IT5005 Project Part 1 Report

1. Paper Reading Questions

1.1 What does the CLIP model learn?

CLIP learns transferable image representations by jointly training an image encoder and a text encoder from scratch. Instead of memorizing fixed object categories such as “cat” or “dog,” CLIP is trained on 400 million image–text pairs (WebImageText), each serving as a weakly supervised signal linking an image with its textual description.

Through training, CLIP aligns visual and textual embeddings so that an image of “a golden retriever” is positioned close to the text “a photo of a dog” and far from “a plate of sushi.” This joint alignment enables CLIP to generalize to unseen categories—recognizing new concepts simply through textual descriptions, much like how humans imagine a “hedgehog” after reading about it.

1.2 Explain in at most 3 sentences what "contrastive learning" means.

Contrastive learning is a method where the model learns representations by distinguishing between similar and dissimilar examples. In CLIP, the goal is to maximize the cosine similarity of matching (image, text) pairs and minimize it for mismatched pairs within a batch.

1.3 Why do you think CLIP's zero-shot performance can sometimes surpass supervised baselines? What does this say about the generalization abilities of representation learning?

CLIP’s strong zero-shot capability comes from from the scale and diversity of its training data. Unlike traditional supervised models that memorize a limited set of labels (e.g., ImageNet’s 1,000 classes), CLIP is trained on a vast and varied dataset covering photos, artworks, memes, and captions from the internet.This wide exposure allows it to learn broad, transferable concepts rather than overfitting to specific datasets.

Consequently, this large-scale, language-guided training leads to robust generalization and sometimes even surpasses narrowly trained supervised models. For example, when faced with a new class like “red panda,” CLIP can infer its meaning through linguistic cues alone. This demonstrates that representation learning can achieve greater generalization when exposed to diverse, multimodal data.

1.4 How do labels in CLIP-based zero-shot classification differ from traditional models?

Traditional models use fixed, predefined labels that carry no semantic meaning for the model. In contrast, CLIP uses natural language descriptions as dynamic, semantic labels (e.g., “a photo of a cat”). This approach allows it to classify virtually any concept that can be described with text, removing the restriction of a closed set of categories.

1.5 What are CLIP's shortcomings and why do they happen?

Despite its remarkable generalization, CLIP still has several weaknesses that mostly come from how it was trained and the kind of data it used.

First, CLIP is not good at fine-grained recognition. For example, it can easily recognize that an image shows a dog, but it may struggle to tell the difference between a “Siberian Husky” and an “Alaskan Malamute.” This happens because the model focuses on broad, general meanings rather than detailed visual differences.

Second, CLIP performs poorly on systematic reasoning tasks such as counting (“three apples on the table”) or spatial reasoning (“the cat is under the chair”), since its learning objective does not encourage compositional understanding.

CLIP also inherits biases from its unfiltered web data, reflecting stereotypes related to gender, culture, and ethnicity.

In addition, CLIP also shows poor efficiency in data and computation. Achieving its current level of performance requires massive amounts of data and compute power.

In short, CLIP’s strengths—scale and diversity, also explain its weaknesses: its learning is broad but imprecise, capturing correlations at massive scale without fine-grained human supervision.

2. Experimental Analysis

This section details the experiments conducted as part of the Part_1_Qns.ipynb .

2.1 Experimental Setup

1. **Environment:** The experiments were conducted using a Python environment with the core dependencies listed in requirements.txt, including torch, transformers, and datasets.
2. **Model:** The pre-trained openai/clip-vit-base-patch32 model was used as the foundation for all tasks.
3. **Dataset:** The experiments utilized a subset of the ImageNet dataset, focusing on a limited number of classes for efficient analysis.

2.2 Zero-Shot Classification Experiments

We investigated two key aspects of zero-shot classification with CLIP: the impact of prompt engineering and the choice of similarity metric.

Step 8: Prompt Engineering

Different text prompts were evaluated to determine their effect on classification accuracy. The results are indicated in the screenshot below.

```
8]  prompt_accuracies = evaluate_zero_shot_prompts(prompts_to_test, class_labels, test_images, test_labels, encoder, metrics)
    for prompt, accuracy in prompt_accuracies.items():
        print(f'Prompt: "{prompt}" -> Accuracy: {accuracy:.4f}')

·   Evaluating different prompts...

·   Prompts:   0%|          | 0/6 [00:00<?, ?it/s]

·   Prompt: "a photo of a {}" -> Accuracy: 0.5885
    Prompt: "a picture of a {}" -> Accuracy: 0.5875
    Prompt: "an image of a {}" -> Accuracy: 0.5841
    Prompt: "{}" -> Accuracy: 0.5396
    Prompt: "a wild animal: {}" -> Accuracy: 0.5383
    Prompt: "Beneath the fading sunset, the curious child wandered along the winding path. This is an image of {}." -> Accuracy: 0.4812
```

From the accuracy shown above for different prompts, we find that:

- The prompts "a photo/a picture/an image of a {}" are the strongest.

This is because such prompts directly align with the domain and can thus fit well.

- The prompt "{}" has a lower accuracy.

This is because the prompt, as a bare label, doesn't give any extra information.

- The prompt "a wild animal: {}" also underperforms.

This is because the prompt gives a biased prior and could mislead the model in other contexts.

- The prompt "Beneath the fading sunset, the curious child wandered along the winding path. This is an image of {}." performs the worst.

This is because such a long, irrelevant prompt might dilute the class token in the text embedding.

Results show that simple, natural prompts such as “a photo of a {class}” perform best. Short and contextually relevant prompts align well with the data CLIP was trained on. In contrast, bare labels lack semantic context, and long or irrelevant descriptions degrade accuracy by diluting the signal of the target concept.

Step 10: Similarity Metric Comparison

```
    X = test_images.to(encoder.device)
    txt = encoder.encode_text([best_prompt.format(c) fo
[9]
```

```
Using prompt: 'a photo of a {}'

Accuracy with dot_product: 0.1608
Accuracy with cosine_similarity: 0.5885
```

From the accuracy shown in above screenshot, for different similarity metrics, we find that cosine similarity performs significantly better than dot product. This is likely due to the fact that by definition, cosine similarity is normalised while dot product is not:

cosine_similarity(a, b) = (a · b) / (|a| |b|)

dot_product(a, b) = a · b

CLIP’s embeddings are normalized and optimized for cosine similarity, which measures the angular closeness between two vectors. As the dot product is unnormalized, it is affected by vector magnitude, leading to inconsistent and less meaningful comparisons.

2.3 Linear Probing vs. Zero-Shot

We then trained a simple linear classifier (ClsNetwork) on top of the frozen CLIP image features to evaluate the effectiveness of linear probing.

Step 13: Performance Analysis and Comparison

1. **final test accuracy and F1-score:**

The notebook results indicate the following performance for the two methods:

```
trainer = Trainer(classifier_model, train_loader, test_loader, device)
trainer.train(epochs = 10, lr = 1e-4)
```

[11] ✓ 58.0s

```
...
...  Epoch 1 Test Metrics: accuracy=0.4258, f1=0.3922
...  Epoch 2 Test Metrics: accuracy=0.5664, f1=0.5482
...  Epoch 3 Test Metrics: accuracy=0.6330, f1=0.6201
...  Epoch 4 Test Metrics: accuracy=0.6601, f1=0.6505
...  Epoch 5 Test Metrics: accuracy=0.6690, f1=0.6603
...  Epoch 6 Test Metrics: accuracy=0.6785, f1=0.6714
...  Epoch 7 Test Metrics: accuracy=0.6808, f1=0.6758
...  Epoch 8 Test Metrics: accuracy=0.6834, f1=0.6757
...  Epoch 9 Test Metrics: accuracy=0.6833, f1=0.6773
...  Epoch 10 Test Metrics: accuracy=0.6818, f1=0.6755

     Training finished!
```

The trained linear classifier achieves a final test accuracy of 0.6855 and an F1-score of 0.6795. This is a notable improvement over the best zero-shot accuracy of 0.5885. The peak performance was observed at epoch 9, with an accuracy of 0.6875 and an F1-score of 0.6815.

2. Summary Table:

Method	Data Requirement	Average Accuracy (Example)
Zero-Shot CLIP	0 labels (only class names)	**0.5885**
Linear Probing	Train labels (80% of dataset)	**0.6875**

3. Analyze the results:

The linear probe outperformed zero-shot classification because linear probing adapts the model to the specific data distribution and classes of the downstream task. While zero-shot classification demonstrates impressive generalization, fine-tuning even a simple linear head on task-specific labeled data allows the model to learn the specific features that are most discriminative for that particular dataset, leading to superior performance.

4. Trade-offs

The two methods present a clear trade-off:

Aspect	Zero-Shot Classification	Linear Probing
Performance	Lower, but strong generalization	Significantly higher on the specific downstream task.
Data Requirement	No labeled data required.	Requires labeled training data.

Aspect	Zero-Shot Classification	Linear Probing
Flexibility	Extremely high. Can classify any arbitrary set of classes at inference time.	Low. The classifier is trained for a fixed set of classes.
Computational Cost	Very low. Only requires a single forward pass for inference.	Higher. Requires a full training phase for the linear head.

2.4 Follow-up Questions Analysis

This section directly addresses the follow-up questions from the Jupyter notebook, based on the experimental results presented in the previous sections.

1. Try different prompts and record them. Which works best?

As recorded in the table in Section 2.2.1, we tested six different prompt templates. The prompt "a photo of a {}" achieved the highest accuracy of **0.5885**, making it the best-performing prompt among the tested options. Simple and direct prompts that are common in web data, such as "a picture of a {}" and "an image of a {}", also performed very well, indicating that the model is optimized for these kinds of natural language descriptions.

2. What happens if we add random things to the front of a simple prompt?

Adding complex, irrelevant, or overly descriptive text to the front of the prompt significantly degrades performance. As shown in the results, the long, narrative prompt "Beneath the fading sunset... This is an image of {}." resulted in the lowest accuracy of **0.4812**.

This happens because the additional, irrelevant text introduces noise and dilutes the core semantic meaning of the class label within the text embedding. The model's attention is spread across the entire sentence, making it harder to isolate the key concept it needs to match with the image.

3. Does using cosine similarity differ from using a dot product? Why or why not?

Yes, using cosine similarity results in dramatically different and superior performance compared to using the dot product. Our experiment showed that cosine similarity achieved an accuracy of 0.5885, while the dot product only achieved **0.1608**.

The reason for this significant difference lies in normalization. CLIP's embeddings are designed to be compared in terms of their angle, not their magnitude. Cosine similarity calculates the cosine of the angle between two vectors, effectively measuring their orientation regardless of their length, because it is normalized by the vector magnitudes. In contrast, the dot product is an unnormalized metric that is sensitive to the magnitude of the vectors, causing unstable and inaccurate similarity scores.

3. References

[1] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763). PMLR.

[2] OpenAI. (2021, January 5). CLIP: Connecting text and images. OpenAI Blog. https://openai.com/index/clip/

[3] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). CLIP: Contrastive Language-Image Pre-Training [Computer software]. GitHub. https://github.com/openai/CLIP

[4] Weng, L. (2021, May 31). Contrastive representation learning. Lil'Log. https://lilianweng.github.io/posts/2021-05-31-contrastive/

[5] Kumar, A. (2023, April 5). Boost foundation model results with linear probing and fine-tuning. Snorkel AI Blog. https://snorkel.ai/blog/boost-foundation-model-results-with-linear-probing-fine-tuning/

[6] Stanford University. (n.d.). Transfer learning. CS231n: Deep Learning for Computer Vision. https://cs231n.github.io/transfer-learning/

AI Tool Declaration:
We used GPT-5 to generate ideas, improve expression, produce drafts, refine our assignment. We are responsible for the content and quality of the submitted work.