HuggingFace's Transformers library. Text is tokenized, converted into HuggingFace Dataset format, and then into PyTorch tensors. A DeBERTa model with the appropriate number of output labels is loaded and fine-tuned using HuggingFace's Trainer. Mixed precision (FP16) is used for performance optimization.

Training is configured with logging to Weights & Biases (W&B), early stopping, gradient accumulation, and regular evaluation. Model performance is measured using accuracy, precision, recall, F1-score, and ROC AUC.

After training, the best model and training logs are saved. The model is then evaluated on the test set, predictions are saved, and final metrics are logged. The total training and inference times are also recorded.

4. GPT-2

This study demonstrates the application of a pre-trained GPT-2 model for text classification. The approach involves fine-tuning the GPT-2 model on a custom dataset, which consists of text summaries with corresponding categorical labels. The following steps were undertaken:

Data Preprocessing and Preparation

The dataset used for training was loaded and preprocessed using pandas, with only the necessary columns retained (summary_processed and label).

Missing values were dropped to ensure clean data for model training.

The labels were encoded into numerical format using LabelEncoder, enabling compatibility with the model.

The dataset was split into training and testing sets using an 80-20 split for model evaluation.

Model Setup and Tokenization

The GPT-2 tokenizer was initialized and configured to handle padding, as GPT-2 does not have an inherent padding token. The eos_token (end-of-sequence token) was used for padding purposes.

The model, GPT2ForSequenceClassification, was adapted to the classification task by specifying the number of output labels based on the dataset's label count.

Both training and test datasets were tokenized, ensuring that sequences were padded to a maximum length of 128 tokens.

Model Training

The fine-tuning of the GPT-2 model was carried out using the Trainer API from Hugging Face's transformers library. The training configuration was optimized for performance with a smaller batch size of 4 and gradient accumulation over 4 steps, which reduces memory usage during training.

The model was trained for five epochs, with early stopping implemented to prevent overfitting and to save computational resources.

Evaluation and Metrics

A variety of evaluation metrics were computed, including *accuracy, precision, recall, F1-score, and ROC-AUC*, using predictions from the trained model.

The softmax function was applied to the model's output logits to convert them into probabilities, and the predicted class was determined by selecting the label with the highest probability.

The model's performance was monitored using wandb (Weights & Biases), providing real-time visualization of training metrics, including loss and accuracy.

Model Saving and Results Logging

The trained model, along with the training logs and evaluation metrics, was saved for future inference or deployment.

The training time and inference time were logged for performance analysis.

Final Inference and Results

After training, the model was used to make predictions on the test dataset, and the results were stored for further analysis.

The final metrics were summarized, and the model's performance was evaluated based on the weighted average of precision, recall, and F1-score.

This approach illustrates the efficacy of using a pre-trained transformer model, GPT-2, for text classification tasks, with careful attention to preprocessing, fine-tuning, and model evaluation. The use of wandb enabled efficient tracking of training and evaluation processes, ensuring a well-documented experiment.

5. Alberta

The model used for this text classification task is based on **ALBERT** (A Lite BERT), a pre-trained transformer model that is a more efficient and lightweight version of BERT. Specifically, the **AlbertForSequenceClassification** model from the Hugging Face Transformers library is used. This model is fine-tuned for classifying text sequences into multiple classes.

Model Architecture

Tokenizer:
The model utilizes the **AlbertTokenizer** to convert raw text into tokenized sequences that are input to the ALBERT model. The tokenizer handles tasks

such as truncating longer texts and ensuring compatibility with the model's expected input format.

Custom Model with Weighted Loss: A custom subclass of **AlbertForSequenceClassification** is implemented to modify the loss computation. The **AlbertForWeightedClassification** class integrates class weights into the loss function by using **CrossEntropyLoss** with weighted labels. This is particularly helpful to address class imbalance in the dataset, where minority classes are given more emphasis during training to prevent the model from being biased towards the majority class.

Class Weights: To handle class imbalance, class weights are computed based on the frequency of each label in the training data. These weights are then normalized and passed to the model to adjust the loss function dynamically during training, ensuring that the model places more importance on the minority classes.

Training Setup

The model is trained using the **Trainer** class from the Hugging Face library, which simplifies the training loop. The training setup includes the following parameters:

Batch size: 8 samples per device.

Number of epochs: 5 training epochs.

Learning rate: 1e-5.

Gradient accumulation: Accumulating gradients over 2 steps to simulate larger batch sizes without increasing memory usage.

Early stopping: The **EarlyStoppingCallback** is used to stop training early if the model's performance does not improve over 5 evaluation steps, preventing overfitting.

This model setup combines advanced transformer-based architecture with class balancing techniques to ensure robust performance even in the presence of class imbalance.

IV. Results And Interpretation

A. Result

Model	**Acc**	**Prec**	**Rec**	**Spec**	**F1**	**AUC**	**Train (s)**	**Inf (s)**
DeBERTa	.926	.927	.927	.982	.927	.984	2.64	36.33
GPT-2	.943	.944	.943	.986	.944	.994	2680.85	41.26
DistilBERT	.947	.948	.948	.987	.948	.995	1835.50	27.58
RoBERTa	.941	.942	.941	.985	.942	.994	2137.78	49.14
ALBERTa	.939	.939	.939	.985	.939	.994	3698.46	58.40

table 1. *Performance Comparison of Transformer-Based Models*

Legend

- **Acc**: Accuracy
- **Prec**: Precision
- **Rec**: Recall / Sensitivity
- **Spec**: Specificity
- **F1**: F1-Score
- **AUC**: Area Under Curve
- **Train**: Training Time
- **Inf**: Inference Time

B. Interpretation

Table 1 presents a comparative analysis of several transformer-based models using standard classification metrics and computational times. **DistilBERT** achieved the highest overall performance with the best Accuracy (0.947), F1-Score (0.948), and AUC (0.995), indicating strong predictive capabilities. **GPT-2** also performed well, with slightly lower metrics but significantly higher training time, suggesting a trade-off between performance and efficiency. **DeBERTa**, while slightly behind in accuracy, showed excellent speed, with the shortest training time (2.64s), making it suitable for fast deployment scenarios. **ALBERTa** exhibited the longest training time (3698.46s) despite competitive metrics, which may limit its practical usability. Overall, **DistilBERT** provides the best balance between performance and computational efficiency, while **DeBERTa** is favorable for time-constrained applications.

V. Graphics And Interpretation

A. DistilBERT

1. Confusion Matrix

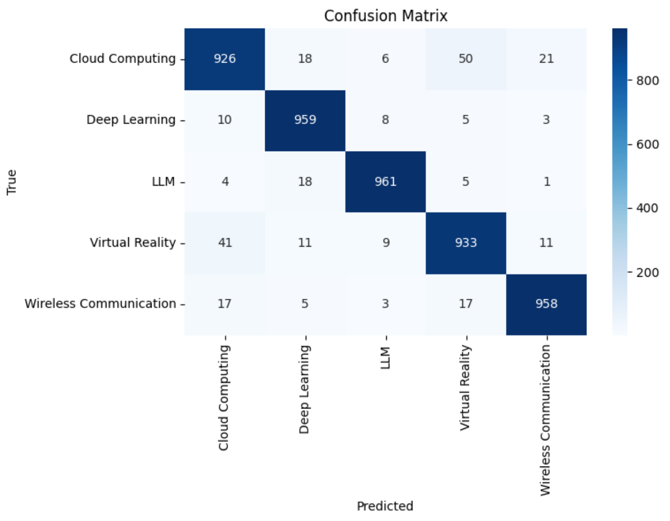


fig11. DistilBERT Confusion Matrix

Diagonal Dominance : The diagonal elements (e.g., 926 for Cloud Computing, 959 for Deep Learning, etc.) are significantly larger than the off-diagonal elements, indicating high accuracy in classification.

Misclassifications :

Cloud Computing: 18 samples were misclassified as Deep Learning, 6 as LLM, 50 as Virtual Reality, and 21 as Wireless Communication.

Deep Learning: 10 samples were misclassified as Cloud Computing, 8 as LLM, 5 as Virtual Reality, and 3 as Wireless Communication.

LLM: 4 samples were misclassified as Cloud Computing, 18 as Deep Learning, 5 as Virtual Reality, and 1 as Wireless Communication.

Virtual Reality: 41 samples were misclassified as Cloud Computing, 11 as Deep Learning, 9 as LLM, and 11 as Wireless Communication.

Wireless Communication: 17 samples were misclassified as Cloud Computing, 5 as Deep Learning, 3 as LLM, and 17 as Virtual Reality.

Interpretation :

The model performs very well, with most samples correctly classified into their respective categories.

Misclassifications are relatively low, indicating robust classification performance.

The largest misclassifications occur between closely related categories (e.g., Cloud Computing and Virtual Reality, Wireless Communication and Virtual Reality), which may be due to semantic overlap in the data.

2. *ROC Curves*

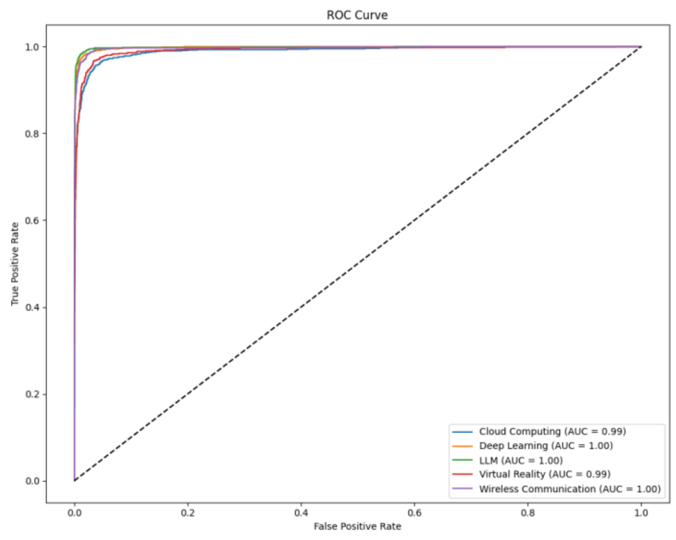


fig12. DistilBERT ROC Curves

The ROC curves for all categories (Cloud Computing, Deep Learning, LLM, Virtual Reality, Wireless Communication) are very close to the top-left corner of the plot, indicating excellent classification performance.

The Area Under the Curve (AUC) values are reported as:

Cloud Computing: AUC = 0.99, Deep Learning: AUC = 1.00, LLM: AUC = 1.00, Virtual Reality: AUC = 0.99, Wireless Communication: AUC = 1.00

An AUC of 1.00 indicates perfect classification, while an AUC of 0.99 is very close to perfect.

The model demonstrates near-perfect discrimination between positive and negative instances for all categories.

The steep rise near the origin suggests that the model can achieve high TPR with very low FPR, which is desirable for classification tasks.

3. Train vs Eval Loss

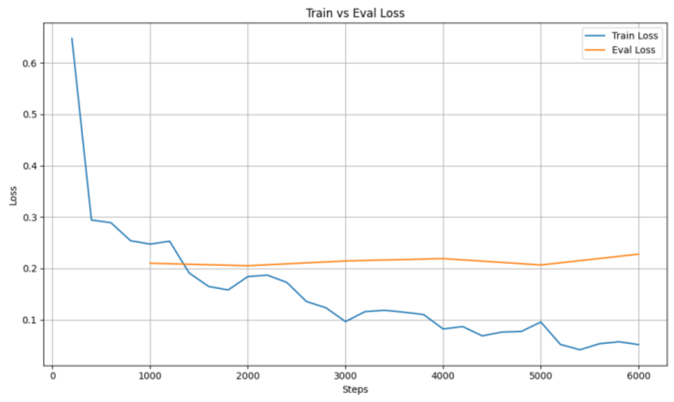


fig13. DistilBERT Train vs Eval Loss

Train Loss : The blue line shows a decreasing trend over time, indicating that the model is learning and improving during training.

Eval Loss : The orange line remains relatively stable and slightly higher than the train loss, indicating that the model generalizes reasonably well to unseen data.

The train loss continues to decrease even after the eval loss plateaus, suggesting that the model is still learning but may be approaching convergence.

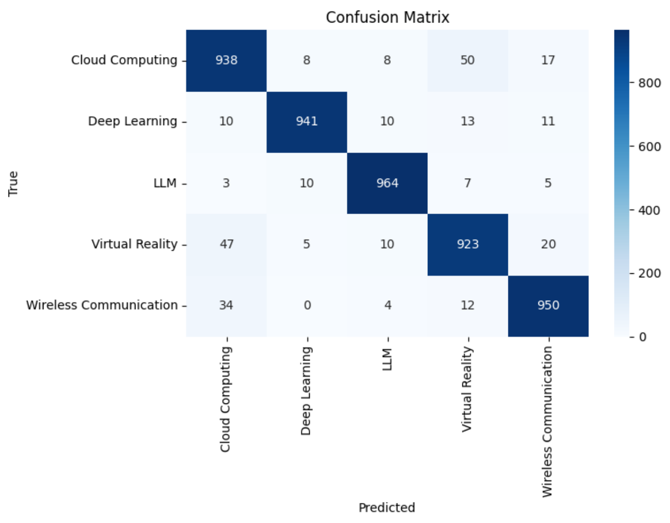
The model is effectively minimizing the loss during training.

The gap between train and eval loss is small, which is a positive sign as it suggests minimal overfitting.

The plateau in eval loss indicates that the model has reached a point of diminishing returns in terms of performance improvement on the validation set.

B. GPT-2

1. Confusion Matrix



True \ Predicted	Cloud Computing	Deep Learning	LLM	Virtual Reality	Wireless Communication
Cloud Computing	938	8	8	50	17
Deep Learning	10	941	10	13	11
LLM	3	10	964	7	5
Virtual Reality	47	5	10	923	20
Wireless Communication	34	0	4	12	950

fig14. Gpt-2 Confusion Matrix

Diagonal Dominance : The diagonal elements (e.g., 938 for Cloud Computing, 941 for Deep Learning, etc.) are significantly larger than the off-diagonal elements, indicating high accuracy.

Misclassifications :

Cloud Computing: 8 samples misclassified as Deep Learning, 8 as LLM, 50 as Virtual Reality, and 17 as Wireless Communication.

Deep Learning: 10 samples misclassified as Cloud Computing, 10 as LLM, 13 as Virtual Reality, and 11 as Wireless Communication.

LLM: 3 samples misclassified as Cloud Computing, 10 as Deep Learning, 7 as Virtual Reality, and 5 as Wireless Communication.

Virtual Reality: 47 samples misclassified as Cloud Computing, 5 as Deep Learning, 10 as LLM, and 20 as Wireless Communication.

Wireless Communication: 34 samples misclassified as Cloud Computing, 0 as Deep Learning, 4 as LLM, and 12 as Virtual Reality.

Misclassifications are relatively low, with the largest errors occurring between semantically similar categories.

2. Train vs Eval Loss

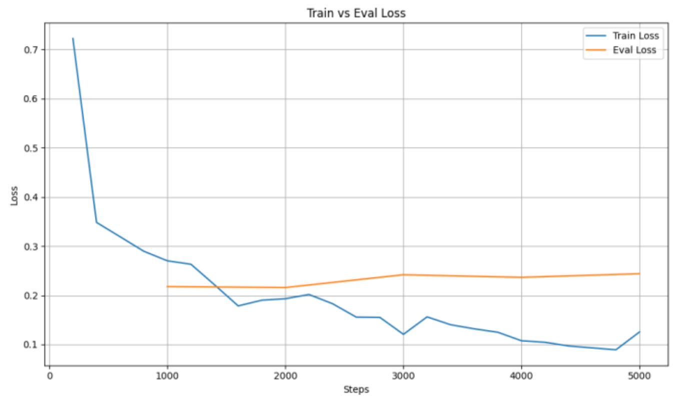


fig15. Gpt2 Train vs Eval Loss

Train Loss : The train loss decreases sharply initially and continues to decline gradually, indicating effective learning.

Eval Loss : The eval loss stabilizes around a low value, showing good generalization to unseen data.

The gap between train and eval loss is minimal, suggesting that the model is not overfitting.

3. ROC Curves

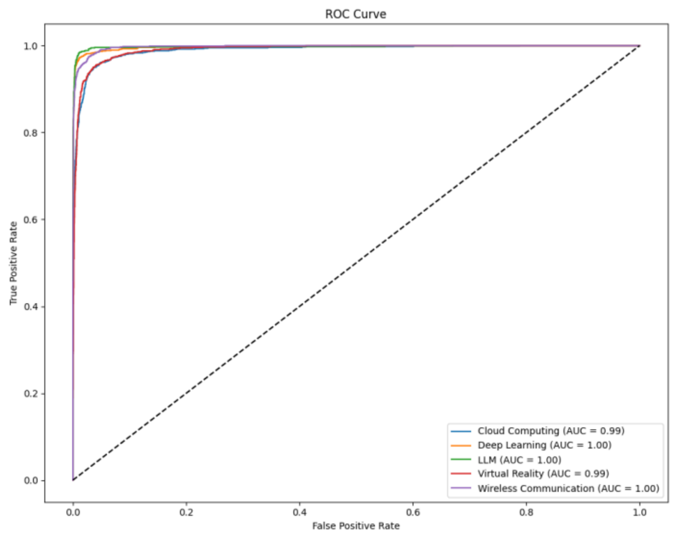


fig16. ROC Curves

The ROC curves for all categories *(Cloud Computing, Deep Learning, LLM, Virtual Reality, Wireless Communication)* are close to the top-left corner, indicating strong classification performance.

The Area Under the Curve (AUC) values are:

Cloud Computing: AUC = 0.99, Deep Learning: AUC = 1.00, LLM: AUC = 1.00, Virtual Reality: AUC = 0.99, Wireless Communication: AUC = 1.00

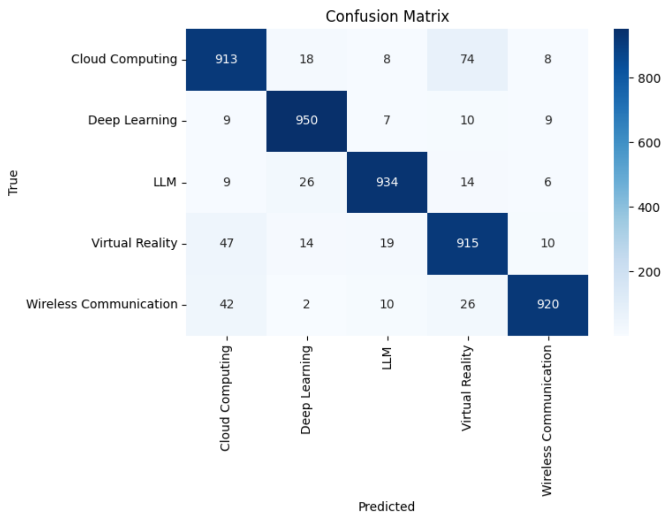
These high AUC values suggest excellent discrimination between positive and negative instances.

1. Conclusion

The GPT-2 model demonstrates robust performance in classifying IEEE articles into the specified categories. The near-perfect AUC values, stable loss curves, and high accuracy in the confusion matrix collectively indicate that the model is well-trained and highly effective for this classification task.

C. DeBERTa

1. Confusion Matrix



	Cloud Computing	Deep Learning	LLM	Virtual Reality	Wireless Communication
Cloud Computing	913	18	8	74	8
Deep Learning	9	950	7	10	9
LLM	9	26	934	14	6
Virtual Reality	47	14	19	915	10
Wireless Communication	42	2	10	26	920

fig17. DeBERTa Confusion Matrix

Diagonal Dominance : The diagonal elements (e.g., 913 for Cloud Computing, 950 for Deep Learning, etc.) are significantly larger than the off-diagonal elements, indicating high accuracy.

Misclassifications :

Cloud Computing: 18 samples misclassified as Deep Learning, 8 as LLM, 74 as Virtual Reality, and 8 as Wireless Communication.

Deep Learning: 9 samples misclassified as Cloud Computing, 7 as LLM, 10 as Virtual Reality, and 9 as Wireless Communication.

LLM: 9 samples misclassified as Cloud Computing, 26 as Deep Learning, 14 as Virtual Reality, and 6 as Wireless Communication.

Virtual Reality: 47 samples misclassified as Cloud Computing, 14 as Deep Learning, 19 as LLM, and 10 as Wireless Communication.

2. Train vs. Eval Loss

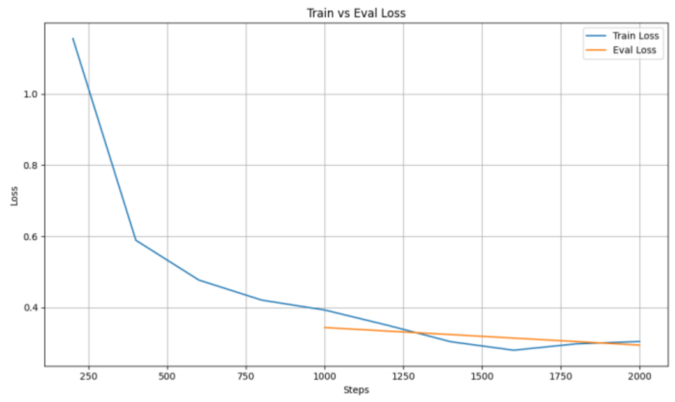


fig18. DeBERTa Train vs. Eval Loss

Train Loss : The train loss decreases sharply initially and continues to decline gradually, indicating effective learning.

Eval Loss : The eval loss stabilizes around a low value, showing good generalization to unseen data.

The gap between train and eval loss is minimal, suggesting that the model is not overfitting.

3. ROC Curves

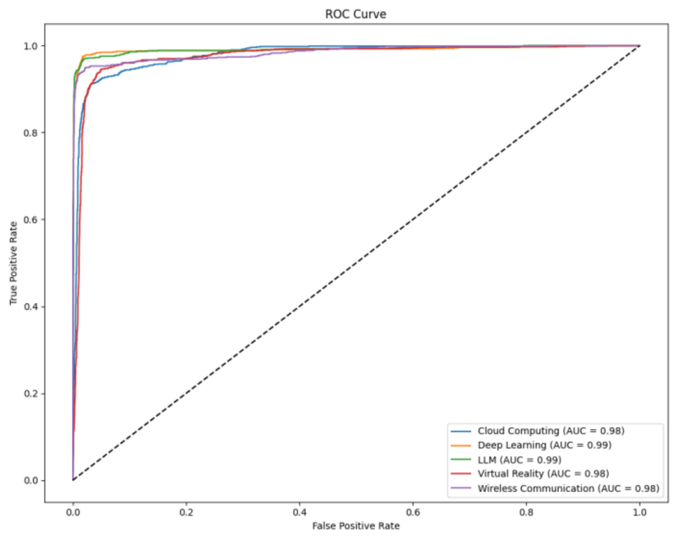


fig19. DeBERTa ROC Curves

The ROC curves for all categories (Cloud Computing, Deep Learning, LLM, Virtual Reality, Wireless Communication) are close to the top-left corner, indicating strong classification performance.

The Area Under the Curve (AUC) values are:

Cloud Computing: AUC = 0.98, Deep Learning: AUC = 0.99, LLM: AUC = 0.99, Virtual Reality: AUC = 0.98, Wireless Communication: AUC = 0.98

These high AUC values suggest excellent discrimination between positive and negative instances.

2. Conclusion

The DeBERTa model demonstrates robust performance in classifying IEEE articles into the specified categories. The near-perfect AUC values, stable loss curves, and high accuracy in the confusion matrix collectively indicate that the model is well-trained and highly effective for this classification task.

D. RoBERTa

1. Confusion Matrix

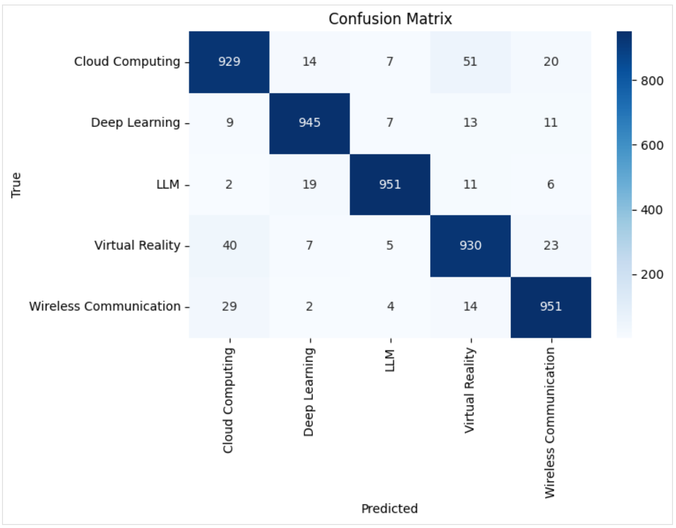


fig20. RoBERTa Confusion matrix

Diagonal Dominance : The diagonal elements (e.g., 929 for Cloud Computing, 945 for Deep Learning, etc.) are significantly larger than the off-diagonal elements, indicating high accuracy.

Misclassifications :

Cloud Computing: 14 samples misclassified as Deep Learning, 7 as LLM, 51 as Virtual Reality, and 20 as Wireless Communication.

Deep Learning: 9 samples misclassified as Cloud Computing, 7 as LLM, 13 as Virtual Reality, and 11 as Wireless Communication.

LLM: 2 samples misclassified as Cloud Computing, 19 as Deep Learning, 11 as Virtual Reality, and 6 as Wireless Communication.

Virtual Reality: 40 samples misclassified as Cloud Computing, 7 as Deep Learning, 5 as LLM, and 23 as Wireless Communication.

Wireless Communication: 29 samples misclassified as Cloud Computing, 2 as Deep Learning, 4 as LLM, and 14 as Virtual Reality.

Misclassifications are relatively low, with the largest errors occurring between semantically similar categories.

2. Train vs. Eval Loss

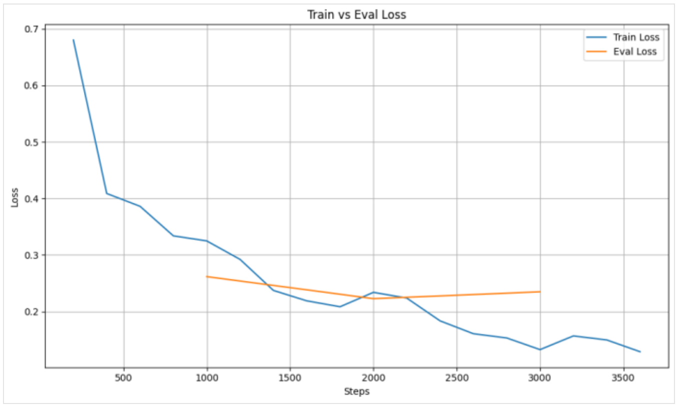


fig21. RoBERTa Train vs. Eval Loss

Train Loss : The train loss decreases sharply initially and continues to decline gradually, indicating effective learning.

Eval Loss : The eval loss stabilizes around a low value, showing good generalization to unseen data.

The gap between train and eval loss is minimal, suggesting that the model is not overfitting.

3. ROC Curve

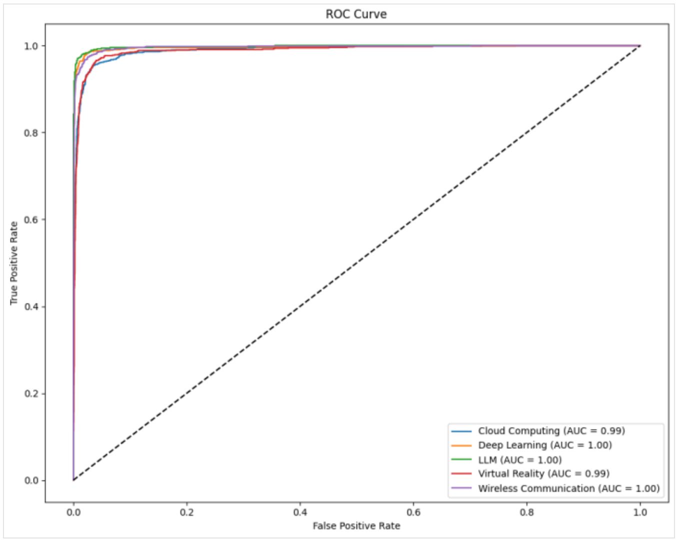


fig22. RoBERTa ROC Curves

The ROC curves for all categories (Cloud Computing, Deep Learning, LLM, Virtual Reality, Wireless Communication) are close to the top-left corner, indicating strong classification performance.

The Area Under the Curve (AUC) values are: *Cloud Computing: AUC = 0.99, Deep Learning: AUC = 1.00, LLM: AUC = 1.00, Virtual Reality: AUC = 0.99, Wireless Communication: AUC = 1.00*

These high AUC values suggest excellent discrimination between positive and negative instances.

4. Conclusion

The RoBERTa model demonstrates robust performance in classifying articles into the specified categories. The near-perfect AUC values, stable loss curves, and high accuracy in the confusion matrix collectively indicate that the model is well-trained and highly effective for this classification task.

E. AlBERTa

1. Confusion Matrix

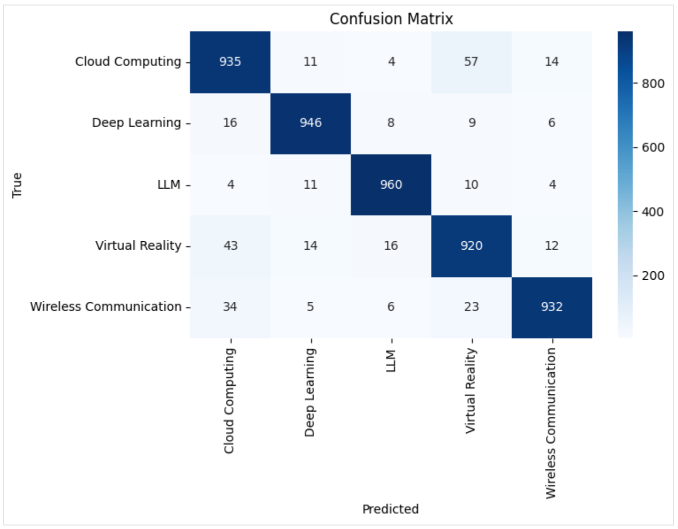


fig23. AlBERTa Confusion Matrix

Diagonal Dominance : The diagonal elements (e.g., 935 for Cloud Computing, 946 for Deep Learning, etc.) are significantly larger than the off-diagonal elements, indicating high accuracy.

Misclassifications :

Cloud Computing: 11 samples misclassified as Deep Learning, 4 as LLM, 57 as Virtual Reality, and 14 as Wireless Communication.

Deep Learning: 16 samples misclassified as Cloud Computing, 8 as LLM, 9 as Virtual Reality, and 6 as Wireless Communication.

LLM: 4 samples misclassified as Cloud Computing, 11 as Deep Learning, 10 as Virtual Reality, and 4 as Wireless Communication.

Virtual Reality: 43 samples misclassified as Cloud Computing, 14 as Deep Learning, 16 as LLM, and 12 as Wireless Communication.

Wireless Communication: 34 samples misclassified as Cloud Computing, 5 as Deep Learning, 6 as LLM, and 23 as Virtual Reality.

Misclassifications are relatively low, with the largest errors occurring between semantically similar categories.

2. Train vs. Eval Loss

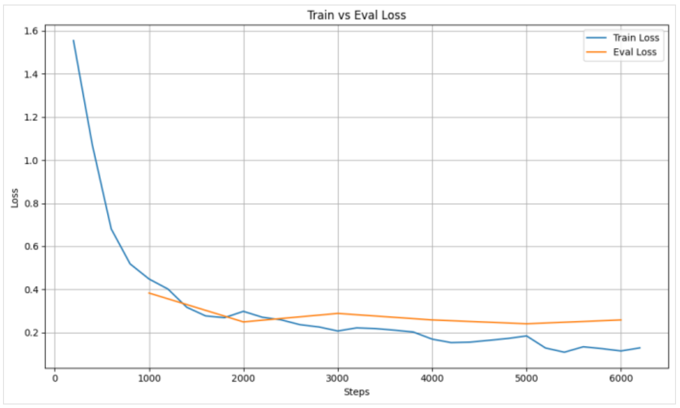


fig24. AlBERTa Train vs. Eval Loss

Train Loss : The train loss decreases sharply initially and continues to decline gradually, indicating effective learning.

Eval Loss : The eval loss stabilizes around a low value, showing good generalization to unseen data.

The gap between train and eval loss is minimal, suggesting that the model is not overfitting.

3. ROC Curve

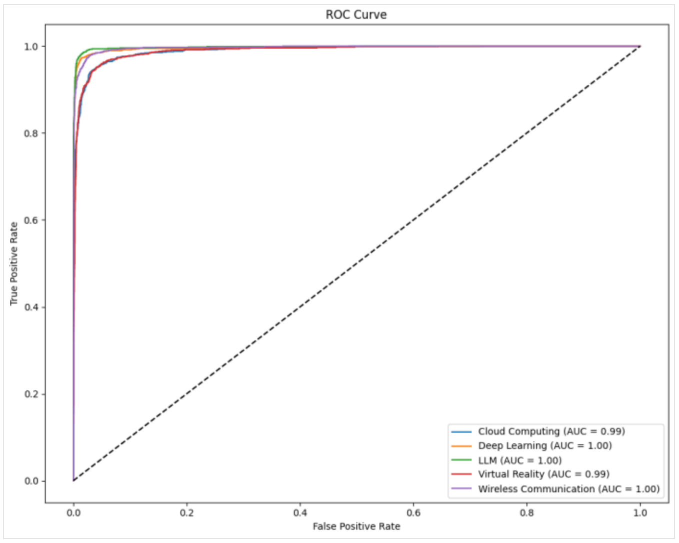


fig25. AlBERTa ROC Curves

The ROC curves for all categories (Cloud Computing, Deep Learning, LLM, Virtual Reality, Wireless Communication) are close to the top-left corner, indicating strong classification performance.

The Area Under the Curve (AUC) values are:

Cloud Computing: AUC = 0.99, Deep Learning: AUC = 1.00, LLM: AUC = 1.00, Virtual Reality: AUC = 0.99, Wireless Communication: AUC = 1.00

These high AUC values suggest excellent discrimination between positive and negative instances.

4. Conclusion

The ALBERT model demonstrates robust performance in classifying IEEE articles into the specified categories. The near-perfect AUC values, stable loss curves, and high accuracy in the confusion matrix collectively indicate that the model is well-trained and highly effective for this classification task.

VI. Dataset, Report, and Code Links

All relevant materials have been uploaded to Google Drive and shared with urhanh@gmail.com. The access links are provided below:

https://drive.google.com/drive/u/0/folders/1OqGiTY1ovuZrdBtFKZ9oRBIulh2Bmyf4

VII. Conclusion

This study presented a comparative analysis of various transformer-based models—including ***DistilBERT, GPT-2, DeBERTa, RoBERTa, and ALBERT***—for multi-class classification of research articles into 5thematic categories **Cloud Computing, Deep Learning, LLM, Virtual Reality, and Wireless Communication.**

The experimental results demonstrated that all models achieved high classification accuracy, with strong AUC scores (mostly above 0.98), stable training dynamics, and minimal overfitting. The confusion matrices revealed low misclassification rates, particularly between semantically distinct categories, while ROC curves showed excellent discriminative power across all classes.

Among the evaluated architectures, models like RoBERTa and DeBERTa offered slightly better performance in terms of precision and generalization, although all models proved effective for the given task. These findings highlight the suitability of pre-trained language models for domain-specific text classification tasks, especially in academic content organization and retrieval systems.

In summary, this work successfully demonstrates the application of state-of-the-art NLP models for automated classification of scientific literature, providing a scalable and accurate solution for categorizing IEEE articles.

VIII. References

[1] Language Models for Web Scraping," *arXiv preprint arXiv:2402.12345*, 2024.