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:
Prof. 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.

Statement

We declare that the original version of this thesis was written in the Italian version.

Use of the ChatGPT generative chatbot model was made to help in the generation and revision of some pieces of code, generation of prompt sets used during the experiments, as well as in the translation of some passages into the English language.

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 vocabulary expressiveness. We experiment with temperature scaling, prompts defined by a larger vocabulary, and high resolution dataset 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 could contribute 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

In the context of multimodal learning models, the emergence of joint learning from visual and textual data arises. The CLIP model is a suggestive example: it attempts to align images with their respective textual descriptions within a shared embedding space, through a contrast loss.

In this way, CLIP enables classification tasks such as zero-shot, where the model is able to recognize never-before-seen classes based on prompts defined through natural language. All this without requiring any ad hoc refinement of the model.

However, if CLIP works well using semantically rich prompts, its behavior under a class-agnostic setting - i.e. where textual prompts are not aligned to the labels of the classes - remains less explored.

Therefore, we will act by analyzing the behavior and functionality of CLIP in contexts in which this model does not seem to excel. The aim of this thesis is therefore to understand whether the embeddings produced by CLIP are semantically structured in such a way as to favor the separability of the classes, even in the absence of explicit supervision.

0.2 Objectives and Goals

This thesis aims to analyze the representational quality of CLIP's image embeddings through an analysis 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.

- 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 to improve classification accuracy.

0.3 Methodology

This study is experimental and implementation-driven, combining supervised and unsupervised evaluations to assess the quality of CLIP's embeddings 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. We will use standard image classification datasets such as CIFAR-10 and CIFAR-100 for our experiments.
2. **Baseline Supervised Classification**: CLIP's image embeddings are extracted and used as input features for classical classifiers (e.g., 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 of CIFAR dataset, 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. **Diagnostic Analysis and Performance Optimization**: different strategies are tested to improve class-agnostic classification accuracy, after analyzing the main associated problems. We will argue about:

- Temperature scaling to adjust the sharpness of cosine similarity distributions.
- Prompt refinement for more consistent and object-centric descriptions.
- Use of high-resolution datasets like Tiny ImageNet 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.

7. **Clustering**: unsupervised learning about image feature classification and class-agnostic logit representation experiments.

All experiments were implemented in Python using PyTorch and HuggingFace Transformers, using Google Colab as development environment. 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

CLIP (Contrastive Language-Image Pretraining) is a multimodal pre-trained model that attempts to semantically align images and natural language descriptions within a shared embedding space. It structurally involves two neural networks: a Vision Transformer (ViT) to encode images and a second model for encoding texts, also based on Transformer technology. In detail, these networks map the tensors representing images and texts into embeddings of fixed length d (typically $d = 512$) that the CLIP model jointly aligns through a contrastive loss InfoNCE (Noise-Contrastive Estimation).

We are saying that CLIP has therefore been perfected for what is known as *inter-modal alignment*, that is, it aligns well pairs of images and texts. On the other hand, using such a contrastive loss, the model may be inefficient if we consider an *intra-modal alignment*, that is, if we ask it to associate for example semantically similar images. Some studies, as described in [2], suggest a technique called *modality-inversion* to fill this gap.

Anyway, mathematically speaking, 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.

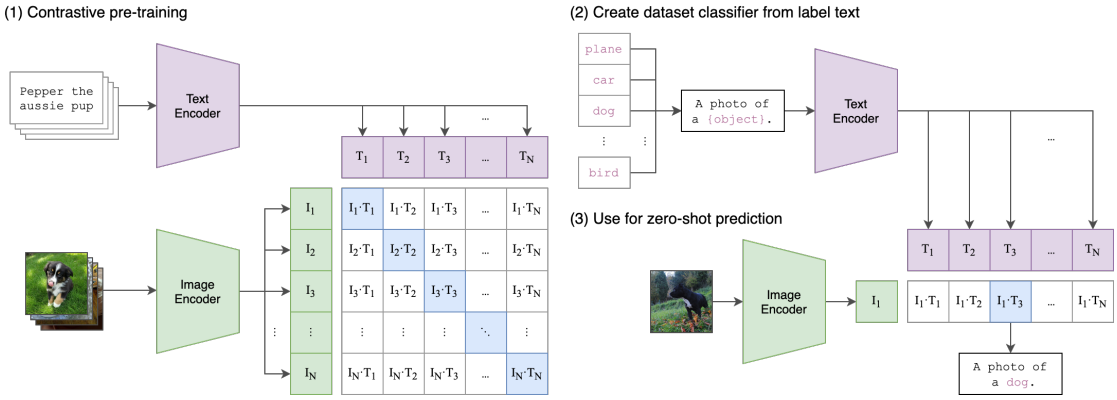

Figure 1.1: Architectural structure of the CLIP model. Image adapted from [1]

1.2 Mathematical Reminders

In this section, a series of mathematical requirements are presented that are useful for better understanding the operational structure of CLIP and its contrastive learning paradigm. A reminder of the linear classification and the accuracy metrics of the model's predictions is also presented. These reminders constitute a good basis for being able to move consciously within the experiments performed.

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). In particular, CLIP accepts as input tensors representing images in the space $\mathbb{R}^{3\times224\times224}$, so for all images in the dataset that do not respect this constraint, a resizing is necessary. The procedure for textual descriptions is different: these must first be tokenized so that, for example, *"a photo of a lion"* becomes a sequence of indices $\{t_1, t_2, \ldots, t_n\}$ of the vocabulary, with $t_i \in \mathbb{Z}$.

At this point then 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 $f_{\text{img}}(x) \cdot f_{\text{text}}(y) \in [-1; 1]$, where 1 indicates perfect similarity. In the context of CLIP, these quantities are called logits.

Contrastive Learning

The model is trained using a contrastive loss, specifically the symmetric InfoNCE loss. 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 final value $\in [0, log(N)]$ where 0 indicates the maximum similarity between image and text. Such symmetric InfoNCE loss has proven effective in learning joint embeddings for vision and language, as described as well in [3].

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. For image-to-text, the loss is:

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

Now let's consider a Batch Size = 2 where:

- $\text{sim}(\text{Img}_1, \text{Text}_1) = 0.746$
- $\text{sim}(\text{Img}_1, \text{Text}_2) = -0.232$
- $\text{sim}(\text{Img}_2, \text{Text}_2) = 0.710$
- $\text{sim}(\text{Img}_2, \text{Text}_1) = -0.250$

Assuming $\tau = 0.07$:

$$\text{sim}_{1,1}/\tau = \frac{0.746}{0.07} \approx 10.657, \quad \text{sim}_{1,2}/\tau = \frac{-0.232}{0.07} \approx -3.314$$
$$Z_1 = \exp(10.657) + \exp(-3.314) \approx 42799.7 + 0.036 = 42799.736$$
$$\mathcal{L}_1 = -\log\left(\frac{\exp(10.657)}{Z_1}\right) = -\log\left(\frac{42799.7}{42799.736}\right) \approx -\log(0.99999916) \approx 8.4\times10^{-7}$$
$$\text{sim}_{2,2}/\tau = \frac{0.710}{0.07} \approx 10.143, \quad \text{sim}_{2,1}/\tau = \frac{-0.250}{0.07} \approx -3.571$$

$$Z_2 = \exp(10.143) + \exp(-3.571) \approx 25478.5 + 0.028 = 25478.528$$

$$\mathcal{L}_2 = -\log\left(\frac{\exp(10.143)}{Z_2}\right) = -\log\left(\frac{25478.5}{25478.528}\right) \approx -\log(0.9999989) \approx 1.1 \times 10^{-6}$$

Of course, real batches are more noisy and larger, but this illustrates the principle.

Classification and Linear Models

Given feature-label pairs (z_i, y_i), we consider a linear model:

$$f(z) = z^\top W + b,$$

where $W \in \mathbb{R}^{d \times C}$, C is weitgh matrix, and $z \in \mathbb{R}^d$ is an embedding (CLIP's output). The classifier must then find W and b that fit well for the concepts already separated in the embedding space. In particular, each column of W is a normal vector to a decision hyperplane for a class, i.e., $W_c \in \mathbb{R}^d$ corresponds to the preferred "direction" of the class c. The scalar product $z^\top W_c$ instead represents how compatible the embedding z is with that class.

Evaluation Metrics

In order to get an idea of how the model behaves in the different experiments that we will perform, we now introduce some metric measures.

Classification Accuracy. This is the primary metric used throughout the experiments, especially for measuring the classifier performance. 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.

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.

Confusion Matrix. For a deeper class 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. Such analysis could tell us how clip embeddings are able to capture the semantics of certain image-text pairs compared to others.

1.3 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 Tiny ImageNet, a high-resolution subset of ImageNet, is used to check 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.
- **Scikit-learn:** Used for training traditional classifiers (e.g., Support Vector Machines) 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.

1.4 Expected Outcomes

Following the methodology presented so far in this thesis, we expect to understand how a multimodal model such as CLIP creates, through its knowledge acquired during its pre-training, embeddings that have a discriminative power. If this power, in the context of class-agnostic settings, is not sufficient to achieve the performance in the benchmark experiments, we then hope to apply refinement techniques that aim to increase the accuracy of the results.

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` version 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 and train a linear Support Vector Machine (SVM) classifier on top of these features.

This setup allows us to check 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 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 inter-modal representational structure, setting the bases 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-10 and 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-10/CIFAR-100 dataset is loaded using `torchvision` and wrapped in a custom dataset class to ensure each image is preprocessed appropriately ($3 \times 224 \times 224$ shape) 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 (i.e. aims to achieve MEAN = 0 and STANDAR DEVIATION = 1 for each 512-dimensional feature), 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 a key factors: increasing the variability of classes, through categories that are semantically and visually similar, makes discrimination difficult if embeddings are used without any fine-tuning. Anyway we have to remember that CIFAR standard dateset uses images of $[3 \times 32 \times 32]$ (RGB channels, height, width), making difficult, also for a model like CLIP, maintaining fine-grained visual information from the transformation $[3 \times 32 \times 32] \longrightarrow [3 \times 224 \times 224] \longrightarrow$ CLIP's embedding.

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 (a human could get wrong too!).

Overall, 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.

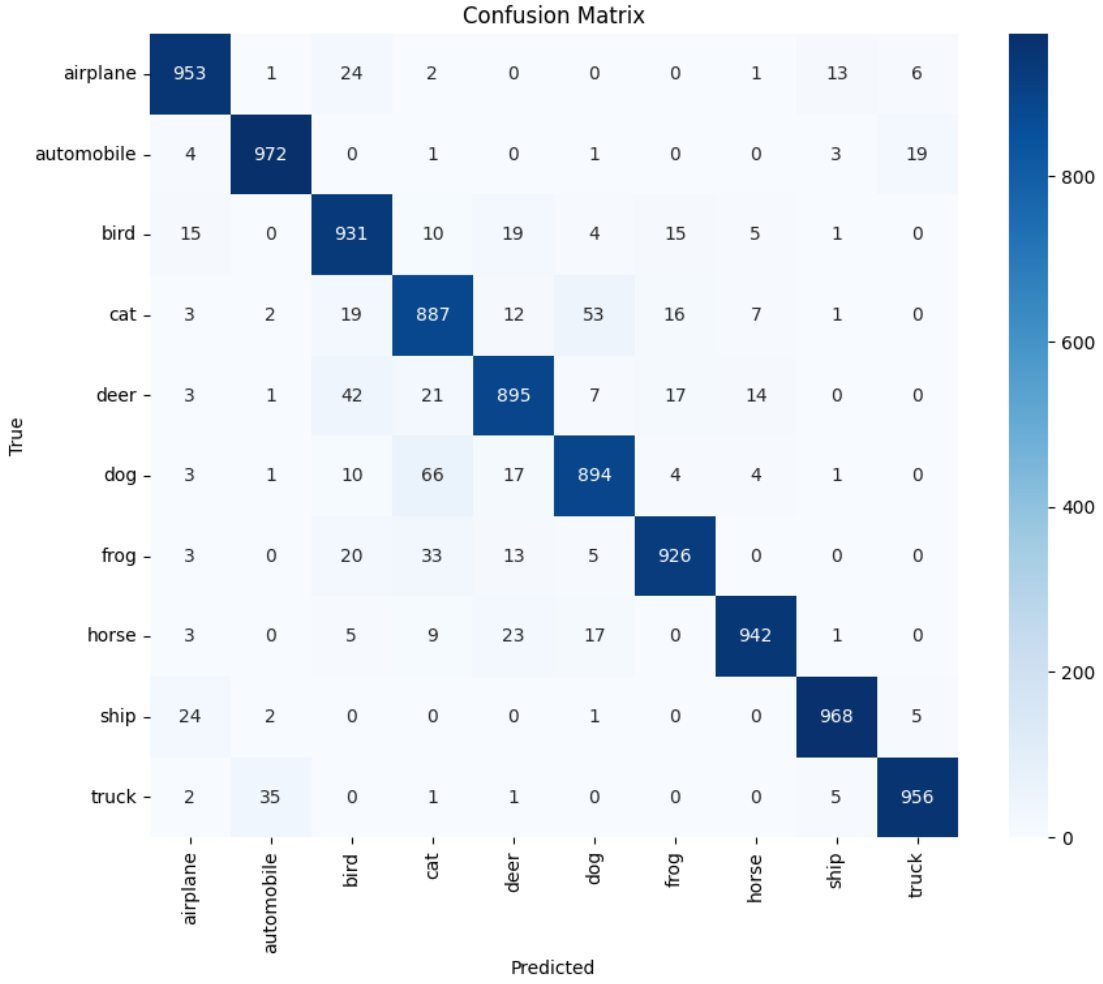

Figure 2.1: Confusion Matrix with CIFAR-10 - Baseline Classification.

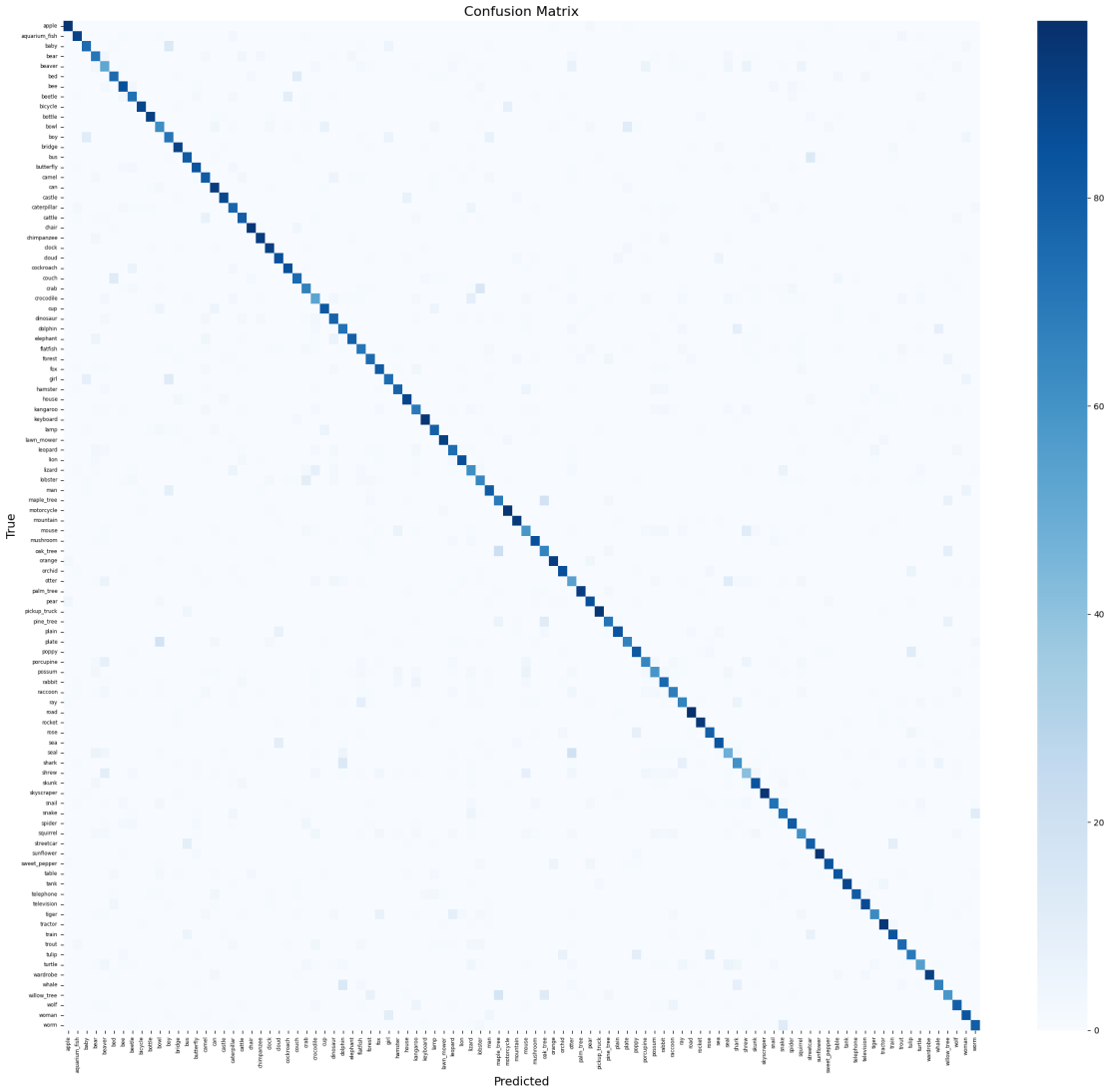

Figure 2.2: Confusion Matrix with CIFAR-100 - Baseline Classification.

2.3 Zero-Shot Classification

We know that CLIP is a pre-trained model on millions of images/texts in order to find the maximum similarity between the image-text pair. One of the most interesting features is to be found in what is called zero-shot prediction classification task. In other words, using CIFAR as a base dataset, we will extract the features associated with the typical descriptions of the labels of that dataset ("*a photo of a* ..."). Then we will compare them with those extracted from the images of the same dataset with the aim of evaluating the cosine similarities. At this point we analyze

how CLIP manages to correctly predict the classes for an inter-modal classification. All this despite the fact that the model has never seen elements of CIFAR during its training. Zero-Shot prediction has to do with the fact that there is no need to train anything, but we are just providing CLIP with an image and a list of possible texts, and we exploit the similarity of the respective extracted features to predict. Obviously, the similarity is more truthful if the space of embeddings produced by CLIP is quite informative and discriminating.

2.3.1 Experiment Setup

Following the baseline classification experiment, we evaluate CLIP's zero-shot capabilities on CIFAR-10/CIFAR-100. CIFAR consists of semantic categories. To align with CLIP's training paradigm, we construct prompts 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 text encoder:

```
text_prompts = [f"a photo of a {label}" for label in
    class_names]

# prompts tokenizing
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: indeed in light of these results, we can state that CLIP is quite good at creating semantic representations that can discriminate between general categories, although it generates confusion when these are extracted from low-resolution datasets and evaluated with lightweight classifiers such as SVM.

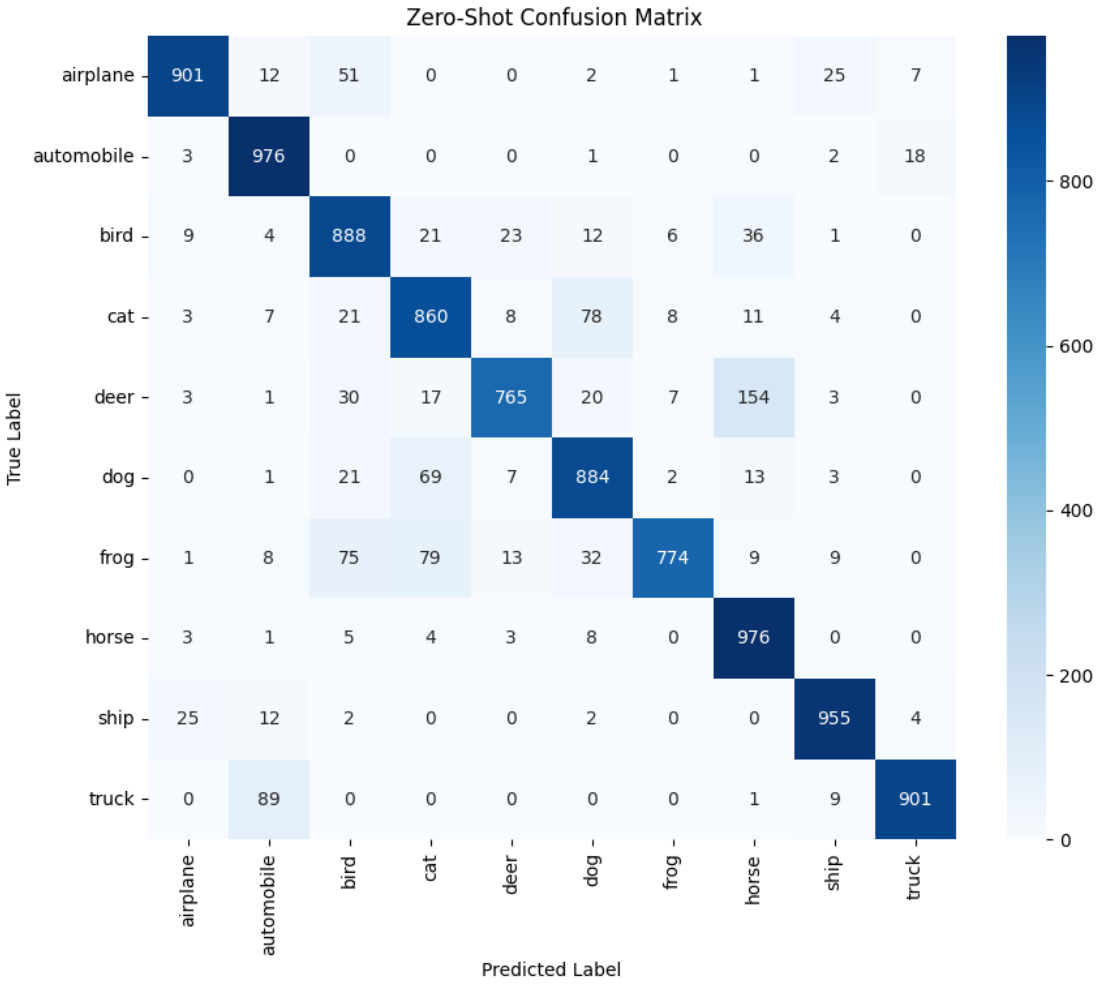

Figure 2.3: Confusion Matrix with CIFAR-10 - Zero-Shot Classification.

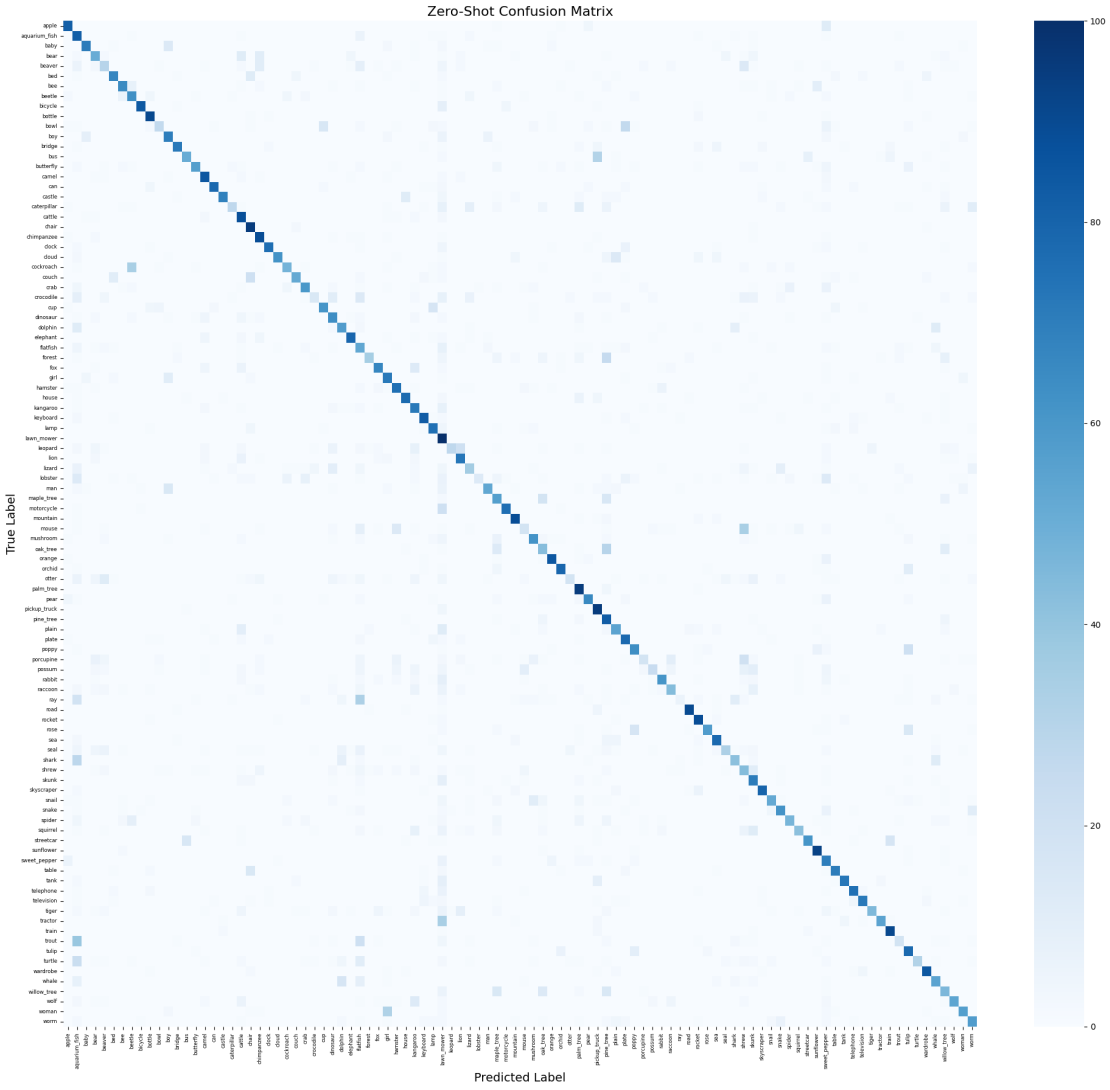

Figure 2.4: Confusion Matrix with CIFAR-100 - Zero-Shot Classification.

Chapter 3

Class-Agnostic Techniques

3.1 Introduction

In this chapter we explore the behavior of the CLIP model under a class-agnostic setting. This is a very useful approach to understand the representational structure of embeddings. In fact, the core of this experiment consists in extracting similarities between images and a set of generic prompts, whose descriptions are semantically similar to the real classes of the CIFAR dataset, then training a linear SVM classifier on these features. This contrasts with the baseline experiments in Chapter 2, where classification was performed directly on CLIP's image embeddings.

In the context of models like CLIP, the term logits refers to raw cosine similarity scores. A class-agnostic logit representation setting, as specified, attempts to use these scores as features, all without constraining the model to know the true classes of the reference dataset.

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?

Again, 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.

3.2 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.

3.2.1 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:

```
# prompt tokenizing
text_inputs = clip_processor(prompts, return_tensors="pt",
    padding=True).to(device)

with torch.no_grad():
    text_features = clip_model.get_text_features(**
       text_inputs)

text_features = text_features / text_features.norm(dim=-1,
    keepdim=True)
```

3.2.2 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)

            # evaluating cosine similarity
            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
```

3.2.3 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}%")
```

3.2.4 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 baseline image classification (~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.

This experiment highlights a critical tension in using pretrained vision-language models for fine-grained classification, although CLIP still manages to align, at least roughly, semantics between images and descriptions even in unsupervised contexts.

Looking ahead, this performance 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.

3.3 Analyzing the Limitations of Class-Agnostic Logits Representations

We saw class-agnostic strategy produced an accuracy of 29.30%; this percentage 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)

      # cosine similarity
      logits = (image_features @ text_features.T)

      # Softmax Function for transforming logits into
         normalized probabilities
      probs = F.softmax(logits, dim = 1)

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

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

```
22
23      image_features_list.append(logits.cpu().numpy())
24      labels_list.append(labels.cpu().numpy())
25
26  features = np.concatenate(image_features_list, axis = 0)
27  labels = np.concatenate(labels_list, axis = 0)
28
29  all_entropy = np.concatenate(entropies)
30
31  print(f"Sparsity analysis through mean entropy: {
        all_entropy.mean():.4f}")
32
33  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.

We have also tested semantic targeted prompts with a bigger cardinality of 99 classes. The probability distribution of logits obtained by CLIP shows an average entropy that remains systematically close to the theoretical maximum,

$$\log(99) \approx 4.5951,$$

signaling a lack of clear decision between classes again. Following the intuition presented in the work of Balanya et al. [4], 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:

$$f(z) = z^\top W + b,$$

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 [5]. 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).

That said, a simple MLP (Multilayer Perceptron) architectural model could already be able to discriminate better within the high-dimensional space of embeddings.

Limitations of Prompts Vocabulary

When using a limited and not very varied set of textual prompts, there is a risk of not covering well the various semantic nuances of the visual concepts contained in the CIFAR-100 images, therefore the embeddings extracted from these do not find an easy alignment with the chosen descriptions. Increasing the variety of the vocabulary, that is, using more prompts, but above all more descriptions for the same concepts could help to achieve greater information separation in CLIP embeddings.

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 [6] 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

4.1 Introduction

In this chapter 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.

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 cardinality of prompts with greater variety of vocabulary, to significantly increase the effectiveness of logits data. 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.

4.2 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 under-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.

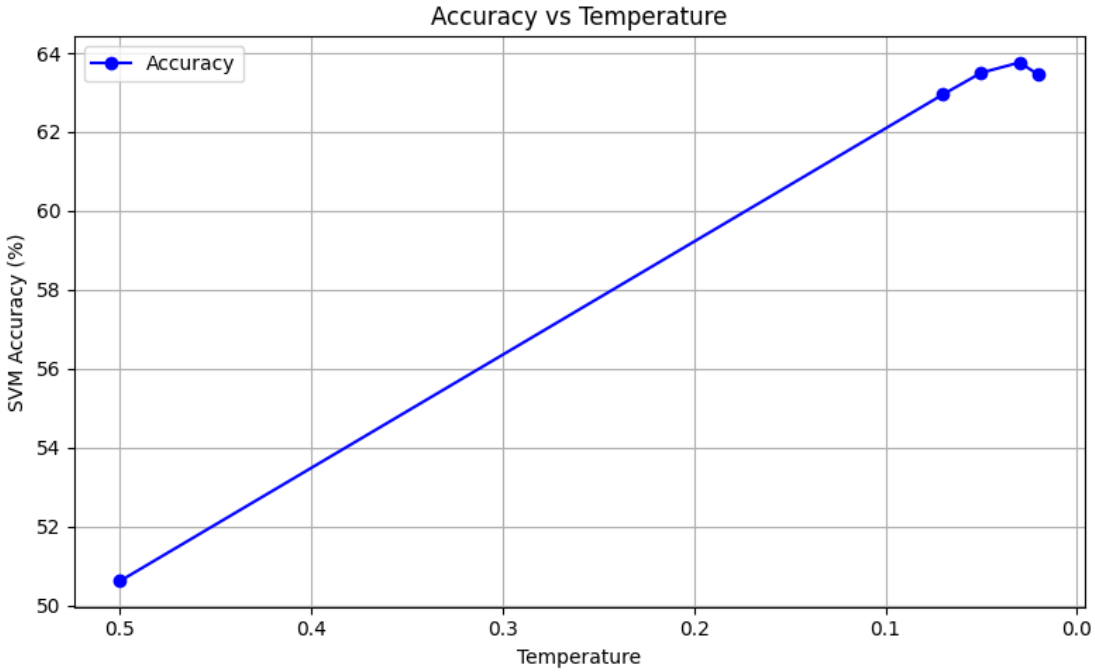

Figure 4.1: Relation between temperature and SVM accuracy.

4.3 Increasing Prompts Vocabulary

We know that CLIP projects images and texts into a shared vector space through embeddings. We can imagine that with few prompts, the space of text projections is very limited and therefore images are described along only a few "semantic directions". So by increasing the number and diversity of prompts, we increase the number of these directions. And we hope to obtain a greater ability to distinguish different classes. This is because, by increasing the number and vocabulary of prompts, we increase the probability that each class in the dataset is relevant to at least some of these prompts, thus improving the quality of the features for the linear classifier.

Having established this, we repeated the class-agnostic logit representation experiment with a progressive set of prompts of cardinality 99, 205, 415. Each prompt has the constraints of not corresponding exactly to the real CIFAR-100 classes, maintaining descriptiveness and visual semantics, being diversified by subject, shape, composition, positions, etc. The experimental results obtained are described in the following table.

Prompt Set	Mean Entropy	SVM Accuracy
32	3.4656	29.30%
99	4.5950	45.45%
205	5.3299	54.59%
415	6.0282	62.86%

Table 4.2: Experimental results on CIFAR-100 using image-text similarity logits obtained by CLIP with different sets of generic prompts.

Observing the clear increase in accuracy (even higher than zero-shot on CIFAR-100 when the prompt set has cardinality 415), we can think that the logit vectors have become richer and less redundant. As a result, we have more discriminative features and similar classes like "*truck*" vs "*car*" could start to have more distinguishable projections.

It is then observed that the improvement between the different sets of prompts is not linear, as expected in saturable systems. In fact, between $99 \longrightarrow 205$ prompts there is an increase in accuracy of $+9.14\%$, while between $205 \longrightarrow 415$ of $+8.2\%$. This means that the model benefits from the semantic expansion, but begins to saturate, that is, the new prompts add increasingly marginal information or are already correlated to the existing ones.

4.4 High-quality image dataset

We have already outlined that one of the limitations in achieving poor performance in the class-agnostic baseline experiment is the poor image quality of CIFAR-100. This leads to receiving very "compressed" input from CLIP, leading to the loss of most of the relevant visual characteristics.

For this reason we tested the same initial class-agnostic experiment, using higher resolution datasets. Specifically, we relied on 64×64 resolution Tiny ImageNet (from which we filtered 100 of the 200 generic classes provided) and 512×512 Food101 (101 food classes like "*apple_pie*", "*beef_carpaccio*", "*churros*"). For the Food101 dataset we repeated the experiment with both the 32 generic CIFAR-based generic prompts and with the new 32 food-based prompts, like "*a photo of baked dessert*", "*a photo of a cold drink*", etc.

The experimental results obtained are presented in the following table.

Prompt Type	Dataset	Mean Entropy	SVM accuracy
Generic/CIFAR-based	Tiny ImageNet	3.4656	35.62%
Generic/CIFAR-based	Food101	3.4656	41.25%
Food-based	Food101	3.4654	62.47%

Table 4.3: Experimental class-agnostic result using higher resolution datasets.

We can immediately see how moving from CIFAR-100 (32×32) to Tiny ImageNet (64×64) results in more expressive CLIP embeddings. The thesis that better image resolution quality leads to more information in the embedding space, is strengthened by the fact that using Food101 (512×512) with CIFAR-based prompts achieves an accuracy higher than 29.30% (class-agnostic baseline), which is extraordinary! Furthermore, when we use Food101 with semantically aligned prompts, a percentage of 62.47% is achieved, even higher than zero-shot (61.71%). On the other hand, we know that CLIP was trained on a massive and highly varied dataset, called *WebImageText* (WIT), which provides an average variable quality that is still higher than that of CIFAR.

Chapter 5

Clustering

5.1 Introduction

As a last experiment we will do clustering, so we are no longer talking about supervised classification as with SVM, but about unsupervised analysis. In fact, this approach does not require knowing in advance what the "clusters" are, but the algorithm used must discover them on its own. This technique is useful for discovering natural patterns in the data produced by CLIP embeddings.

There are several clustering algorithms, which are based on the structure of the data and the objectives of the analysis. In our case we are working with CLIP embeddings, i.e. normalized vector which reside in a 512-dimensional space and our goal is to have an overview of how semantically structured these are already, even without supervising anything. We therefore plan to use the `K-means` partitional algorithm which is based on the a priori choice of the number K of clusters to create. Once K centroids are then randomly chosen, the algorithm assigns each observation to the cluster whose centroid is closest and updates the centroids by calculating the average of the observations assigned to each cluster.

The clustering accuracy metric we use is purity, defined as

$$\text{Purity} = \frac{1}{N} \sum_{k} \max_{j} |C_k \cap L_j| \in [0, 1]$$

where C_k are the images in the cluster k, L_j are the images belong to the true class j and N is the total images number. In other words this is a metric that measures how homogeneous the clusters found are compared to the true labels.

5.2 Baseline and Class-Agnostic Clustering

Starting from the clustering on the features extracted from the Baseline Image Feature Classification experiment, we group the images without using any labels, then evaluating how much the clusters found correspond to the real classes. The pipeline used involves using `K-Means` on the image features already extracted, evaluating the clusters found by comparing them with the original labels, then calculating the purity.

The same pipeline is also used for Class-Agnostic, where however now we will use the logits vectors to do clustering. We will test purity both when extracting features with the original 32 generic-based prompts and with the more specific set of 415 prompts with a greater expressive vocabulary.

```
def purity_score(y_true, y_pred):
    # we define a matrix [key, value] where key is the ID of the cluster and value is the
    # list of the associated true lables to the cluster element
    # example: y_true = ["cat", "dog", "cat", "dog", "dog"]
    #          y_pred = [0, 1, 0, 1, 1]

    # truelabel_matrix {
    #   0: ["cat", "cat"],
    #   1: ["dog", "dog", "dog"]
    # }

    truelabel_matrix = {}
    for true, pred in zip(y_true, y_pred):
        truelabel_matrix.setdefault(pred, []).append(true)

    total = 0
    for cluster in truelabel_matrix.values():
        most_common_label = Counter(cluster).most_common(1)[0][1]
        total += most_common_label
    return total / len(y_true)

# clustering
def cluster_and_evaluate(features, true_labels, n_clusters):
    print(f"\nRunning KMeans with {n_clusters} clusters...")

    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
```

```
    pred_labels = kmeans.fit_predict(features)

    purity = purity_score(true_labels, pred_labels)

    print(f"\nClustering evaluation:")
    print(f"Purity: {purity:.4f}")

    return pred_labels, purity
```

5.3 Clustering Insights

Testing the functions above we ended up in the following clustering table results:

Prompt Set	Features	Purity
32	Baseline Image	0.4455
32	Class-Agnostic	0.2976
415	Class-Agnostic	0.3321

Table 5.1: Experimental clustering results.

We observe that the results seem to indicate that CLIP's pure image embeddings are semantically structured. In fact we can consider a purity of 0.45 out of 100 clusters as a decent result, given that CIFAR-100 contains 100 classes that can easily overlap like "*truck*" or "*car*". We can state that CLIP's image features alone already contain a semantic structure that is quite sufficient to discriminate. While in the case of Class-Agnostic with 32 generic prompts there is a collapse of purity. This suggests that features extracted with CLIP are more confusing and less informative for distinguishing classes. So generic prompts do not help the model discriminate similar classes. However, by increasing the number to 415 and variety of prompts, we provide more context for CLIP to refine embeddings.

General Conclusion

From the results of the experiments conducted, it is observed that CLIP has semantically rich visual embeddings, which allow high performance in both supervised and zero-shot classification.

However, we have found that, in a class-agnostic setting, their usefulness strongly depends on the linguistic quality, quantity and semantic coherence of the prompts. We also achieved significant improvements with temperature scaling techniques and large, well-designed prompt sets.

Clustering also confirms that pure image embeddings are intrinsically more semantically organized than features obtained via generic prompts. All this can shed light on how deeply CLIP actually "understood" visual semantics during its pretraining of millions of images extracted from the web.

Bibliography

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

[2] M. Mistretta, A. Baldrati, L. Agnolucci, M. Bertini, and A. D. Bagdanov, "Cross the gap: Exposing the intra-modal misalignment in clip via modality inversion," *arXiv:2502.04263*, Feb. 2025, accepted for publication at ICLR 2025. [Online]. Available: https://arxiv.org/abs/2502.04263

[3] 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.

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

[5] 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*, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/22/11690

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