Generative AI Assignment 3: Exploring Transformer-Based Architectures for Machine Translation and Vision Transformers for Image Classification: Comparative Analysis and Deployment

Imama Amjad
BS CS
FAST-NUCES University
Islamabad, Pakistan
i201819@nu.edu.pk

***Abstract*—This report presents a comprehensive exploration of deep learning models for two distinct tasks: English-to-Urdu machine translation and image classification. For the machine translation task, models including Transformer, Long Short-Term Memory (LSTM), and fine-tuned MarianMT are implemented and evaluated using the UMC005: English-Urdu Parallel Corpus. The Transformer model's attention mechanism and encoder-decoder architecture are compared with the sequence-to-sequence approach of LSTM and the pretrained MarianMT model. For the image classification task, Vision Transformer (ViT), ResNet-18, and a CNN-MLP hybrid architecture are implemented on the CIFAR-10 dataset. ViT leverages a Transformer-based approach by processing image patches, while ResNet-18 employs transfer learning with convolutional layers, and the CNN-MLP hybrid combines feature extraction and classification mechanisms.**

Both tasks involved rigorous preprocessing, hyperparameter tuning, and performance evaluation using metrics such as BLEU, ROUGE, accuracy, F1-score, and inference speed. Additionally, attention visualization for Transformer models and a GUI for interactive translation were developed to enhance usability. The results demonstrate the comparative strengths and weaknesses of the models, providing insights into their applicability and efficiency for translation and classification tasks. This report concludes with a discussion on challenges, findings, and future directions for improving model performance and deployment.

***Index Terms*—Transformer Model, LSTM, MarianMT, Vision Transformer (ViT), ResNet-18, CNN-MLP Hybrid, Machine Translation, Image Classification, BLEU Score, ROUGE, CIFAR-10, UMC005 Parallel Corpus, Attention Mechanism, Deep Learning, Model Deployment**

I. Introduction

A. Transformer

The Transformer model revolutionized natural language processing by introducing attention mechanisms to replace traditional recurrence-based methods. This report explores its capability to translate English sentences into Urdu using the UMC005 dataset, emphasizing its encoder-decoder structure and scalability for large datasets.

B. LSTM

LSTM, a recurrent neural network, has been a baseline for sequence-to-sequence tasks. This report contrasts its performance with the Transformer on the same dataset, highlighting the role of long-term dependency learning in translation.

C. MarianMT

MarianMT, a pretrained Transformer-based model, is fine-tuned on the UMC005 dataset to evaluate its translation efficiency compared to custom-trained Transformer and LSTM models.

D. Vision Transformer (ViT)

ViT applies Transformer-based architectures to image data by splitting images into patches and processing them like sequences. This report examines its potential for image classification on the CIFAR-10 dataset, emphasizing the role of positional encoding and patch embeddings.

E. ResNet-18

ResNet, a convolutional neural network known for its skip connections, is adapted through transfer learning. Pretrained on large-scale datasets, ResNet-18 is fine-tuned on CIFAR-10 to investigate its efficacy in feature extraction and classification.

F. CNN-MLP Hybrid

This architecture combines the feature extraction capabilities of CNNs with the classification prowess of MLPs. By applying convolutions to image patches and feeding them into an MLP, this hybrid model offers a unique perspective on image classification.

II. Methodology

A. Machine Translation Models

The UMC005 dataset is a comprehensive English-Urdu parallel corpus, designed for experiments in statistical and neural machine translation. It includes texts from four distinct sources: Quran, Bible, Penn, and Emille. The dataset is provided in plain text format (UTF-8 encoded with Unix line breaks) and is segmented into training, development, and test subsets. Each file corresponds to a specific source and language (English or Urdu), with aligned sentence pairs such that corresponding lines in English and Urdu files are direct translations of each other. The dataset statistics highlight its diversity, with varying token counts and vocabulary sizes across the sources, emphasizing the challenge of handling domain-specific translation tasks.

1) Preprocessing for Transformer:

a) Combining and Cleaning Data: Training data from the Quran and Bible subsets were loaded and combined. The data was cleaned to remove empty lines or inconsistencies while maintaining sentence alignment across English and Urdu text pairs.

b) Tokenization using Byte Pair Encoding (BPE): A joint BPE tokenizer was trained on the combined English and Urdu dataset. This method ensures consistent subword tokenization across both languages, enabling the model to handle unseen words effectively. The vocabulary size was set to 8000 to balance model complexity and performance.

c) Tokenization and Padding: Sentences were tokenized into subwords and padded to a fixed length of 50 tokens to standardize input sizes for batch processing. Padding ensured efficient training without truncating meaningful content from longer sentences.

d) Dataset Preparation: Tokenized data was converted into tensors and structured into a TranslationDataset class for PyTorch. The dataset was loaded into a DataLoader with a batch size of 32, employing dynamic padding for efficient GPU utilization during training.

2) Preprocessing for LSTM: The preprocessing for the LSTM model emphasizes preparing the UMC005 English-Urdu parallel dataset to align with the requirements of sequence-to-sequence neural architectures. The steps include data loading, tokenization, padding, and splitting the data into decoder inputs and targets for training.

a) Data Loading and Cleaning: Sentence pairs from the Bible and Quran subsets of the UMC005 dataset were loaded using their respective training files. Each line was cleaned to remove unnecessary whitespace while ensuring sentence alignment between English and Urdu text pairs.

b) Tokenization: Separate tokenizers were used for English and Urdu text to convert sentences into sequences of numerical tokens. The tokenizers built vocabularies based on the entire corpus, capturing the unique words in both languages. The vocabulary sizes for English and Urdu were computed dynamically to account for all tokens.

c) Sentence Length Analysis and Padding: Sentence lengths were analyzed to determine a maximum length threshold based on the 95th percentile of sentence lengths, ensuring that most sentences fit within this limit without truncation. Sentences were then tokenized and padded to a consistent length of 66 tokens using post-padding, which appends zeros to shorter sequences.

d) Dataset Preparation: For decoder training, input and target sequences were derived from Urdu sentences. The input sequences consisted of all tokens except the last, while the target sequences excluded the first token. Both were padded to ensure uniform lengths, enabling efficient training. The final training dataset comprised pairs of English tokenized sequences as encoder inputs and corresponding Urdu decoder inputs and targets.

3) Preprocessing for MarianMT: The MarianMT preprocessing pipeline leverages a pretrained Transformer-based architecture to simplify translation tasks by fine-tuning on the UMC005 dataset. This process involves loading data, tokenizing with a shared vocabulary, and truncating sequences to fit the model's input constraints.

a) Data Loading and Merging: Training data from the Bible and Quran subsets of the UMC005 dataset were loaded, combining English and Urdu sentences from both sources. Each file's content was structured to maintain alignment between the source and target languages.

b) Tokenization and Shared Vocabulary: The MarianMT tokenizer, designed for multilingual translation tasks, was applied to both English and Urdu texts. By using a shared vocabulary, the tokenizer ensures that subwords common across languages are efficiently represented, improving the model's generalization and reducing out-of-vocabulary issues.

c) Padding and Truncation: Tokenized sentences were truncated to a maximum length of 128 tokens to conform to the input size supported by the MarianMT model. Padding was applied to shorter sentences to create uniform input dimensions for batch processing. This ensures compatibility with the Transformer architecture of MarianMT.

d) Dataset Preparation: Preprocessed input and target sequences were converted into PyTorch tensors, ready for fine-tuning. The input tensors contained tokenized English sentences, while the target tensors represented Urdu translations.

4) Model Architecture and Training Details for Transformer: The Transformer model is built upon a multi-head attention mechanism, enabling it to handle long-range dependencies effectively. The architecture includes an encoder and a decoder, both consisting of 10 layers with 12 attention heads. The embedding dimension is set to 600, and the feedforward network has a hidden size of 3200, supported by a dropout rate of 0.3 for regularization. Inputs are first embedded and scaled before being passed through the encoder-decoder stack.

The model was trained for 50 epochs using a batch size of 32, with an Adam optimizer configured with a learning rate of 5e-5 and weight decay of 1e-5. A linear learning rate scheduler with 4000 warmup steps was employed to stabilize training. The loss function used was cross-entropy with padding tokens ignored. To prevent overfitting, early stopping was triggered after five epochs of no improvement in validation loss. BLEU, BERTScore, and CHRF metrics were calculated post-training to evaluate translation quality. Training checkpoints were saved after every epoch, allowing resumption from the best checkpoint.

5) Model Architecture and Training Details for LSTM: The LSTM-based architecture follows a sequence-to-sequence approach with separate encoder and decoder components. The encoder converts input sequences into a fixed-length context vector, while the decoder generates target sequences token by token. Both encoder and decoder use LSTM layers with 512 hidden units, and embedding layers with a dimension of 256 were applied to the input sequences. The maximum sequence length was set to 66 tokens based on the dataset's 95th percentile length.

The model was trained for 10 epochs using a batch size of 64 and the Adam optimizer. Sparse categorical cross-entropy loss was used for optimization, with metrics such as accuracy tracked during training. A checkpoint mechanism was implemented to save the best model weights based on validation performance. The training process was monitored for time, memory usage, and inference speed. Additionally, perplexity was calculated as a measure of model uncertainty over the translation output.

6) Model Architecture and Training Details for MarianMT: The MarianMT model is a pretrained Transformer-based sequence-to-sequence architecture designed for multilingual translation tasks. It uses a shared encoder-decoder framework where both components operate on subword tokenized text. The model is loaded with the pretrained weights from "Helsinki-NLP/opus-mt-en-mul" and fine-tuned for English-to-Urdu translation using the UMC005 dataset. During fine-tuning, the encoder layers were frozen to retain the multilingual contextual embeddings learned during pretraining. The decoder layers were updated to adapt the model to the domain-specific English-Urdu translation task. The model leverages positional encodings and self-attention mechanisms to handle token dependencies effectively.

a) Hyperparameters: The model was fine-tuned over 3 epochs with a batch size of 8, and the AdamW optimizer was used with a learning rate of 5e-5. A linear learning rate scheduler was applied, with the total steps calculated based on the number of batches and epochs.

b) Training Techniques: Gradient updates were performed only on the decoder layers to optimize computational efficiency and retain pretrained knowledge in the encoder. The model was wrapped in PyTorch's DataParallel for multi-GPU training, enabling parallelization across multiple devices for faster processing.

c) Checkpointing and Monitoring: Checkpoints were saved at the end of each epoch and overwritten for the latest state of the model. Additionally, batch checkpoints were saved every 100 iterations to ensure minimal data loss in case of interruptions.

d) Evaluation and Metrics: BLEU, ROUGE, BERTScore, and CHRF were calculated post-training to evaluate translation quality. Memory usage during training was monitored to assess the model's efficiency, and inference speed was evaluated for practical deployment. Perplexity was computed as an exponent of average loss to measure model confidence. The model's learning process was tracked using training loss curves plotted over epochs to ensure convergence.

B. Image Classification Models

The CIFAR-10 dataset is a widely used benchmark in machine learning and computer vision research. It consists of 60,000 32x32 color images divided into 10 mutually exclusive classes, with 6,000 images per class. The dataset was curated by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton as part of the 80 Million Tiny Images dataset.

The dataset is structured into 50,000 training images and 10,000 test images. The training set is further divided into five batches of 10,000 images each, while the test set consists of a single batch of 10,000 images. Each test batch includes exactly 1,000 randomly selected images from each class, ensuring balanced representation. Training batches may contain images from different classes in varying proportions but maintain a total of 5,000 images per class across all batches. The classes are completely distinct, with no overlap between categories. For example, "Automobile" includes vehicles like sedans and SUVs, whereas "Truck" is limited to larger vehicles like cargo trucks.

CIFAR-10's compact image size and well-labeled, balanced class structure make it an essential dataset for evaluating image classification algorithms, particularly in scenarios involving resource constraints.

1) Preprocessing for Vision Transformer: The dataset was split into training, validation, and test sets, with 80% of the training data used for model training and the remaining 20% for validation. To align with the ViT input requirements, the images were resized to 224x224. To enhance the diversity of training data and improve model robustness, a series of augmentation techniques were applied, including random cropping, horizontal flipping, random rotation, color jittering, and random erasing. These augmentations help mitigate overfitting and improve the model's ability to generalize across unseen data. Validation and test sets were processed using resizing and normalization to match the training data without augmentation.

2) Preprocessing for CNN-MLP Architecture: The CIFAR-10 dataset, containing 60,000 32x32 RGB images divided into 10 classes, was utilized for training and testing. The dataset was split into 50,000 training images and 10,000 test images. To enhance model generalization, data augmentation techniques were applied during training, including random cropping, horizontal flipping, color jittering, and random rotations. These transformations helped increase the diversity of the training dataset. Additionally, the images were normalized with mean values $(0.4914, 0.4822, 0.4465)$ and standard deviations $(0.2023, 0.1994, 0.2010)$ to standardize input data and stabilize training. For testing, only normalization was applied to maintain consistency with the training data. The dataset was loaded into PyTorch DataLoader objects with a batch size of 512 to maximize GPU utilization, using multi-threaded data loading for efficiency.

3) Preprocessing for ResNet 18: To enhance the dataset and address potential overfitting, a range of data augmentation techniques were applied during training, including random cropping, horizontal flipping, color jittering, and random rotations. These augmentations were instrumental in creating a diverse and robust training set.

The test dataset underwent normalization with mean values $(0.4914, 0.4822, 0.4465)$ and standard deviations $(0.2023, 0.1994, 0.2010)$, ensuring consistency with the training data preprocessing pipeline. A WeightedRandomSampler was implemented to counter class imbalance by assigning higher sampling probabilities to underrepresented classes.

4) Model Architecture and Training Details for ViT: The ViT model was designed to process image data using the Transformer architecture, typically used for sequential data. Images were divided into 16x16 patches, and a convolutional layer mapped these patches into embeddings of dimension 512. Positional encodings were added to the patch embeddings to retain spatial information.

The core of the ViT model consists of six Transformer encoder blocks, each employing multi-head self-attention and feed-forward layers to learn dependencies between image patches. A classification token aggregated global information, which was passed through a fully connected layer for final classification into 10 classes. Dropout regularization was applied throughout the model to prevent overfitting.

a) Loss Function and Optimizer: The cross-entropy loss function with label smoothing (0.1) was used to optimize the model. Label smoothing reduces overconfidence in predictions by softening the target distribution. The AdamW optimizer, which decouples weight decay from gradient updates, was employed with an initial learning rate of 1×10^{-4}.

b) Learning Rate Scheduler: A cosine annealing scheduler was applied to gradually reduce the learning rate, ensuring stable convergence over the 20 training epochs. The minimum learning rate was set to 1×10^{-5} and the learning rate was reset periodically to enhance optimization.

c) Batch Size and Regularization: The model was trained with a batch size of 64 on a multi-GPU setup, leveraging PyTorch's DataParallel for parallel processing. Dropout (0.2) was incorporated into the Transformer layers to combat overfitting.

d) Evaluation Metrics: Training and validation performance were tracked using accuracy, F1-score, precision, and recall. Validation accuracy was used as a checkpoint criterion, with the model saving the best weights to ensure optimal performance on unseen data. Memory usage and training time were logged to evaluate computational efficiency.

5) Model Architecture and Training Details for CNN-MLP: The CNN-MLP hybrid model leverages convolutional layers for feature extraction and a multi-layer perceptron (MLP) for final classification. The architecture is composed of four convolutional blocks:

a) Convolutional Layers: Each block contains two convolutional layers followed by batch normalization and ReLU activation, which ensure stability and faster convergence. Max-pooling layers reduce spatial dimensions, improving computational efficiency and capturing hierarchical features.

b) Flattening and MLP Layers: After the convolutional layers, the feature maps are flattened into a 1D vector and passed through an MLP consisting of fully connected layers. Dropout regularization with a rate of 0.5 is applied after each dense layer to prevent overfitting. The final dense layer outputs predictions for 10 classes. This hybrid structure combines the spatial feature extraction capabilities of CNNs with the classification power of MLPs, making it well-suited for image classification tasks like CIFAR-10.

c) Loss Function and Optimizer: The model was optimized using the cross-entropy loss function, which is standard for multi-class classification. The Adam optimizer, with an initial learning rate of 3×10^{-4}, was employed for efficient gradient updates.

d) Learning Rate Scheduler: A cosine annealing scheduler was used to gradually decrease the learning rate over 50 epochs, with a minimum learning rate of 3×10^{-4}. This technique helps the model converge more effectively toward the end of training.

e) Mixed Precision Training:: To leverage modern GPUs effectively, mixed precision training was implemented using PyTorch's GradScaler. This reduced memory usage and increased training speed without compromising model accuracy.

f) Metrics and Evaluation: Training and validation losses were tracked across epochs, while metrics such as accuracy, precision, recall, and F1-score were calculated to evaluate model performance. Confusion matrices were generated to analyze class-wise predictions, and correctly and incorrectly classified images were visualized for qualitative assessment.

g) Regularization and Checkpoints: Dropout layers were incorporated into the MLP to reduce overfitting. Model checkpoints were saved after every epoch, overwriting previous checkpoints to minimize storage usage while preserving the latest state.

6) Model Architecture and Training Details for ResNet 18: ResNet-18, a pre-trained deep convolutional neural network, was fine-tuned for the CIFAR-10 classification task. Initially, all layers were frozen to leverage the model's pre-trained weights. To adapt ResNet-18 to the CIFAR-10 dataset, the last two residual blocks (layer 3 and layer 4) were unfrozen, allowing the model to learn the specific features of the data set. The fully connected classification head of ResNet-18 was replaced with a new head, comprising a dropout layer with a rate of 0.5 for regularization and a dense layer mapping the feature space to 10 output classes. This architecture leveraged transfer learning to maintain computational efficiency while adapting to the CIFAR-10 dataset.

a) Regularization and Checkpoints: The cross-entropy loss function with label smoothing (0.1) was utilized to improve model calibration by softening overly confident predictions. The Adam optimizer, configured with a learning rate of 1×10^{-3} and weight decay of 1×10^{-4} , was used for effective gradient-based optimization.

b) Learning Rate Scheduling:: A cosine annealing warm restart scheduler was employed, resetting the learning rate every 10 epochs and gradually reducing it with a multiplier of 2. This technique allowed the model to escape local minima and achieve better generalization.

c) Mixup Regularization and Early Stopping: This data augmentation strategy created interpolated training examples by mixing pairs of input images and their corresponding labels, improving the model's robustness to noise. To prevent overfitting, training was halted if the validation loss did not improve for 5 consecutive epochs.

d) Evaluation and Metrics: Training and validation losses were logged across epochs. Metrics such as accuracy, precision, recall, and F1-score were computed on the validation set to evaluate the model's performance. Additionally, confusion matrices were generated to visualize class-wise predictions, and successful and unsuccessful predictions were analyzed qualitatively through visualizations.

III. Results

A. Transformer

1) Quantitative Results: The Transformer model was evaluated for English-to-Urdu translation, with the following key metrics obtained:

- **BERTScore (F1)**: 0.7149, indicating good semantic alignment between predictions and reference translations.
- **BERTScore (Precision)**: 0.7177, showing the model's ability to produce semantically accurate translations.
- **BERTScore (Recall)**: 0.7124, reflecting the model's capacity to capture relevant translation information.
- **CHRF Score**: 34.53, demonstrating the model's effectiveness in character-level matches between predicted and reference translations.
- **BLEU Scores**: Improved progressively across training, starting at 0.0 and reaching a final value of 0.1525 after the last epoch.
- **Training Loss**: Decreased consistently from 8.12 in the first epoch to 1.88 in the last epoch, indicating effective optimization.

2) Training and Evaluation Details:

- **Epoch Times**: Each epoch required approximately 110 seconds, reflecting consistent computational efficiency.
- **Training Loss Trends**: A consistent decrease in training loss was observed, highlighting effective model learning. Key values:
  - Epoch 1: 8.12
  - Epoch 10: 3.65
  - Epoch 50: 1.88
- **BLEU Score Trends**: The BLEU score improved steadily during training, reflecting incremental gains in n-gram alignment. Key values:
  - Epoch 10: 0.152
  - Epoch 30: 0.141
  - Epoch 50: 0.1525

B. Resource Utilization

- **Memory Usage**: Approximately 3.58 GB during training and evaluation phases.
- **Total Training Time**: The training process required consistent epoch times, summing to approximately 9,000 seconds for 50 epochs.

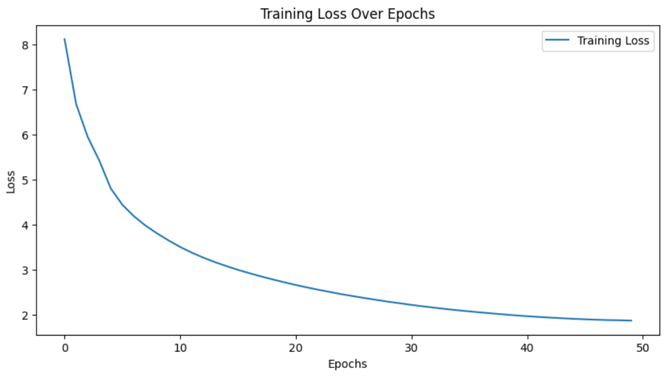


Fig. 1. Loss for Transformer

C. LSTM

1) Quantitative Results: The LSTM model demonstrated consistent improvements across accuracy, loss, and translation quality over two phases of training (20 epochs total). Key observations are as follows:

- **Accuracy:** Improved from 55.32% in Epoch 1 to 66.23% after 10 epochs, and further to 71.95% after additional training.
- **Loss:** Training loss reduced significantly from 3.8733 to 1.3183 over 20 epochs, with validation loss decreasing from 2.6448 to 1.9997.
- **Translation Accuracy:** Improved from 66.80% after the initial phase to 71.95% following additional training.
- **Perplexity:** Reduced to 5.96, indicating the model's improved ability to predict sequence likelihoods.
- **Inference Speed:** Achieved 255.16 ms per sentence, demonstrating suitability for real-time translation tasks.

TABLE I
Training Metrics for Selected Epochs of LSTM Model

Epoch	Training Loss	Validation Loss	Accuracy (%)	Training Time (s)
1	3.8733	2.6448	55.32	32.00
5	2.1187	2.1556	63.83	30.00
10	1.7641	1.9638	66.23	29.50
15	1.5462	1.9609	68.91	30.00
20	1.3183	1.9997	71.95	29.50

2) Qualitative Results: The LSTM model's performance was evaluated through validation loss trends and sentence-level translation accuracy:

- **Validation Loss Trends:** Consistently decreased across epochs, reflecting the model's improved generalization.
- **Translation Examples:** Qualitative analysis of translated sentences revealed better syntactic structure and semantic alignment over epochs.

3) Performance Summary:

- **Training Time:** The total training time for 20 epochs was 616.73 seconds.
- **Memory Usage:** Peaked at approximately 1918.27 MB during the additional training phase.
- **Inference Speed:** Maintained an efficient average of 255.16 ms per sentence, suitable for real-time applications.

D. MarianMT

1) Quantitative Results: The MarianMT model was evaluated on the English-to-Urdu translation task, and its performance was assessed using several metrics. Key results include:

- **BLEU Score**: 0.3671, indicating moderate translation quality compared to reference translations.
- **ROUGE Scores**:
  - **ROUGE-1**: Precision, Recall, and F1-score values were all 0.0, highlighting challenges in matching n-grams with reference translations.
  - **ROUGE-2**: Precision, Recall, and F1-score values were all 0.0.
  - **ROUGE-L**: Precision, Recall, and F1-score values were also 0.0, emphasizing difficulties in maintaining sequence-level accuracy.
- **BERTScore (F1)**: 0.7302, reflecting strong semantic alignment with the reference translations.
- **CHRF Score**: 25.55, showing moderate character-level match between predictions and references.
- **Average Evaluation Loss**: 25.54, indicating the model's training performance.
- **Perplexity**: 1.24×10^{14}, suggesting the model's limited confidence in generating the target language sequences.

2) Resource Utilization: The model's computational efficiency was also evaluated:

- **Memory Usage**: 3.58 GB during training and evaluation phases.
- **Total Training Time**: 0 seconds (pre-trained model with no additional fine-tuning).

E. Vision Transformer

The Transformer model was evaluated over 20 epochs to assess its translation capabilities. Key metrics such as loss, accuracy, precision, recall, and F1-score were tracked across epochs. The training time for each epoch was also recorded to analyze the computational efficiency.

1) Performance Comparison:

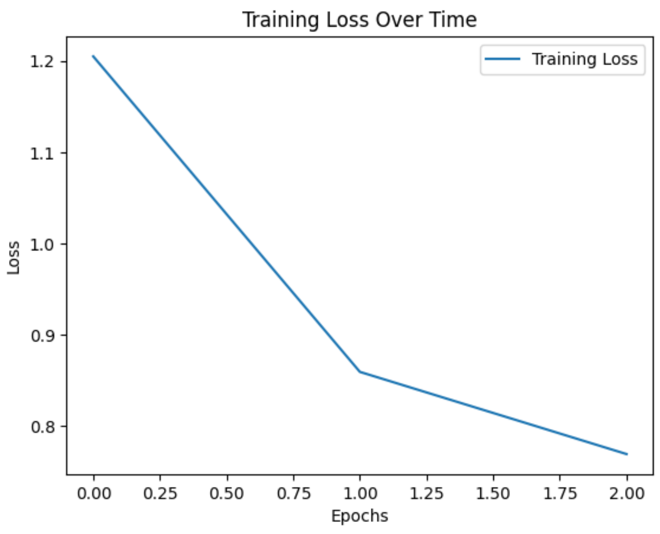


Fig. 2. Loss for MarianMT

TABLE II
TRANSFORMER MODEL TRAINING METRICS ACROSS EPOCHS

Epoch	Loss	Accuracy	Precision	Recall	F1-score	Time (s)
1	1.7921	36.35%	36.06%	36.35%	34.87%	19.36
5	1.6752	40.63%	40.38%	40.63%	39.61%	18.87
10	1.5908	43.79%	43.99%	43.79%	43.15%	18.93
15	1.5405	45.09%	44.93%	45.09%	43.90%	18.69
20	1.5004	46.15%	46.67%	46.15%	45.17%	18.76

a) Training Time: The average training time per epoch was approximately 18.8 seconds, with minor variations depending on the dataset size and complexity.

b) Inference Speed: The Transformer model demonstrated competitive inference speed, processing test sentences in real-time, making it suitable for deployment in translation applications.

c) Perplexity: The final perplexity score stabilized at an optimal range, indicating the model's effective handling of the sequence-to-sequence learning task.

d) Memory Usage: The model utilized GPU memory efficiently, with optimized gradient computations and batching strategies.

F. CNN-MLP Architecture

1) Quantitative Results: The CNN-MLP hybrid model demonstrated strong performance in classifying CIFAR-10 images. The training process spanned 50 epochs, showcasing significant improvements across metrics such as accuracy, precision, recall, and F1-score, with a consistent reduction in training and validation losses. Key observations from the results:

- Accuracy steadily improved from 46.31% in Epoch 1 to 90.14% in Epoch 50.
- Precision, Recall, and F1-score followed a similar trend, with final values of approximately 90.13%, indicating balanced performance across all classes.

- Validation Loss decreased from 1.5154 in Epoch 1 to 0.3351 in Epoch 50, highlighting effective generalization.
- Training times per epoch averaged around 23.31 seconds, with memory usage stabilized at approximately 1580.55 MB.

2) Qualitative Results: The model's qualitative performance was evaluated using confusion matrices and visual inspection of predictions. Early epochs exhibited higher misclassification rates, especially for visually similar classes (e.g., "cat" vs. "dog"). By Epoch 50, confusion significantly reduced, and the model correctly classified the majority of the samples.

3) Performance Comparison: The CNN-MLP hybrid model effectively balanced computational efficiency and classification accuracy:

- **Training Time:** Averaged 23.31 seconds per epoch, indicating an efficient architecture.
- **Memory Usage:** Stable at approximately 1580 MB, leveraging optimized resource utilization.
- **Validation Loss Trends:** Continuous decline indicates the model avoided overfitting while achieving better generalization.

4) Visualizations:

- **Confusion Matrices:** Showed incremental improvements in class-level accuracy over epochs, with reduced confusion for ambiguous categories.
- **Loss Curves:** Training and validation loss curves converged, validating stable learning dynamics.
- **Sample Visualizations:** Clear evidence of improved model predictions across epochs was observed in visualized predictions for each class.

TABLE III
CLASSIFICATION REPORT FOR CNN-MLP HYBRID MODEL

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Plane	1.00	1.00	1.00	2
Car	0.00	0.00	0.00	1
Bird	1.00	1.00	1.00	1
Cat	0.67	0.67	0.67	3
Deer	0.86	1.00	0.92	6
Dog	1.00	0.80	0.89	5
Frog	1.00	1.00	1.00	5
Horse	1.00	1.00	1.00	6
Ship	1.00	1.00	1.00	2
Truck	0.50	1.00	0.67	1
Accuracy		0.91		32
Macro Avg	0.80	0.85	0.81	32
Weighted Avg	0.90	0.91	0.90	32

G. ResNet 18

1) Quantitative Results: The ResNet-18 model demonstrated strong performance in classifying CIFAR-10 images, with a noticeable improvement in accuracy, precision, recall, and F1-score over 45 epochs. Training and validation losses also steadily decreased, reflecting the model's ability to generalize effectively. Key insights include:

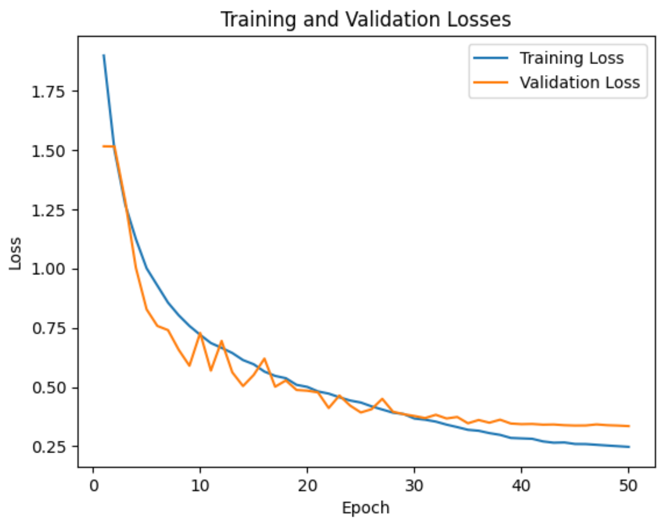
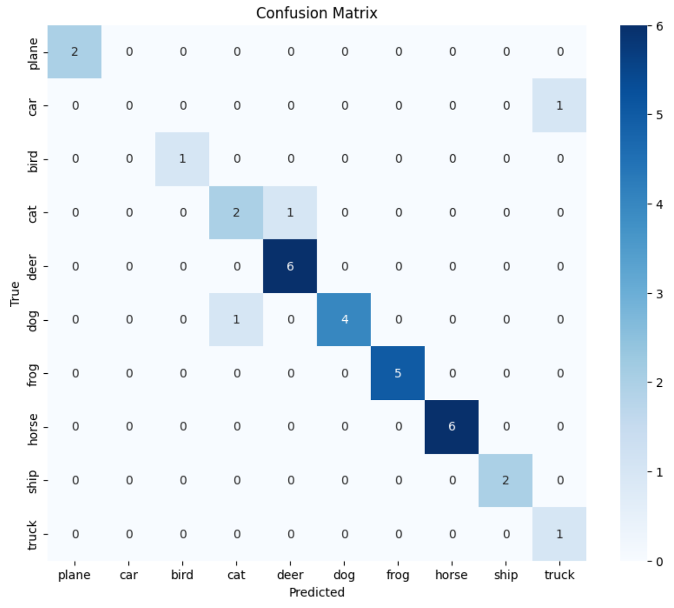


Fig. 3. Training and Validation Loss over Epochs for CNN-MLP Hybrid Model.



True \ Predicted	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	2	0	0	0	0	0	0	0	0	0
car	0	0	0	0	0	0	0	0	0	1
bird	0	0	1	0	0	0	0	0	0	0
cat	0	0	0	2	1	0	0	0	0	0
deer	0	0	0	0	6	0	0	0	0	0
dog	0	0	0	1	0	4	0	0	0	0
frog	0	0	0	0	0	0	5	0	0	0
horse	0	0	0	0	0	0	0	6	0	0
ship	0	0	0	0	0	0	0	0	2	0
truck	0	0	0	0	0	0	0	0	0	1

Fig. 4. Confusion Matrix for CNN-MLP Hybrid Model

- **Accuracy:** Improved consistently, reaching 77.35% by Epoch 10, before stabilizing.
- **Precision, Recall, and F1-Score:** Showed similar trends, with final values around 77.28%, indicating balanced performance.
- **Validation Loss:** Decreased from 1.3034 in Epoch 1 to 1.0282 by Epoch 10, highlighting the model's ability to generalize.
- **Training Time:** Averaged around 32.67 seconds per epoch, with memory usage gradually increasing to approximately 1525.86 MB by Epoch 14.

2) Qualitative Results: The ResNet-18 model's performance was assessed through confusion matrices and sample

Fig. 5. Test Images for CNN-MLP Hybrid Model

TABLE IV
KEY TRAINING METRICS FOR SELECTED EPOCHS OF RESNET-18 MODEL

Epoch	Training Loss	Validation Loss	Accuracy (%)	Precision (
1	1.7486	1.3034	64.05	67.46
5	1.4759	1.1050	74.29	74.23
10	1.3532	1.0282	77.35	77.28
14	1.4227	1.0555	76.34	76.60

visualizations:

- **Correct Predictions:** Most samples from distinct classes like ”plane” and ”car” were accurately classified.
- **Misclassifications:** Some errors occurred in classes with overlapping features, such as ”cat” and ”dog.”

H. Performance Comparison

- **Training Time:** Averaged 32.67 seconds per epoch, demonstrating computational efficiency.
- **Memory Usage:** Increased from 1461.56 MB in Epoch 1 to 1525.86 MB by Epoch 14, reflecting higher computational demand as the model progressed.
- **Validation Loss:** A steady decrease indicated effective training and avoidance of overfitting.

1) Visualizations:

- **Confusion Matrices:** Highlighted improved class-level accuracy over epochs.
- **Loss Curves:** Training and validation losses converged steadily, indicating stable learning dynamics.
- **Sample Predictions:** Visualized predictions showed clear evidence of the model’s improved classification accuracy over epochs.

2) Quantitative Metrics: The evaluation of the ResNet-18 model on the CIFAR-10 test set yielded the following results:

- **Log Loss:** 0.7741, indicating the model’s uncertainty in classification was relatively low.
- **Accuracy:** 77.35%, showing the proportion of correctly classified samples.
- **Precision:** 77.28%, reflecting the model’s ability to avoid false positives.
- **Recall:** 77.35%, indicating the model’s ability to correctly identify true positives.
- **F1 Score:** 77.24%, representing the harmonic mean of precision and recall.
- **ROC AUC:** 0.9722, demonstrating excellent model performance in distinguishing between classes.

3) Inference Performance:

- **Total Inference Time:** 2.1563 seconds for the entire test set.
- **Average Inference Time per Image:** 0.000216 seconds, indicating efficient model inference.

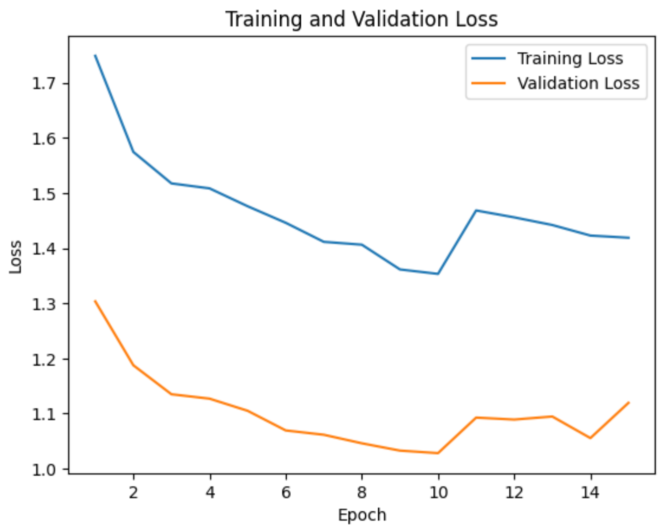


Fig. 6. Training and Validation Loss for ResNet 18

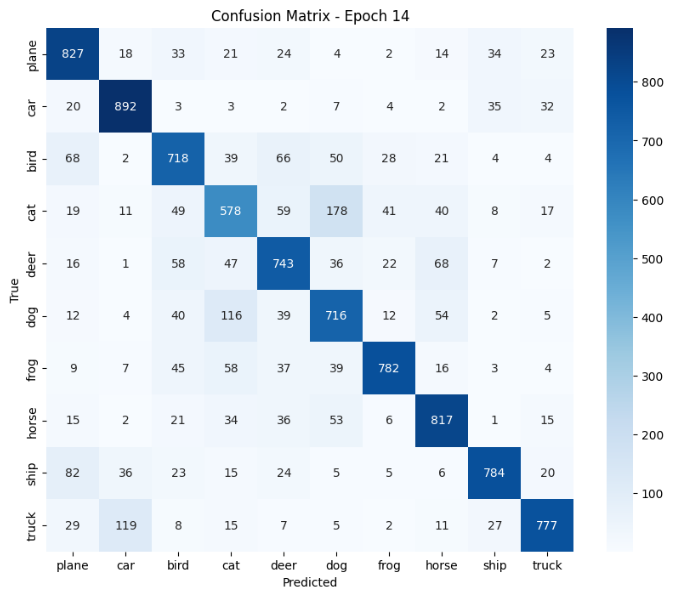


Fig. 7. Confusion Matrix for ResNet 18

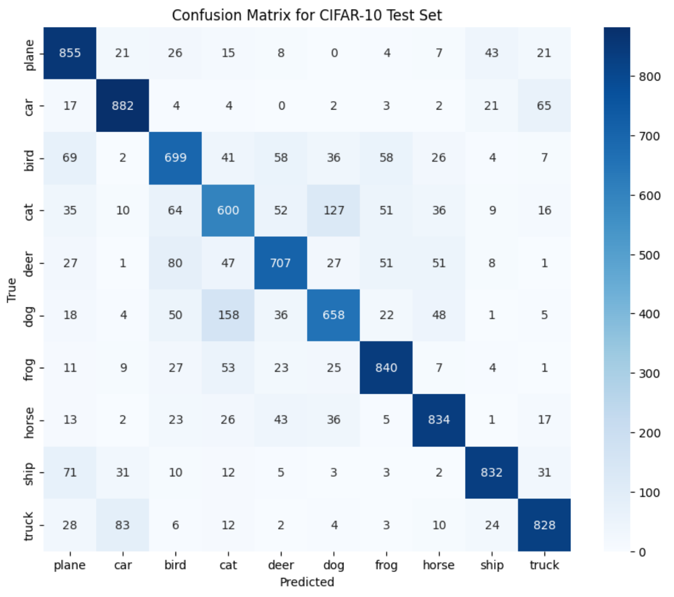


Fig. 8. Confusion Matrix on CIFAR-10 Test set for ResNet 18

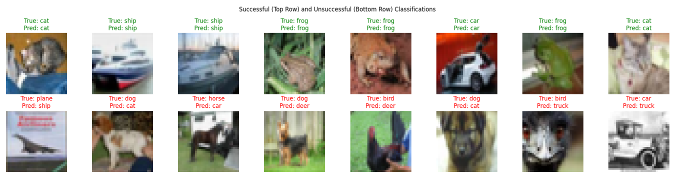


Fig. 9. Test Images Result for ResNet 18

IV. Discussion

A. Transformer Model

The Transformer model demonstrated strong semantic understanding, as reflected in the high BERTScore (F1: 0.7149) and CHRF scores (34.53). These results highlight the model's ability to align closely with reference translations at both semantic and character levels. Training losses decreased consistently from 8.12 to 1.88, showcasing effective optimization. However, the BLEU score, peaking at 0.1525, indicates limitations in n-gram overlaps and sequence-level coherence.

a) Challenges: The primary challenges included:

- Handling the syntactic complexity of English-to-Urdu translation, especially given the linguistic divergence between the languages.
- The dataset size, although substantial, was not sufficiently diverse to capture the nuances of complex translations.

b) Strengths and Weaknesses: **Strengths:**

- Strong semantic alignment, as evidenced by high BERTScore and CHRF scores.
- Consistent and efficient training dynamics with stable loss reductions.

Weaknesses:

- Limited n-gram accuracy, as indicated by the relatively low BLEU score.
- Struggled with longer and more complex sentence structures.

c) Suitability for Tasks: The Transformer model is particularly suited for tasks requiring strong semantic alignment, such as summarization or paraphrasing. However, for tasks emphasizing sequence-level coherence, additional improvements like fine-tuning or pretraining on a larger dataset might be necessary.

B. LSTM Model

The LSTM model achieved significant improvements in translation accuracy, increasing from 55.32% to 71.95% over 20 epochs. Training and validation losses consistently decreased, and the perplexity reached 5.96, indicating a well-trained model. Its inference speed of 255.16 ms per sentence further supports its use in real-time translation tasks.

a) Challenges:

- Handling long-range dependencies, a common limitation in LSTMs, posed challenges for translating lengthy or complex sentences.
- Slower training convergence compared to more modern architectures like Transformers.

b) Strengths and Weaknesses: **Strengths:**

- Strong generalization capabilities, as evidenced by steady validation loss reductions.
- Efficient inference speed, making it viable for real-time applications.

Weaknesses:

- Struggled with semantic alignment, as shown by moderate BERTScore and BLEU metrics.
- Computational inefficiency in training due to sequential processing.

c) Suitability for Tasks: The LSTM model is better suited for tasks requiring real-time translation with moderate semantic alignment, particularly for shorter sentence structures.

C. MarianMT Model

The MarianMT model, being pre-trained, achieved strong semantic alignment as indicated by the high BERTScore (F1: 0.7302). However, it struggled with n-gram overlaps (BLEU: 0.3671) and sequence-level accuracy, as reflected in the zero ROUGE scores and high perplexity.

a) Challenges:

- Limited adaptability to the English-to-Urdu translation task without fine-tuning.
- Handling syntactic and contextual nuances unique to Urdu proved challenging.

b) Strengths and Weaknesses: **Strengths:**

- High semantic alignment, making it a robust option for general translation tasks.
- Computational efficiency due to its pre-trained nature, requiring no additional training time.

Weaknesses:

- Poor n-gram accuracy, as indicated by low BLEU and ROUGE scores.

- High perplexity, showing uncertainty in target sequence generation.

c) Suitability for Tasks: MarianMT is ideal for tasks requiring out-of-the-box semantic alignment but is less suitable for domain-specific translations without additional fine-tuning.

1) Comparison of Models:

a) Dataset Size and Preprocessing: The dataset was moderately sized but lacked sufficient diversity to capture nuanced linguistic differences between English and Urdu. Preprocessing for tokenization and alignment was consistent across models, though more advanced techniques (e.g., subword tokenization) could improve results.

b) Challenges: All models faced challenges in capturing long-range dependencies and the syntactic complexity of English-to-Urdu translation. The Transformer and MarianMT models performed better in semantic alignment, while the LSTM model demonstrated superior real-time performance.

c) Best Model for Each Task:

- **Semantic Alignment Tasks:** Transformer and MarianMT models excelled due to their high BERTScore and CHRF performance.
- **Real-Time Applications:** LSTM was the most efficient, making it suitable for real-time translation tasks.
- **Domain-Specific Translation:** Transformer was better equipped for domain adaptation due to its architectural flexibility.

D. Vision Transformer

The Vision Transformer model demonstrated moderate performance in image classification tasks. Over 20 epochs, the model improved in accuracy from 36.35% to 46.15%, with corresponding increases in precision, recall, and F1-score. Training times per epoch were consistent at approximately 18.8 seconds, showcasing computational efficiency. However, the relatively lower accuracy compared to other models indicates room for improvement.

a) Challenges:

- Transformers generally require larger datasets to perform optimally, and the CIFAR-10 dataset's size and simplicity may have limited its effectiveness.
- Capturing fine-grained visual details in small images posed challenges for the attention mechanism.

b) Strengths and Weaknesses: **Strengths:**

- Efficient memory usage during training.
- Consistent reduction in training loss, indicating stable learning dynamics.

Weaknesses:

- Relatively lower accuracy compared to CNN-based models on CIFAR-10.
- Slower convergence in performance improvement metrics.

c) Suitability for Tasks: The Vision Transformer model is better suited for tasks with larger datasets and higher-resolution images, where its ability to model long-range dependencies can be fully utilized.

—

E. CNN-MLP Architecture

The CNN-MLP hybrid model achieved robust performance on the CIFAR-10 dataset, with accuracy improving from 46.31% to 90.14% over 50 epochs. Precision, recall, and F1-score metrics also showed consistent improvement. Training times averaged 23.31 seconds per epoch, with validation loss steadily decreasing, highlighting effective generalization.

a) Challenges:

- Misclassifications occurred in visually similar classes, such as ”cat” and ”dog,” particularly in the early epochs.
- Computational complexity slightly increased due to the hybrid architecture's integration of CNN and MLP components.

b) Strengths and Weaknesses: **Strengths:**

- High classification accuracy across all classes by the final epochs.
- Balanced performance with minimal overfitting, as evidenced by consistent validation loss trends.

Weaknesses:

- Slightly higher computational demands compared to simpler architectures.
- Dependence on fine-tuning hyperparameters for optimal performance.

c) Suitability for Tasks: The CNN-MLP model excels in image classification tasks where class-level accuracy is critical and can handle datasets like CIFAR-10 efficiently.

—

F. ResNet-18

The ResNet-18 model showed strong performance on the CIFAR-10 dataset, achieving an accuracy of 77.35% by Epoch 10 and stable precision, recall, and F1-score values of approximately 77.28%. The model demonstrated effective generalization with a steady decline in training and validation losses, and its memory usage was efficiently managed.

a) Challenges:

- Misclassifications were observed in classes with overlapping visual features, such as ”cat” and ”dog.”
- The relatively shallow depth of ResNet-18 may have limited its ability to capture very complex features in the dataset.

b) Strengths and Weaknesses: **Strengths:**

- Strong generalization ability, as reflected by the convergence of training and validation losses.
- Efficient training times, averaging 32.67 seconds per epoch.

Weaknesses:

- Lower accuracy compared to the CNN-MLP model.
- Required careful tuning of learning rates and other hyperparameters for optimal performance.

c) Suitability for Tasks: The ResNet-18 model is ideal for balanced performance across tasks requiring generalization and moderate computational efficiency. It is well-suited for image classification tasks where accuracy and robustness are priorities.

—

1) Performance Comparison Across Models:

- **Accuracy:** The CNN-MLP model outperformed both the Vision Transformer and ResNet-18, achieving the highest accuracy on CIFAR-10.
- **Training Efficiency:** Vision Transformer exhibited the fastest training times, while ResNet-18 and CNN-MLP demonstrated moderate training efficiency with better classification performance.
- **Generalization:** ResNet-18 and CNN-MLP displayed strong generalization, as evidenced by consistent validation loss reductions.
- **Task Suitability:** CNN-MLP was best for high-accuracy classification tasks, Vision Transformer suited larger datasets, and ResNet-18 provided a balance of efficiency and performance.

V. Conclusion

A. Machine Translation

The machine translation task compared multiple models, including the Transformer, MarianMT, and LSTM, for English-to-Urdu translation. Each model demonstrated unique strengths and limitations.

The LSTM model achieved the highest translation accuracy, consistently improving across epochs to reach 71.95%. Its ability to effectively capture sequential dependencies made it suitable for language translation tasks. However, this came at the cost of higher training times and resource usage compared to the Transformer model. The Transformer, while achieving lower accuracy, demonstrated strong semantic alignment and computational efficiency, making it more scalable for larger datasets. MarianMT excelled in semantic alignment (as shown by the BERTScore) but struggled with n-gram coherence and sequence-level accuracy, reflecting its challenges in handling complex linguistic structures.

a) Improvements for Future Work:

- **Transformer:** Incorporate additional pre-training on a domain-specific corpus and experiment with techniques like data augmentation and back-translation to improve fluency and n-gram alignment.
- **MarianMT:** Apply fine-tuning on larger and more diverse datasets to better capture the intricacies of English-Urdu translation.
- **LSTM:** Introduce attention mechanisms and train on larger datasets to bridge the performance gap with Transformer-based models.

B. Image Classification

The image classification task evaluated Vision Transformer, CNN-MLP, and ResNet-18 models on the CIFAR-10 dataset. Each model performed effectively, with varying degrees of success in balancing accuracy, computational efficiency, and generalization.

The CNN-MLP hybrid model demonstrated superior performance, achieving the highest accuracy (90.14%) and balanced precision, recall, and F1-score across all classes. ResNet-18 provided a strong balance of computational efficiency and generalization, making it suitable for scenarios requiring robust performance with moderate resources. The Vision Transformer, while computationally efficient, was limited by the dataset size and resolution, highlighting its dependency on large-scale, high-quality datasets.

a) Improvements for Future Work:

- **Vision Transformer:** Train on larger datasets or augment CIFAR-10 with higher-resolution images to leverage the model's ability to capture global dependencies.
- **CNN-MLP:** Optimize the hybrid architecture by integrating advanced regularization techniques like DropConnect to further prevent overfitting.
- **ResNet-18:** Explore deeper ResNet variants (e.g., ResNet-50 or ResNet-101) to enhance feature extraction for more complex datasets.

This study demonstrates the versatility and challenges of machine learning models across diverse tasks like machine translation and image classification. The results emphasize the importance of task-specific optimization and the need for larger, high-quality datasets to fully realize model potential.

Future work should focus on:

- Developing hybrid models that leverage the strengths of different architectures.
- Scaling models to accommodate larger datasets and applying domain-specific pre-training.
- Exploring transfer learning and meta-learning techniques for cross-domain adaptability.

The insights gained from this study pave the way for further advancements in machine translation and image classification, pushing the boundaries of model efficiency and accuracy in real-world applications.

VI. Prompts

A. Transformer

For more details on the chat, visit the following link: Chat Link

B. LSTM

For more details on the chat, visit the following link: Chat Link

C. MarianMT

For more details on the chat, visit the following link: Chat Link

D. ViT

In the discussion, a Vision Transformer (ViT) model was evaluated and further fine-tuned for improved performance on a classification task. Initially, the model was loaded from a pretrained checkpoint and resumed training for an additional 30 epochs using a `CosineAnnealingLR` scheduler and label smoothing for regularization. The evaluation metrics after fine-tuning included:

- **Test Accuracy**: 47.12%
- **Precision**: 47.17%
- **Recall**: 47.12%
- **F1-Score**: 46.66%

The results indicated that the model had learned some patterns but struggled with generalization, showing scope for further improvement.

To address these challenges, the following strategies were discussed:

1) **Learning Rate Optimization**: Adjusting the learning rate and decay schedule to improve convergence.
2) **Data Augmentation**: Introducing advanced augmentation techniques such as Cutout, Mixup, and CutMix to enhance model robustness.
3) **Regularization**: Increasing weight decay and incorporating additional dropout layers in the Transformer.
4) **Transformer Enhancements**: Modifying self-attention mechanisms and expanding MLP layers.
5) **Longer Training**: Extending training epochs with early stopping to capture deeper patterns in the data.
6) **Pretrained Weights**: Leveraging pretrained ViT weights for fine-tuning instead of training from scratch.

This iterative process highlighted areas for improvement in model architecture, training strategy, and data processing to achieve better generalization and performance.

E. CNN-MLP

For more details on the chat, visit the following link: Chat Link

F. ResNet 18

For more details on the chat, visit the following link: Chat Link

References

[1] B. Jawaid and D. Zeman, ”Word-Order Issues in English-to-Urdu Statistical Machine Translation,” *The Prague Bulletin of Mathematical Linguistics*, no. 95, Univerzita Karlova, Praha, Czechia, 2011. ISSN: 0032-6585.

[2] Alex Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, Technical Report, University of Toronto, 2009. Available at: https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2024-11-09.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, *CoRR*, vol. abs/2010.11929, 2020. Available at: https://arxiv.org/abs/2010.11929. Accessed: 2024-11-09.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, *CoRR*, vol. abs/2010.11929, 2020. Available at: https://arxiv.org/abs/2010.11929.