# Progress Report 1: Evaluation of GPU Impact on Fake News Classification

Mary Nathalie Dela Cruz

The research objective is to evaluate the impact of running GPUs for a fake news classification task. Note that the methodology is in the development phase and results will be increased to larger input ranges. For now, model hyperparameters were constant. Batch sizes of 16 and 32 were run to evaluate GPU memory utilization. Increasing the batch size can improve throughput and accelerate training since GPUs can handle data in parallel. The programming language is Python.

## Methodology

1. GPU Utilization Monitoring

The environment was set up in COLAB to monitor GPU utilization. The NVIDIA Management Library (NVML) was used to track GPU memory usage. With this, we can get information on the GPU resource consumption of different stages of model training.

Possible Next Steps:

- Monitor GPU Utilization in every step of the methodology, not only in model training, to check areas of improvement.

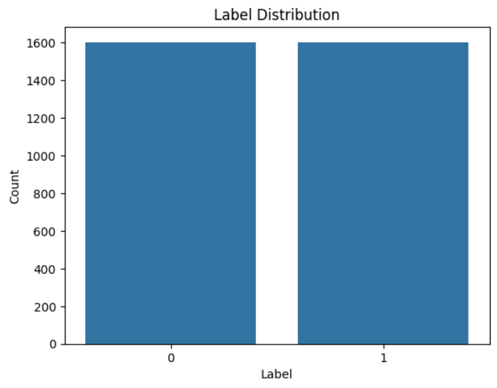
2. Data Preparation

Necessary linguistic datasets and stopwords were downloaded using the Natural Language Toolkit (NLTK) and Advertools library. Stopwords from the Filipino and English languages were collected to create a comprehensive list that contains common but insignificant words.

The dataset used for this study was the 'fake_news_filipino' dataset in HuggingFace. This was loaded and converted into a pandas DataFrame to facilitate easier data manipulation and analysis.

3. Exploratory Data Analysis (EDA)

Label distribution reveals no imbalances that can affect model training.



The typical size of the input is depicted by the mean article length of **182.73**.

Common words in the articles were identified and filtered using the comprehensive stopword list. These common words can affect model performance.

Most common words: **[('si', 5267), ('naman', 2478), ('kay', 2226), ('ayon', 1733), ('lang', 1650), ('duterte', 1605), ('daw', 1571), ('noong', 1541), ('rin', 1279), ('nang', 1249)]**

Possible Next Steps:

- Double-check the languages used in the articles.
- Manually add common words in the stopword list.

4. Preprocessing and Tokenization

The text was preprocessed by converting it to lowercase, removing non-alphanumeric characters, and excluding stopwords. Tokenizers from the Transformers library, specifically BERT and DistilBERT tokenizers, were used. The tokenized dataset was then appropriately padded and truncated to fit the requirement of each model (BERT and DistilBERT).

Possible Next Steps:

- Improve the preprocessing stage by stemming and/or lemmatizing. As far as we have researched, there are no available Python packages for this step suitable for Tagalog.
- Check on other preprocessing steps in NLP.

5. Model Training and Evaluation

The Trainer class from the Transformers library was utilized to train and fine-tune the model. Hyperparameters such as learning rate, batch size, and the number of epochs can be varied. K-fold cross-validation was used to ensure the robustness and reliability of the results.

System metrics, including GPU utilization and memory usage, were logged during training to get real-time insights into resource usage.

Both BERT and DistilBERT models were trained and evaluated. Performance metrics such as accuracy, precision, recall, and F1 score were recorded. The training time for each model on GPU for varying batch sizes was measured.

Possible Next Steps:

- Compare the training time of a model run on GPU and a model run on CPU.
- Explore the impact of learning rate and number of epochs to model performance.
- Compare performance when the dataset is distributed across multiple GPUs using libraries like `torch.nn.DataParallel` or `torch.distributed`
- Compare performance when mixed precision training is implemented to when it is not implemented. Check the performance of GPUs with reduced precision.
- Explore impact of gradient accumulation steps.
- Explore other models like ROBERTA
- Check out TensorParallel (TP), PipelineParallel (PP), Zero Redundancy Optimizer (ZeRO), Sharded DDP and the combinations of these strategies if we have the time.
- Compare performance with other papers.

**Initial Results**

Note:

- DistilBERT is a smaller and faster version of BERT designed to maintain most of the performance of BERT with a reduced model size.
- Learning Rate is set to 1.00E-05. Epochs is set to 2. They will be varied in the final analysis.

- **Blue** highlights best score and **red** highlights worst score.

Training Time:

| model | batch_size | fold | training_time_gpu |
|---|---|---|---|
| bert | 16 | 1 | 182.1875 |
| bert | 16 | 2 | **183.5882** |
| bert | 32 | 1 | 169.4072 |
| bert | 32 | 2 | 169.2093 |
| distilbert | 16 | 1 | 119.2122 |
| distilbert | 16 | 2 | 119.3199 |
| distilbert | 32 | 1 | 111.663 |
| distilbert | 32 | 2 | **111.6337** |

Performance Metrics:

| model | batch_size | fold | accuracy_gpu | f1_gpu | precision_gpu | recall_gpu | loss_gpu |
|---|---|---|---|---|---|---|---|
| bert | 16 | 1 | 0.915783 | 0.917933 | 0.876887 | 0.96301 | 0.386777 |
| bert | 16 | 2 | 0.981285 | 0.981818 | 0.974729 | 0.989011 | 0.060054 |
| bert | 32 | 1 | 0.964442 | 0.964218 | 0.94932 | 0.979592 | 0.157575 |
| bert | 32 | 2 | **0.995009** | **0.995122** | **0.99391** | **0.996337** | **0.01832** |
| distilbert | 16 | 1 | **0.886463** | **0.886957** | **0.864407** | **0.910714** | **0.282322** |
| distilbert | 16 | 2 | 0.936369 | 0.936488 | 0.955527 | 0.918193 | 0.179229 |
| distilbert | 32 | 1 | 0.922645 | 0.924574 | 0.883721 | 0.969388 | 0.200645 |
| distilbert | 32 | 2 | 0.969432 | 0.969846 | 0.977667 | 0.962149 | 0.097109 |

GPU Utilization:

| model | batch_size | fold | gpu_memory_before | gpu_memory_after |
|---|---|---|---|---|
| bert | 16 | 1 | 15.42557 | 20.07956 |
| bert | 16 | 2 | 20.07956 | 20.07956 |
| bert | 32 | 1 | 20.07956 | 20.18375 |
| bert | 32 | 2 | 21.0173 | 20.07956 |
| distilbert | 16 | 1 | 20.07956 | 21.12149 |
| distilbert | 16 | 2 | 21.12149 | 21.34725 |
| distilbert | 32 | 1 | 21.34725 | 24.07365 |
| distilbert | 32 | 2 | 24.07365 | 24.21258 |

| model | batch_size | fold | virtual_memory_before | virtual_memory_after |
|---|---|---|---|---|
| bert | 16 | 1 | 24.9 | 25.3 |
| bert | 16 | 2 | 25.3 | 25.3 |
| bert | 32 | 1 | 25.3 | 25.3 |
| bert | 32 | 2 | 25.3 | 25.1 |
| distilbert | 16 | 1 | 24.6 | 24.8 |
| distilbert | 16 | 2 | 24.9 | 24.7 |
| distilbert | 32 | 1 | 24.7 | 24.7 |
| distilbert | 32 | 2 | 24.7 | 24.7 |

| **model** | **batch_size** | **fold** | **gpu_util_before** | **gpu_util_after** |
|---|---|---|---|---|
| bert | 16 | 1 | 30 | 97 |
| bert | 16 | 2 | 97 | 97 |
| bert | 32 | 1 | 100 | 100 |
| bert | 32 | 2 | 98 | 100 |
| distilbert | 16 | 1 | 70 | 93 |
| distilbert | 16 | 2 | 93 | 93 |
| distilbert | 32 | 1 | 97 | 98 |
| distilbert | 32 | 2 | 100 | 97 |