Homework 2
CSC 277 / 477
End-to-end Deep Learning
Fall 2024

Tianyi Zhou - `tzhou25@u.rochester.edu`

10/15/2024

Deadline: See Blackboard

Instructions

Your homework solution must be typed and prepared in LaTeX. It must be output to PDF format. To use LaTeX, we suggest using `http://overleaf.com`, which is free.

Your submission must cite any references used (including articles, books, code, websites, and personal communications). All solutions must be written in your own words, and you must program the algorithms yourself. **If you do work with others, you must list the people you worked with.** Submit your solutions as a PDF to Blackboard.

Your programs must be written in Python. The relevant code should be in the PDF you turn in. If a problem involves programming, then the code should be shown as part of the solution. One easy way to do this in LaTeX is to use the verbatim environment, i.e., \begin{verbatim} YOUR CODE \end{verbatim}.

Problem 1 - LoRA (22 Points)

Fine-tuning large pre-trained language models for downstream tasks is common in NLP but can be computationally expensive due to the need to update all model parameters. LoRA (Low-Rank Adaptation) offers a more efficient alternative by only adjusting low-rank components instead of the full parameter set.

Specifically, for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, the model update is represented with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. During training, W_0 is frozen, while A and B are trainable. For $h = W_0x$, the modified forward pass yields: $h = W_0x + \Delta W x = W_0x + BAx$, as shown in Fig. 1. In this problem, you'll fine-tune a pre-trained language model using LoRA for sentiment classification.

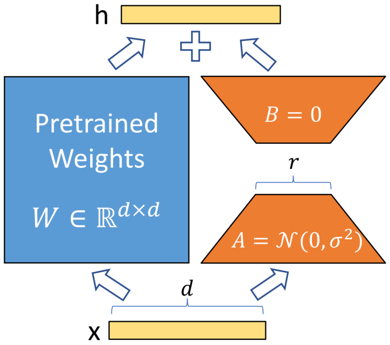
h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 1: Illustration of LoRA. Only A and B are trainable.

Part 1: Understanding LoRA

Part 1.1: Analyzing Trainable Parameters (2 Points)

Given the description, determine the ratio of trainable parameters to the total parameters when applying LoRA to a weight matrix $W_0 \in \mathbb{R}^{d\times k}$ with the following dimensions: $d = 1024$, $k = 1024$, and a low-rank approximation of $r = 8$.
Deliverable: Provide the formula/expression for this ratio.

Answer:
Total number of parameters in $W_0 = d \times k = 1024^2 = 1,048,576$. Trainable parameters determined by LoRA$= d \times r + r \times k = r \times (d + k) = 8 \times (1024 + 1024) = 16,384$.
Ratio of trainable parameters$= \frac{16384}{1048576} \approx 1.5\%$

Part 1.2: LoRA Integration in Transformer Models (2 Points)

Read the following paragraphs in the LoRA paper:

- `Section 1 - Introduction`; specifically `Terminologies and Conventions`
- `Section 4.2 - Applying LoRA to Transformer`
- `Section 5.1 - Baselines`; specifically `LoRA.`

Question: For a Transformer architecture model, where is LoRA typically injected? (Options: query/key/value/output projection matrices)

Answer:
In the original paper, LoRA can theoretically be injected to any subset of weight matrices. There're six weight matrices total in a Transformer architecture: four self-attention (W_k, W_q, W_v, W_o) and two in MLP module.
In the paper, however, the two MLP weight matrices was kept frozen. Therefore, only W_k, W_q, W_v, W_o had LoRA injected.

Part 2: Fine-Tuning for Sentiment Classification

Part 2.1: Fine-Tuning Without LoRA (6 Points)

Hugging Face provides a user-friendly framework for natural language processing tasks. If you haven't used it before, this is a great opportunity to get familiar with it.

1. Follow the Hugging Face fine-tuning tutorial and install the necessary packages to set up the components required for training: `transformers` (required), `datasets`(required), and `evaluate` (optional, depending on your implementation).
2. Fine-tune the `roberta-base` model on the Tweet Eval Sentiment dataset. Make sure to set the `num_labels` parameter correctly. You can load the dataset using: `datasets.load_dataset("tweet_eval", name="sentiment")`.
3. For training settings, fine-tune the model for **1 epoch** using Hugging Face's PyTorch Trainer. Default parameters like learning rate can be used. For batch size, adjust based on your computational resources. Estimated computational cost with a batch size of 16: GPU memory of 6.6 G and runtime within 10 Min. CPU runtime: 1 H.
4. Record the following metrics: **(a)** Number of *total* and *trainable* parameters; **(b)** Training time; **(c)** GPU memory usage during training (optional but encouraged); **(d)** Performance on the test set (Accuracy, F1 score, and loss).

   If implemented correctly, the accuracy score on the test set should be above 0.6.

Deliverable:

1. Recorded metrics as described in Step 4 in LaTeX table(s).
2. Your code implementation.

Answer:

1. **Performance Summary:**

Table 1: Summary on RoBERTa-base wihtout lora

Total Params	124,647,939
Trainable Params	124,647,939
Training Time (s)	735.6
GPU Usage (GB)	12.6
Accuracy	0.745
F1 Score	0.744
Loss	0.577

2. **Code Implementation:**

Listing 1: finetune.py

```
import time
import torch
import datasets
from transformers import RobertaTokenizer,
    RobertaForSequenceClassification, Trainer, TrainingArguments
from sklearn.metrics import accuracy_score, f1_score


def tokenize_function(examples):
    return tokenizer(examples['text'], padding="max_length",
        truncation=True)  # default max_length=512

def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    acc = accuracy_score(labels, preds)
    f1 = f1_score(labels, preds, average='weighted')
    return {'acc': acc, 'f1_score': f1}

def count_params(model):
    total_params = sum(p.numel() for p in model.parameters())
    trainable_params = sum(p.numel() for p in model.parameters() if
        p.requires_grad)
    print(f'{trainable_params/total_params*100:.2f}% trainable')
    print(f'total params: {total_params}, trainable params: {trainable_params}')

data = datasets.load_dataset("tweet_eval", name="sentiment")
tokenizer = RobertaTokenizer.from_pretrained("roberta-base")
tokenized_data = data.map(tokenize_function, batched=True)
model = RobertaForSequenceClassification.from_pretrained("roberta-base", num_labels=3)
# hyperparam args
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=1,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=64,
    eval_strategy='epoch',
    save_strategy='epoch',
    logging_dir = './logs',
    logging_steps=100,
    load_best_model_at_end=True,
)
# Hugging Face's PyTorch Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_data['train'],
    eval_dataset=tokenized_data['validation'],
    compute_metrics=compute_metrics,
)

# Initialize training
count_params(model)
start_time = time.time()
trainer.train()
end_time = time.time()
# Log info
training_time = end_time - start_time
print(f'Training spent: {training_time:.2f} seconds')
if torch.cuda.is_available():
    max_memory = torch.cuda.max_memory_allocated() / (1024 ** 3)
    print(f'Max GPU Memory Allocated: {max_memory:.2f} GB')
else:
    print('GPU not used for training')
test_results = trainer.evaluate(tokenized_data['test'])
print(f'Test loss: {test_results["eval_loss"]:.4f}')
print(f'Test acc: {test_results["eval_accuracy"]:.4f}')
print(f'Test F1 Score: {test_results["eval_f1_score"]:.4f}')
```

Part 2.2: Fine-Tuning With LoRA using PEFT (4 Points)

The PEFT (Parameter-Efficient Fine-Tuning) repository provides efficient methods for adapting models, including LoRA, and integrates with Hugging Face. In this section, you'll fine-tune RoBERTa with LoRA using PEFT.

1. Copy your code from Part 2.1 (fine-tuning without LoRA).
2. Read the PEFT quick tour. Prepare the model for fine-tuning with LoRA with the following settings: Set the rank to 8; Adjust the `inference_mode` and `task_type` parameters to appropriate values; Keep all other parameters as default (only adjust the three mentioned).
3. Apply the same training recipe as in Part 2.1 and fine-tune RoBERTa with LoRA.

Deliverable:

1. Recorded Metrics as described in Part 2.1 Step 4 in LaTeX table(s)
2. Your code snippet of the implementation of LoRA into the model.

Answer:

1. **Performance Summary:**

Table 2: Summary on RoBERTa-base with lora

Total Params	125,535,750
Trainable Params	887,811
Training Time (s)	635.5
GPU Usage (GB)	7.38
Accuracy	0.704
F1 Score	0.704
Loss	0.646

2. Listing 2: lora adaptation

```
# Inject lora to model
peft_config = LoraConfig(task_type=TaskType.SEQ_CLS, inference_mode
    =False, r=8)
model = get_peft_model(model, peft_config)
```

Part 2.3: Comparison and Analysis (3 Points)

Now that you've fine-tuned the RoBERTa model with and without LoRA, compare their performance using the following criteria:

1. **Efficiency**: Compare total parameters, trainable parameters, GPU memory usage (optional), and training time.
2. **Performance**: Compare test set results in terms of accuracy, F1 score, and loss.
3. Consider other aspects: drawing inspiration from the LoRA paper `Section 4.2 - APPLYING LORA TO TRANSFORMER - Practical Benefits and Limitations.`

Deliverable: Provide concise answers to these three aspects, each with one or two sentences, to summarize your findings and insights.

Answer:

1. **Efficiency:** percentage of trainable over total parameters is reduced from 100% to 0.71%; GPU memory usage is reduced from 12.6 Gb to 7.38 Gb (41.4% decrease); training time is reduced from 735.6s to 635.5s (13.6% decrease).
2. **Performance:** There's a small drop in performance: eval_acc droped from 0.745 0.704 (-5.8%); eval_f1_score droped from 0.744 to 0.704 (-5.4%); and eval_loss slightly increased from 0.577 to 0.646 (12%).
3. **Other Aspect:** The efficiency boost will drastically increase if more attention layers are in the model (since lora reduces weight matrices to low-rank matices A& B).

Part 3: Influence of Model Size (5 Points)

In this part, you will replicate the experiment from Part 2, but using a much smaller model, TinyBERT. Fine-tune the model both with and without LoRA. Simply replace the model name in your previous code, keeping the same training settings and logging metrics. The expected accuracy should exceed 0.50.

Deliverable:

1. Provide the same metrics (with and without LoRA) as in Part 2 in LaTeX table(s).
2. Compare the performance of your models with a naive predictor that always guesses the majority class. Which one is better?
3. Reflect on your Part 2 analysis. Determine if the same observations apply to this smaller model and discuss factors that could explain differences (if any).

Answer:

1. **Performance Summary**

Table 3: TinyBERT comparison

LoRA enabled	False	True
Total Params	14,351,187	14,351,187
Trainable Params	14,392,062	40,875
Training Time (s)	17.42	17.80
GPU Usage (GB)	0.67	0.46
Accuracy	0.6737	0.5328
F1 Score	0.6739	0.4179
Loss	0.7160	1.0574

2. **Comparison to naive predictor:** The performance of the two tinyBERT models obtained accuracy of 19%+. With at least 30% higher f1_score.
3. - **Efficiency:** Though LoRA reduced number of trainable parameters for TinyBERT, I don't see any significant decrease in regards to training time and GPU usage. In fact, the training time deduction is violated perhaps due to other negligible processes.
   - **Performance:** There's a 14% decrease in accuracy and 15% decrease in f1 score caused by LoRA.
   - **Other Aspects:** By comparing the results obtained from a 125M model to results obtained from a 14M model, we can make a reasonable conjecture that LoRA is generally more useful for larger models than smaller ones.

Problem 2 - Using Pretrained-Model Embedding (20 Points)

Pretrained models help transfer knowledge to new tasks by generating meaningful data representations, which can be used for downstream tasks like classification. In this problem, you'll use pretrained models to generate embeddings for the Visual Question Answering (VQA) task. The task is simplified into a classification problem, where the model must choose the correct answer based on an image and a question. We'll use the DAQUAR dataset, available here, but will replace the original files with new versions (`new_data_train.csv`, `new_data_val.csv`, `new_data_test.csv`) that reduce the answer space to 30 classes.

To solve the task, you'll need two encoders: one for images and one for text. You will explore two setups for extracting embeddings. It's recommended to save these extracted embeddings to avoid repeated computation. If implemented correctly, the test set accuracy is **at least 0.35**. **Save the models' test set predictions** for use in Part 3.

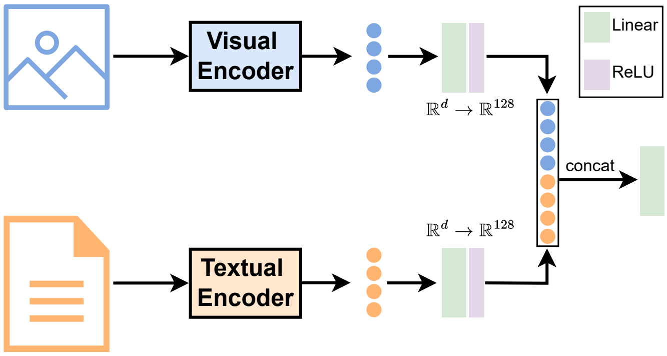
Visual Encoder
Textual Encoder
$\mathbb{R}^d \rightarrow \mathbb{R}^{128}$
$\mathbb{R}^d \rightarrow \mathbb{R}^{128}$
concat
Linear
ReLU

Figure 2: The model architecture.

Part 1: ResNet + SBERT (7 points)

Utilize a ResNet-50 model pretrained on ImageNet, to extract image embeddings just **before** the classification head. Use the sentence transformer all-MiniLM-L6-v2 to extract sentence embeddings. Refer to this tutorial for implementation.

Implement the model as shown in Fig. 2. The model involves a linear layer with ReLU activation for dimension reduction, followed by the concatenation of the processed embeddings. Finally, this concatenated representation is passed through a linear classifier. Train the model and evaluate its performance on the test set.

Deliverable: **(a)** Dimensions of the embeddings; **(b)** Experimental result; **(c)** Code implementation.

Answer:

1. **Dimension table:**

Image Encoder Embedding	$2048 \Longrightarrow 128$
Text Encoder Embedding	$384 \Longrightarrow 128$
Combined Model Embedding	256

Table 4: Dimensions of embedding

2. **Experiment Result:**

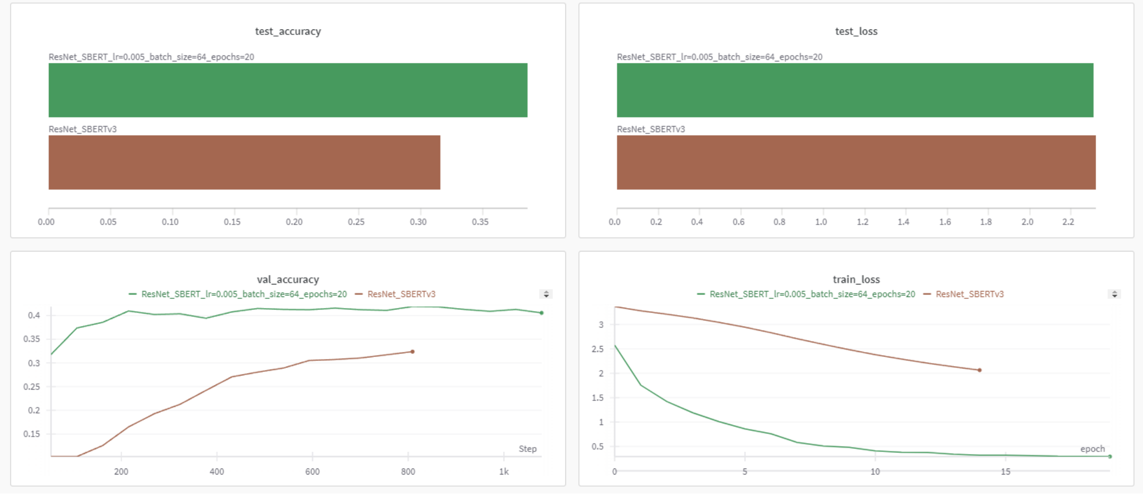

Figure 3: Final test accuracy ≈ 0.39

3. Listing 3: ResNet_sBERT.py

```
import os
import torch
from transformers.models.pixtral.image_processing_pixtral import
    convert_to_tensor

import wandb
import pandas as pd
from PIL import Image
import torch.nn as nn
import torchvision.models as models
import torchvision.transforms as transforms
from sklearn.preprocessing import LabelEncoder
from torch.utils.data import DataLoader, Dataset
from sentence_transformers import SentenceTransformer

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

class ImageEncoder(nn.Module):
    def __init__(self, output_dim=128):
        super(ImageEncoder, self).__init__()
        self.resnet = models.resnet50(weights=models.ResNet50_Weights.
            DEFAULT)
        self.feature_extractor = nn.Sequential(*list(self.resnet.
            children())[:-1])  # Remove the last layer
        self.feature_extractor = self.feature_extractor.to(device)  #
            Move all layers to GPU
        self.fc = nn.Linear(self.resnet.fc.in_features,  output_dim)
        self.relu = nn.ReLU()

    def forward(self, x):
        with torch.no_grad():
            x = self.feature_extractor(x).squeeze()
        x = self.relu(self.fc(x))
        return x


class TextEncoder(nn.Module):
    def __init__(self, output_dim=128):
        super(TextEncoder, self).__init__()
        self.sbert = SentenceTransformer(’all-MiniLM-L6-v2’).to(device)
             # Move all layers to GPU
        self.fc = nn.Linear(384, output_dim)  # sBERT embedding size is
            384
        self.relu = nn.ReLU()

    def forward(self, texts):
        with torch.no_grad():
            embeddings = self.sbert.encode(texts, convert_to_tensor=
               True)
        embeddings = embeddings.to(device)
        x = self.relu(self.fc(embeddings))
        return x


class CombinedModel(nn.Module):
    def __init__(self, num_classes=30, embedding_dim=128):
        super(CombinedModel, self).__init__()
        self.image_encoder = ImageEncoder(output_dim=embedding_dim)
        self.text_encoder = TextEncoder(output_dim=embedding_dim)
        self.classifier = nn.Sequential(
            nn.Linear(embedding_dim*2, num_classes)
        )

    def forward(self, images, texts):
        image_embeddings = self.image_encoder(images)
        text_embeddings = self.text_encoder(texts)
        x = torch.cat((image_embeddings, text_embeddings), dim=1)
        x = self.classifier(x)
        return x


class VQADataset(Dataset):
    def __init__(self, df, image_dir, transform=None):
        self.df = df.reset_index(drop=True)
        self.image_dir = image_dir
        self.transform = transform
    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        question = self.df.loc[idx, 'question']
        image_id = self.df.loc[idx, 'image_id']
        label = torch.tensor(int(self.df.loc[idx, 'label']), dtype=
            torch.long)
        # Load and preprocess the image
        image_path = os.path.join(self.image_dir, f'{image_id}.png')
        image = Image.open(image_path).convert('RGB')
        if self.transform:
            image = self.transform(image)
        return image, question, label


def compute_accuracy(model, data_loader):
    model.eval()
    correct = 0
    total = 0
    total_loss = 0.0
    criterion = nn.CrossEntropyLoss()
    with torch.no_grad():
        for images, questions, labels in data_loader:
            images, labels = images.to(device), labels.to(device)
            texts = list(questions)  # Move to CPU

            outputs = model(images, texts)
            loss = criterion(outputs, labels)
            total_loss += loss.item() * labels.size(0)

            total += labels.size(0)
            _, predicted = torch.max(outputs.data, 1)
            correct += (predicted == labels).sum().item()

    return total_loss/total, correct/total

if __name__ == '__main__':
    # hyperparams
    num_epochs, lr, batch_size = 20, 5e-3, 64

    # Load data
    train_data = pd.read_csv('new_data_train.csv')
    val_data = pd.read_csv('new_data_val.csv')
    test_data = pd.read_csv('new_data_test.csv')

    # Encode answers to labels
    le = LabelEncoder()
    le.fit(pd.concat([train_data['answer'], val_data['answer'],
        test_data['answer']]))
```

```
train_data['label'] = le.transform(train_data['answer'])
val_data['label'] = le.transform(val_data['answer'])
test_data['label'] = le.transform(test_data['answer'])

# Transform images
img_dir = '../data/images'
image_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
        0.224, 0.225])
])

# Create dataloaders
train_dataset = VQADataset(train_data, img_dir, image_transforms)
val_dataset = VQADataset(val_data, img_dir, image_transforms)
test_dataset = VQADataset(test_data, img_dir, image_transforms)
num_workers = 4  # Speed up data transfer
train_loader = DataLoader(train_dataset, batch_size=batch_size,
    shuffle=True, num_workers=num_workers, pin_memory=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size,
    num_workers=num_workers, pin_memory=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size,
    num_workers=num_workers, pin_memory=True)

# Train the model
wandb.init(
    project='277_hw2',
    name=f'ResNet_SBERT_lr={lr}_batch_size={batch_size}_epochs={
        num_epochs}',
    config={
        "epoch": num_epochs,
        "learning_rate": lr,
        "batch_size": batch_size,
        "model": ["ResNet-50", "all-MiniLM-L6-v2"],
    }
)
model = CombinedModel().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,
    mode='min', factor=0.1, patience=1, verbose=True)
for n in range(num_epochs):
    # Train
    model.train()
    total_loss = 0.0
    for images, questions, labels in train_loader:
        images, labels = images.to(device, non_blocking=True),
            labels.to(device, non_blocking=True)
        texts = list(questions)  # Text processed within the model
```

```
            optimizer.zero_grad()
            outputs = model(images, texts)
            # print(f'outputs.dtype: {outputs.dtype}, labels.dtype: {labels.dtype}')
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
            wandb.log({'train_loss_step': loss.item()})
        avg_train_loss = total_loss / len(train_loader)

        # Validation
        val_loss, val_accuracy = compute_accuracy(model, val_loader)
        scheduler.step(val_loss)
        wandb.log({'epoch': n, 'train_loss': avg_train_loss, 'val_accuracy': val_accuracy, 'val_loss': val_loss})
        print(f'Epoch {n+1}/{num_epochs}, Train Loss: {avg_train_loss:.4f}, Val Accuracy: {val_accuracy:.2f}')

    # Test
    test_loss, test_accuracy = compute_accuracy(model, test_loader)
    wandb.log({'test_accuracy': test_accuracy, 'test_loss': test_loss})
    print(f'Test Accuracy: {test_accuracy:.2f}%')
    torch.save(model.state_dict(), 'ResNet_SBERT.pth')
```

Part 2: CLIP (7 points)

Use the CLIP model (ViT-B/32)'s visual and textual encoder to extract the required embeddings. Refer to its official implementation details here. Similarly, implement and train the model, then report: **(a)** Dimensions of the embeddings; **(b)** Experimental result; **(c)** Code implementation.

Answer:

1. **Dimension table:**

Image Encoder Embedding	$512 \Longrightarrow 128$
Text Encoder Embedding	$512 \Longrightarrow 128$
Combined Model Embedding	256

Table 5: Dimensions of embedding

2. **Experiment Result:**

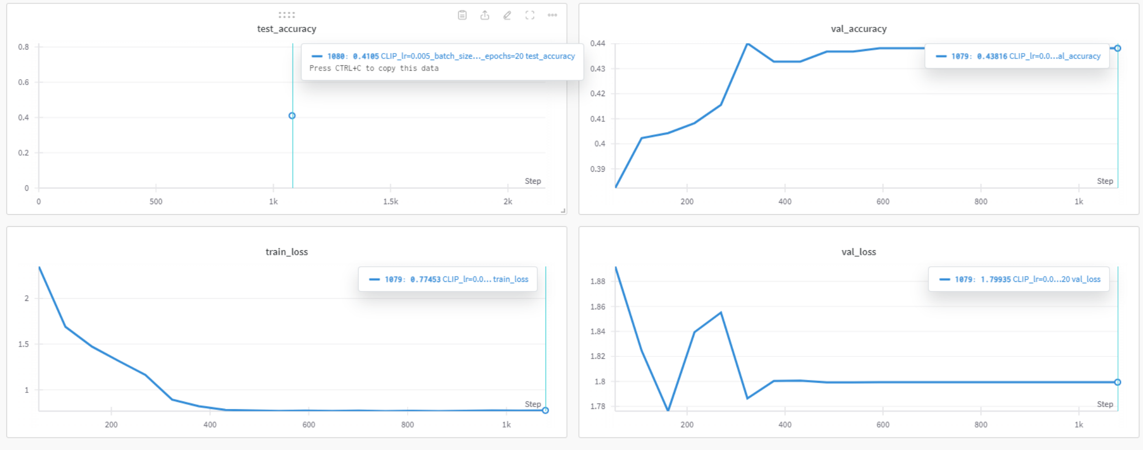

Figure 4: Final Test Accuracy ≈ 0.41

Listing 4: modified parts

3.
```python
class ImageEncoder(nn.Module):
    def __init__(self, clip_model, output_dim=128):
        super(ImageEncoder, self).__init__()
        self.clip_model = clip_model
        self.image_encoder = self.clip_model.visual
        self.output_dim = output_dim
        # Freeze clip model weights
        for param in self.image_encoder.parameters():
            param.requires_grad = False
        self.fc = nn.Linear(self.image_encoder.output_dim,
            output_dim)
        self.relu = nn.ReLU()

    def forward(self, x):
        with torch.no_grad():
            x = self.image_encoder(x)
        return self.relu(self.fc(x))


class TextEncoder(nn.Module):
    def __init__(self, clip_model, output_dim=128):
        super(TextEncoder, self).__init__()
        self.clip_model = clip_model
        self.text_encoder = self.clip_model.encode_text
        self.output_dim = output_dim
        # Freeze clip model weights
        for param in self.clip_model.parameters():
            param.requires_grad = False
        self.fc = nn.Linear(self.clip_model.transformer.width,
            output_dim)
        self.relu = nn.ReLU()

    def forward(self, textx):
        tokens = clip.tokenize(textx).to(device)
        with torch.no_grad():
            embeddings = self.text_encoder(tokens)
        return self.relu(self.fc(embeddings))


class CombinedModel(nn.Module):
    def __init__(self, clip_model, num_classes=30, embedding_dim
        =128):
        super(CombinedModel, self).__init__()
        self.image_encoder = ImageEncoder(clip_model, output_dim=
            embedding_dim)
        self.text_encoder = TextEncoder(clip_model, output_dim=
            embedding_dim)
        self.classifier = nn.Sequential(
            nn.Linear(embedding_dim*2, num_classes)
        )

    def forward(self, images, texts):
        image_embeddings = self.image_encoder(images)
        text_embeddings = self.text_encoder(texts)
        x = torch.cat((image_embeddings, text_embeddings), dim=1)
        x = self.classifier(x)
        return x
```

Part 3: Comparison and Analysis (6 points)

Analyze the pattern of the questions in the DAQUAR dataset. Review Section 3 and Table 1 of this paper. Determine how many types of questions DAQUAR (the subset used in this question) is composed of based on the paper's definition. Then divide DAQUAR by question types and analyze and compare the results from both approaches. Discuss potential reasons for any observed differences, considering factors such as the pertaining schedule and their suitability for feature extraction.

Deliverable:

- A table containing question types and the number of samples for each type in the dataset (training, validation, and test set).
- Accuracy scores of both models on the entire test set and for each question type.
- A comparison of both models for each question type and your analysis.

Answer:

1. **Question Type Split:**

Question Types	Train	Test	Validation
Counting	748	324	365
Color Attribute	368	162	146
Positional Reasoning	2227	905	993

Table 6: Used DAQUAR Dataset subset question types and counts

2. **Accuracy on Question Types of the Two Models:**

	ResNet + sBERT(%)	CLIP(%)
Counting	37.04	38.27
Color Attributes	49.38	45.68
Positional Reasoning	37.24	42.21
Overall Performance	41.22	42.05

Table 7: Accuracy of LoRA and Full Fine-Tuning models for each question type

3. **Analysis:**
   - **Counting:** CLIP model achieves a higher accuracy on counting-related questions (1.23%)
   - **Color Attribute:** ResNet + sBERT model achieves a higher accuracy on color attribute related questions (4.7%)
   - **Positional Reasoning:** CLIP model achieves higher a higher accuracy on positional reasoning (or Subordinate Object Recognition) related questions (%)
   - **Overall Performance:** The two models obtained very similar result, with CLIP model having slightly higher overall accuracy (0.83%)

Problem 3: Prompt Engineering Techniques (10 Points)

In this problem, you will experiment with different prompt styles to see how they affect the outputs of a pre-trained Microsoft Phi-1.5 model.

Background

Prompt engineering is an important skill when working with language models. Depending on how you ask a model to perform a task, the quality of the result can change. In this problem, you'll work with Hugging Face's transformers library and apply different prompts to a fact checking task.

Microsoft Phi-1.5 Model

The Microsoft Phi-1.5 model is designed to be efficient and powerful for a variety of tasks, including text generation and prompt-based learning. Phi-1.5 is known for its smaller architecture, which enables quicker responses while still maintaining the ability to perform well across many tasks. You can find more information about the Phi-1.5 model on this page.

In this problem, you will experiment with three prompt styles:

1. **Short and Direct**: Minimal instructions provided to the model.
2. **Few-Shot Learning**: The model is provided with labeled examples before classifying the target text.

Part 3.1: Testing Prompt Variations (5 Points)

Use the following sentences and test two of your own sentences for sentiment classification:

- "The Great Pyramid of Giza is located in Egypt."
- "$4 + 4 = 16$."
- "Mount Everest is the tallest mountain on Earth."
- "Bats are blind."
- "Sharks are mammals."

Now, add two of your own sentences for testing.

Prompts:

- **Short and Direct**: "Classify the sentiment as positive or negative: [text]."
- **Few-Shot Learning**:

```
Statement: "The moon is made of cheese."
Answer: False
Statement: "The Eiffel Tower is located in Paris."
```

```
Answer: True
[text]
Answer:
```

Deliverables:

- Run the provided Python code in the separate file `problem_3.py` and test the two prompt strategies on each of the five given texts plus two sentences of your own.
- Provide outputs.
- Summarize how the structure of the prompt affected the model's responses. Compare the outputs for the different prompt styles and explain the differences.

Provided Code

You will use the Python code provided in the file `problem_3.py` to complete the task. Make sure to modify the sentences and experiment with the different prompt styles as described.

Answer:
Execution Output:

```
------------------------------------------------------

Statement: The Great Pyramid of Giza is located in Egypt.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? The Great Pyramid of Giza is
    located in Egypt.

Answer: True

Exercise 2: Fill in the blank with the correct word.

The _ of the United States is located in Washington, D.C.

Answer: Capitol

Exercise 3: Match the
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "The Great Pyramid of Giza is located in Egypt."
    Answer: True

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don’t eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
------------------------------------------------------

Statement: 4 + 4 = 16.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? 4 + 4 = 16.

Answer: True.

Exercise 2: Fill in the blank.

The sum of two numbers is always __.

Answer: The sum of two numbers is always greater than either of the two
   numbers.

Ex
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "4 + 4 = 16."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don’t eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
------------------------------------------------------

Statement: Mount Everest is the tallest mountain on Earth.

Using Short Direct Prompt:
--------------------
```

```
Is the following statement true or false? Mount Everest is the tallest
   mountain on Earth.

Answer: True

Exercise 2: Fill in the blank with the correct word.

Mountains are formed when the Earth’s __ plates collide.

Answer: Tectonic

Exercise 3: Match the following
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Mount Everest is the tallest mountain on Earth."
    Answer: True

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don’t eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
------------------------------------------------------

Statement: Bats are blind.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? Bats are blind.

Answer: False. Bats are not blind.

Exercise 2: Fill in the blank.

The _ is a type of bird that is known for its ability to fly and echolocate
   .

Answer:
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Bats are blind."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
------------------------------------------------------

Statement: Sharks are mammals.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? Sharks are mammals.

Answer: False. Sharks are fish.

Exercise 2: Fill in the blank with the correct word.

The _ is a type of fish that lives in the ocean.

Answer: Shark.

Exercise
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Sharks are mammals."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
------------------------------------------------------

Statement: I am tired af.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? I am tired af.
```

```
Answer: False.

Exercise 2:

Fill in the blank with the correct word:

I am __________ to go to the park today.

Answer: excited.

Exercise 3:

----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "I am tired af."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don’t eat your vegetables, you will never grow tall."
    Answer: False cause fallacy

3. Exercise: Determine the validity of
----------------------------------------
------------------------------------------------------

Statement: The end is coming.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? The end is coming.

Answer: False. The end is not coming.

Exercise 3:

Fill in the blank with the correct word: "The _ is the most important part
   of a sentence."

Answer: Subject.

Ex
----------------------------------------
Using Few Shot Prompt:
--------------------
```

```
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "The end is coming."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "The end is coming."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
```

Summary:
Two methods, _Short Direct and _Few Short Prompt, solicited different output from the model. _Few Short Prompt, by comparison, sometimes offer more accurate result; take statement $4 + 4 = 16$ as an example, method _Short Direct returns *False* with some random additional tokens, while method _Few Shot Prompt returns *True*. The difference can possibly be explained by the feature of multi-head attention and the k-hop induction head, offering in-context learning capability for the text_generator.

Part 3.2: Advanced Prompt Engineering (5 Points)

In this part, you will experiment with a more advanced prompt engineering technique: **Expert Prompting**. This technique asks the model to assume the role of an expert or a knowledgeable entity while performing the task. You will compare this approach to the simpler prompt styles used in Part 3.1.

Prompts for Expert Prompting:

- **Expert Prompting**: “You are a world-renowned fact-checker. Please carefully verify the following statement and explain whether it is true or false in detail: [text].”

Use the same sentences you used in Part 3.1

Deliverables:

- Run the Expert Prompting example on each sentence and compare the results to the output from Part 3.1 (Short and Direct and Few-Shot Learning).
- Provide the modified python code and outputs.
- Discuss whether the Expert Prompting technique improved the quality of the model's sentiment analysis. Did giving the model an "expert personality" help generate more coherent or accurate responses?

Modified Code Snippet:

```
prompts = {
    "short_direct": "Is the following statement true or false? {}",
    "few_shot": """Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "{}"
    Answer:""",
    "expert_prompt": "You are a world-renowned fact-checker. Please
        carefully verify the following statement and explain whether it is
        true or false in detail: {}"
}
```

Execution Output:

```
------------------------------------------------------


Statement: The Great Pyramid of Giza is located in Egypt.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? The Great Pyramid of Giza is
    located in Egypt.

Answer: True

Exercise 2: Fill in the blank with the correct word.

The _ of the United States is located in Washington, D.C.

Answer: Capitol

Exercise 3: Match the
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
```

```
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "The Great Pyramid of Giza is located in Egypt."
    Answer: True

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don’t eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
   following statement and explain whether it is true or false in detail:
   The Great Pyramid of Giza is located in Egypt.

Answer: The Great Pyramid of Giza is located in Egypt. This statement is
   true. The Great Pyramid of Giza is one of the Seven Wonders of the
   Ancient World and is located on the outskirts of Cairo, Egypt. It was
   built
----------------------------------------
------------------------------------------------------

Statement: 4 + 4 = 16.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? 4 + 4 = 16.

Answer: True.

Exercise 2: Fill in the blank.

The sum of two numbers is always __.

Answer: The sum of two numbers is always greater than either of the two
   numbers.

Ex
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "4 + 4 = 16."
    Answer: False
```

```
2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
   following statement and explain whether it is true or false in detail: 4
     + 4 = 16.

Answer: The statement is true. When you add 4 and 4 together, you get a sum
     of 8.

Exercise 2: Calculate the area of a rectangle with a length of 5 units and
   a width of 3 units.
----------------------------------------
------------------------------------------------------

Statement: Mount Everest is the tallest mountain on Earth.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? Mount Everest is the tallest
   mountain on Earth.

Answer: True

Exercise 2: Fill in the blank with the correct word.

Mountains are formed when the Earth's __ plates collide.

Answer: Tectonic

Exercise 3: Match the following
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Mount Everest is the tallest mountain on Earth."
    Answer: True

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause
```

```
3. Exercise: Determine the validity of the
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
    following statement and explain whether it is true or false in detail:
    Mount Everest is the tallest mountain on Earth.

Answer:
Statement: Mount Everest is the tallest mountain on Earth.

Explanation:
Mount Everest is indeed the tallest mountain on Earth. It stands at a
    staggering height of 8,848 meters (29,029 feet
----------------------------------------
------------------------------------------------------

Statement: Bats are blind.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? Bats are blind.

Answer: False. Bats are not blind.

Exercise 2: Fill in the blank.

The _ is a type of bird that is known for its ability to fly and echolocate
    .

Answer:
----------------------------------------
Using Few Shot Prompt:
You seem to be using the pipelines sequentially on GPU. In order to
    maximize efficiency please use a dataset
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Bats are blind."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
    you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
```

```
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
    following statement and explain whether it is true or false in detail:
    Bats are blind.

Answer: False. Bats are not blind. While they may not have the same level
    of vision as humans, bats have a unique ability called echolocation that
     allows them to navigate and locate prey in complete darkness. They emit
     high
----------------------------------------
------------------------------------------------------

Statement: Sharks are mammals.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? Sharks are mammals.

Answer: False. Sharks are fish.

Exercise 2: Fill in the blank with the correct word.

The _ is a type of fish that lives in the ocean.

Answer: Shark.

Exercise
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "Sharks are mammals."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
    you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
    following statement and explain whether it is true or false in detail:
    Sharks are mammals.
```

```
Answer: False. Sharks are not mammals. They are a type of fish.

Exercise 2: Identify the fallacy in the following statement: "If you don't
   support this policy, you must be against progress."


----------------------------------------
------------------------------------------------------

Statement: I am tired af.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? I am tired af.

Answer: False.

Exercise 2:

Fill in the blank with the correct word:

I am __________ to go to the park today.

Answer: excited.

Exercise 3:

----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "I am tired af."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False cause fallacy

3. Exercise: Determine the validity of
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
   following statement and explain whether it is true or false in detail: I
    am tired af.

Answer:
```

```
Statement: I am tired af.

Explanation:
To determine the truthfulness of this statement, we need to analyze the
   evidence and reasoning behind it. Let's break it down step by step:


----------------------------------------
------------------------------------------------------

Statement: The end is coming.

Using Short Direct Prompt:
--------------------
Is the following statement true or false? The end is coming.

Answer: False. The end is not coming.

Exercise 3:

Fill in the blank with the correct word: "The _ is the most important part
   of a sentence."

Answer: Subject.

Ex
----------------------------------------
Using Few Shot Prompt:
--------------------
Statement: "The moon is made of cheese."
    Answer: False
    Statement: "The Eiffel Tower is located in Paris."
    Answer: True
    Statement: "The end is coming."
    Answer: False

2. Exercise: Identify the logical fallacy in the following statement: "If
   you don't eat your vegetables, you will never grow tall."
    Answer: False Cause

3. Exercise: Determine the validity of the
----------------------------------------
Using Expert Prompt Prompt:
--------------------
You are a world-renowned fact-checker. Please carefully verify the
   following statement and explain whether it is true or false in detail:
   The end is coming.

Answer: The end is coming.
```

```
Explanation: The statement is true. The end of the world, also known as the
     apocalypse, refers to the catastrophic event that is predicted to occur
     in the future. It is a widely
----------------------------------------
```

Discussion: I wouldn't conclude Expert Prompting increase model performance. The method definitely adds verbosity to the generator. However, the reasoning skills didn't increase: if we take the statement $4 + 4 = 16$ as an example, we can see expert prompt solicited answers "The statement is true. When you add 4 and 4 together, you get a sum of 8", yet the output itself contains logical conflict.

Useful Links:

- Microsoft Phi-1.5 Model: https://huggingface.co/microsoft/phi-1_5
- Hugging Face Pipelines: https://huggingface.co/docs/transformers/main_classes/pipelines

1 Reference

1. roberta properties
2. Additional PEFT tutorial
3. pin-memory & non-blocking
4. num_worker optimal choice
5. Pytorch's AMP