University of Florence

Engineering School of Florence

Department of Information Engineering

What Do Vision-Language Models Understand? Dissecting CLIP Model from Baseline Supervised Classification to Class-Agnostic Similarity Features

Authors:

Niccolò Benedetto

Supervised by:

Mr. Andrew David Baghdanov

Academic Year: 2024-2025

Dedication

A voi, mia famiglia, che mai avete vacillato nella fede che riponevate in me, che con sguardi silenziosi e sorrisi discreti mi avete sempre sorretto: il vostro supporto è stato, e sarà, la mia roccia taciturna.

Ma oggi, in questa pagina che segna la fine di un lungo capitolo, rivolgo il mio più sentito ringraziamento a me stesso.
A me, che ho scelto di intraprendere questo cammino impervio, accettando la fatica come unica alleata.
A me, che tra svariati fallimenti e qualche incertezza, mai ho smesso di credere nella forza della mia testa, nella fermezza di ogni mio passo e nel valore di ciascun sacrificio.
A me, che ho costruito pietra dopo pietra, con pazienza e ostinazione, ciò che oggi posso chiamare conquista.

Questa tesi non è soltanto la prova tangibile del sapere acquisito, tuttavia è anche il simbolo di un cammino compiuto con dignità e coraggio. È il punto culminante di una fase e, al contempo, l'alba di una nuova partenza, più ambiziosa e consapevole. Che ogni fatica fin qui sostenuta sia il fondamento su cui poggiare il futuro, e che ogni sogno diventi progetto, ogni progetto, realtà.

A chi crede nel valore del lavoro silenzioso, dell'impegno incrollabile e della volontà che non chiede aiuto ma cerca solo se stessa: questa pagina è per voi.
Ma, prima di tutti, è per me.

Nonno, ce l'ho fatta. So che hai visto tutto, e che oggi sorridi con me.

Acknowledgments

I would like to express my sincere gratitude to Professor Andrew Bagdanov, whose guidance and expertise were instrumental throughout this thesis. I am also thankful to the Computer Science Department of Florence for the educational resources and support.

I extend heartfelt thanks to my family for their constant encouragement and understanding. Finally, I acknowledge the open-source communities and platforms like Hugging Face and OpenAI for making cutting-edge tools like CLIP accessible for research and experimentation.

Abstract

Contrastive Language–Image Pre-training (CLIP) models have shown remarkable success in aligning vision and language modalities through contrastive learning. However, understanding what these models truly encode about visual categories - especially under limited supervision - remains an open question.

This thesis investigates the representational quality of CLIP's image embeddings through a series of classification experiments on standard datasets such as CIFAR-10 and CIFAR-100. We begin with a baseline image classification pipeline using CLIP embeddings and supervised linear models. We then evaluate CLIP in a zero-shot setting using textual prompts to explore its performance without any training.

The core of our work explores class-agnostic similarity representations: rather than using class names, we generate generic prompts to probe CLIP's ability to separate visual categories based on cross-modal similarity alone. To assess and improve this approach, we identify and mitigate key challenges such as sparse logits, low image resolution, and limited classifier expressiveness. We experiment with temperature scaling, prompt engineering, and multi-layer classifiers to better leverage CLIP's pretrained features.

Our results provide new insights into how and what CLIP understands from images, highlighting the model's strengths and limitations in encoding class-discriminative features even without fine-tuning. This analysis contributes to a deeper understanding of the effectiveness and boundaries of vision-language models in classification tasks under various supervision levels.

Contents

List of Figures

List of Tables

General introduction

0.1 Background and Context

Recent advances in multimodal learning have led to the emergence of models that jointly learn from visual and textual data. A prominent example is CLIP (Contrastive Language–Image Pretraining), which is trained to align images with their corresponding textual descriptions in a shared embedding space using a contrastive loss objective. By doing so, CLIP enables strong cross-modal capabilities such as zero-shot classification, where the model can recognize unseen classes based on natural language prompts - without requiring task-specific fine-tuning.

CLIP's architecture and training paradigm have significantly influenced research in vision-language modeling, retrieval, and representation learning. Its ability to generalize across a wide range of tasks stems from leveraging massive and diverse web-scale image-text pairs during pretraining.

However, CLIP was not designed with supervised classification in mind, and it is not immediately clear how well its learned image representations support conventional classification tasks. Furthermore, while CLIP excels when using semantically rich prompts, its behavior under class-agnostic settings - where textual prompts are generic and not tied to class labels - remains underexplored.

This thesis investigates CLIP's representations from multiple perspectives: first through standard supervised classification using its image embeddings, then through zero-shot classification with textual prompts, and finally through class-agnostic similarity features. In doing so, we aim to better understand the extent to which CLIP's embeddings are structured in a way that supports class separability, even in the absence of explicit class supervision.

0.2 Objectives and Goals

This thesis aims to analyze the representational quality of CLIP's image embeddings through the lens of both supervised and unsupervised classification tasks. The central objective is to understand what CLIP "knows" about visual categories when

evaluated in standard classification pipelines, including zero-shot and class-agnostic scenarios. Specifically, the goals of this work are:

- To understand CLIP's architecture, training objective, and representational behavior in downstream tasks, particularly classification.
- To evaluate baseline classification performance using CLIP's image embeddings on standard datasets (CIFAR-10 and CIFAR-100) with supervised classifiers. classification using embeddings.
- To explore CLIP's zero-shot classification ability by measuring performance without fine-tuning, using natural language prompts as class descriptors.
- To study class-agnostic similarity-based representations, where generic prompts (not tied to class names) are used to generate logit features for classification.
- To identify and address potential bottlenecks in class-agnostic performance, such as sparse logits, suboptimal prompts, and resolution mismatch.
- To test various enhancement strategies - including temperature scaling, alternative classifiers (e.g., MLP, SVM), and high-resolution datasets - to improve classification accuracy.
- To derive insights into the strengths and limitations of CLIP's learned features, with a particular focus on whether they naturally encode class-discriminative structure.

0.3 Methodology

This study is experimental and implementation-driven, combining supervised and unsupervised evaluations to assess the quality of CLIP's image representations. The methodology consists of the following steps:

1. Model and Dataset Preparation: the pretrained CLIP model (`ViT-B/32`) is loaded using the Hugging Face transformers library. Standard image classification datasets such as CIFAR-10 and CIFAR-100 are used as benchmarks.
2. Baseline Supervised Classification: CLIP's image embeddings are extracted and used as input features for classical classifiers (e.g., Logistic Regression and SVM) to measure performance in a fully supervised setting.

3. Zero-Shot Classification Evaluation: the model's zero-shot capabilities are evaluated by generating text prompts corresponding to class labels, encoding them into the shared latent space, and computing cosine similarity with image embeddings.

4. Class-Agnostic Logit Representation: a set of general-purpose, class-agnostic prompts (e.g., "*a photo of a small object*", "*a photo of a blurry environment*") is introduced to derive logit features. These features are evaluated using linear classifiers to assess whether CLIP's representations contain useful, discriminative structure even without class-specific supervision.

5. Performance Optimization and Diagnostic Analysis: several strategies are tested to improve class-agnostic classification accuracy

    - Temperature scaling to adjust the sharpness of cosine similarity distributions.
    - Prompt refinement for more consistent and object-centric descriptions.
    - Alternative classifiers, such as MLPs, to better capture nonlinear patterns.
    - Use of high-resolution datasets like Imagenette to test resolution sensitivity.

6. Evaluation and Interpretation: accuracy metrics, entropy analysis, and visualizations are used to interpret the behavior of different configurations. The results are compared to both zero-shot and baseline supervised benchmarks to understand the limits of CLIP's implicit visual understanding.

All experiments described in this chapter were implemented in Python using PyTorch and HuggingFace Transformers. The complete code is available on GitHub at the following address:
`https://github.com/NiccoBene00/DissectingCLIP`

Chapter 1

Project Overview

1.1 Reviewing Literature on CLIP

The CLIP (Contrastive Language–Image Pre-training) model, introduced by Radford et al. [2], represents a pivotal advancement in multimodal learning by aligning visual and textual representations within a shared embedding space. CLIP's architecture is composed of two separate neural networks: a Vision Transformer (ViT) or ResNet for image encoding, and a Transformer-based model for text encoding. These networks map their respective inputs - images and natural language descriptions - into fixed-length embeddings, which are trained jointly using a contrastive loss objective.

At the core of CLIP's training paradigm lies *contrastive learning*, specifically a symmetric InfoNCE (Noise-Contrastive Estimation) loss applied across batches. Given a batch of N image-text pairs, CLIP computes a full $N \times N$ similarity matrix between all image and text embeddings. The model is then optimized to maximize the similarity of correct (diagonal) pairs while minimizing the similarity of incorrect (off-diagonal) ones. This learning strategy enables CLIP to associate a wide range of visual concepts with natural language prompts, achieving state-of-the-art zero-shot performance on various vision-language tasks - including classification, retrieval, and captioning - without the need for task-specific fine-tuning.

However, CLIP's training is fundamentally geared toward *inter-modal alignment* - i.e., learning correspondences between images and text - rather than optimizing relationships within a single modality. This lack of intra-modal structure introduces a challenge known as *intra-modal misalignment*, wherein semantically similar images (or texts) may be embedded far apart in the latent space. Such limitations can hinder CLIP's effectiveness in unimodal tasks, such as image-to-image retrieval or

similarity-based clustering, where a coherent intra-modal structure is crucial.

Recent studies [3, 4] have critically examined this issue, demonstrating that CLIP's embeddings - while effective for cross-modal reasoning - are suboptimal for unimodal downstream tasks. This has led to growing interest in developing techniques that either repurpose or refine CLIP representations to better capture intra-modal semantic structure, laying the groundwork for this thesis.

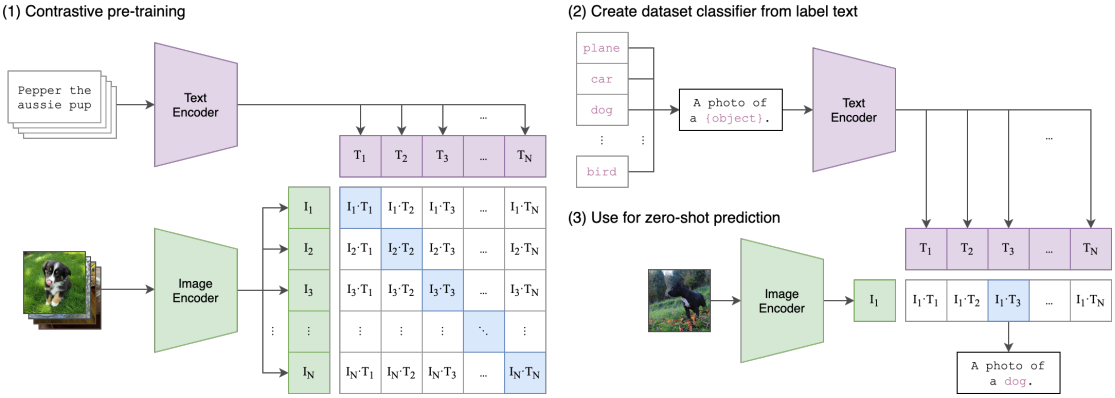

Figure 1.1: Architectural structure of the CLIP model. Image adapted from [1]

1.2 Experimental Framework

This thesis is structured around a series of experiments designed to analyze and understand the intra-modal capabilities of the CLIP model. The project is divided into five key phases:

1. **CLIP Model Exploration:** An initial investigation into the architecture, training paradigm, and functional behavior of the CLIP model, using the Hugging Face implementation. This includes an understanding of its vision and text encoders, as well as its contrastive learning objective.

2. **Baseline Intra-modal Classification:** Establishing a performance benchmark by extracting CLIP's image embeddings from datasets such as CIFAR-10 and CIFAR-100, and evaluating their suitability for traditional supervised classification tasks using linear classifiers such as SVM or logistic regression.

3. **Zero-Shot Classification:** Assessing CLIP's original zero-shot inference capability using class-specific prompts to evaluate cross-modal alignment performance, and comparing it to supervised baseline performance.

4. **Improving Class-Agnostic Representations:** Investigating various techniques to improve the quality of CLIP's class-agnostic similarity-based features. This includes adjusting prompt design, applying temperature scaling to mitigate logit sparsity, experimenting with multilayer perceptrons, expanding the prompt vocabulary, and evaluating high-resolution datasets such as Imagenette.

5. **Evaluation and Analysis:** Comparing the performance of baseline and enhanced methods in terms of classification accuracy and representation quality. This phase aims to draw conclusions on what CLIP representations capture well and where they fall short in intra-modal contexts.

1.3 Mathematical Reminders

This section outlines the key mathematical concepts and tools that form the theoretical backbone of this thesis. These reminders are essential for understanding the CLIP model architecture, its contrastive learning paradigm, and the inversion techniques used for intra-modal retrieval and classification.

Vector Spaces and Embeddings

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input spaces for images and text, respectively. CLIP encoders $f_{\text{img}} : \mathcal{X} \to \mathbb{R}^d$ and $f_{\text{text}} : \mathcal{Y} \to \mathbb{R}^d$ project inputs into a shared d-dimensional embedding space $\mathbb{R}^d$ (generally d=512) [5]. The encoded vectors are ℓ_2-normalized so that $\|f(x)\|_2 = 1$ for all x, enabling cosine similarity to be used as a similarity measure:

$$\text{sim}(x, y) = \frac{f_{\text{img}}(x) \cdot f_{\text{text}}(y)}{\|f_{\text{img}}(x)\| \, \|f_{\text{text}}(y)\|}.$$

Hence, since image embeddings f_{img} and text embeddings f_{text} are projected into a shared space, cosine similarity provides a scale-invariant way to measure their alignment. It ensures that only directional alignment matters, not magnitude - ideal for comparing high-dimensional semantic representations. In practice, both embeddings are ℓ_2-normalized, so the cosine similarity reduces to the dot product $f_{\text{img}}(x) \cdot f_{\text{text}}(y)$.

Contrastive Learning

The model is trained using a contrastive loss, specifically the symmetric InfoNCE loss [6]. This is used in contrastive learning to bring positive pairs (correct image-text) closer together in embedding space, while pushing negative pairs (incorrect matches) further apart. Given a batch of N image-text pairs $\{(x_i, y_i)\}_{i=1}^{N}$, the loss for an image x_i and its corresponding text y_i is:

$$\mathcal{L}_{\text{contrast}}^{\text{(image-to-text)}} = -\log \frac{\exp(\text{sim}(f_{\text{img}}(x_i), f_{\text{text}}(y_i))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f_{\text{img}}(x_i), f_{\text{text}}(y_j))/\tau)},$$

where $\tau > 0$ is a temperature parameter, i.e. a learnable scalar that adjusts the concentration level of the distribution. A lower τ sharpens the distribution, making the model more confident in its predictions. All this is equivalent to a cross-entropy loss where the correct text is the target class for the image input. In addition a symmetric term (text-to-image) is also computed by swapping modalities, ensuring alignment in both directions:

$$\mathcal{L}_{\text{contrast}}^{\text{(text-to-image)}} = -\log \frac{\exp(\text{sim}(f_{\text{text}}(y_i), f_{\text{img}}(x_i))/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(f_{\text{text}}(y_i), f_{\text{img}}(x_j))/\tau)},$$

This enforces that not only should images retrieve their correct texts, but texts should also retrieve their corresponding images. The final loss used in training is the average of the two directional losses:

$$\mathcal{L}_{\text{total}} = \frac{1}{2}\left(\mathcal{L}_{\text{contrast}}^{\text{(image-to-text)}} + \mathcal{L}_{\text{contrast}}^{\text{(text-to-image)}}\right)$$

This symmetric InfoNCE loss has proven effective in learning joint embeddings for vision and language, as shown in [2].

Numerical Example

In CLIP, both image and text embeddings are normalized to unit length. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, the cosine similarity between them is:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \mathbf{u}^\top \mathbf{v}$$

since $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$ by normalization.

Let

$$\mathbf{u} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}, \quad \mathbf{v}_1 = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$$

Then,

$$\text{sim}(\mathbf{u}, \mathbf{v}_1) = 0.6 \cdot 0.707 + 0.8 \cdot 0.707 = 0.9898$$
$$\text{sim}(\mathbf{u}, \mathbf{v}_2) = 0.6 \cdot (-0.707) + 0.8 \cdot 0.707 = 0.1414$$

These values represent the model's similarity between an image and two text prompts.

CLIP uses the InfoNCE loss for contrastive learning. Given a batch of N image-text pairs $(\mathbf{u}_i, \mathbf{v}_i)$, the loss encourages matched pairs to have high similarity, and unmatched pairs to have low similarity. For image-to-text, the loss is:

$$\mathcal{L}_{\text{img}\to\text{text}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp\left(\text{sim}(\mathbf{u}_i, \mathbf{v}_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(\mathbf{u}_i, \mathbf{v}_j)/\tau\right)}$$

Let's consider a Batch Size = 2 where:

- $\text{sim}(\text{Img}_1, \text{Text}_1) = 0.946$
- $\text{sim}(\text{Img}_1, \text{Text}_2) = -0.432$
- $\text{sim}(\text{Img}_2, \text{Text}_2) = 0.910$
- $\text{sim}(\text{Img}_2, \text{Text}_1) = -0.150$

Assuming $\tau = 0.07$:

$$\text{sim}_{1,1}/\tau = \frac{0.946}{0.07} \approx 13.514, \quad \text{sim}_{1,2}/\tau = \frac{-0.432}{0.07} \approx -6.171$$
$$Z_1 = \exp(13.514) + \exp(-6.171) \approx 735392.8 + 0.0021 = 735392.8021$$
$$\mathcal{L}_1 = -\log\left(\frac{\exp(13.514)}{Z_1}\right) = -\log\left(\frac{735392.8}{735392.8}\right) = -\log(1) = 0.0$$
$$\text{sim}_{2,2}/\tau = \frac{0.910}{0.07} \approx 13.00, \quad \text{sim}_{2,1}/\tau = \frac{-0.150}{0.07} \approx -2.14$$
$$Z_2 = \exp(13.00) + \exp(-2.14) \approx 442413.4 + 0.118 = 442413.518$$
$$\mathcal{L}_2 = -\log\left(\frac{\exp(13.00)}{Z_2}\right) \approx -\log(1) = 0.0$$

Of course, real batches are more noisy and larger, so the loss won't be zero, but this illustrates the principle.

Classification and Linear Models

In the inter-modal baseline, we extract image embeddings and use them as features to train a supervised classifier. Given feature-label pairs (z_i, y_i), we consider a linear model:

$$f(z) = Wz + b,$$

where $W \in \mathbb{R}^{C \times d}$, C is the number of classes, and $z \in \mathbb{R}^d$ is an embedding. Models such as Support Vector Machines (SVM) or logistic regression [7] are commonly used here. The cross-entropy loss is employed for training:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{N} \log \frac{\exp(f_{y_i}(z_i))}{\sum_{j=1}^{C} \exp(f_j(z_i))}.$$

Evaluation Metrics

To assess the performance of both traditional and class-agnostic classification pipelines, we rely on several evaluation metrics:

Classification Accuracy. This is the primary metric used throughout the experiments. Given a set of predicted labels $\hat{y}_i$ and ground-truth labels y_i for N samples, accuracy is defined as:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i)$$

where $\mathbb{1}(\cdot)$ is the indicator function that returns 1 if the prediction is correct, and 0 otherwise. Accuracy is suitable for balanced datasets such as CIFAR-10 and CIFAR-100.

Entropy of Logits. To analyze the confidence and sparsity of the class-agnostic logit features, we compute the entropy of the softmax-normalized logit vectors. Given a logit vector $y_i \in \mathbb{R}^C$ for an input image (where C is the number of class-agnostic text prompts), we first apply the *softmax function*, which transforms the raw similarity scores into a probability distribution:

$$p_i = \frac{\exp(y_i/\tau)}{\sum_{k=1}^{C} \exp(y_k/\tau)}$$

Next, we compute the Shannon entropy of the resulting probability vector p:

$$H(p) = -\sum_{i=1}^{C} p_i \log p_i$$

Entropy quantifies the uncertainty of the distribution. A lower entropy indicates a more confident or sparse prediction (i.e., the model strongly favors a few classes), whereas a higher entropy corresponds to greater uncertainty, where the logits are spread more evenly across classes. It is important to remember that entropy only measures the "concentration" of the probability distribution, not its correctness. In other words, entropy alone is not sufficient to explain the effectiveness of the model, but it is a descriptive metric, not a predictive one.

It is important as well to distinguish that the softmax function itself merely normalizes logits to a probability simplex and does not perform learning. The *loss function*, such as the cross-entropy or InfoNCE loss, operates on the softmax outputs and compares them to target distributions to guide model optimization during training [8]. In our case, we are not training, but analyzing the model's implicit confidence via entropy over the temperature-scaled softmax output of frozen logits.

Confusion Matrix. For in-depth class-wise analysis, confusion matrices can be used to visualize true versus predicted labels. This can be particularly helpful in identifying which classes the model struggles with.

Visualization of Logit Distributions. In addition to scalar metrics, we also include visual diagnostics such as line plots of cosine similarity logits and histograms of entropy values. These help qualitatively evaluate feature separability and model confidence.

1.4 Tools and Framework

This project leverages a variety of open-source tools and libraries to conduct experiments, analyze results, and implement classification pipelines. The key components include:

- **CLIP (Contrastive Language–Image Pre-training):** The core model used throughout the thesis. Pretrained CLIP variants are accessed via the `transformers` library by Hugging Face, providing both image and text encoders for multimodal representation.
- **Datasets:** Experiments are primarily conducted on CIFAR-10 and CIFAR-100, which consist of low-resolution natural images across diverse categories. Additional testing on Imagenette, a high-resolution subset of ImageNet, is used to assess the impact of image quality on representation.

- **PyTorch:** The primary deep learning framework used for all tensor operations, model inference, and feature extraction. PyTorch's flexibility enables integration with both the CLIP architecture and custom processing pipelines.
- **Hugging Face Transformers and Datasets:** Provides easy access to pretrained CLIP models and dataset loaders, streamlining experimental setup.
- **Scikit-learn:** Used for training traditional classifiers (e.g., Support Vector Machines, Logistic Regression) on CLIP embeddings and for computing evaluation metrics such as accuracy and confusion matrices.
- **NumPy and Matplotlib:** Employed for numerical computation, result aggregation, and visualization of features, similarity distributions, and entropy metrics.
- **Seaborn:** Utilized for enhanced data visualization, especially in analyzing the structure and confidence of logit distributions through entropy histograms and similarity heatmaps.

1.5 Expected Outcomes

This thesis aims to contribute both empirical findings and methodological insights into the structure and usability of vision-language models, particularly CLIP. The expected outcomes include:

- **Improved classification performance through embedding refinement:** Performance enhancement in class-agnostic classification tasks through techniques such as prompt engineering, temperature scaling, and model-based projection.
- **Understanding cross-modal embedding behavior:** Practical insights into how inter-modal alignment influences intra-modal structure, and how modality-agnostic features can approximate or even match zero-shot performance.
- **Methodological contributions:** Development of a structured evaluation framework for probing CLIP's internal representations, including reusable tools for visualizing logits, analyzing sparsity, and tuning similarity-based representations.

- **Open-source and reproducibility:** All code, models, and experimental results are made available to support further research on embedding quality, model interpretation, and lightweight evaluation of multimodal representations.

Conclusion

In conclusion, this chapter has provided a foundational overview of the project, outlining its objectives and scope. By establishing this context, the subsequent chapters will delve into the specific methodologies, findings, and analysis that contribute to achieving the project's goals.

Chapter 2

CLIP Benchmarks

2.1 Introduction

This chapter presents an initial empirical investigation into the performance of the CLIP model on standard image classification tasks. Specifically, we utilize the `openai/clip-vit-base-patch32` variant of CLIP as a frozen image encoder and evaluate its extracted representations on the CIFAR-100 dataset.

The primary goal of this phase is to establish a performance baseline that reflects the quality of CLIP's learned visual features. To do so, we extract 512-dimensional image embeddings from CLIP without updating the model weights, and train a linear Support Vector Machine (SVM) classifier on top of these features. While this approach does incorporate supervised learning through the SVM, the core CLIP model remains untouched. As such, this experiment approximates a few-shot learning setting, where only a lightweight classifier is trained on top of a general-purpose embedding space.

This setup allows us to assess how effectively CLIP captures meaningful visual semantics that generalize across tasks. If the SVM achieves strong performance despite limited model complexity and no feature tuning, it would suggest that CLIP's pre-trained features are highly informative — a desirable property for zero-shot and few-shot applications.

In addition to this supervised baseline, we also evaluate CLIP in its native zero-shot classification mode, using text prompts to match images against class labels without any training. This contrast highlights the difference between CLIP's cross-modal reasoning capabilities and its intra-modal representational structure, setting the stage for deeper analyses in subsequent chapters.

2.2 Baseline Image Feature Classification

In this section, we establish a baseline by leveraging CLIP as a frozen feature extractor. We apply this model to the CIFAR-100 image classification task, using a linear Support Vector Machine (SVM) trained on the extracted image embeddings. The objective is to evaluate the quality of CLIP's visual representations for intra-modal classification tasks without modifying the model's weights.

2.2.1 Model and Dataset Setup

We use the pretrained CLIP model and processor from Hugging Face:

```
clip_model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
clip_processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
clip_model = clip_model.to(device).eval()
```

The CIFAR-100 dataset is loaded using `torchvision` and wrapped in a custom dataset class to ensure each image is preprocessed appropriately using CLIP's processor:

```
class CLIPImageDataset(Dataset):
    def __init__(self, cifar_dataset):
        self.dataset = cifar_dataset

    def __getitem__(self, idx):
        image, label = self.dataset[idx]
        processed = clip_processor(images=image,
           return_tensors="pt")
        return processed["pixel_values"].squeeze(0), label
```

2.2.2 Embedding Extraction

Image embeddings are extracted by forwarding preprocessed images through the frozen CLIP model. Each image is mapped to a 512-dimensional embedding vector:

```
def extract_embeddings(dataloader):
    all_embeddings, all_labels = [], []
    with torch.no_grad():
        for images, labels in dataloader:
            images = images.to(device)
```

```
            embeddings = clip_model.get_image_features(
                images)
            all_embeddings.append(embeddings.cpu().numpy())
            all_labels.extend(labels.numpy())
    return np.vstack(all_embeddings), np.array(all_labels)
```

2.2.3 Classification and Evaluation

The extracted features are standardized using `StandardScaler` to ensure fair treatment across dimensions, and a linear SVM is trained and evaluated:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(train_features)
X_test_scaled = scaler.transform(test_features)

svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
svm_clf.fit(X_train_scaled, train_labels)
y_pred = svm_clf.predict(X_test_scaled)
```

Classification accuracy is then computed as:

```
accuracy = accuracy_score(test_labels, y_pred)
print(f"Baseline CLIP image classification: {accuracy *
    100:.2f}%")
```

2.2.4 Final Insights

The CIFAR-100 dataset presents a greater challenge than CIFAR-10 due to its 100 fine-grained categories, which often include visually similar subtypes (e.g., various animal species or object variants). The observed drop in classification accuracy from 93.24% on CIFAR-10 to 78.01% on CIFAR-100 reflects two key factors:

- **Increased intra-class variability and inter-class similarity:** CIFAR-100 contains categories that are semantically and visually closer to one another, making it harder to distinguish them using general-purpose embeddings.
- **Low-resolution complexity:** CIFAR images are small (32×32 pixels), which reduces the amount of fine-grained visual information available to both the feature extractor and the classifier.

A detailed classification report highlights the following trends:

- **High-performing classes** (F1-score ≥ 0.90): Typically include objects with distinctive shapes or appearances such as *pickup truck*, *wardrobe*, *keyboard*, and *chair*.
- **Low-performing classes** (F1-score ≤ 0.60): Often involve visually similar natural categories like *beaver*, *shrew*, *seal*, and *crocodile*, which are harder to distinguish at low resolution.
- **Moderate performance in human categories:** Classes such as *boy*, *girl*, *man*, and *woman* show F1-scores in the range of 0.75–0.82. This may indicate that CLIP captures coarse-grained semantic information related to human identity but struggles with nuanced differences in such visually constrained settings.

Overall, the experiment suggests that CLIP's pretrained vision encoder may be less equipped for distinguishing fine-grained natural classes, particularly under limited resolution. The underperformance of many animal-related categories, compared to objects or vehicles, aligns with the hypothesis that CLIP's training on web-scale image-text pairs favors distinctive and context-rich entities over biologically subtle distinctions. This highlights the importance of considering different dataset resolution and semantic content when evaluating intra-modal representation quality.

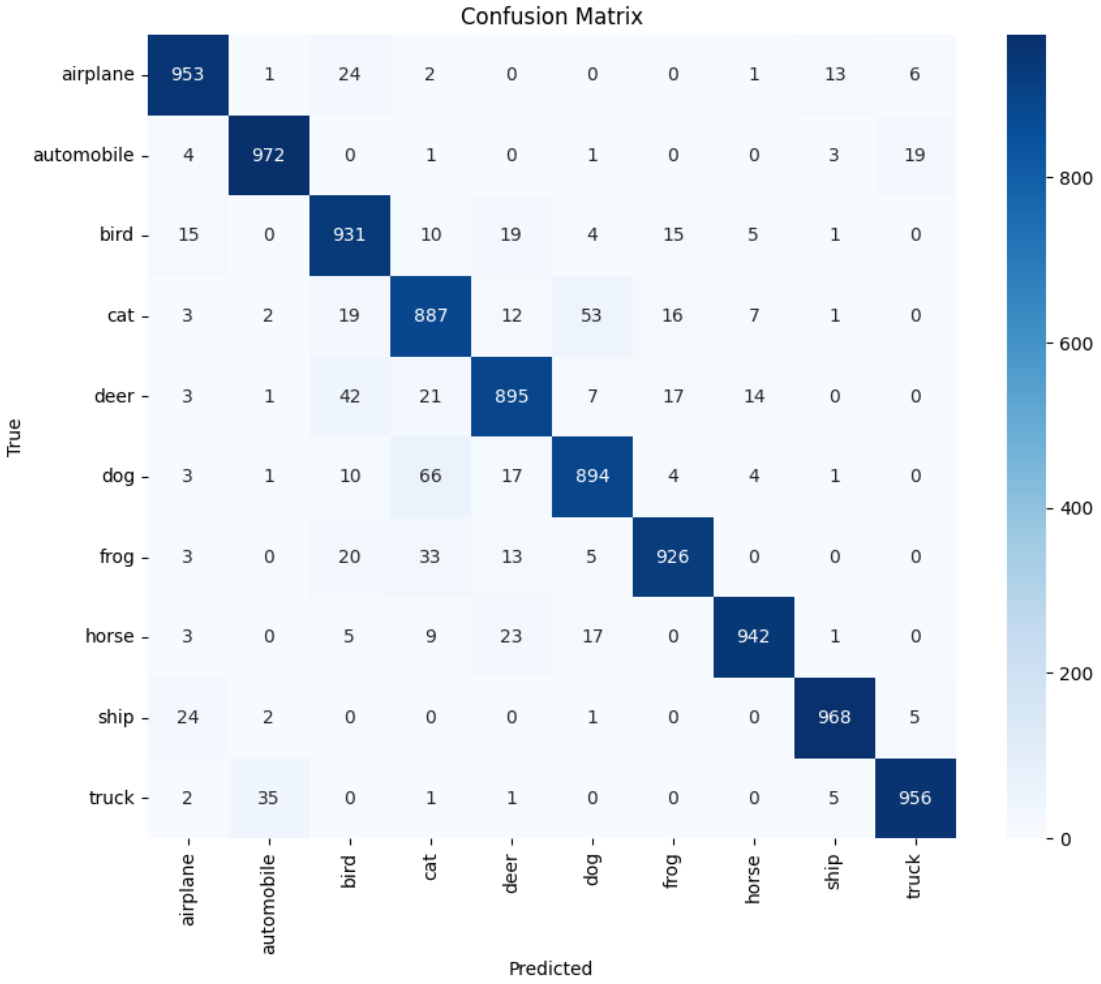

Figure 2.1: Confusion Matrix with CIFAR-10 - Baseline Classification

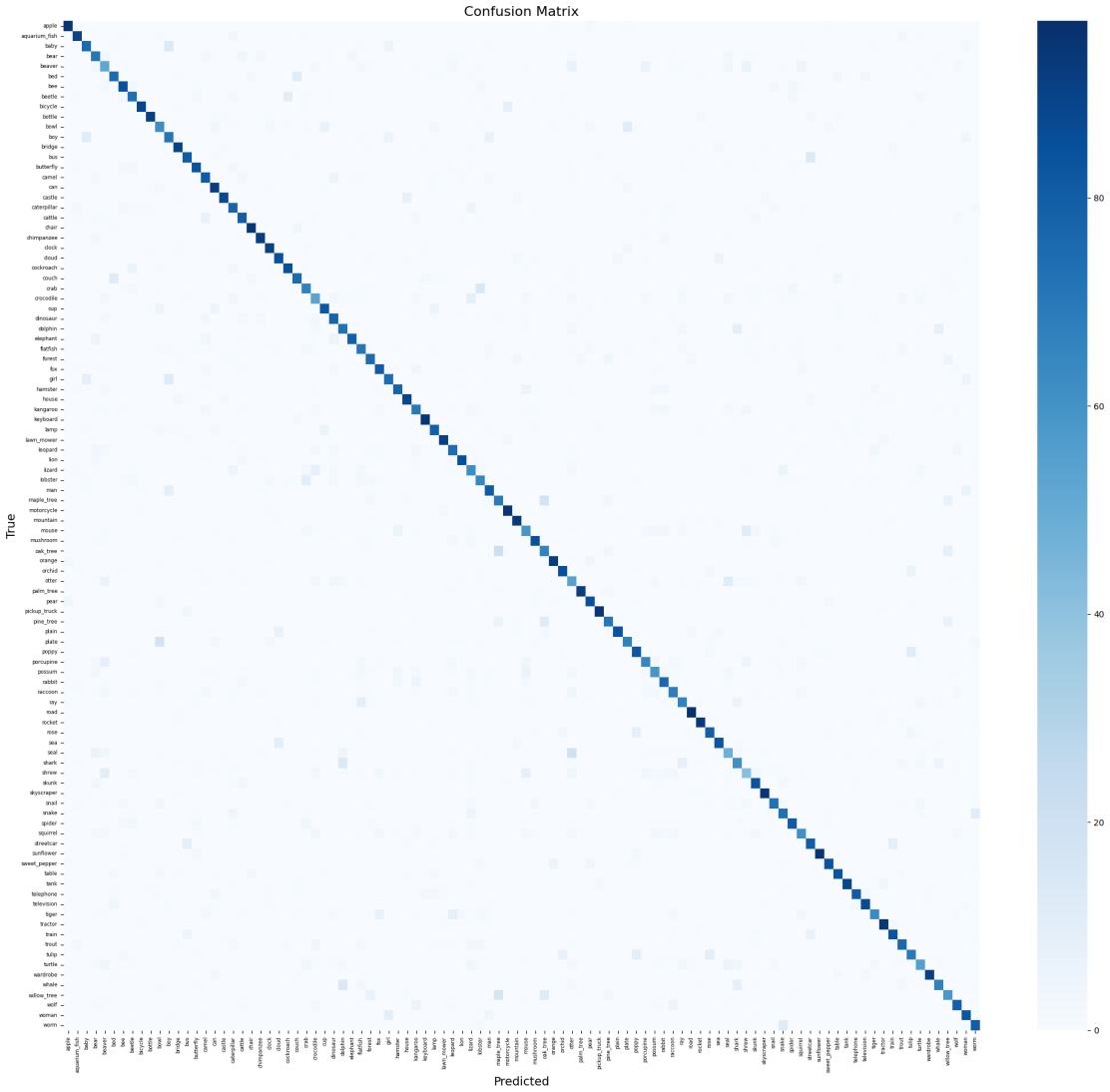

Figure 2.2: Confusion Matrix with CIFAR-100 - Baseline Classification

2.3 Zero-Shot Classification

Zero-shot classification represents one of CLIP's most compelling capabilities. Unlike traditional supervised models, CLIP does not require retraining or fine-tuning on specific downstream tasks. Instead, it leverages its training on large-scale image–text pairs to directly align visual inputs with natural language prompts. Given an image and a set of candidate descriptions - such as *"a photo of a cat"*, *"a photo of a truck"*, and so on - CLIP computes the similarity between the image embedding and each text embedding, selecting the most similar one (i.e. the one

with the bigger similarity) as its prediction.

This architecture allows CLIP to generalize beyond the set of classes it has explicitly encountered during training. It relies on semantic alignment between visual content and textual cues rather than predefined class labels, making it highly flexible and versatile for novel classification scenarios.

2.3.1 Experiment Setup

Following the baseline classification experiment, we evaluate CLIP's zero-shot capabilities on CIFAR-100. This involves no additional training or fine-tuning: classification is performed by directly comparing the similarity between image embeddings and a set of handcrafted text prompts using CLIP's dual encoders.

CIFAR-100 consists of 100 semantic categories. To align with CLIP's training paradigm, we construct prompt phrases such as `"a photo of a {class}"` for each label. These natural language descriptions are tokenized (directly through the clip processor) and encoded into 512-dimensional text embeddings using CLIP's frozen text encoder:

```
text_prompts = [f"a photo of a {label}" for label in
    class_names]
text_inputs = clip_processor(text=text_prompts,
    return_tensors="pt", padding=True).to(device)
text_features = clip_model.get_text_features(**text_inputs)
text_features = text_features / text_features.norm(dim=-1,
    keepdim=True)
```

Each test image is then passed through the image encoder, and cosine similarity is computed between the image features and all class prompt embeddings:

```
inputs = clip_processor(images=image, return_tensors="pt").
    to(device)
image_features = clip_model.get_image_features(**inputs)
image_features = image_features / image_features.norm(dim
    =-1, keepdim=True)

similarity = (image_features @ text_features.T).unsqueeze(0)
pred_label = similarity.argmax().item()
```

2.3.2 Final Insights

Compared to the CIFAR-10 experiment, we observe a clear performance degradation on CIFAR-100:

- On CIFAR-10, CLIP achieves 88.8% accuracy using zero-shot classification with natural language prompts.
- On CIFAR-100, CLIP still performs competitively, reaching 61.71% accuracy despite the increased difficulty- 100 classes, many of which are fine-grained and visually similar, and low-resolution images (32×32).

These findings underscore both the strengths and limitations of CLIP:

- Its semantic representations are rich and transferable, supporting generalization to unseen categories.
- However, confusion increases when label distinctions are subtle or underrepresented in its pretraining data.

In conclusion, CLIP's zero-shot performance confirms the power of contrastive multimodal learning. Yet, when evaluated on tightly constrained or noisy datasets such as CIFAR-100, a lightweight supervised classifier (e.g., linear SVM) trained on CLIP embeddings can yield higher accuracy by better adapting to dataset-specific decision boundaries.

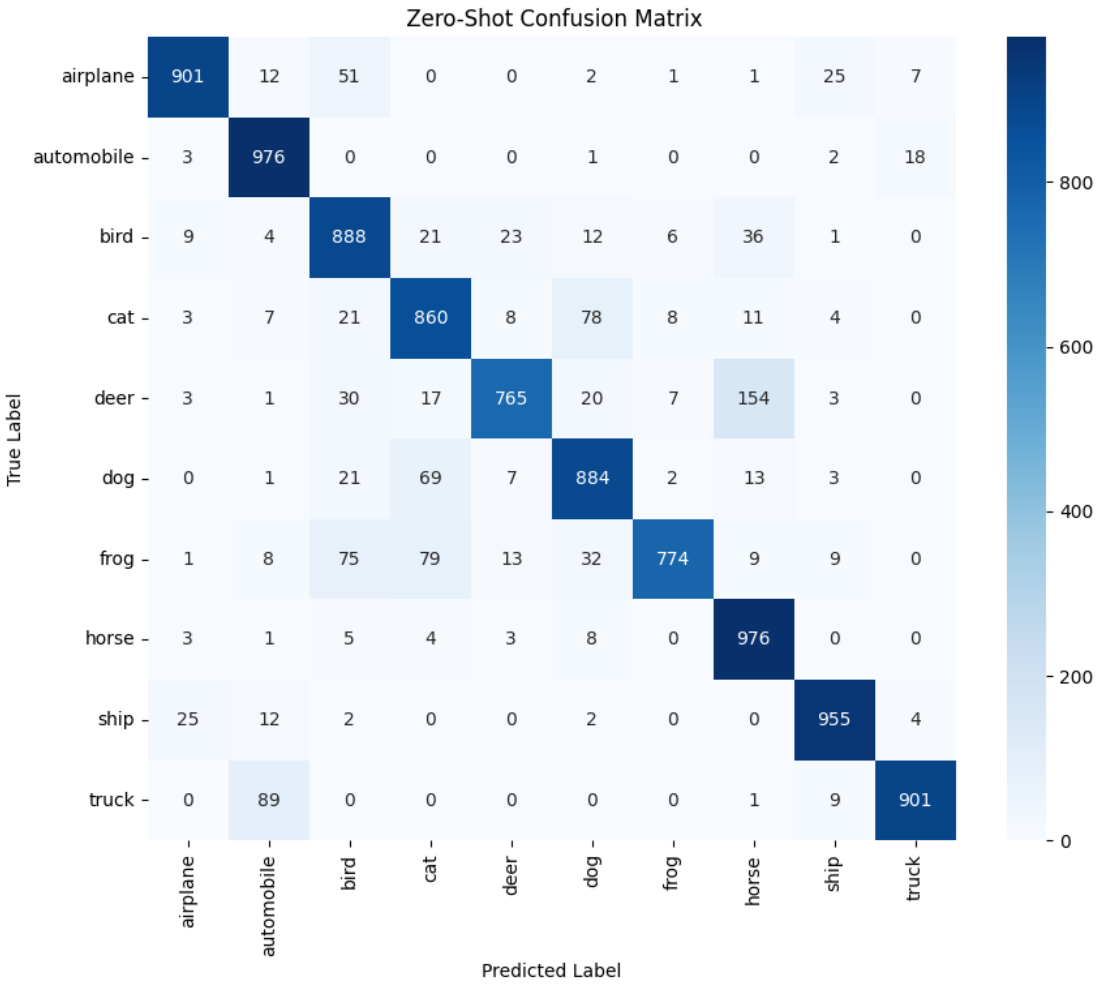

Figure 2.3: Confusion Matrix with CIFAR-10 - Zero-Shot Classification

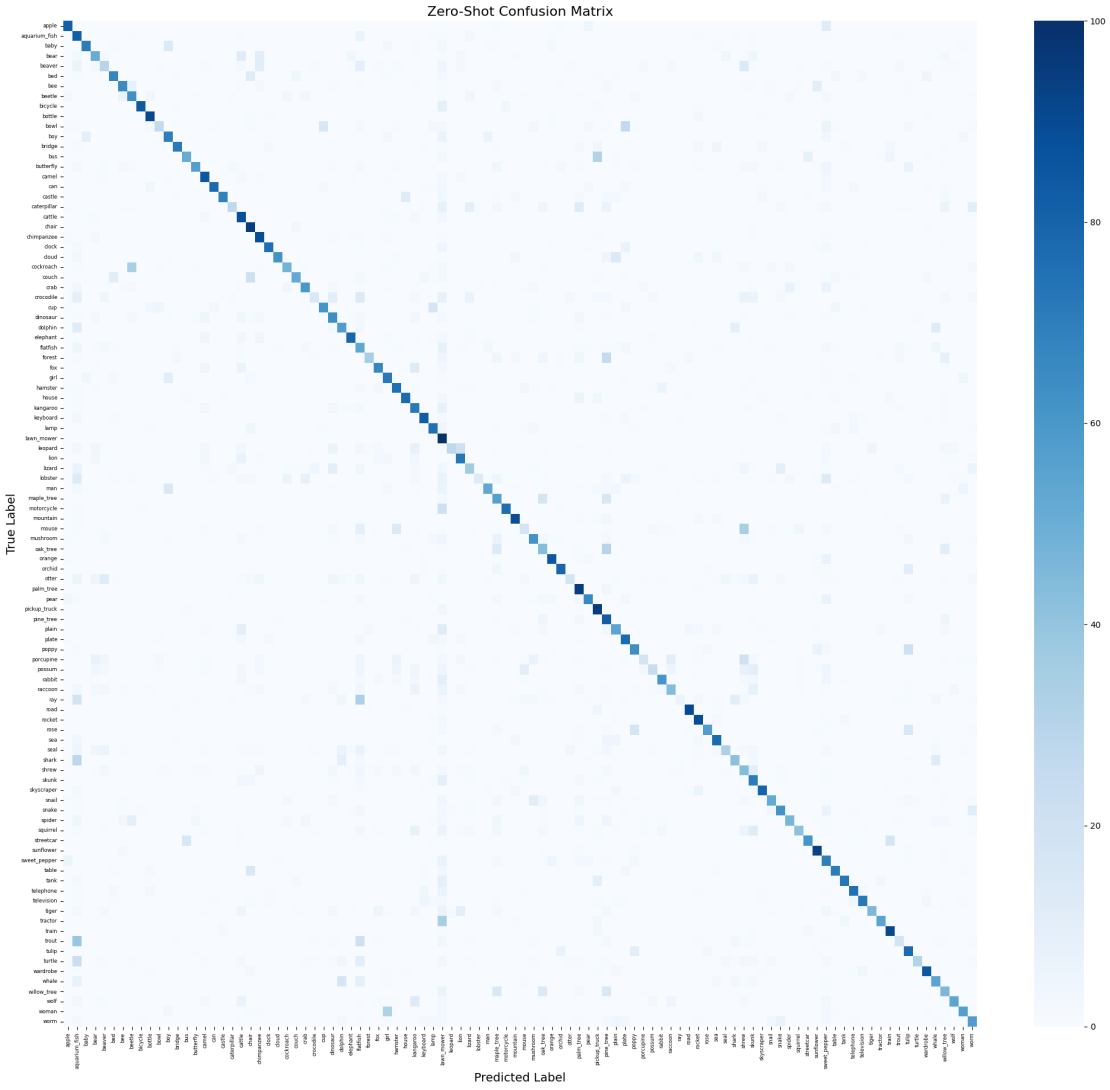

Figure 2.4: Confusion Matrix with CIFAR-100 - Zero-Shot Classification

Chapter 3

Class-Agnostic Techniques

Introduction

In this chapter, we explore class-agnostic representation techniques as a means to improve or better understand the representational structure of CLIP embeddings. A class-agnostic approach implies that the learned features or representations are not explicitly tied to predefined class labels during feature extraction. Rather than crafting representations aligned with known categories (e.g., *airplane*, *cat*, *dog*), we aim to extract generic, task-independent features.

In the context of models like CLIP, *logits* refer to the raw output scores - such as the cosine similarity between image and text embeddings (in zero-shot classification) or the output of a linear classifier like SVM - before applying a softmax function. A *class-agnostic logit representation* thus refers to using these similarity scores as feature vectors, without anchoring them to any specific class label during extraction. Instead, we treat these logits as rich semantic indicators of how an image aligns with a set of conceptual prompts.

The core idea of this chapter is to use the image-to-text similarity logits - computed between images and a collection of generic prompts - as input features for a downstream classifier, such as a Support Vector Machine (SVM). This contrasts with the baseline experiments in Chapter 2, where classification was performed directly on CLIP's image embeddings.

Our investigation is structured around the following guiding questions:

- Are CLIP's similarity logits semantically rich enough to support object classification even in the absence of class labels during feature extraction?
- Can CLIP's internal similarity structure generalize to new tasks or datasets without additional fine-tuning?

- Where is the most useful semantic information encoded in CLIP - within its image embeddings or within the similarity logits?

This exploration falls under the name of probe-based supervised evaluation, where a lightweight classifier (here, an SVM) is trained on top of fixed, pretrained representations to assess how well those representations encode class-discriminative information. Importantly, we do not update or fine-tune CLIP itself; instead, the SVM acts as a diagnostic tool - a *probe* - to measure the effectiveness and generality of the pretrained features.

The subsequent sections present our methodology, experiments, and insights gained from this class-agnostic probing of CLIP's internal representations.

3.1 Class-Agnostic Logits for CLIP Classification

In this section, we investigate whether class-agnostic similarity logits - computed between CIFAR-100 images and a set of generic, non-class-specific prompts - can serve as effective features for downstream classification. Unlike traditional approaches which rely on image embeddings or class-specific labels, our method leverages CLIP's ability to reason about high-level semantic concepts through text-image similarity.

3.1.1 Prompt Design

We construct a list of 32 prompts designed to reflect general visual concepts rather than dataset-specific categories. These include semantic attributes like *"a flying thing"*, *"a photo taken outdoors"*, or *"a blurry photo"*. This approach enables us to probe whether CLIP can align images with abstract concepts that generalize beyond specific class boundaries:

```
prompts = [
    "a photo of an object",
    "a photo of something natural",
    "a photo of something man-made",
    "a blurry photo",
    "a close-up photo",
    "a photo taken during the day",
    "a photo taken outdoors",
    "a photo taken indoors",
    "a small object",
    "a large object",
    "an animal",
    "a machine",
```

```
    "a thing that moves",
    "a stationary object",
    "a photo of a living thing",
    "a photo of a synthetic thing",
    "a colorful object",
    "a grayscale image",
    "a smooth surface",
    "a textured object",
    "a plastic object",
    "a metallic object",
    "a flying thing",
    "a photo of something round",
    "a rectangular object",
    "a noisy image",
    "an object with wheels",
    "an object with wings",
    "an underwater object",
    "a photo from a low angle",
    "a photo from a top view",
    "a centered object",
]
```

The prompts are tokenized and encoded via CLIP's text encoder. The resulting embeddings are normalized to enable cosine similarity comparisons:

```
text_inputs = processor(prompts, return_tensors="pt",
    padding=True).to(device)
with torch.no_grad():
    text_features = model.get_text_features(**text_inputs)
text_features = text_features / text_features.norm(dim=-1,
    keepdim=True)
```

3.1.2 Feature Extraction via Class-Agnostic Logits

To extract features for classification, we compute cosine similarities (logits) between each image and the entire prompt set. These logits are then used as feature vectors for a downstream classifier:

```
def extract_logit_features(data_loader, text_features):
    image_features_list = []
    labels_list = []

    for inputs, labels in tqdm(data_loader):
        with torch.no_grad():
```

```
            image_features = model.get_image_features(**
                inputs)
            image_features = image_features / image_features
                .norm(dim=1, keepdim=True)
            logits = image_features @ text_features.T

        image_features_list.append(logits.cpu().numpy())
        labels_list.append(labels.cpu().numpy())

    features = np.concatenate(image_features_list, axis=0)
    labels = np.concatenate(labels_list, axis=0)
    return features, labels
```

3.1.3 Training and Evaluation

A linear Support Vector Machine (SVM) is trained on the logit features extracted from the CIFAR-100 training set. The model is then evaluated on the test set to assess the quality of these class-agnostic representations:

```
svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
svm_clf.fit(train_features, train_labels)

test_preds = svm_clf.predict(test_features)
test_acc = accuracy_score(test_labels, test_preds)
print(f"\nTest accuracy using Linear SVM: {test_acc * 100:.2
    f}%")
```

3.1.4 Final Insights

The class-agnostic experiment yielded a test accuracy of 29.30%, which - while significantly better than random chance (1% for CIFAR-100) - falls well short of both our linear probe baseline (~78%) and CLIP's own zero-shot performance (~61.71%).

This result offers a nuanced perspective on the effectiveness of raw similarity logits as representations:

- The model can extract *some* class-discriminative information from generic, semantically high-level prompts, even without any class-specific guidance.
- However, the performance gap suggests that class-agnostic logits may lack sufficient resolution or specificity to robustly distinguish among fine-grained classes like those in CIFAR-100.

- Unlike image embeddings or class-specific zero-shot prompts, these logits are aligned with *broad visual attributes*, which might not correspond well with the subtle differences between visually similar categories (e.g., "*shrew*" vs. "*otter*").

This experiment highlights a critical tension in using pretrained vision-language models for fine-grained classification: while CLIP's generic alignment capabilities are powerful, they may not be directly suited for high-resolution classification without class-aware supervision or tuning.

Looking ahead, this performance ceiling motivates the next section, where we investigate potential improvements to the class-agnostic strategy.

These explorations aim to diagnose and mitigate the limitations uncovered here, offering paths toward more robust generalization from pretrained features.

3.2 Analyzing the Limitations of Class-Agnostic Logits Representations

While the class-agnostic strategy produced an accuracy of 29.30%, this is considerably lower than both the zero-shot classification (61.71%) and the supervised baseline (78.01%). In this section, we investigate the underlying reasons behind this performance gap and outline several key limitations that emerged from our analysis.

Sparse Logit Distributions

We might think that the cosine similarity scores between image and prompt embeddings - used as features for classification - tend to produce sparse distributions. Calculating the sparsity of logits - through the analysis of entropy as a metric presented in the first chapter - helps us to have a clearer overview. We are now going to enrich the feature extraction function, presented above, in order to implement an entropy analysis.

```
def extract_logit_features(data_loader, text_features):
  image_features_list = []
  labels_list = []

  entropies = []

  for inputs, labels in tqdm(data_loader):
    with torch.no_grad():
```

```
        image_features = model.get_image_features(**inputs)
        image_features = image_features / image_features.norm(
            dim = 1, keepdim = True)

        logits = (image_features @ text_features.T)

        # Softmax Function
        probs = F.softmax(logits, dim = 1)

        # Entripy Anlysis for Sparsity Metrics
        entropy = -torch.sum(probs * torch.log(probs + 1e-9),
            dim = 1)

        entropies.append(entropy.cpu().numpy())

    image_features_list.append(logits.cpu().numpy())
    labels_list.append(labels.cpu().numpy())

features = np.concatenate(image_features_list, axis = 0)
labels = np.concatenate(labels_list, axis = 0)

all_entropy = np.concatenate(entropies)

print(f"Sparsity analysis through mean entropy: {
    all_entropy.mean():.4f}")

return features, labels
```

Since we have 32 generic classes (or prompts), the theoretical maximum entropy - i.e., when the distribution is perfectly uniform - is

$$\log(32) \approx 3.4657.$$

From the entropy analysis described above, we obtain an average entropy of about 3.4656, which is very close to the theoretical maximum. This means that the logits are not sparse at all, but rather broadly distributed. In other words, the images have similar similarity (cosine similarity) with all 32 classes and therefore the model does not distinguish well between the generic classes we have chosen.

Such a result was more than predictable, since the generic text classes are perhaps semantically too similar to each other to be discriminative with respect to the CIFAR-100 images. Even a human could get confused between *"a photo of something round"* and *"a centered object"* when dealing with 32×32 low-level images.

However, we have also tested semantic targeted prompts with a reduced cardinality of 16 classes. The probability distribution of logits obtained by CLIP shows an average entropy that remains systematically close to the theoretical maximum,

$$\log(16) \approx 2.7726,$$

signaling a lack of clear decision between classes again. Following the intuition presented in the work of Balanya et al. [9], which shows a positive correlation between entropy and the need for calibration via temperature scaling, it suggests experimenting with the use of temperatures. In fact, the use of low temperatures could increase the sharpness of the distribution, i.e., amplify the discriminative differences in activation between classes.

Limitations of Linear Classifiers

A linear SVM, though simple and interpretable, may not be expressive enough to exploit the non-linear structure embedded in the logit space. Since CLIP embeddings reside in a high-dimensional, potentially curved manifold, a linear model might fail to capture the true class boundaries. Indeed a linear classifier (like SVM with a linear kernel) does this:

$$f(z) = Wz + b.$$

In other words it can only draw a straight-line (or hyperplane) decision boundaries in feature space. However in our case class-agnostic features are high-dimensional but not linearly separable: the logit features come from cosine similarities with vague prompts and these don't directly correspond to any of the 100 CIFAR-100 labels. So, objects from different classes can have very similar logit vectors, making linear boundaries ineffective. In our support we can consider the work done in [10]. The authors propose a model called CASVM, which involves a CNN enriched with a coordinate attention (CA) mechanism to extract better features and an SVM classifier as a final layer. Experimenting on Fashion-MNIST, CIFAR-10, CIFAR-100 and Animal10 datasets, the work shows that CASVM outperforms standard softmax models, offering better accuracies, higher robustness and lower final loss. Thus the paper confirms that nonlinear features extracted by CNN are often not linearly separable in high-dimensional feature space. CASVM faces the same criticality that occurs in our scenario: making the extracted features as discriminative as possible for classification (thanks to CA).

Low Image Resolution

CIFAR-100 images are only 32×32 pixels, which significantly limits the amount of visual detail available for representation. In fact, there could be a large loss of information in the transition between the CIFAR image and the embedding produced

by CLIP (512-dimensional vector). So, this reduction in resolution may hinder the CLIP encoder's ability to produce semantically rich embeddings, especially when combined with abstract prompts. See how the authors also demonstrate in [11] how fine-grained classification on datasets with low-resolution images suffers a drastic drop in performance, due to the loss of relevant details. Now, they try to solve the problem by integrating a super-resolution module into the pipeline that restores detail and precision. Their model in fact uses a loss that incorporates additional attributes (attributed-assisted loss) to make the features more discriminative and less confusing between similar classes. Evaluating the model on resolutions of 32×32, 64×64, 128×128 and 224×224 they observe how the classification quality increases as the resolution increases.

These findings guide our next steps in improving the class-agnostic setup. In the following section, we explore specific strategies aimed at mitigating these issues, enhancing the quality of extracted features, and ultimately bridging the gap to zero-shot performance.

Chapter 4

CLIP Logits Enhancements

Introduction

In this section we try to mitigate the limitations of the logits representation calculated in Chapter 3 to have more discriminative features needed for a more precise classification. In fact, from the Class-Agnostic experiments conducted so far, it would seem that we are not fully exploiting the semantic power of CLIP in classification contexts. Therefore, it is appropriate to intervene at the quality level of the feature space, trying to enhance the characteristics of the logits both on the image side and on the classifier side.

The following chapter presents some calibration techniques, such as the use of temperature scaling factors, which act directly on the shape of the logits with the aim of recalibrating the naive probabilities of the model. And at the same time it analyzes the use of different architectures, compared to the linear SVM classifier, to significantly increase the effectiveness of logits data, not perfectly separable with hyperplanes. Finally, we observe how the features extracted using CLIP, from datasets containing images at a higher resolution than those used so far, have an impact on the effectiveness of the discriminative power of the logits representation.

4.1 Temperature Scaling

In this section, we analyze the effectiveness of integrating the temperature scaling technique into the CLIP pipeline to move from raw logits to more eliable outputs. The entropy values measured in Chapter 3 report a production of over-confident and poorly calibrated logits. Applying temperature scaling as a post-calibration technique to the logits we obtain a probability distribution of these defined by

$$\hat{p}_k = \frac{\exp(z_k/\tau)}{\sum_{j=1}^{K} \exp(z_j/\tau)}.$$

Now, in our scenario, logits are generated via cosine similarity between visual embeddings and textual prompts, semantically aligned to the class names. However, if the prompts are not perfectly discriminative, the probability distribution of logits tends to be flat, as we have already observed through entropy. In other words, the model assigns similar decision values to many classes. The goal is therefore to observe whether, for $\tau < 1$, this distribution is amplified, thus managing to also sharpen the differences between classes. We also experiment how the use of this technique can influence the geometric space of logit features by observing whether the linear SVM classifier is more able to find hyperplanes that discriminate classes.

Furthermore, of particular interest is to study the behavior of the model for different τ parameters. This analysis allows us to locate a region of values for τ that we can define as cutoff, that is, below which scaling the logits with even smaller temperatures leads to defining a distorted feature space, where the classes appear very different even if semantically they are not. The experimental results obtained are summarized in the following table and plot.

τ	**Mean Entropy**	**Accuracy SVM**
0.50	3.4649	50.62% ↑
0.07	3.4206	62.95% ↑
0.03	3.2099	63.76% ↑
0.02	2.8957	63.45% ↓

Table 4.1: Results by varying the temperature parameter on the CLIP logits

Note that when the temperature drops below τ=0.07 the accuracy of the linear classifier seems to converge towards the upper limit of 64%. There is even a reversal of direction between τ=0.03 and τ=0.01, that is, the tau factor decreases and the accuracy stops increasing. This is a symptom that temperature scaling is no longer useful to the classifier in terms of logit discrimination. We have therefore identified the cutoff region.

When $\tau < 0.07$, even small differences in the similarity values z_{ij} (obtained by cosine similarity between visual embedding i and text embedding j) are exaggerated, inducing a strongly peaked and overconfident output distribution. This phenomenon results in distortion of the feature space. Since the discrimination is driven by a numerical artifact (the low value of τ), the geometric relations between embeddings are distorted: semantically close classes (e.g. "*animal*" vs "*creature*") appear artificially distant in the logit space.

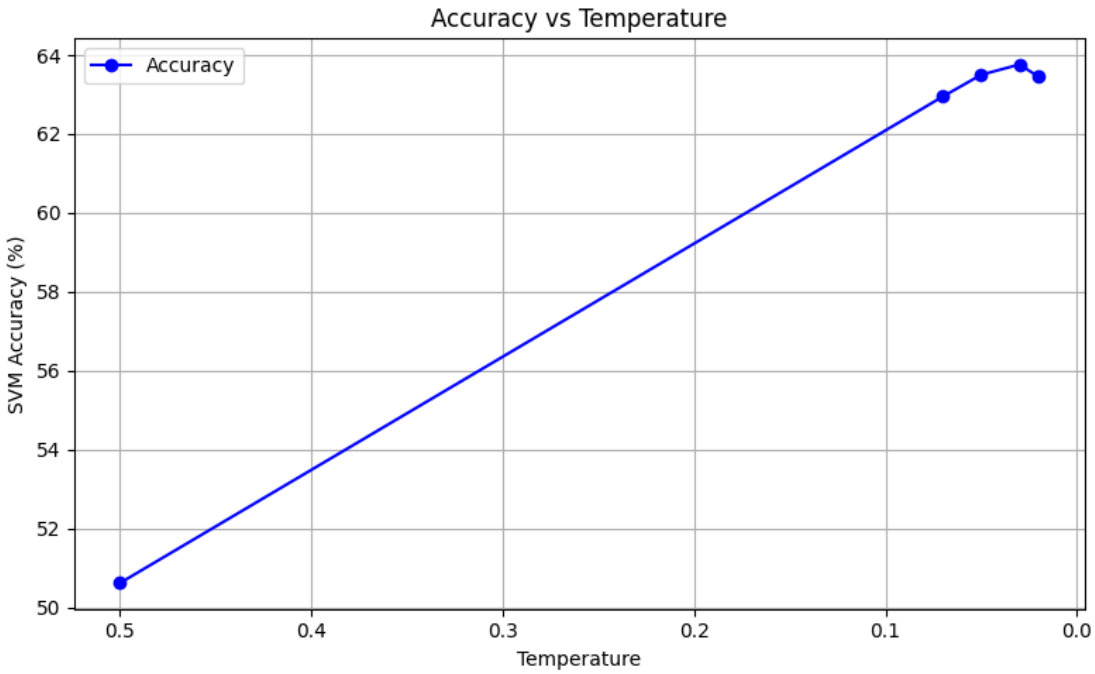

Figure 4.1: Relation between temperature and SVM accuracy

4.2 MLP

4.3 High-quality image dataset

General Conclusion

Writing a general conclusion involves summarizing the main points of your work, reflecting on its significance, and offering any final thoughts or recommendations. Here's a structured approach to writing a general conclusion:

1. Begin by summarizing the key findings or results of your work. Highlight the most important discoveries, insights, or conclusions that you have reached throughout your project or study.

2. Remind readers of the objectives or goals you set out to achieve at the beginning of your work. Discuss how well you have met these objectives and whether you have successfully addressed the problems you identified.

3. Acknowledge any limitations or constraints of your study. Discuss any challenges you encountered, such as methodological limitations, data constraints, or unexpected obstacles, and how these may have affected your results or conclusions.

4. Identify areas for future research or further investigation based on the findings of your work. Suggest potential approaches that could improve your work and open further perspectives.

Bibliography

[1] OpenAI, "Clip: Contrastive language–image pre-training," https://github.com/openai/CLIP, 2021.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[3] G. Goh and et al., "Clip2latent: Towards zero-shot image-to-image retrieval," *arXiv preprint arXiv:2301.09662*, 2023.

[4] W. Zhang, X. Wang, T. Xu, Y. Wu, and J. Yang, "Contrastive learning is not just for cross-modal: Improving clip with intra-modal alignment," *arXiv preprint arXiv:2304.02643*, 2023.

[5] G. Strang, *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 1993.

[6] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT press, 2016.

[9] D. Balanya *et al.*, "Adaptive temperature scaling for confidence calibration in deep learning," *arXiv preprint arXiv:2206.XXXX*, 2022.

[10] Q. Tan, Z. He, X. Wang, S. Pan, L. Li, and X. Wang, "CASVM: An Efficient Deep Learning Image Classification Method Combined with SVM," *Applied Sciences*, vol. 12, no. 22, p. 11690, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/22/11690

[11] K. Singh, R. Agarwal, and C. Jawahar, "Enhancing fine-grained classification for low resolution images," *arXiv preprint arXiv:2105.00241*, 2021. [Online]. Available: https://arxiv.org/abs/2105.00241