HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Leveraging Pretrained Unimodal Models for Efficient Vision-Language Pretraining

Tim Cares

Master's thesis in Applied Computer Science

9. Oktober 2024

Autor Tim Cares
Matrikelnummer
Email Adresse

Erstprüfer Prof. Dr. Vorname Name
Abteilung Informatik, Fakultät IV
Hochschule Hannover
Email Adresse

Zweitprüfer Prof. Dr. Vorname Name
Abteilung Informatik, Fakultät IV
Hochschule Hannover
Email Adresse

I hereby declare that I have written and submitted this thesis independently, without any external help or use of sources and aids other than those specifically mentioned by me. I also declare that I have not taken any content from the works used without proper citation and acknowledgement.

Hannover, 9. Oktober 2024 Tim Cares

Acknowledgements

Abstract

Multimodal models, especially vision-language models, have gained increasing popularity due to their wide range of applications on both unimodal and multimodal tasks. However, existing approaches often require large-scale models, extensive data, and substantial compute, limiting their accessibility for smaller research groups and individuals. This thesis address this issue by introducing an efficient self-supervised vision-language model that is significantly cheaper to train and smaller in size. We leverage pretrained unimodal encoders and introduce a randomly initialized shared encoder to align representations using a contrastive loss function. A self-supervised image model is employed for simultaneous knowledge distillation, guiding the alignment through high-level image representations. Our proof-of-concept demonstrates competitive performance with popular vision-language models like CLIP and FLAVA on retrieval tasks, outperforming them on certain metrics while using only 0.75% of the data used by CLIP and 4.3% by FLAVA. These finding underscore the potential for designing efficient multimodal models, and therefore lay the foundation for future research on financially accessible models, promoting broader participation in multimodal learning. To promote transparency and facilitate further research, we have made our code for training and evaluating our model publicly available.

Contents

1 Introduction

1.1 Motivation

Supervised Learning has been the fundamental technique for training deep learning models since the development of the backpropagation algorithm. This is not surprising, as it provides a model with clear, domain-specific learning signals, making it is the most direct and efficient way to solve problems or learn specific tasks.

However, the development of artificial intelligence using labeled data has its limitations. As we aim to teach machine learning models increasingly complex tasks, more labeled data is required. Naturally, generating millions of labeled examples becomes more difficult as tasks grow in complexity, making the development of such models less feasible.

Due to these challenges, **self-supervised learning** has gained increased attention from the scientific community in the last few years. The approach does not rely on creating labeled data by hand, e.g. through human annotation, but instead derives them from the data itself. For example, to train a large language model (LLM) to understand written text, words in a sentence are masked or deleted, and the model's task is to predict these missing words.

Figure 1: An example for creating labels out of raw data. Any word in a sentence can potentially be masked, which is why training examples (including labels) can be created just from the data itself, without any human annotation. The labels for this example would be "moon" and "bright" (adapted from [LM21]).

This has three advantages: First, labels do not need to be created by hand, it is straightforward to randomly mask words in a sentence and use them as targets during training. Second, massive amounts of text available on the internet enable the generation of massive training datasets. Last, and most important, the model learns to understand and represent the world we live in. This becomes clear with the example shown in Figure 1. Here, the model must predict the words "moon" and "bright" based on the context/words remaining after masking. To do so successfully, the model must learn that at night, it is the moon that shines, not the sun, and that if the moon shines, it is usually bright. This example illustrates an important

Figure 2: The same real-world concept, expressed in different modalities. Most AI models are not able to understand that both expressions represent the same concept. Data comes from the COCO test set [Lin+14].

characteristic of self-supervised learning: It encourages the model to learn common sense and the world that we humans live in.

As a result of such self-supervised training, the model learns to represent input data in a way that it can interpret and understand, typically as an n-dimensional vector, which is a process known as **representation learning**.

This is practical on tasks like text classification or image classification, where knowledge gained from self-supervision can be transferred to downstream tasks. For instance, if a model has learned to extract the meaning of a sentence and express it as a representation, then it becomes significantly simpler to classify the sentiment of the sentence using this representation.

1.1.1 Multimodality

While representation learning has seen success in areas like computer vision and natural language processing, creating interpretable representations of the respective input data, these models often lack the ability to understand that real-world concepts are not bound to the method in which they are expressed. Consider Figure 2: For a human, both the image and the text express the same concept, a sheep on grass. To us, it does not matter whether this concept is expressed in language or in an image, we understand that both represent the same idea. AI models, however, usually do not possess this ability.

Most AI models are **unimodal**, meaning they can only understand one type of data, e.g. either text or images, but not both. An image model, for instance, is able to reliably generate a representation of the image in Figure 2, but cannot process the text. While a text model can generate a representation of the text, this representations will not be the same as the one generated by the image (from the image model). This is because, although both modalities express the same concept, the models have been trained independently and have no incentive to produce matching representations for the same concept, expressed in the other respective modality.

However, if the goal of representation learning is to capture the underlying concept, then we would expect the representations to be the same (or at least very similar), regardless of

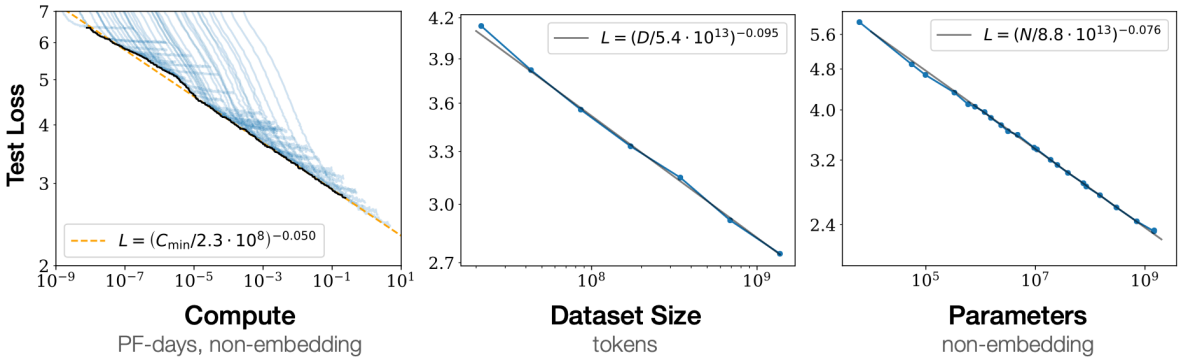

Figure 3: Scaling compute, model parameters, and dataset size for large language models shows a continuous improvement in performance. Compute and parameters required for embedding tables is not considered, and "PF-days" refers to the number of petaFLOP-days required to train the model. Figure is taken directly from [Kap+20].

the modality in which the concept is expressed. Again, Figure 2 shows the same concept in different modalities, so the representations should be aligned. This is achieved through **multimodal representation learning**, where "multimodal" refers to the ability of the model to understand and process multiple types of data, such as text and images.

1.1.2 Efficiency

In recent years, the idea of multimodal representation learning has received increased attention from the scientific community. Methods like OpenAI's CLIP [Rad+21], Meta's FLAVA [Sin+21], or Microsoft's BEiT-3 [Wan+23] have shown that models are able to learn a joint representation of multiple modalities, in this case text and images. However, these models, and deep learning models in general, have also become increasingly large and computationally expensive to train. This trend can be partially attributed to the neural scaling laws formulated by researchers from OpenAI [Kap+20] (seen in Figure 3).

Moreover, multimodal models generally require more parameters than their unimodal counterparts, as they need to process multiple types of data. Even though image and text representations should be aligned for the same concept, both modalities are inherently different and require different processing steps. Consequently, multimodal models are usually larger and more computationally expensive to train than unimodal models.

While this is not a problem for large enterprises or research institutions like OpenAI, Meta, or Microsoft, it poses a challenge for smaller research groups or individual researchers. They rely increasingly on the progress made by large institutions, as they do not have the resources to train such large models themselves. This situation limits the research on multimodal representation learning that can be conducted by smaller groups and individuals, thereby reducing the diversity of research in the field.

1.2 Goals and Contributions

Given the challenges associated with training large multimodal models from scratch, we identify the opportunity to utilize the vast amount of existing pretrained unimodal models. By extracting and combining knowledge from these models, we can generate multimodal models, for the proof-of-concept developed in this work vision-language models, that are *smaller* in size and significantly *cheaper* to train. Importantly, the approach is not limited to vision and language, meaning it can be adapted to other combinations of modalities wherever pretrained unimodal models are available.

Pretrained unimodal models, such as those for computer vision and natural language processing, have already learned rich representations within their respective domains through extensive (pre-)training. By aligning and integrating these representations, we can create a multimodal model without the need for large-scale training from scratch. This methodology can be extended to other modalities, such as audio, video, by leveraging existing pretrained models in those domains.

It is particularly interesting to use an end-to-end self-supervised approach because it has the advantage that it avoids the scalability issues associated with supervised learning, such as the need for large labeled datasets. Self-supervised learning allows us to leverage the raw data available across modalities, making the training process more scalable and cost-effective. Additionally, using self-supervised methods throughout, including the pretrained parts, ensures that the entire end-to-end process benefits from scalable learning techniques, regardless of the specific modalities involved.

The goal is to develop a multimodal model that performs well enough to serve as a successful proof-of-concept in the vision-language domain, demonstrating that this approach is viable. While it may not match the performance of large enterprise models—a goal that is generally unrealistic for individual researchers without access to extensive computational resources—achieving competitive results with significantly less computational expensive would proof the concept. Moreover, success in the vision-language domain suggests that this approach can be applied to other modality combinations, further broadening its impact.

This strategy would make multimodal learning more accessible to smaller research groups and individual researchers, increasing the diversity of research in the field across various modalities.

Based on this, our contributions are as follows:
We develop an end-to-end self-supervised learning approach for generating multimodal models from unimodal components, which is significantly **cheaper** to train and **smaller** in size compared to existing multimodal (vision-language) models. We show that while our approach does not reach the performance of state-of-the-art multimodal models, it is **competitive** with other vision-language models in some benchmarks.

As part of our research, we also find a promising new approach to generate smaller unimodal models that are to some extent competitive with their larger counterpart.

1.3 Required Background Knowledge

This work assumes that the reader has a thorough understanding of machine learning, deep learning, and neural networks. In particular, familiarity with concepts such as loss functions, activation functions (e.g. softmax), backpropagation, and optimization algorithms in the context of neural networks is essential. Since this work focuses on vision-language models, a basic understanding of computer vision, and most importantly, natural language processing (NLP) is required. The latter includes knowledge about tokenization, word embeddings, and while RNNs are not used in this work, familiarity with them and the **self-attention mechanism** is recommended.

Because the work involves utilizing pretrained models, the reader should be aware of concepts like transfer learning, finetuning pretrained models, and using pretrained models as feature extractors.

A deep understanding of multimodal models and how they are trained is not required, as this work provides a detailed explanation of the methodology used.

1.4 Overview

Building upon the established motivation and goals, this thesis is organized as follows:

Background
We begin by introducing the fundamental loss functions utilized in this work for ease of reference. These include essential loss functions such as cross-entropy loss and Kullback-Leibler divergence, which are critical for understanding the concept of **knowledge distillation**—a key component of our approach. Following this, we provide an introduction to the Transformer architecture, which serves as the backbone for all models in this work. We discuss how Transformers can be employed in multimodal models and detail how self-supervised learning is applied to train these models effectively.

We continue by exploring relevant approaches to multimodal models, including the paper "See, Hear, and Read: Deep Aligned Representation" [AVT17], which introduces the fundamental approach we build on.

Data and Methodology
We present all datasets used in this work, explaining their collection processes and providing detailed descriptions where necessary. Technical details regarding hardware, software, and costs are provided in the appendix to ensure reproducibility.

Here, we also provide additional detials on how we set up our experiments, and how we proceed with our research.

Experiments

Unimodal Knowledge Distillation

In section Section 5.1, we present the first part of our research, where we explore knowledge distillation on unimodal image and text models. This section is essential for validating that the knowledge distillation process works effectively in a unimodal setting and for providing a baseline for the subsequent multimodal models. It lays the foundation for our approach to **multimodal knowledge distillation**, presented in the following chapter, allowing us to incrementally increase complexity.

Multimodal Knowledge Distillation

Section 5.2 introduces our approach to multimodal knowledge distillation. We begin by adapting the approach from "See, Hear, and Read: Deep Aligned Representation" [AVT17] to the Transformer architecture. We then present our method of transforming this approach into an **end-to-end self-supervised learning process**, emphasizing the contributions and innovations of our work.

Iterative Enhancements

Following the adaptation of an end-to-end self-supervised approach, we introduce additional techniques aimed at improving the performance of our approach, presented in an iterative and incremental manner. Sections such as "Token-Type Embeddings" and "Contrastive Distillation" detail these enhancements. Throughout these sections, we repeatedly evaluate our approach on various multimodal benchmarks and compare it with other state-of-the-art multimodal methods.

Evaluation on Unimodal Benchmarks

With our final approach established, we evaluate its performance on unimodal image and text benchmarks, directly comparing it to the unimodal distilled models from Section 5.1. This evaluation is crucial for determining whether the multimodal models not only excel at aligning modalities but also perform effectively on unimodal tasks. Demonstrating parity or superiority would suggest that our multimodal models can serve as replacements for the earlier unimodal models.

Teacher Ablation Studies

Here we conduct ablation studies on the impact of different teacher models on the performance of our approach. This will provide insights on if our method can be viewed as a general approach to efficient multimodal training or if it is specific to the models that we use.

Limitations and Insights

It follows an evaluation of the limitations of our approach, in which we highlight shortcomings and provide insights into potential improvements.

Discussiuon, Conclusion, Outlook, and Impact
Finally, we discuss the results of our work, providing insights into the implications of our findings. We provide an outlook on future research and discuss the potential impact of our contributions on multimodal representation learning.

2 Background

2.1 Basic Loss Functions

Throughout this work we will make use of various loss functions and extend them if necessary. To avoid redundancy, we will define the basic concepts of them here for easier reference.

We denote bold symbols (e.g. $\boldsymbol{v}$) as vectors, and v_i as the i-th element of the respective vector. Upper-cased bold symbols (e.g. $\boldsymbol{M}$) denote matrices, and M_{ij} the element in the i-th row and j-th column of the respective matrix.

2.1.1 Mean Squared Error

The Mean Squared Error (MSE) is a loss function used in regression tasks and describes the average of the squared differences between the prediction $\hat{\boldsymbol{y}} \in \mathbb{R}^D$ and the target $\boldsymbol{y} \in \mathbb{R}^D$. Since in this work the predictions and targets will exclusively be in the form of D-dimensional vectors, the MSE is defined as:

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \frac{1}{D} \sum_{d=0}^{D-1} (y_d - \hat{y}_d)^2 \tag{1}$$

2.1.2 Kullback-Leibler Divergence

The kullback-keibler divergence (KL-Divergence) is used to measure the difference between two probability distributions. Specifically, in the context of machine learning, we are comparing a predicted probability distribution $\boldsymbol{q} \in \mathbb{R}^C$ with a target distribution $\boldsymbol{p} \in \mathbb{R}^C$. Since we are using the KL-Divergence in the context of classification tasks, which are discrete distributions over C classes, the KL-Divergence is defined as:

$$\mathcal{L}_{\text{KD}}(\boldsymbol{p} \parallel \boldsymbol{q}) = D_{\text{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) = \sum_{j=0}^{C-1} p_j \log \frac{p_j}{q_j} \tag{2}$$

p_j and q_j are the probabilities of class j according to the target and predicted distribution, respectively. For both distributions, there are potentially multiple classes with a non-zero probability:

$$\forall j(p_j \in [0,1]) \wedge \sum_j p_j = 1 \tag{3}$$

Equation 3 is defined analogously for $\boldsymbol{q}$.

2.1.3 Cross-Entropy Loss

The cross-entropy Loss (CE) is quite similar to the KL-Divergence in that it compares two probability distributions in classification tasks. It is defined as:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}, \boldsymbol{q}) = H(\boldsymbol{p}) + D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) = -\sum_j p_j \log p_j + \sum_j p_j \log \frac{p_j}{q_j} \tag{4}$$

Here $H(\boldsymbol{p})$ denotes the entropy of the target distribution $\boldsymbol{p}$, and $D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q})$ the KL-Divergence between the target and predicted distribution.

The difference between KL-Divergence and cross-entropy is that the latter is used in traditional classification tasks, where the target distribution $\boldsymbol{p}$ is fixed and one-hot encoded, meaning that there is only one correct class:

$$\exists! i(p_i = 1) \wedge \forall j\big(j \neq i \rightarrow p_j = 0\big) \tag{5}$$

This strengthens the condition of the KL-Divergence, which we defined previously in Equation 3. Since the goal is to minimize the cross-entropy loss $H(\boldsymbol{p}, \boldsymbol{q})$, and $\boldsymbol{p}$ is always one-hot encoded, meaning one class i has a probability of 1, all others 0 (see Equation 5), the entropy of the target distribution $H(\boldsymbol{p})$ will remain constant and does not affect the minimization. Moreover, again given the constraint in Equation 5, only one term in the sum of the KL-Divergence is non-zero. Consequently, we can simplify the cross-entropy loss, so that the training objective for classification tasks is:

$$\begin{aligned} \min H(\boldsymbol{p}, \boldsymbol{q}) &= H(\boldsymbol{p}) + D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) \\ &= D_{\mathrm{KL}}(\boldsymbol{p} \parallel \boldsymbol{q}) \\ &= \sum_j p_j \log \frac{p_j}{q_j} \\ &= \log \frac{1}{q_i} \\ &= -\log q_i \end{aligned} \tag{6}$$

The cross entropy loss therefore minimizes the negative log-likelihood of the correct class i.

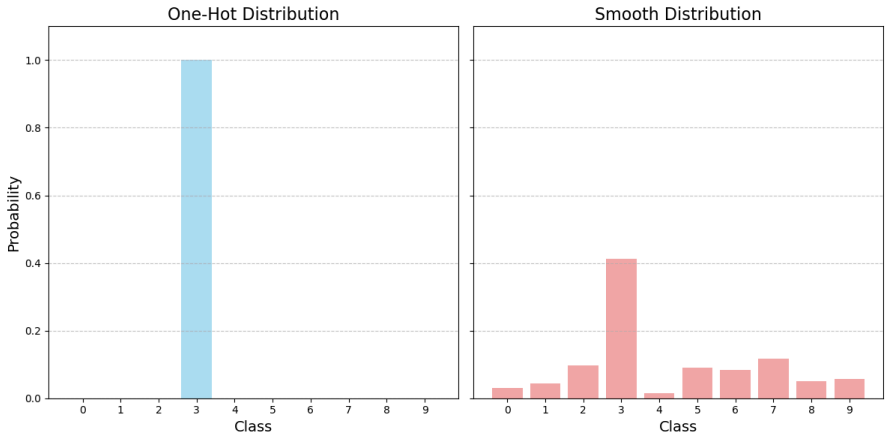

Figure 4: Comparison between different distributions over $C = 10$ classes. The one-hot distribution (left) is used for classification tasks with the cross-entropy loss. The KL-Divergence is used when predicting a smooth distribution (right). A smooth distribution usually results from a model prediction, and is a popular target distribution for knowledge distillation, introduced in Section 2.2.

Often times, the prediction $\boldsymbol{x}$ of a model is returned as raw logits, and not as probabilities. To convert logits into probabilities the softmax function is used. For ease of use, without having to mention a softmax-normalization every time we make use of the cross-entropy loss, we redefine the cross-entropy loss *actually used in this work* as:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{x}) = H(\boldsymbol{p}, \boldsymbol{x}) = -\log \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{7}$$

We denote $\boldsymbol{x}$ as the raw logits (the model prediction), and $\boldsymbol{p}$ as the one-hot encoded target distribution. i is the index of the correct class, and hence each element in $\boldsymbol{x}$ corresponds to the raw logit for one class.

A comparison between the target distribution $\boldsymbol{p}$ for cross-entropy and KL-Divergence is shown in Figure 4.

2.2 Knowledge Distillation

Training large deep learning models is computationally expensive, and therefore finanically infeasible for researchers outside of large corporations. Models often require more than 500 million parameters to achieve state-of-the-art (SOTA) performance, and training those models requires a lot of computational resources, e.g. GPUs, time and data. For example, CoCa, a vision model reaching SOTA performance of 91% validation accuracy on ImageNet-1K

[Rus+15], has 2.1 billion parameters, and was trained on more than 3 billion images [Yu+22]. Based on our approximation, it should have cost over 350 thousand USD to train[1].

One strategy to avoid high computational costs is transfer learning. Here, a potentially large pretrained model is used as a starting point and finetuned on a specific task, for a potentially different use case. That way, features learned by the model during pretraining can be reused, and the model can be adapted to the new task with less data. The disadvantage of this approach is that the model size does not change, so finetuning is still computationally expensive, especially for large models. A viable strategy would be to only use a few layers from the pretrained model, but since the layers are then used in a different model, they have to adapt to the new environment during finetuning. This, while the model will be smaller and more efficient, requires longer training times.

Another option is knowledge distillation (KD). Here, a smaller model is trained to replicate, or rather predict, the outputs of a larger model for a given sample. The larger model is called the teacher, and the smaller model the student. There are two strategies of KD usually used in practice: Response-based KD and feature-based KD [Gou+21], and we will make use of both in this work.

Knowledge distillation has the advantage that the student model can be much smaller and have a different architecture compared to the teacher model. Since the teacher is running in inference mode no backpropagation is needed, and thus no gradients have to be computed (this is still required in simple transfer learning). This makes KD faster and requires less memory compared to finetuning. Most importantly, it has been empirically shown that student models much smaller than their teachers can achieve similar performance. For example, the distilled model of BERT, DistilBERT, reduced the model size by 40% while retraining 97% of the performance of the original model [San+19].

2.2.1.1 Response-based Knowledge Distillation

In response-based KD, the teacher must provide a probability distribution over a set of classes for a given sample, which is the teacher's prediction. The student model tries to replicate this probability distribution. This is also called soft targets, because the probability distribution is, unless the teacher is 100% sure, not one-hot encoded, but rather a smooth distribution over the classes. This increases the relative importance of logits with lower values, e.g. the classes with the second and third highest logits, and Hinton et al. [HVD15] argue that this makes the model learn hidden encoded information the teacher model has learned, which are not represented when focusing on just the class with the highest logit/probability. This helps the student model to generalize better, especially on less data, compared to a model trained from scratch [Gou+21, HVD15].

[1] Calculation done based on the price per TPU hour of the CloudTPUv4 on Google Cloud Platform with a three year commitment. CoCa was trained for 5 days using 2048 CloudTPUv4s [Yu+22]. At a price of 1.449 USD per TPU hour (as of August 2024), the total cost is $1.449\ \mathrm{USD/h} * 24\ \mathrm{h/day} * 5\ \mathrm{days} * 2048\ \mathrm{TPUs} = 356,106.24\ \mathrm{USD}$.

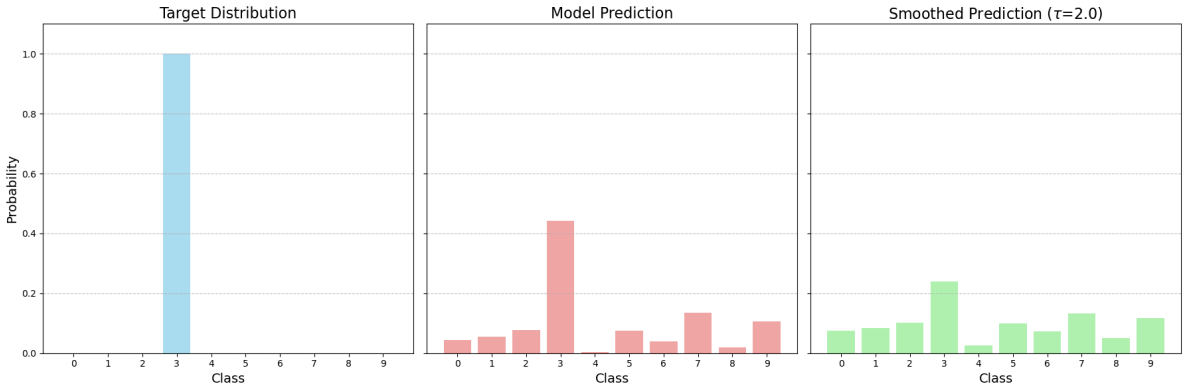

Figure 5: We present a similar figure as Figure 4. A temperature parameter τ further smoothens an already soft distribution. In response-based KD, this soft distribution is the prediction of a teacher model for a given sample (middle). Further smoothing (right) increases the relative importance of classes with lower scores. Especially in distributions with large number of classes some of them will have semantic similarities. Consider the classes "German Shorthaired Pointer" and "Labrador Retriever" of ImageNet-1K [Rus+15], which are both dog breeds. The temperature parameter brings the scores for those classes closer together, which helps the student model to learn the hidden encoded information the teacher model has learned. In the setting of the dog breeds that means: Both classes are related/ similar.

The loss function typically used in response-based KD is the KL-Divergence, introduced in Section 2.1.2, measuring the difference between two probability distributions. The mathematical formulation is as follows: Let $f(\cdot)$ be the teacher model, $g(\cdot)$ the student model, and $\boldsymbol{x}$ the input sample of the modalitiy used, e.g. an image. We define $\boldsymbol{u} = g(\boldsymbol{x})$ and $\boldsymbol{z} = f(\boldsymbol{x})$ as the output of the teacher and student model, respectively. Those are the logits, and for a classification task of e.g. 1000 classes, vectores of length 1000 ($\boldsymbol{u} \in \mathbb{R}^{1000} \wedge \boldsymbol{z} \in \mathbb{R}^{1000}$). A best practice is to divide the logits by a temperature parameter τ, before applying the softmax function [Gou+21, HVD15], which smooothes the probability distribution further, as illustrated in Figure 5.

The temperature τ is usually a tuneable hyperparameter, but research has shown that it can also be a learned parameter, especially in other settings such as contrastive learning [Bao+22], introduced later.

$$p_i = \frac{\exp\left(\frac{u_i}{\tau}\right)}{\sum_j \exp\left(\frac{u_j}{\tau}\right)} \tag{8}$$

$$q_i = \frac{\exp\left(\frac{z_i}{\tau}\right)}{\sum_j \exp\left(\frac{z_j}{\tau}\right)} \tag{9}$$

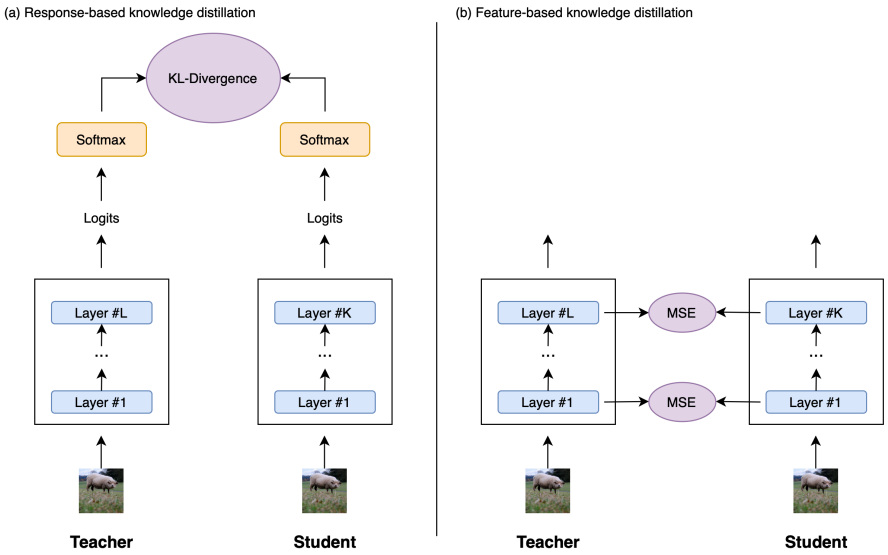

Figure 6: Response-based knowledge distillation (a) requires a teacher to provide logits to generate a probability distribution, which is predicted by the student. Feature-based knowledge distillation (b) is used for predicting the actual activations of the teacher's layer(s). In both cases the weights of the teacher are frozen and the teacher is running in evaluation/ inference mode. It is important to note that in most cases the number of layers between teacher and student differs ($L \neq K$), making the direct regression of the teacher's activations non-trivial. Figure adapted and inspired by [Gou+21], image is taken from COCO train set [Lin+14].

i and j denote indices of the classes, and p_i and q_i the probabilities of class i according to the teacher $g(\cdot)$ and student model $f(\cdot)$, respectively. The goal is to minimize the difference between both distributions (the probabilities over all classes), computed by the KL-Divergence.

Consequently, recalling the definition of the KL-Divergence in Equation 2, the training objective of response-based knowledge distillation is to bring the probability distributions of the student model for a given sample, or rather for all sample in the training set, as close as possible to the probability distributions of the teacher model for the respective sample(s).

2.2.1.2 Feature-based Knowledge Distillation

In feature-based KD, the teacher model does not need to provide a probability distribution over classes. Instead, the student model tries to replicate the (intermediate) activations of the teacher model, and therefore does not have to predict the final output (probability distribution) of the teacher.

The activations of the teacher model are usually regressed using the mean squared error (MSE) as the loss function (defined in Section 2.1.1). The mean absolute error (MAE) can also be used as a criterion, although it is less common [Bae+23, Bae+22, Gou+21].

How the activations of the teacher model are regressed by the student model can be adjusted to the specific use case. However, this choice is greatly influenced by the architecture of the student model. For example, if the student model has the same architecture as the teacher, but only half the number of layers, then the student model can't replicate the activations of the teacher 1:1. In this case, other strategies have to be used, and we will introduce one of them in Section 5.1. An illustration of response-based vs. feature-based KD is shown in Figure 6.

2.3 Transformer

While we previously introduced concepts to train smaller models in a cost-efficient way, we now introduce the architecture of the models used in this work. We will exclusively make use of the Transformer architecture [Vas+17], which has been shown to be highly effective across vision [Bao+22] and language [Dev+19]. Consequtively, we will also define the interpretation of image and text in the context of the Transformer.

2.3.1 Language Transformer

2.3.1.1 Text Representation

In Transformers, text is represented as a sequence of discrete tokens, which are, as described in word2vec [Mik+13], embedded into D-dimensional vectors using an embedding matrix. A single token i, being a (sub)word like "hell", "hello", "a", is represented as $\boldsymbol{e}_i^w \in \mathbb{R}^D$, which is the embedding of the token resulting from the embedding matrix. A text or sentence is represented as a sequence of M token embeddings/represenations $\boldsymbol{E}_w = [\boldsymbol{e}_1^w, \boldsymbol{e}_2^w, ..., \boldsymbol{e}_M^w] \in \mathbb{R}^{M \times D}$. w denotes the modality being text.

In addition, each sequence is prepended with a $[\texttt{T\_CLS}] \in \mathbb{R}^D$ token, and appended with a $[\texttt{T\_SEP}] \in \mathbb{R}^D$ token, often referred to as the cls and end-of-sequence token, respectively. As with all word embeddings, they are learnable and part of the embedding matrix. The purpose of the $[\texttt{T\_CLS}]$ token is to aggregate the global information/content of the text, while the $[\texttt{T\_SEP}]$ token is used to indicate the end of the text sequence. The resulting text representation is:

$$\boldsymbol{E}_w = \left[\boldsymbol{e}_{[\texttt{T\_CLS}]}^w, \boldsymbol{e}_1^w, \boldsymbol{e}_2^w, ..., \boldsymbol{e}_M^w, \boldsymbol{e}_{[\texttt{T\_SEP}]}^w\right] \in \mathbb{R}^{(M+2) \times D} \tag{10}$$

The sequence length M is not fixed, but is usuall set to a specific value when training a Transformer model, a popular choice being $M + 2 = 512$.

As will be explained later, unlike RNNs, Transformers do not process a sequence step by step, but all at once. As a result, the Transformer does not have any sense of order in the sequence. It is therefore necessary to add a so called positional encoding to the text embed-

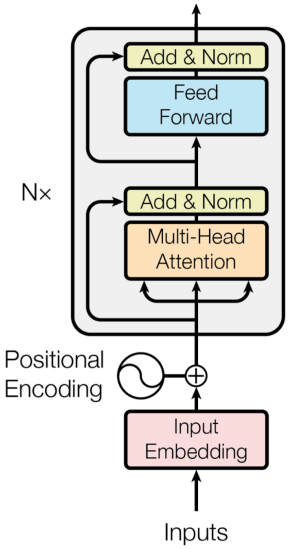

Figure 7: The architecture of the Transformer encoder. The original architecture also includes a Transformer decoder, which is not relevant for our work and therefore omitted. The encoder consists of N (we use L) Transformer layers. The output of a layer is the input to the subsequent layer. Figure is adjusted from the original Transformer paper [Vas+17].

dings, giving the Transformer a sense of order in the text sequence. This positional encoding $\boldsymbol{T}^{\text{pos}}$ is a sequence of D-dimensional vectors with the same length as $\boldsymbol{E}_w$, and can therefore simply be added to the text embeddings. The positional encoding, i.e. the embedding of each time step, is either learned or fixed, and we refer to the original "Attention is all you need paper" [Vas+17] for more details.

$$\boldsymbol{T}_w^{\text{pos}} = \left[\boldsymbol{t}^w_{\text{pos}_{\texttt{[T\_CLS]}}}, \boldsymbol{t}^w_{\text{pos}_1}, \boldsymbol{t}^w_{\text{pos}_2}, ..., \boldsymbol{t}^w_{\text{pos}_M}, \boldsymbol{t}^w_{\text{pos}_{\texttt{[T\_SEP]}}}\right] \tag{11}$$

The final text representation is defined as:

$$\boldsymbol{H}_{w,0} = \left[\boldsymbol{h}_{w,0,\texttt{[T\_CLS]}}, \boldsymbol{h}_{w,0,1}, ..., \boldsymbol{h}_{w,0,M}, \boldsymbol{h}_{w,0,\texttt{[T\_SEP]}}\right] = \boldsymbol{E}_w + \boldsymbol{T}_w^{\text{pos}} \tag{12}$$

We refer to 0 as the layer number of the Transformer, which we will clarify in the next section.

2.3.1.2 Transformer Layer

The actual Transformer consists of a set of L layers, also referred to as blocks, which are stacked on top of each other. The input to a Transformer layer l is a sequence of token embeddings $\boldsymbol{H}_{w,l-1}$ produced by the previous layer, and the input to the first layer $l = 1$ is therefore the text representation $\boldsymbol{H}_{w,0}$ as defined in Equation 12. Correspondingly, the output of a Transformer layer l is denoted as $\boldsymbol{H}_{w,l}$, and the length of the sequence does not change across layers. The architecture of a Transformer is displayed in Figure 7.

Independent of modality, we define the operations performed by one Transformer layer as follows:

$$\begin{aligned} \boldsymbol{H}'_l &= \mathrm{LN}(\mathrm{MHA}(\boldsymbol{H}_{l-1}) + \boldsymbol{H}_{l-1}) \\ \boldsymbol{H}_l &= \mathrm{LN}(\mathrm{FFN}(\boldsymbol{H}'_l) + \boldsymbol{H}'_l) \end{aligned} \tag{13}$$

$\mathrm{LN}(\cdot)$ denotes layer normalization, or short LayerNorm, as defined in [BKH16]. Simply put, it normalizes each embedding (or time step respectively) of a sequence individually, so that the mean is zero and the variance is one.

$$h'_j = \frac{h_j - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}, \qquad \mu_k = \frac{1}{D}\sum_{d=0}^{D-1} h_d, \qquad \sigma_k^2 = \frac{1}{D}\sum_{d=0}^{D-1} (h_d - \mu_k)^2 \tag{14}$$

ε is a small constant to avoid division by zero, and D is the embedding dimension of the tokens. The operation is applied to each time step $\boldsymbol{h}$ individually, and its relation to other normalization techniques, like BatchNorm, is shown in Figure 34.

The addition between two sequences of embeddings is a residual connection originally introduced in ResNets [He+16], and adds each time step of one sequence to the corresponding time step of the other sequence, also seen in Equation 12 when adding the positional encoding to the text embeddings.

$\mathrm{FFN}(\cdot)$ denotes a timestep-wise application of a feed-forward network. In essence, it is a 2-layer MLP with Layer Norm and GELU activation, where the first layer usually consists of D_{ff} neurons, and the second layer of D neurons, with D being the embedding dimension of the tokens. A popular choice for the intermediate dimension D_{ff} is $D_{\text{ff}} = 4 \times D$. The operation for a single time step $\boldsymbol{h}$, or embedding respectively, is defined as:

$$\mathrm{FFN}(\boldsymbol{h}) = \mathrm{LN}(\mathrm{GELU}(\boldsymbol{h}\boldsymbol{W}_1^T + \boldsymbol{b}_1))\boldsymbol{W}_2^T + \boldsymbol{b}_2 \tag{15}$$

For a sequence of embeddings $\boldsymbol{H}$, the operation is applied to each time step $\boldsymbol{h}$ individually, which includes the [T_CLS] and [T_SEP] token. Since the output of $\mathrm{FFN}(\cdot)$ is also in $\mathbb{R}^D$, the embedding dimension of the tokens, referred to as the *hidden size* or *hidden dim* of a Transformer, does not change across layers.

$$\mathrm{FFN}(\boldsymbol{H}) = \left[\mathrm{FFN}\left(\boldsymbol{h}_{\texttt{[T\_CLS]}}\right), \mathrm{FFN}(\boldsymbol{h}_1), \mathrm{FFN}(\boldsymbol{h}_2), ..., \mathrm{FFN}(\boldsymbol{h}_M), \mathrm{FFN}\left(\boldsymbol{h}_{\texttt{[T\_SEP]}}\right)\right] \tag{16}$$

For the definition of Multi-Head Attention $\mathrm{MHA}(\cdot)$, performing self-attention between the tokens, we refer to the original Transformer paper [Vas+17] for a detailed explanation.

The output of the last layer L is the final text representation $\boldsymbol{H}_{w,L}$. To use this representation for downstream tasks like classification, the sequence of embeddings has do be reduced to a single representation of the input. This is done by taking the representation of the [T_CLS] token, which is then passed to a subsequent classification head, represented by a single linear layer:

$$\hat{\boldsymbol{y}}_w = \boldsymbol{h}_{w,L,[\texttt{T\_CLS}]}{\boldsymbol{W}_{[\texttt{T\_CLS}]}}^T + \boldsymbol{b}_{[\texttt{T\_CLS}]} \in \mathbb{R}^C \tag{17}$$

2.3.2 Vision Transformer

After the success of the Transformer architecture in NLP, breaking various benchmarks with architectures like BERT [Dev+19], which led to its widespread adoption especially in Large Language Models (LLMs) like GPT-2 [Rad+19], researchers explored the potential of this architecture beyond text. This exploration led to the development of the vision Transformer (ViT) [Dos+21], marking a significant shift in computer vision.

Diffent from traditional Convolutional Neural Networks (CNNs), which have been the dominant architecture in computer vision before the ViT, the ViT processes images as 1D sequences of patches, instead of 2D grids of pixels.

We define an image as a 3-dimensional tensor $\boldsymbol{v} \in \mathbb{R}^{C \times H \times W}$. Throughout this work, we will exclusively use an image size of 224×224 pixels and 3 color channels, so $\boldsymbol{v} \in \mathbb{R}^{3 \times 224 \times 224}$. Before being passed to the Transformer layers, the image first needs to be converted into a sequence of embeddings, similar to text. This is done by first dividing the image into 14×14 patches, each being a square of size 16×16 pixels. Each patch i is then flattened into a 256-dimensional vector, and projected into a 768-dimensional embedding $\boldsymbol{e}_i^v \in \mathbb{R}^{768}$ using a fully connected layer. v denotes the image modality.

Similar to text, the resulting sequence of patches is prepended with a special learnable $[\texttt{I\_CLS}] \in \mathbb{R}^{768}$ token, which is used to aggregate the global information/content of the image. The image representation is defined as a sequence of N patch embeddings and the $[\texttt{I\_CLS}]$ token:

$$\boldsymbol{E}_v = \left[\boldsymbol{e}^v_{[\texttt{I\_CLS}]}, \boldsymbol{e}^v_1, \boldsymbol{e}^v_2, ..., \boldsymbol{e}^v_N\right] \tag{18}$$

To again give the Transformer a sense of order in the image patches, a unique positional encoding is added to each patch embedding, which is either learned or fixed and also represented as a sequence of 768-dimensional vectors:

$$\boldsymbol{T}^{\text{pos}}_v = \left[0, \boldsymbol{t}^v_{\text{pos}_1}, \boldsymbol{t}^v_{\text{pos}_2}, ..., \boldsymbol{t}^v_{\text{pos}_N}\right] \tag{19}$$

Notice that the positional encoding for the $[\texttt{I\_CLS}]$ token is set to zero. This is because the $[\texttt{I\_CLS}]$ token is not actually part of the image, and can be seen as a type of meta token. The input to a ViT is defined as:

$$\boldsymbol{H}_{v,0} = \left[\boldsymbol{h}_{v,0,[\texttt{I\_CLS}]}, \boldsymbol{h}_{v,0,1}, ..., \boldsymbol{h}_{v,0,N}\right] = \boldsymbol{E}_v + \boldsymbol{T}^{\text{pos}}_v \tag{20}$$

For an image size of 224×224 pixels, and a patch size of 16×16 pixels, the number of patches N is 196.

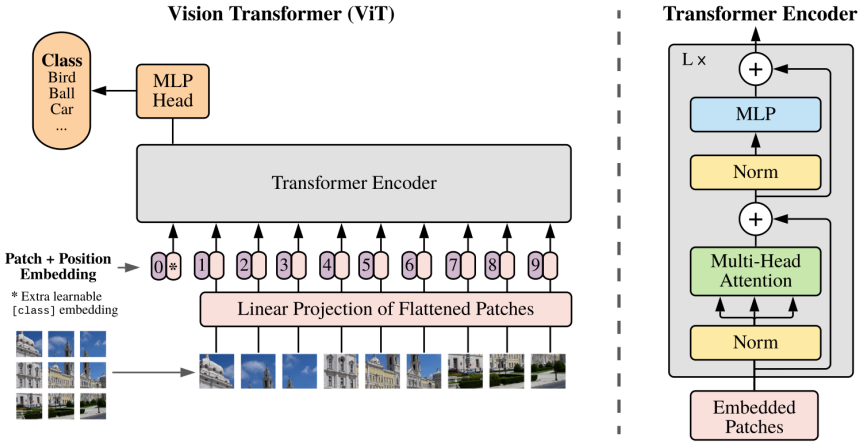

Figure 8: The architecture of the vision Transformer. Its key difference to the language Transformer is the patch embedding and the position of LayerNorm in a layer [Dos+21].

The architecture of a vision Transformer layer is almost identical to a language Transformer layer, with the only difference being that the LayerNorm operation $\mathrm{LN}(\cdot)$ is applied before Multi-Head Attention $\mathrm{MHA}(\cdot)$ [Dos+21], as illustrated in Figure 8. This type of Transformer is referred to as the Pre-LN Transformer, and the operations performed by one layer changes to:

$$\begin{aligned} \boldsymbol{H}'_l &= \mathrm{MHA}(\mathrm{LN}(\boldsymbol{H}_{l-1})) + \boldsymbol{H}_{l-1} \\ \boldsymbol{H}_l &= \mathrm{FFN}(\mathrm{LN}(\boldsymbol{H}'_l)) + \boldsymbol{H}'_l \end{aligned} \tag{21}$$

For downstream tasks like classification, the ViT follows the procedure of the original Transformer, and passes the representation of the [`I_CLS`] token to a classification head, which is a single linear (i.e. feed-forward) layer [Dos+21] (see Equation 17).

With the introducion of the vision Transformer came the division into three different model variants, each having the same architecture but different scales. The smallest model is the ViT-B/16, followed by the ViT-L/16. The largest model is the ViT-H/14. The number of layers L and the hidden size D are different for each variant, and the ViT-B/16 has $L = 12$ layers and $D = 768$ hidden dim, the ViT-L/16 has $L = 24$ layers and $D = 1024$ hidden dim, and the ViT-H/14 has $L = 32$ layers and $D = 1280$ hidden dim [Dos+21]. Both ViT-B/16 and ViT-L/16 have a patch size of 16×16 pixels, while the ViT-H/14 has a patch size of 14×14 pixels. As might have come apparent, our explanation of the Transformer architecture is based on the ViT-B/16 model, which is the model we will make use of in this work.

2.4 Multimodal Models

Multimodal models are characterized by their ability to process multiple modalities, such as text, images, audio, or video, within a single model. The motivation behind these models lies

in the idea that models should be able to understand real-world concepts in a way similar to humans. Humans can express the same concept across different modalities, “a cat”, for example, can be represented in text, image, or audio, and regardless of how the concept is expressed, the interpretation and understanding remains the same.

Please note that since our focus is on vision-language models, all further explanations of multimodality will be in the context of vision and language.

In the context of deep learning this means that the representations, the embedding/representation of e.g. the [I_CLS] and [T_CLS] token, of a concept should be the same (or at least close to each other), no matter if is expressed through image or text, which is also called alignment. However, in most existing models this is not the case. These models are typically unimodal, meaning they process only one modality, making alignment of multiple modalities impossible. A naive approach would be to pass an image into an image model, and its caption into a (seperate) text model. Even though the generated representations describe the same concept, they will not be the same, as both models are not related to each other. Each model will have a seperate latent space, as there has been no incentive for the models to learn a representation that is aligned across modalities (Figure 9), resulting in different representations for the same concept. While it is possible to compare the representations of two unimodal models, e.g. through cosine similarity, a similarity close to 1 (the maximum) does not necessarily mean that the concepts expressed in the representations are the same. There simply is no semantic relationship between the representations of the same concept produced by two unimodal models. A practical example of this will be shown in Section 5.2.1.1.4.

To overcome this limitation, a model is required that can produce modality-invariant representations, i.e. representations that are independent of the modality of the input. They should map the input of different modalities into a common representation space, where the representations of the same concept are aligned, i.e. close to each other, since they describe the same concept.

Multimodal models consist of both unimodal encoders and multimodal components. Unimodal encoders are needed because of the inherent differences between modalities, e.g. image and text: Images are 2D and composed of pixels, while text is 1D and composed of words. Unimodal encoders encode the input into a modality-specific representation space, so they are normal unimodal models, e.g. a ResNet for images. In this work, all encoders will be based on the Transformer architecture.

Multimodal models require components that enforce a shared representation space for the modalities. There are two options: A multimodal (or shared) encoder, or a loss function (training objective).

The multimodal encoder is responsible for mapping the modality-specific representations into a unified/shared representation space, where representations should be independent of the modality. That means, the representations should not contain any modality-specific

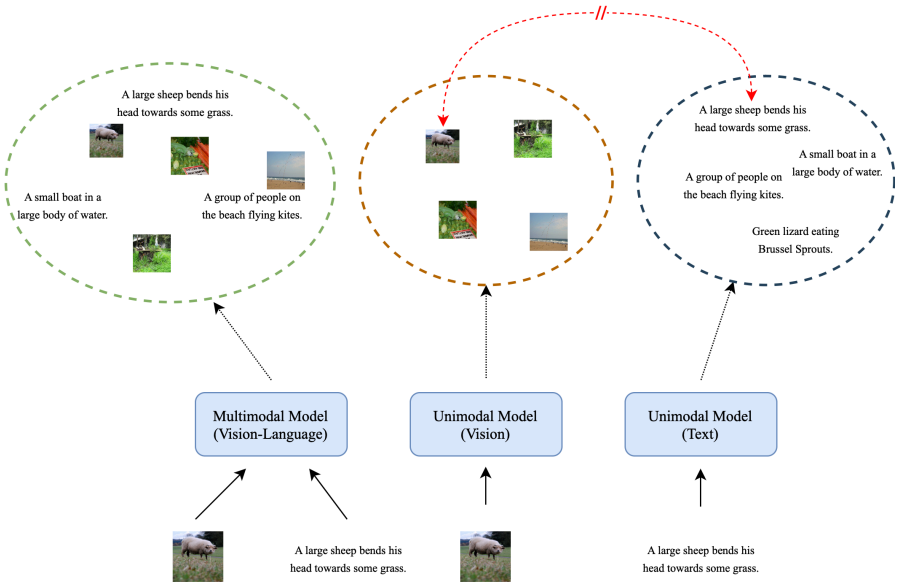

Figure 9: A multimodal model maps multiple modalities into a common representation space, where the representations of the same concept are aligned. In contrast, unimodal models map the input of a single modality into a modality-specific representation space. There is no alignment between the representations of the same concept produced by two unimodal models (indicated by the double slashed [//] arrow). While a comparison between the representations of two unimodal models is numerically possible, e.g. through cosine similarity, the similarity cannot be interpreted in a meaningful way.

information, e.g. pixel information in images or single-word information in text. Only then representations of the same concept can be aligned, or close to each other under some distance metric, respectively.

To actually ensure that the representations of the same concept are aligned, and not only in the same space, a training objective is needed that pushes the representations of the same concept closer together in representation space, while pushing the representations of different concepts further apart. For vision-language models this translates to pushing the representations of an image and its caption closer together, while pushing the representations of an image and an unrelated caption (or vice versa) further apart. To quantify the similarity between two representations, a distance metric is used, e.g. cosine similarity. The loss function is usually the contrastive loss, and its implementation for vision-language models will be introduced in Section 2.5.2.2.

When it comes to the actual implementation of multimodal models, Transformers are a very suitable choice, as they can be used for both vision and language, and even other modalities not covered in this work, like audio. Furthermore, both the language and vision Transformer require their input to be a sequence of embeddings, which makes alignment more straightforward: For each unimodal encoder of a multimodal (vision-language) model one distinct Transformer can be used, and the output of an unimodal encoder, which is the respective

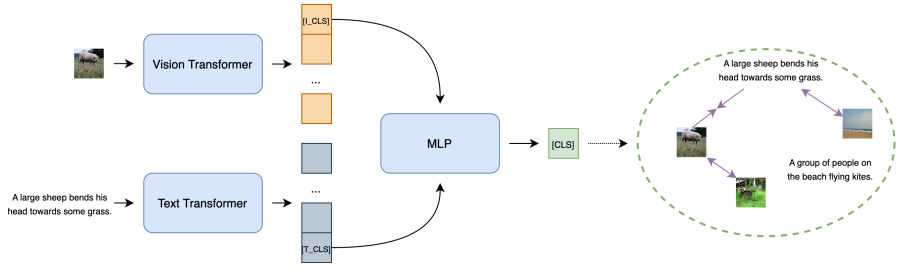

Figure 10: An abstract illustration of a vision-language model. Image and text are first passed through unimodal Transformer encoders, the cls tokens are extracted and passed seperately into the MLP that maps the modality-specific representations into a common representation space. A contrastive loss ensures the alignment and repulsion of similar and dissimilar concepts, respectively. We indicate this through purple arrows.

cls token ([I_CLS] or [T_CLS]) can then be passed (seperately) to a shared encoder, which can be implemented as a simple feed forward network (MLP). The output of the shared encoder is then still *one* representation for the image, and *one* for the text. However, both represenations will be close to each other under cosine similarity, which is useful for multimodal tasks like image-text retrieval, introduced in Section 2.6. An abstract illustration of a vision-language model with Transformer encoders and a shared encoder MLP is shown in Figure 10.

2.5 Self-Supervised Learning

2.5.1 Motivation

While we previously identified large deep learning models as generally expensive to train, we now focus on the problem of scalability in the context of supervised learning, the most common form of training AI models.

Supervised models, while powerful, are not inherently scalable. Although their architecture can be extended to create larger models that achieve better performance, these larger models require more data for training. In the context of supervised learning, this data must be labeled, which presents a significant challenge. Labeled data is scarce and expensive to obtain, as it requires human annotation, thereby limiting the scalability of supervised models.

The primary objective of self-supervised learning is to learn representations of data without relying on human-annotated labels. However, self-supervised learning is not unsupervised learning. Unsupervised learning operates without any form of supervision, meaning that no labels are required at all, as seen in clustering methods like K-means. In contrast, self-supervised learning requires, as supervised learning, labeled data, but in contrast to supervised learning labels are generated directly from the data itself.

A prominent example of self-supervised learning is Masked Language Modeling (MLM) in Natural Language Processing (NLP), which is used in the popular NLP model BERT, being

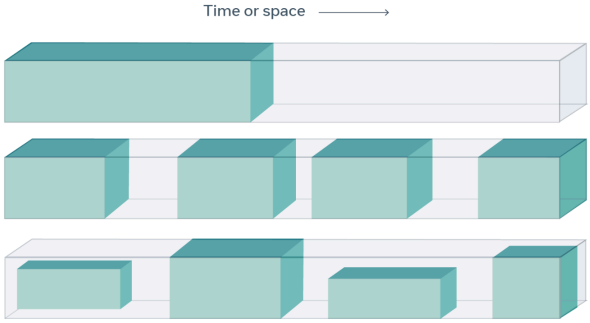

Figure 11: In self-supervised learning parts of the data are masked (grey), and the task of a model is to predict the masked parts using the visible data (green) (Figure is taken directly from[LM21]).

one of the first models trained using self-supervised methods to achieve state-of-the-art performance in NLP [Dev+19]. In BERT, certain tokens, or words, are masked, i.e. removed from a sentence, and the model is tasked with predicting the masked tokens (see Figure 11). Since the labels are derived from the data itself — the words to predict are part of the original data — no human annotation is needed [Dev+19]. This allows for the utilization of large amounts of unlabeled data, as any text data can be used.

What makes self-supervised learning particularly powerful is its applicability to any type of data with a hierarchical structure, such as text, images, audio, or video. In these cases, part of the data can be masked, and the model must predict the masked part based on the context provided by the remaining data. An intuitive example, presented by Yann LeCun and Ishan Misra of Meta, illustrates why this approach is effective. Consider the sentence "The lions chase the wildebeests in the savanna." If "lions" and "wildebeests" are masked, the input becomes "The [MASK] chases the [MASK] in the savanna.". To successfully predict the masked words, the model must understand the real-world concepts expressed by the sentence. While "The cat chases the mouse in the savanna" might be a valid prediction in the context of "chase," the word "savanna" provides additional context, as it is not a typical habitat for cats and mice, but rather for lions and wildebeests. Thus, the model must understand that lions and wildebeests are animals that inhabit savannas, in order to make a correct prediction. Through this process of predicting masked words, the model learns about the concepts of the world we live in [LM21].

While this example is specific to text data, the same principle can be applied to other types of hierarchical data, e.g. images and audio.

Consequently, self-supervised learning makes models scalable, and allows/forces them to understand the meaning and underlying concepts of the data they are trained on. This is particularly important when aligning multiple modalities.

2.5.2 Contrastive Learning

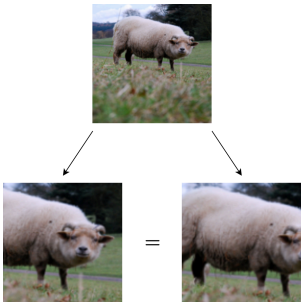

Figure 12: Adding small translations to an image, e.g. a random crop, as illustrated in the figure, will retrain high-level semantic features while changing pixel-level information. The content of the image stays the same, and the same should therefore hold for the representations produced by the model. Image in the figure has been taken from the COCO train set [Lin+14].

2.5.2.1 Vision

In settings where masking discrete tokens and predicting them based on a set of possible tokens, as in language models, is not possible, contrastive learning can be used as a self-supervised method. This is especially useful in vision models, as images are continuous, so there is no discrete set of possible tokens/words to predict.

Contrastive learning, or the contrastive loss, is a method to learn representations of data without the need for labels, and used in computer vision models like MoCo [He+20], SimCLR [Che+20], and CLIP [Rad+21].

In computer vision, contrastive learning exploits the fact that the high-level semantics of an image are invariant to small (or moderate) changes in pixel-level information. This is achieved by augmenting the input image, e.g., by cropping, rotating, or flipping it. Provided the augmentation is not too drastic (e.g. the crop size too large), the high-level semantics of the image will remain the same after augmentation, even though pixel-level information do not. The goal of the image model is then to maximize the cosine similarity between the global representations of two augmented versions of the same image. In Transformers, the global representation is usually the `[I_CLS]` token retuned by the final layer of the model, which is a vector that can be compared with the `[I_CLS]` token of another image using the cosine similarity. The augmented versions are often referred to as different *views* of the same image [CH21], as shown in Figure 12.

However, this alone is not sufficient, as the model will collapse to a trivial solution by simply returning the same representation for all inputs, as demonstrated in the papers MoCo [He+20] and SimSiam [CH21]. Producing the same representation for all inputs is the simplest way to maximize the cosine similarity between the original image and its augmented versions, because the representation produced for an image would always be the same, therefore maximizing the cosine similarity (a value of 1). To prevent this, negative samples are introduced. Negative samples are other images that do not contain the same content

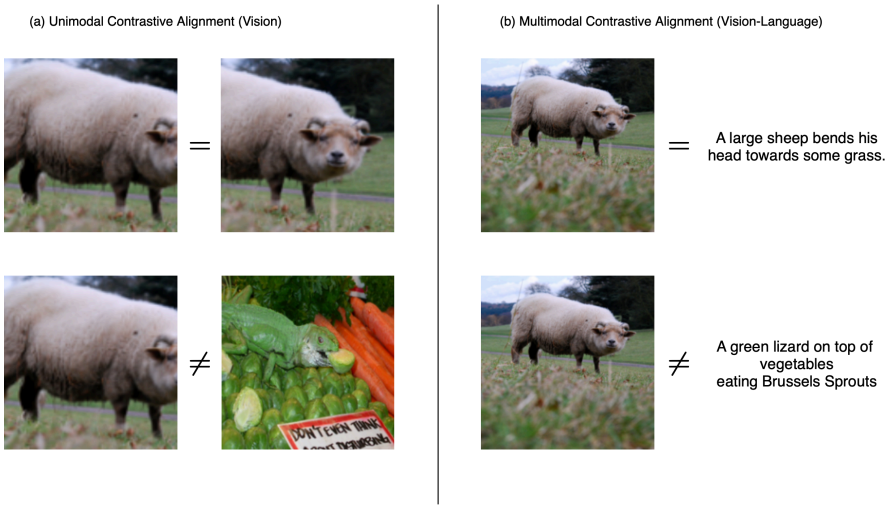

Figure 13: Contrastive learning aims to align the same (or similar) real-world concepts in representation space, while pushing different concepts apart. Multimodal contrastive learning (b) requires existing pairs, e.g. image-text, while for the unimodal case (a) pairs are synthetically created by augmenting the input. Images and text in the figure have been taken from the COCO train set [Lin+14].

as the original image, and the cosine similarity between the original image and these negative samples should therefore be minimized. This prevents the model from collapsing to a constant representation, as it would not minimize the cosine similarity between negative samples, and thus not minimize the loss. A simple yet expressive visualization can be found in [CH20]. This makes self-supervised training of image models possible, and the learned representations represent the high-level semantics of the images, learned without the need for labels.

An implementation and mathematical formulation of the contrastive loss will be introduced on the example of vision-language models in the next section.

2.5.2.2 Vision-Language

Introduced as a method for self-supervised learning of image models, contrastive learning can be extended from unimodal (image) to multimodal applications, such as image and text. As mentioned in the previous section, we aim to maximize the cosine similarity between an image and its corresponding text (i.e., caption), and vice versa. Augmentation is not needed, as we always have pairs: one image and one text. Negative samples for images are captions of other images, and vice versa. In this setting, the model learns to produce similar representations for an image and its caption, describing the same real-world concept, and dissimilar representations for an image and caption that are unrelated. A conceptual example for both vision and vision-language contrastive learning can be seen in Figure 13.

Contrastive learning requires a (global) representation of the input, which is then used to compare it with other inputs. Since the introduction of the vision Transformer in 2020

by Dosovitskiy et al. [Dos+21], most vision-language models are exclusively based on the Transformer architecture, which is why the cls token is used as the global representation for both image ([I_CLS]) and text ([T_CLS]), respectively. There have been other approaches, such as Cross-Modal Late Interaction introduced in FLILP [Yao+21], but they usually require significantly more compute [Yao+21] and do not outperform global contrastive learning [Wan+23], which is what we use here.

The representations are generated by passing the image sequence $\boldsymbol{H}_{v,0}$ and text sequence $\boldsymbol{H}_{w,0}$ through the vision-language model, and extracting the representations for both tokens ($\boldsymbol{h}_{v,L,[\texttt{I\_CLS}]}$ and $\boldsymbol{h}_{w,L,[\texttt{T\_CLS}]}$) from the output of the final layer $\boldsymbol{H}_{v,L}$ and $\boldsymbol{H}_{w,L}$, which is the output of the Transformer. For the resulting batch of image and text representations $\left\{\boldsymbol{h}_{(v,L,[\texttt{I\_CLS}]),k}, \boldsymbol{h}_{(w,L,[\texttt{T\_CLS}]),k}\right\}_{k=1}^{B}$, where B is the batch size, the cosine similarity between all possible image-text pairs is computed. The cosine similarity between two vecotrs $\boldsymbol{a}$ and $\boldsymbol{b}$ is given by:

$$\cos(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}\boldsymbol{b}^T}{\|\boldsymbol{a}\|_2 * \|\boldsymbol{b}\|_2} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2}\frac{\boldsymbol{b}^T}{\|\boldsymbol{b}\|_2} \tag{22}$$

$\boldsymbol{a}\boldsymbol{b}^T$ denotes the simple dot product between both representations. $\|\boldsymbol{a}\|_2$ and $\|\boldsymbol{b}\|_2$ denote the L2-norm of the representations.

The cosine similarity between all possible image-text pairs can be computed efficiently by organizing all image and text representations in a matrix, which is already given in a batch-wise training, and normalizing every representation, which we denote by the function $\delta(\cdot)$.

$$\delta(\boldsymbol{h}) = \frac{\boldsymbol{h}}{\|\boldsymbol{h}\|_2} \tag{23}$$

$$\boldsymbol{I} = \left[\delta\left(\boldsymbol{h}_{(v,L,[\texttt{I\_CLS}]),1}\right), \delta\left(\boldsymbol{h}_{(v,L,[\texttt{I\_CLS}]),2}\right), ..., \delta\left(\boldsymbol{h}_{(v,L,[\texttt{I\_CLS}]),B}\right)\right] \in \mathbb{R}^{B\times D} \tag{24}$$

$$\boldsymbol{T} = \left[\delta\left(\boldsymbol{h}_{(w,L,[\texttt{T\_CLS}]),1}\right), \delta\left(\boldsymbol{h}_{(w,L,[\texttt{T\_CLS}]),2}\right), ..., \delta\left(\boldsymbol{h}_{(w,L,[\texttt{T\_CLS}]),B}\right)\right] \in \mathbb{R}^{B\times D} \tag{25}$$

$\boldsymbol{I}$ denotes the batch/matrix of image representations, and $\boldsymbol{T}$ contains the text representations. D is the dimensionality of the representations.

A matrix multiplication of both batches of representations then computes the dot product between every image with every text, and vice versa. Since the representations are normalized, the result will be the cosine similarity between all possible image-text pairs in the batch.

$$\boldsymbol{L} = \boldsymbol{I}\boldsymbol{T}^T, \boldsymbol{L} \in \mathbb{R}^{B\times B} \tag{26}$$

$L_{i,j}$ then denotes the similarity between image i and text j in the batch. The diagonal of the matrix contains the similarity between positive pairs, i.e. the correct image-text pairs (i, i), with $L_{i,i}$ describing their similarity. For an image, all other texts in the batch are considered

as negative samples, and vice versa for text. The superscript T denotes the transpose of a matrix, and is not to be confused with the batch of text representations $\boldsymbol{T}$.

For a batch size of 256 ($B = 256$), each image has 255 negative samples (captions of other images) and one positive sample (its own caption), the same holds vice versa. This can be seen as a classification problem with 256 classes, where the model has to predict the correct class out of 256 classes, and each class representing one caption or image, respectively. For an image, the logit for the correct class is the similarity (cosine) to its own caption, and the logits for the negative classes are the similarities to the captions of other images. The same holds vice versa for text.

To calculate the loss, the cross-entropy loss is used. For a batch, the loss for selecting the correct caption for each image is given by:

$$\mathcal{L}_{\text{CL}}^{\text{i2t}} = \frac{1}{B} \sum_{i=0}^{B-1} -\log \frac{\exp(L_{i,i})}{\sum_{k=0}^{B-1} \exp(L_{i,k})} \tag{27}$$

$\frac{\exp(L_{i,i})}{\sum_{k=0}^{B-1} \exp(L_{i,k})}$ denotes the softmax-normalized similarity between an image and its correct caption, which is the usual way to calculate the cross-entropy. The result of this normalization is a probability distribution for each image, where each caption in the batch has a probability of being the correct caption for the image, and vice versa. The probability that the correct caption belongs to the current image is then used to calculate the negative log-likelihood, which is the loss.

Accordingly, the loss for selecting the correct image for each caption is given by:

$$\mathcal{L}_{\text{CL}}^{\text{t2i}} = \frac{1}{B} \sum_{i=0}^{B-1} -\log \frac{\exp(L_{i,i})}{\sum_{k=0}^{B-1} \exp(L_{k,i})} \tag{28}$$

Here, the softmax-normalization is with respect to the similarity of a text with all other images in the batch. The final loss is the mean of the image-to-text and text-to-image loss:

$$\mathcal{L}_{\text{CL}} = \frac{1}{2} * (\mathcal{L}_{\text{CL}}^{\text{i2t}} + \mathcal{L}_{\text{CL}}^{\text{t2i}}) \tag{29}$$

Returning to the concept of contrastive learning, this process ensures that the similarity between the representation of an image and its caption is maximized, i.e. close to each other, while the similarity between an image and an unrelated caption is minimized, i.e. far apart. Only this would appropriately minimize the loss, and thus the model learns to align the representations of the same concept across modalities. An illustration of multimodal contrastive learning can be found in Figure 14. Intuitively, it can be thought of as *finding the correct caption for an image among all captions in the current batch*, and vice versa.

The performance of contrastive learning is highly dependent on the number of negative samples available, which directly translates to the batch size. For instance, with a batch size

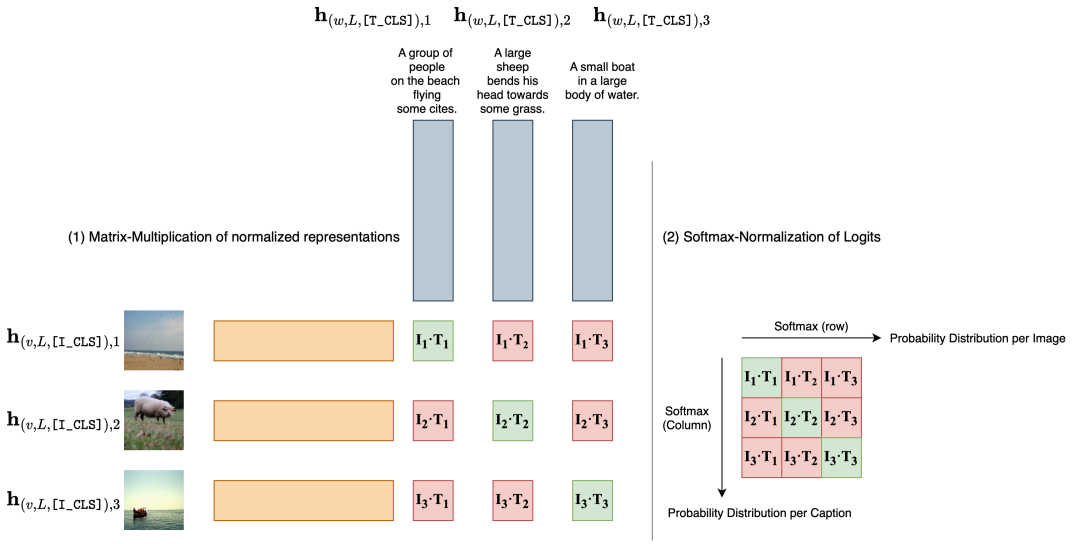

Figure 14: Contrastive Learning is performed using matrix multiplication of normalized representations (1), and the result is matrix $\boldsymbol{L}$ described in Equation 26. The diagonal of the resulting matrix contains the cosine similarity between positive samples. The softmax operation along the rows yields a probabilty distribution for each image over all captions, and the softmax operation along the columns vice versa (2). The cross-entropy loss is then used to calculate the loss for the distributions. The final loss is the mean of both losses. Image-text pairs in the figure have been taken from the COCO train set [Lin+14].

of two, the model only needs to differentiate between one caption that belongs to the image and one that does not (a negative sample), and vice versa. This task is significantly simpler than with 255 negative samples or more, where there might be captions that are semantically similar to the image, but do not belong to it. With increased negative samples, the probability of encountering hard-negative examples increases, forcing the model to aggregate as much information as possible in [I_CLS] and [T_CLS] to even differentiate between semantically similar concepts.

The results improve with an increased number of negative examples [He+20, Wan+23], which we will also show later More negative samples are usually achieved by using larger batch sizes [He+20, Rad+21, Wan+23], however, this typically requires higher VRAM GPUs, or multiple GPUs, which is costly.

2.6 Image-Text Retrieval

The goal of image-text retrieval (ITR) is to find the matching caption for a given image in a set of captions, and likewise, finding the matching image for a given caption in a set of images. This is exactly what is learned through contrastive learning, where we try to maximize the similarity between an image and its paired caption among other captions, and vice versa. The process begins with embedding and normalizing a set of samples, which become a set

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
FLAVA [Sin+21]	42.74	76.76	-	38.38	67.47	-	67.7	94.0	-	65.22	89.38	-
CLIP [Rad+21]	58.4	81.5	88.1	37.8	62.4	72.2	88.0	98.7	99.4	68.7	90.6	95.2
BEiT-3 [Wan+23]	84.8	96.5	98.3	67.2	87.7	92.8	98.0	100.0	100.0	90.3	98.7	99.5

Table 1: Benchmarks of different vision-language models on the MSCOCO and Flickr30K datasets for image-text retrieval. Metrics are in percent, so the maximum score is 100(%).

of keys. For some normalized candidate representation, called the query, the most similar key among the set of keys is the retrieved sample. In the context of vision-language models, the query could be an image, and the keys a set of captions. For the normalization and comparison the same batch-wise computation introduced for contrastive loss can be used. The retrieved sample is the one with the highest cosine similarity to the query among all keys.

Image-text retrieval is the direct application of vision-language contrastive learning, and can be viewed as a form of semantic search, which has significant practical relevance in areas like recommendation systems, e.g. to find fitting images based on a given text query. This is precisely what is learned through multimodal contrastive learning.

Image-text retrieval is a simple and efficient way to benchmark the quality of the learned representations of a vision-language model, as it does not require any finetuning, just the embeddings produced by the model. The metric used for benchmarking is Rank@K (R@K), where K determines at which rank the paired/correct sample has to be in the ranking of keys, in order for the retrieval to be considered correct. We use R@1, R@5, and R@10, where R@1 is the normal accuracy, i.e., the paired sample has to be the most similar one. R@5 means that the paired sample has to be in the top 5 most similar samples, and for R@10, it has to be in the top 10 most similar samples. The higher the R@K scores, the better the model has learned to align the representations of the same concept across modalities.

In this thesis, we use the 5K test set of MSCOCO [Lin+14], and the 1K test set of Flickr30k [You+14] for benchmarking, which are the standard benchmarking dataset for multimodal models like FLAVA [Sin+21], CLIP [Rad+21], VLMo [Bao+22], and BEiT-3 [Wan+23]. MSCOCO contains 5K images with 5 captions for each image [Lin+14], and Flickr30k contains 1K images with 5 captions each [You+14]. For both datasets, all images and all texts are embedded and normalized, so that each image and each text is represented by the respective cls token returned by the model. Then, matrix multiplication between all images and all captions of a dataset is performed, resulting in a matrix of shape (F, W), where F is the number of images and W is the number of captions in the dataset. So for MSCOCO, the matrix is of shape (5K, 25K), and for Flickr30k, the matrix is of shape (1K, 5K).

For each image, R@1, R@5, and R@10 are computed. The mean of R@1, R@5, and R@10 over all images are then called text-retrieval of the respective metrics (e.g. R@1-text-retrieval).

We call this text-retrieval, because we are trying to retrieve the correct caption for a given image. The same is done for each caption, resulting in image-retrieval of the respective metrics (e.g. R@1-image-retrieval). For each dataset, we have 6 metrics in total: R@1, R@5, and R@10 for text-retrieval and image-retrieval, respectively. We will report the results of image-text retrieval in the format seen in Table 1.

2.7 Related Work

2.7.1 Deep Aligned Representations

The motivation for the knowledge distillation driven approach in this work is provided by the paper "See, Hear, and Read: Deep Aligned Representations" by Aytar et al. [AVT17]. For simplicity, this paper will be referred to as "SHRe" (for **S**ee, **H**ear, **Re**ad) in this work.

In SHRe, the authors propose a method to align representations of image, text, and audio through knowledge distillation from a supervised image model. The student model is a multimodal model with separate modality-specific encoders for image, text, and audio, with a shared encoder on top. The approach utilizes 1D convolutions for audio and text, and 2D convolutions for images. The output feature maps of these encoders are flattened and then passed separately through a shared encoder, consisting of 3 linear layers [AVT17]. Notice how the aforementioned is similar to the definition of a multimodal model as defined in Section 2.4. The approach is generally independent of the specific architecture of the components (encoders), meaning that any architecture can be used (also the Transformer architecture).

The teacher model was trained in a supervised manner, with the authors utilizing a model pretrained on ImageNet-1K, though it is not specified what exact model was used. The training objective is to minimize the KL-Divergence between the teacher and student model.

Specifically, the method involves using image-text $\{\boldsymbol{x}^v, \boldsymbol{x}^w\}$ and image-audio $\{\boldsymbol{x}^v, \boldsymbol{x}^a\}$ pairs. $\boldsymbol{x}^v$ is a 2D image, $\boldsymbol{x}^w$ a sequence of text tokens, and $\boldsymbol{x}^a$ a spectrogram of audio. For each pair, the image $\boldsymbol{x}^v$ is passed through the teacher model $g(\cdot)$, producing a probability distribution over the ImageNet-1K classes (1000 classes), denoted as $g(\boldsymbol{x}^v)$. The same image $\boldsymbol{x}^v$ is also passed through the image encoder $f_v(\cdot)$ of the student model, followed by the shared encoder $s(\cdot)$, also resulting in a probability distribution over the ImageNet-1K classes, defined as $s(f_v(\boldsymbol{x}^v))$.

The other part of the pair, for example, the text $\boldsymbol{x}^w$ in an image-text pair, is passed through the text encoder $f_w(\cdot)$ of the student model, and then through the shared encoder $s(\cdot)$. Since the shared encoder is the same as the one used for the image, the output is, again, a probability distribution over the ImageNet-1K classes, represented as $s(f_w(\boldsymbol{x}^w))$.

The probability distribution generated by the teacher model for the image can be compared with the probability distribution produced by the student model for the same image, using KL-Divergence. This is the usual, response-based, approach to knowledge distillation, defined in Section 2.2.1.1. What makes the approach unique, however, is that the probability distribution of the teacher model for the image can be compared with the probability distribution of the student model for the text. For a single image-text pair, the loss is defined as:

$$\mathcal{L}_{\text{KD}}^{vw} = \frac{1}{2} * D_{\text{KL}}(g(\boldsymbol{x}^v) \parallel s(f_v(\boldsymbol{x}^v))) + \frac{1}{2} * D_{\text{KL}}(g(\boldsymbol{x}^v) \parallel s(f_w(\boldsymbol{x}^w))) \tag{30}$$

With D_{KL} being the KL-Divergence. The loss changes accordingly for image-audio pairs, where the probability distribution over audio is defined as $s(f_a(\boldsymbol{x}^a))$.

$$\mathcal{L}_{\text{KD}}^{va} = \frac{1}{2} * D_{\text{KL}}(g(\boldsymbol{x}^v) \parallel s(f_v(\boldsymbol{x}^v))) + \frac{1}{2} * D_{\text{KL}}(g(\boldsymbol{x}^v) \parallel s(f_a(\boldsymbol{x}^a))) \tag{31}$$

The goal of this approach is to make the probability distributions between teacher and student as similar as possible. Since an image and its corresponding text in an image-text pair describe the same real-world concept, the distribution of the teacher model for the image over the ImageNet-1K classes can directly be transferred to the caption of the image. That way, the model can learn to output the same probabilities over the ImageNet-1K classes for both the image and the text. This enables the alignment of modalities at the level of real-world objects. The same process can be applied to image-audio pairs, allowing the model to align representations across multiple modalities. A visualization of this is shown in Figure 37.

Even though all modalities share the same shared encoder $s(\cdot)$, the output of the intermediate layers in the shared encoder will still differ for each modality. This is because KL-Divergence only ensures alignment at the level of classes, which corresponds to the output layer (the last linear layer of the shared encoder outputs the probability distribution over ImageNet-1K classes). The internal representations in $s(\cdot)$, meaning the first two layers, can still be different between the modalities of a pair. They can vary, as long as the resulting probability distribution of the last linear layer is the same as the teacher's output.

However, the shared encoder is meant to have the same internal representation for e.g. an image and its caption/text: Since they describe the same concept, the activations in the shared encoder should be similar, which is, as described in Section 2.5.2, crucial for tasks such as retrieval. To achieve this, the authors add a ranking loss to the training, which functions similarly to a contrastive loss. This ranking loss drives the representations of inputs from the same pair closer together, while pushing the representations of inputs from different pairs further apart. It is defined as:

$$\mathcal{L}_{\text{Rank}} = \sum_{i=1}^{B} \sum_{j \neq i} \max\{0, \Delta - \cos(\boldsymbol{x}_i^v, \boldsymbol{x}_i) + \cos(\boldsymbol{x}_i^v, \boldsymbol{x}_j)\} \tag{32}$$

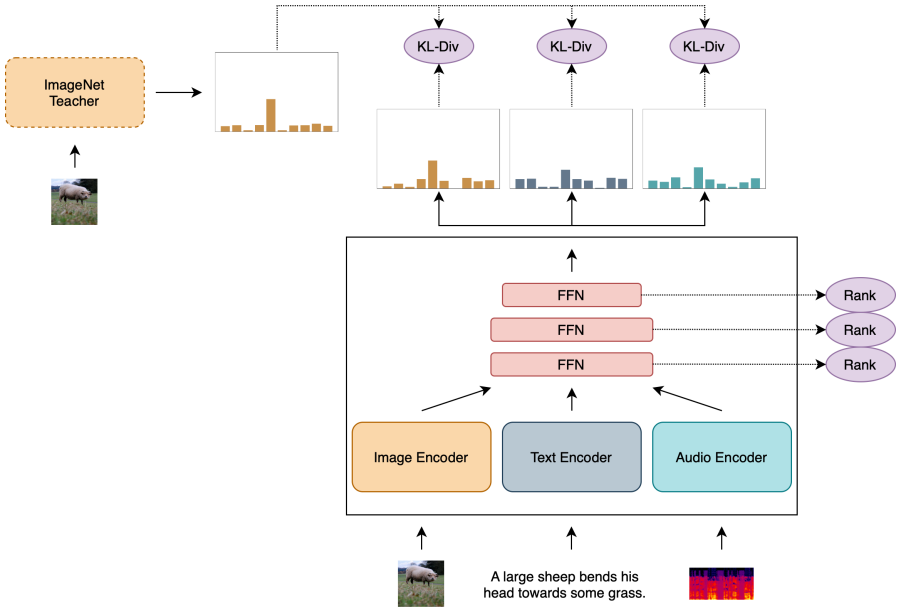

Figure 15: Illustration of the SHRe approach. The model is trained to output the same probability distribution over ImageNet-1K classes between images, image-text pairs, and image-audio pairs. Internal representations are aligned using a ranking loss [AVT17]. Image, text, and audio are always passed individually through the model. For each input, the model outputs a probability distribution over the ImageNet-1K classes (illustrated distributions are smaller for better visibility). The figure does not originate from the original paper, but is a custom visualization of the concept. Image and text example is taken from the MSCOCO train set [Lin+14], the spectogram originates from the SHRe paper [AVT17].

Here, B represents the batch size, $\boldsymbol{x}_i^v$ is an image, and $\boldsymbol{x}_i$ is the corresponding text or audio, depending if an image-text or image-audio pair is used. j iterates over negative samples in the batch ($j \neq i$).

Different from contrastive loss, for a given input, e.g. an image, the ranking loss does not normalize the similarity score of a positive pair (e.g. image-text) with respect to all other possible pairings (all other texts) for a sample (image) in the batch. The authors did not provide intuitions for the choice of the ranking loss over the contrastive loss, and we can only assume that since the paper was published in 2017 [AVT17], the contrastive loss was not as widely adapted as it is today.

The final loss is a combination of the KL-Divergence loss and the ranking loss:

$$\mathcal{L}_{\text{SHRe}} = \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{Rank}} \tag{33}$$

The authors evaluate SHRe on retrieval tasks, and the results (Table 2) show that SHRe performs significantly better than a random baseline. Interestingly, even though the model is only trained on image-text and image-audio pairs, the alignment also generalizes to text-audio pairs, and the model can retrieve text-audio pairs, albeit not as well as between the

Model	MSCOCO		Flickr (Custom)[2]		Unspecified[3]	
	Image ↓ Text	Text ↓ Image	Image ↓ Sound	Sound ↓ Image	Text ↓ Sound	Sound ↓ Text
Random	500	500	500	500	500	500
SHRe	5.8	6.0	47.5	47.8	135.0	140.5

Table 2: Retrieval results of SHRe on different datasets. Each dataset contains 5k sample pairs (e.g. image-text pairs) for evaluation, and is splitted into 5 chunks of 1k samples each. Retrieval is then performed only between the samples inside of each chunk, and the metric used is the median rank. The median rank denotes the position of the correct pair in the list of possible retrieval candidates (1k candidates), the lowest rank is 1, meaning that the correct pair has the highest similarity score to the query. The median rank is averaged over all chunks for each datasets, so the results seen describe the average median rank over all chunks for each dataset. The results are taken from the SHRe paper [AVT17].

modalities it was trained on [AVT17]. This indicates that the image modality acts as an anchor between text and audio, enabling the model to align representations between modalities it was not explicitly trained on. The alignment between modalities becomes transitive.

The approach is illustrated in Figure 15. It is important to note that SHRe is only trained with image-text and image-audio pairs, and not, how it might seem from the figure, with image-text-audio triplets.

The SHRe approach is a crucial foundation for this work, as it demonstrates how the knowledge from a **supervised** unimodal (image) model can be *extracted* and used to train a multimodal model.

2.7.2 CLIP

2.7.2.1 Method

CLIP is a method developed by OpenAI to train a vision-language model using contrastive learning. CLIP stands for (**C**ontrastive **L**anguage-**I**mage **P**retraining). The architecture consists of a separate image encoder $f(\cdot)$ and text encoder $g(\cdot)$, both of which can be any architecture, and a linear projection (linear layer without bias and activation function) on top of the modality-specific encoders.

[2]Datasets used consists of videos collected from Flickr, from which frames were extracted and used as images with the corresponding audio [AVT17].

[3]Data has been collected and annotated using Amazon Mechanical Turk [AVT17, SF08]. Where the data originates from is not specified in the paper.

The forward pass works as follows: For a batch of image-text pairs, the images $\left\{\boldsymbol{H}_{(v,0),i}\right\}_{i=1}^{B}$ (B denotes the batch size) are passed through the image encoder, resulting in an image representation $\boldsymbol{I} \in \mathbb{R}^{B \times D}$. Similarly, the texts $\left\{\boldsymbol{H}_{(w,0),i}\right\}_{i=1}^{B}$ are passed through the text encoder, producing text representations $\boldsymbol{T} \in \mathbb{R}^{B \times D}$. Recall that $\boldsymbol{I}$ and $\boldsymbol{T}$ correspond to the batched representations of the [I_CLS] and [T_CLS] tokens as defined in Section 2.5.2.2.

Both the image and text representations produced by the encoders are in separate embedding spaces — one for images and one for text - they are not related to each other initially. However, for contrastive learning to be effective, the embeddings should exist in the same latent space. After all, the embedding for an image and its corresponding text should be the same (or at least very close to each other).

In SHRE, discussed in the previous section, this shared latent space is achieved through a shared encoder on top of the modality-specific encoders, and through a ranking loss [AVT17]. CLIP maps the image and text representations into a shared latent space using linear projections $\boldsymbol{o}_v$ and $\boldsymbol{o}_w$ for image and text, respectively.

The image representations in the shared embedding space are denoted as $\boldsymbol{I}' = \|\boldsymbol{o}_v \boldsymbol{I}^T\|_2$, and the text representations are given by $\boldsymbol{T}' = \|\boldsymbol{o}_w \boldsymbol{T}^T\|_2$. Since cosine similarity is used as the similarity metric in the contrastive loss, the embeddings are normalized, which is indicated by the l2 norm $\|\cdot\|_2$ around the result of the linear projections. It is important to note that the superscript T denotes the transpose of a matrix, not the batch of text representations.

Then, it is sufficient to perform matrix multiplication of the normalized representations in order to compute the cosine similarity between each pair. The result is given by:

$$\boldsymbol{L} = \exp(t) * \boldsymbol{I}' \boldsymbol{T}'^T, \boldsymbol{L} \in \mathbb{R}^{B \times B} \tag{34}$$

The result of this operation is essentially the batched cosine similarity operation for the contrastive loss. However, it is notable that the cosine similarities $\boldsymbol{L}$ are scaled by $\exp(t)$, where t is a temperature parameter. This parameter is used to control the smoothness of the softmax function, and is a scalar applied element-wise to the cosine similarities, which should be a familiar concept from knowledge distillation.

In knowledge distillation, the temperature was introduced as a tunable hyperparameter [AVT17, Gou+21]. However, in CLIP it is a learnable parameter that is optimized during training, just like any other parameter in the model, eliminating the need for manual tuning. The temperature t is optimized in log-space, which is why the actual temperature by which logits are scaled, is given by $\exp(t)$ [Rad+21].

Although the authors did not provide a specific reason for the optimization in log-space, it is likely that this approach ensures that the temperature is always positive, since $\exp(t)$ always returns a positive value. Optimizing in log-space may also contribute to greater nu-

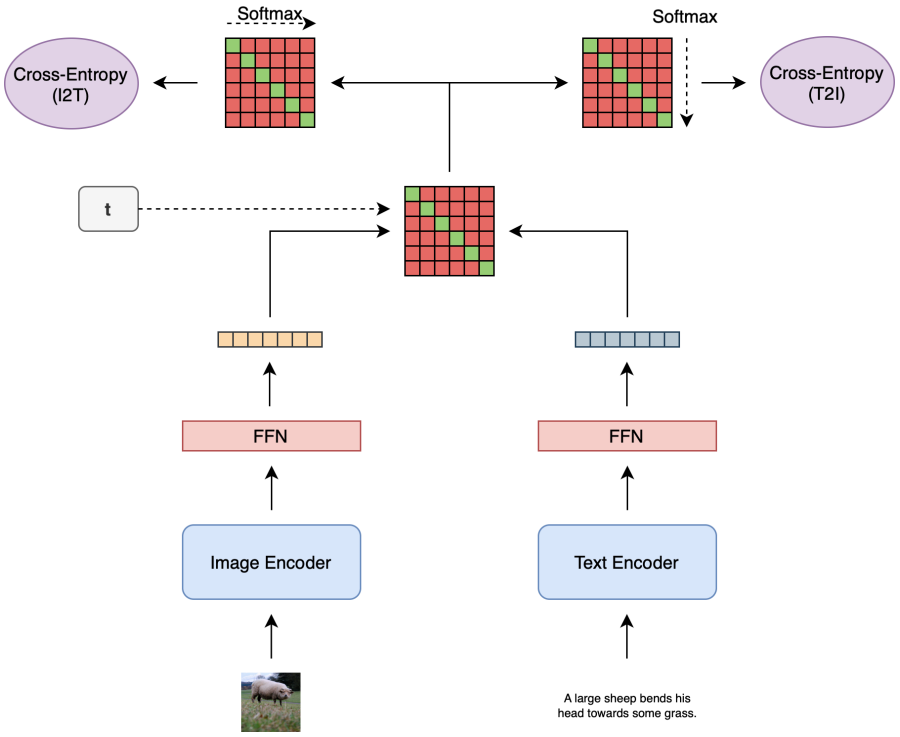

Figure 16: Illustration of CLIP training. A batch of image-text pairs is passed through the model and embedded into a shared latent space. The cosine similarity between all pairs is computed and softmax-normalized to calculate the image-to-text and text-to-image loss. The final loss is the mean of both losses [Rad+21]. The example is shown with a batch size of 6. The figure does not originate from the original paper, but is a custom visualization of the concept. Image-Text pair is taken from the MSCOCO train set [Lin+14], and do not refer to the contrastive loss of 6 pairs at the top of the figure. They are merely indicators of the intput to the model.

merical stability (the logarithm grows at a low rate), resulting in less drastic changes in the temperature during optimization and thereby making training more stable.

In the matrix $\boldsymbol{L}$, the cosine similarity between image i and text j in the batch is denoted by $L_{i,j}$, where the diagonal elements contain the similarity for positive pairs. To maximize the similarity between positive pairs (i, i), and minimize the similarity between negative pairs (i, j), with $i \neq j$, cross-entropy loss is yet again. The loss is defined exactly the same as for the vision-language contrastive loss (see Equation 27 and Equation 28).

$$\mathcal{L}_{\text{CLIP}} = \mathcal{L}_{\text{CL}} = \frac{1}{2} * \left(\mathcal{L}_{\text{CL}}^{\text{i2t}} + \mathcal{L}_{\text{CL}}^{\text{t2i}}\right) \tag{35}$$

CLIP only relies **only** on contrastive learning to train a vision-language model, and therefore requires a high batch size to achieve good results. The authors use a very large batch size of 32,768 [Rad+21]. An abstract illustration of the end-to-end training process of CLIP is shown in Figure 16.

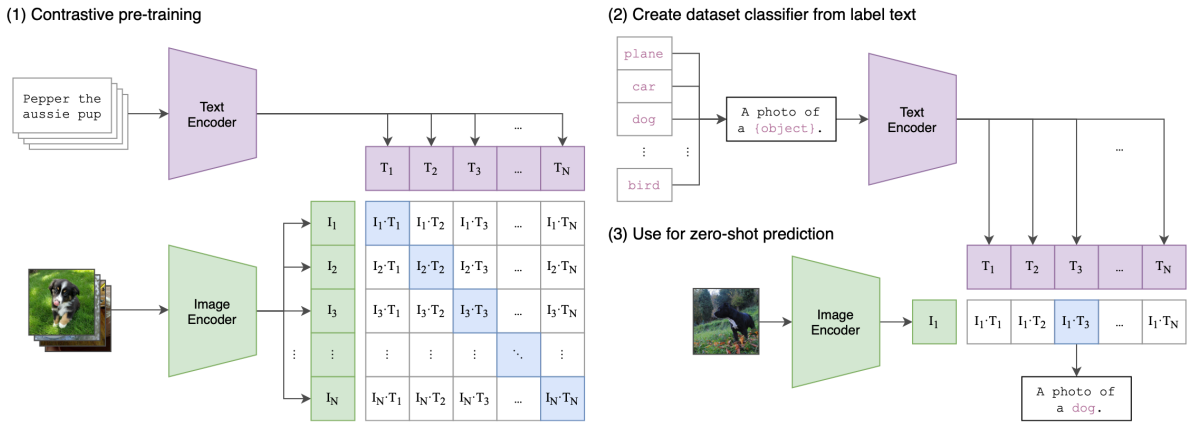

Figure 17: For zero-shot image classification, CLIP uses prompt engineering to create one classifier per class (2). The class whose classifier has the highest similarity (cosine) with the image representation is the predicted class (3) for the image [Rad+21].

2.7.2.2 Zero-Shot Image Classification

What makes CLIP special is its method of zero-shot image classification using the trained model. This capability is achieved through prompt engineering on the text encoder. For each class in the dataset, where image classification is desired, the name of the class is injected into a prompt template. The prompt template follows a structure like this: “a photo of a {class name}.”.

CLIP uses 80 different prompts, so for each class in the dataset, 80 distinct prompts are generated (similar to the example shown above). These 80 prompts are passed through the text encoder and text projection, resulting in 80 different text embeddings for one class. These embeddings are then averaged and normalized, yielding a single embedding per class. This embedding captures the semantic meaning of the class name, which the model learned through contrastive pretraining.

To classify an image, the image is passed through the image encoder and image projection, resulting in an image embedding. The cosine similarity between this image embedding and all class embeddings is calculated. The class corresponding to the text embedding with the highest similarity to the image representation is the predicted class for the image, as demonstrated in Figure 17.

The approach reaches a zero-shot accuracy of 76.2% on the validation set of ImageNet-1K [Rus+15], with a top-5 accuracy of 95% [Rad+21]. This is particularly impressive given that the model has never seen any images from the ImageNet-1K dataset during training, nor has it been trained on any image classification task. It merely achieves this accuracy through its cross-modal understanding between text and image. The model effectively “knows” how the ImageNet-1K classes look visually.

However, it is important to note that these results were based on a vision Transformer following the ViT-L/14@336px architecture for the image encoder. This architecture consists

of 24 layers, 16 attention heads, a hidden size of 1024, and processes images at a resolution of 336×336 pixels [Rad+21]. For the text encoder, a 12-layer Transformer was used, consisting of 12 attention heads and a hidden size of 768 [Rad+21]. According to HuggingFace, the model is 428 million parameters large[4]. Additionally, the model was trained on a custom dataset specifically developed for CLIP, consisting of 400 million image-text pairs [Rad+21]. Therefore, while impressive, which such a large model and dataset, the results are to be expected.

[4] https://huggingface.co/openai/clip-vit-large-patch14

3 Experimental Approach

With the background in place, we adopt an incremental experimental that begins with the simplest possible approach and incrementally builds upon the results and knowledge gained from each step. Our primary aim is to validate the effectiveness of knowledge distillation (KD), which is central to our approach throughout this thesis.

We start by testing KD in a unimodal setting to ensure that the technique functions correctly within our experimental framework. Specifically, we perform knowledge distillation of both an image model (Data2Vec2 [Bae+23]) and text model (BERT [Dev+19]). This step leverages extensively researched methods in unimodal KD and serves to confirm that our implementation is effective.

Building on the success of unimodal KD, we then advance to our main objective: creating a multimodal model using knowledge distillation from a unimodal teacher. Recognizing that distilling a multimodal model from a unimodal teacher is more challenging than distilling between unimodal models of the same architecture, we begin this part by employing a supervised teacher model. This means that the teacher model has been trained on labeled data and provides logits that the student model can regress. This approach effectively reproduces the method presented in SHRe [AVT17] to some extend, which has been demonstrated to work as a proof-of-concept.

Once we validate that this approach works similarly for our models, we proceed to the next phase by advancing to a self-supervised teacher. This step is crucial because our ultimate goal is to develop a model and procedure for efficiently creating a multimodal model entirely reliant on unlabeled data. Consequently, the teacher model and any pretrained modules used must not have been trained on labeled data. The aim of our experimental approach is to demonstrate the feasibility of this concept, serving as a proof-of-concept for creating efficient multimodal models without the need for any labeled data in the end-to-end training process.

4 Data Collection

Before we start, we first need to collect data for our experiments, which has to be both unimodal and multimodal. The requirement for multimodal data is obvious: We aim to align image and text, which requires a dataset of image-text pairs. Unimodal data is required for preliminary tests of classic unimodal knowledge distillation, on which we can then build. Further, we will utilize unimodal data for the evaluation of multimodal models on downstream tasks. After all, a multimodal model should not only excel in aligning modalities, but also in tasks that only involve on of the aligned modalities (vision and language). This also gives us the opportunity to compare the performance of unimodal and multimodal distilled models on the same tasks.

4.1 Unimodal Data

Collecting unimodal data does not pose an obstacle, as there are many highly curated and large datasets available. For image data, we select ImageNet-1K [Rus+15], which is an intuitive choice, as is features a high variety of content, is widely used for image classification, and, with 1.2 million training images, can be considered a medium-sized dataset. For comparison, current (August 2024) SOTA vision-language models have been trained on datasets spanning at least 14 million samples [Bao+22, Sin+21, Wan+23].

We will use this dataset for both knowledge distillation, and, most importantly, for the evaluation of image models using the ImageNet-1K validation accuracy metric, which is the most popular benchmark for computer vision models by far. We utilize the full dataset of the 2012 version, which contains 1.2 million images for training and 50,000 for validation [Rus+15]. The data can be downloaded from Huggingface's dataset hub[5] without any costs, merely requiring an account.

For raw text data, used in unimodal knowledge distillation of our text model, we select OpenWebText (OWT) [GC19]. This data was developed to replicate the datasets used to train GPT-2, and is also publicly available on HuggingFace[6]. The dataset consists of raw unstructured text, without any labels, which are not necessary for our distillation process. It is pub-

[5] https://huggingface.co/datasets/ILSVRC/imagenet-1k
[6] https://huggingface.co/datasets/Skylion007/openwebtext

lished as 21 chunks, and we select the first 6 for training and the 7th for validation, which is around 33% of the data. We do not collect the full dataset, as our subset already consists of more than 2.5 billion tokens, which we consider sufficient for our purposes. Even though the data is already preprocessed and cleaned, we further preprocess it by removing empty lines and null bytes, which we found to be quite common and leads to problems during encoding and training, as they provide no learnable information.

For benchmarking language models, including our multimodal models, on downstream tasks, we will use the GLUE benchmark [Wan+18]. GLUE, short for **G**eneral **L**anguage **U**nderstanding **E**valuation, is a collection of NLP datasets spanning four different tasks: Sentiment analysis (SST-2), grammar error detection (CoLA), sentence similarity (STS-B, MRPC, QQP), and natural language understanding (QNLI, MNLI, RTE). All 8 datasets are publicly available, and can also be accessed through HuggingFace[7].

The 8 datasets measure the performance of language models on the following tasks:

SST-2

Sentence classification of rotten tomatoes movie reviews into "negative" (1), "somewhat negative" (2), "somewhat positive" (3), and "positive" (4) [Soc+13].

CoLA

Is a binary classification tasks to test a models understanding of grammar: A model should output whether a sentence is grammatically correct (label: "acceptable" -> 1) or not (label: "unacceptable" -> 0) [WSB19].

STS-B

A regression task. The model is tasked with predicting the similarity between two sentences. The similarity score is in the interval $[0, 5] \subset \mathbb{R}$ [Cer+17].

MRPC

Is a binary classification taks. The training objective is paraphrase detection, meaning whether two sentences describe the same semantic concept [DB05].

QQP

The same as MRPC, instead of a simple sentence pair, the goal is to detect whether two questions are semantic duplicates, i.e. ask the same thing[8].

QNLI

A binary classification task, where the model has to predict whether a sentence is the answer to a question. Examples are of the form (question, sentence) [Raj+16, Wan+18].

[7] https://huggingface.co/datasets/nyu-mll/glue
[8] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Dataset	Training Examples
ImageNet-1K [Rus+15]	1.28M
OpenWebText (subset) [GC19]	13M[9]
GLUE [Wan+18]	990K (total)
Total	**15.27M**

Table 3: Unimodal datasets and their sizes used in this work.

RTE

A dataset of text pairs, where the model has to predict whether a hypothesis (sentence 2) can be inferred from a text (sentence 1) [DGM05, Wan+18]. The task is binary classification (hypothesis can be inferred, or not).

MNLI

A classification task, where the model has to predict whether a hypothesis can be inferred from the premise (entailment), contradicts the premise (contradiction), or is neutral (neutral). There a two versions available, MNLI matched and MNLI mismatched. Both consists of the same training dataset, but test set of MNLI mismatched consists of out-of-domain data, so sentence pairs about concepts not seen during training. It is therefore a better measure of generalization, compared to MNLI matched [WNB18].

MNLI

A classification task, where the model has to predict whether a hypothesis can be inferred from the premise (entailment), contradicts the premise (contradiction), or is neutral (neutral). There a two versions available, MNLI matched and MNLI mismatched. Both consists of the same training dataset, but the test set of MNLI mismatched consists of out-of-domain data, so sentence pairs about concepts not seen during training. It is therefore a better measure of generalization, compared to MNLI matched [WNB18].

WNLI

A binary classification task, where the model has to predict whether a hypothesis can be inferred from the premise (entailment) or not (contradiction) [LDM12].

A concrete example for each task can be found in Table 38. In total, we collect approximately **15.27 million** training examples, most of which are from OpenWebText. While the amount of training examples from OpenWebText is indeed correct, it is important to note that text data is significantly cheaper to obtain, and easier to handle, as it requires less disk space than image data, which is why we were able to collect that much text data without any problems.

[9]Number was calculated with a sequence length of 256 tokens. All the text we collected was split into slices of 256 tokens, and each slice was considered as one training example.

Dataset	Avg. Caption Length	Avg. Captions per Image	Images	Image-Text Pairs
COCO [Lin+14]	11.0	5.0	82,783	566,747
CC3M (subset) [Sha+18]	12.0	1.0	1,516,133	1,516,133
CC12M (subset) [Cha+21]	10.3	1.0	1,181,988	1,181,988
Total	-	-	2,780,904	**3,264,868**

Table 4: Multimodal Dataset used for aligning image and text.

4.2 Multimodal Data

For training multimodal models, specifically image-text models, datasets containing image-text pairs are required. We orient ourselves on the BEiTv3 paper, currently achieving SOTA performance on multimodal benchmarks [Wan+23]. The paper uses the datasets COCO [Lin+14], Visual Genome [Kri+17], Conceptual Captions 3M [Sha+18] and 12M [Cha+21], and SBU captions [OKB11], of which we select COCO, being the most popular and widely used dataset and therefore an intuitive choice, and a subset of both Conceptual Captions 3M and 12M.

While the COCO dataset can be downloaded in its entirety from the COCO website[10], both variants of Conceptual Captions, created by Google, only provide the url and caption for each image. This is because the images used come from a variety of sources on the internet, and have been uploaded by humans all over the world. The images stem from blog posts, news articles, social media, and other sources. Since Google does not own the rights to the images, they cannot provide them in a dedicated dataset, which is why there is no guarantee that all images will be available at the time of download. Because of this, we have to utilize not only Conceptual Captions 3M, but also the 12M variant to collect enough data. A favorable side effect this has is that our approach becomes more scalable due to the uncurated nature of CC12M [Cha+21, Sha+18], which we will elaborate on in the next section. The index of CC3M is available on the official Conceptual Captions website (training split)[11], and the index of CC12M can be found in the corresponding GitHub repository[12].

In total, we collect approximately **1TB** of data.

4.3 On Curated Datasets

The goal of this work is to develop a multimodal model that is cheap to train and does not rely on labeled data in the end-to-end process. That means not only should our multimodal model not require labeled data for training, but also any pretrained models and components used in the process.

[10] https://cocodataset.org/#download
[11] https://ai.google.com/research/ConceptualCaptions/download
[12] https://github.com/google-research-datasets/conceptual-12m

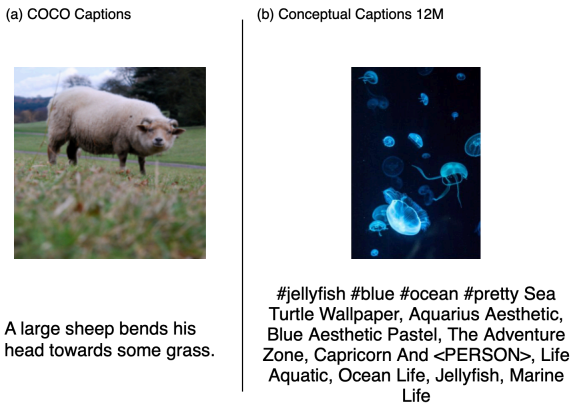

Figure 18: Side-by-side comparison of examples seen in COCO (a) and CC12M (b). While COCO features high quality images and detailed annotations, CC12M consists of in-the-wild image-text pairs from the internet. The latter enables scalability, as more data can be collected without the need for human annotation. The caveat is that the quality of the data is not guaranteed, and image-text pairs might be less correlated. Images and text in the figure have been taken from the COCO train set [Lin+14] and CC12M [Cha+21], respectively.

Whether image-text datasets, and, in fact, any multimodal dataset consisting of pairs of data, can be seen as curated or even labeled data is a matter of perspective. The difference between curated and labeled data lies in the purpose and level of human involvement: curated data focuses on the careful selection, organization, and cleaning of data to ensure quality and relevance, while labeled data involves explicit tagging or annotation of each example to provide a ground truth for training supervised models (which implies that the data is curated as well).

While image-text datasets are not labeled in the traditional sense, as in having a label for an image or text, the pairs themselves can be seen as labels. Single images or texts can be considered as in-the-wild data, i.e. data that appears naturally in the real world, like in books, articles, or on the internet, image-text pairs however require image and text to be paired together. This can be seen as less natural, as it requires a human to create the caption for an image, or vice versa, which is a form of labeling. The COCO dataset, for example, can be seen as labeled, as for each image a human created a caption with the specific intention of training machine learning models [Lin+14]. Consequently, whether multimodal learning can be seen as self-supervised learning, as it is often referred to in the literature [Bao+22, Sin+21, Wan+23], is debatable. With this in mind, creating a multimodal model that is scalable in the sense that it does not rely on labeled data, which is one of the most challenging aspects of AI research, is, if multimodal data is seen as labeled data, not possible.

However, there are multimodal data sources that are at the very least uncurated. One example is the alt-text of images on the internet. Even though the alt-text is created by humans, it is not created with the intention of creating data for machine learning, but rather to provide a description of the image for visually impaired people. Consequently, the data was gener-

ated naturally as a byproduct of a different task, and we therefore refer to any uncurated dataset as unlabeled data in this work.

This is exactly why we select both CC3M and CC12M, as they, especially CC12M, consists of in-the-wild image-text pairs from the internet [Cha+21, Sha+18]. This way of collecting data and training models is therefore significantly more scalable than using curated datasets specifically created for Machine Learning, and ensures that our approach to multimodal models can be applied to a wide range of tasks and domains without any explicit human intervention. A comparison between curated and labeled samples, and in-the-wild samples can be seen in Figure 18.

5 Experiments

5.1 Unimodal Knowledge Distillation

To validate whether traditional unimodal knowledge distillation, an undoubtedly simpler task than multimodal knowledge distillation, even works, we will first conduct experiments on unimodal knowledge distillation. We will then build on the results to develop multimodal knowledge distillation.

5.1.1 Vision

5.1.1.1 Method

Our approach to vision KD involves using a pretrained Data2Vec2 [Bae+23] image model as the teacher, and distilling a shallow version of this model, which is the student. We attribute our choice of Data2Vec2 to its effectiveness and consistency in self-supervised learning across image, text and audio. Data2Vec2 is a general framework to pretrain *unimodal* image, text, and audio models using self-supervised learning [Bae+23, Bae+22], which fits our philosophy of aligning modalities.

We approach the distillation by taking the first 6 Transformer blocks of the *pretrained* teacher model, which are exactly half of the 12 Transformer blocks in the teacher, and organize them into a smaller model. This smaller model also includes the pretrained cls token, patch projection, and positional encodings. Consequently, the student model is not trained from scratch, but already initialized with a subset of the teacher's weights. This is inspired by DistilBERT [San+19], a small BERT variant distilled from the normal BERT model [Dev+19]. As mentioned before, we use the first 6 Transformer blocks of the teacher model, which we found leads to better results than using every second layer. The resulting student model is with 43.1M parameters almost half the size of the teacher model, which has 85.9M parameters.

Data2Vec2 is a self-supervised model [Bae+23], and therefore does not provide a probability distribution over classes that can be predicted using KL-Divergence. Instead, we only have access to the model's activations for each layer, so we have to resort to feature-based knowledge distillation. One option would be to predict the teacher's output for the cls token $\boldsymbol{h}^t_{v,L,[\texttt{I\_CLS}]}$, which aggregates the high level content of the image, and then use the mean

squared error as the loss function. However, this neglects the activations for individual image patches and activations of intermediate layers.

This argument is quite similar to that of Data2Vec. The authors introduce "contextualized representations", which are the activations of all layers for each time step (image patch) of the input. Because of self-attention in Transformers, the activations for each image patch are influenced by all other image patches, and therefore not only encode information about a patches content, but also about its context in the image, i.e. the relationship to other patches [Bae+23, Bae+22]. Consequently, contextualized representations are more informative than a single cls token, as they encode information about the image at different levels of abstraction, and how the model aggregates low level features to high level concepts. Since the goal of KD is to "mimic" the behavior of a teacher model for a given input in a compressed way, this is the exact information that should be transferred from the teacher to the student. Simply predicting the cls token would only "mimic" what information the teacher extracts from the image, but not how the information is extracted.

While the dimensions of our student model match those of the teacher model, they both have a hidden size of $D = 768$ and intermediate size of $D_{\text{ff}} = 3072$, the number of layers in the student model ($L_s = 6$) is only half of that of the teacher model ($L_t = 12$). It is therefore not possible for each layer of the student model to mimic the behavior of the corresponding layer in the teacher model. Fortunately, experiments of the Data2Vec authors show that predicting the mean of all layer activations for each time step (or image patch, respectively) works as well as predicting the activations of each layer individually [Bae+22]. This suits our approach well, as the only mismatch between the teacher and student model is the number of layers, which is irrelevant when predicting the mean of all layer activations for each time step. The authors apply instance normalization to the activations of each layer before averaging, which is a normalization technique that works on each dimension of a sequence independently.

For a sequence of embeddings/representations with length T, instance normalization is defined as follows:

$$h'_{j,d} = \frac{h_{j,d} - \mu_d}{\sqrt{\sigma_d^2 + \varepsilon}}, \qquad \mu_k = \frac{1}{T}\sum_{t=1}^{T} h_{t,k}, \qquad \sigma_k^2 = \frac{1}{T}\sum_{t=1}^{T}\left(h_{t,k} - \mu_k\right)^2 \tag{36}$$

Even though the formula might look complicated, it is quite simple in practice. For each embedding dimension d, the mean μ_d and standard deviation σ_d are calculated over all time steps T. In the case of an embedding dimension of $D = 768$, this means for one sample (e.g. a sequence representing an image) 768 means and standard deviations are calculated, one for each embedding dimension. Then, for each time step j, the embedding at time step j is normalized by normalizing each dimension of the embedding independently, using the corresponding mean and standard deviation computed for that dimension [UVL17]. During the normalization, a small epsilon, e.g. $\varepsilon = 1e - 8$, is added to the standard deviation to prevent

division by zero. For an illustrative comparison between instance normalization, batch normalization and layer normalization, see Figure 34 in the Appendix. We define the operation $\mathrm{IN}(\cdot)$ as instance normalization on a sequence of embeddings $\boldsymbol{H}$.

$$\mathrm{IN}(\boldsymbol{H}) = [\boldsymbol{h}'_1, \boldsymbol{h}'_2, ..., \boldsymbol{h}'_T] \tag{37}$$

After instance norm and averaging, parameter-less layer normalization, denoted as $\mathrm{LN}(\cdot)$, is performed [Bae+23, Bae+22]. We perform all three operations likewise. The target and prediction are therefore given by:

$$\begin{aligned} \boldsymbol{H'}^{t}_{v,l} &= \mathrm{IN}\Big(\boldsymbol{H}^{t}_{v,l}\Big), l \in \{1, 2, ..., L_t\} \\ \boldsymbol{H'}^{s}_{v,l} &= \mathrm{IN}\Big(\boldsymbol{H}^{s}_{v,l}\Big), l \in \{1, 2, ..., L_s\} \end{aligned} \tag{38}$$

$$\begin{aligned} \widehat{\boldsymbol{H}}^{t}_{v} &= \frac{1}{L_t} \sum_{l=1}^{L_t} \boldsymbol{H'}^{t}_{v,l} \\ \widehat{\boldsymbol{H}}^{s}_{v} &= \frac{1}{L_s} \sum_{l=1}^{L_s} \boldsymbol{H'}^{s}_{v,l} \end{aligned} \tag{39}$$

$$\begin{aligned} \boldsymbol{Y} &= \Big[\boldsymbol{y}_{[\texttt{I\_CLS}]}, \boldsymbol{y}_1, ..., \boldsymbol{y}_N\Big] = \mathrm{LN}\Big(\widehat{\boldsymbol{H}}^{t}_{v}\Big) \\ \widehat{\boldsymbol{Y}} &= \Big[\hat{\boldsymbol{y}}_{[\texttt{I\_CLS}]}, \hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N\Big] = \mathrm{LN}\Big(\widehat{\boldsymbol{H}}^{s}_{v}\Big) \end{aligned} \tag{40}$$

The loss for a single sample (image) is defined in the following:

$$\begin{aligned} &\mathcal{L}_{\mathrm{KD}}\Big(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}\Big) = \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|_2^2 = \\ &\frac{1}{N+1}\left(\sum_{n=1}^{N} \mathcal{L}_{\mathrm{MSE}}(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) + \mathcal{L}_{\mathrm{MSE}}\Big(\boldsymbol{y}_{[\texttt{I\_CLS}]}, \hat{\boldsymbol{y}}_{[\texttt{I\_CLS}]}\Big)\right) \end{aligned} \tag{41}$$

We denote $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ as the average representation for image patch i over all layers from the teacher and student model, respectively. This includes instance norm before averaging, and layer norm after averaging. $\mathcal{L}_{\mathrm{MSE}}(\cdot, \cdot)$ is the mean squared error between two vectors, defined in Equation 1.

5.1.1.2 Distillation

We distill the student model by minimizing the loss defined in Equation 41 using the AdamW optimizer [LH19] with a base learning rate of 5e-4. We train for 10 epochs with a batch size of 256 on the training set of ImageNet-1K [Rus+15], and run validation after every epoch on the validation set of ImageNet-1K. As with Data2Vec2, our approach does not involve labels, so we also use the loss defined in Equation 41 for validation. The total number of parameters involved in the distillation process is 129M, of which 43.1M trainable belong to the student model, and 85.9M frozen parameters to the teacher model.

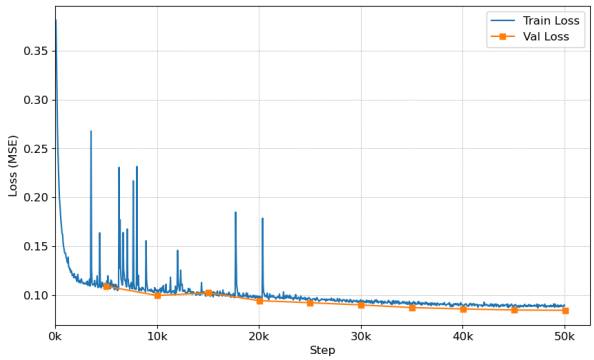

Figure 19: Training loss vs. validation loss during distillation of the Data2Vec2 image model.

Since we use the same architecture as Data2Vec2 for our teacher model, the images, being of size 224×224, which is the size we will consistently use for all experiments, are split into 16x16 patches, which results in a sequence length of $N = 196$. A cls token is added to the sequence, which results in a total sequence length of $N + 1 = 197$. All embeddings have a dimension of $D = 768$.

For data augmentation we decide to use the same augmentation strategy using during the training of the teacher model. This ensures that we get the training targets from the same distribution the teacher has seen, and that we do not generate data for which the teacher might give inaccurate representations. The augmentations involve (1) cropping a random portion of an image and resizing it back to the original image resolution (224×224), (2) performing a random horizontal flip with probability 0.5, and (3) normalizing the image RGB channels with the mean and standard deviation of the ImageNet-1K dataset [Bae+23].

Detailed hyperparameters are provided in Table 32.

We show the evolution of the training and validation loss during training in Figure 19. We observe the traditional convergence behavior of a model during training, and the validation loss is consistently lower than the training loss, which is a sign of good generalization. There are some peaks in the training loss, which are likely due to a high learning rate, but they do not affect the validation loss, which is why do not investigate them further. As we do not have access to any other metric than the MSE loss during training, we have to evaluate the student by finetuning on downstream tasks, which follows in the next section. This will answer whether the distillation actually yields a model competetive in performance to the teacher model, and if the knowledge transfer was successful.

5.1.1.3 Finetuning

To get a sense of how well the student model has learned from the teacher, we evaluate the student model by finetuning it on the downstream image classification tasks of CIFAR-10 [Kri12], CIFAR-100 [Kri12] and ImageNet-1K [Rus+15]. For that, we load the trained student model and add Layer Normalization and a linear classifier on top of it. The output of the student model is a sequence of embeddings, one embedding for each image patch, and one

Layer no.	0	*1*	*2*	*3*	*4*	*5*	*6*	7
Learning rate	2.3e-4	2.8e-4	3.5e-4	4.3e-4	5.3e-4	6.6e-4	8.1e-4	1e-3

Table 5: Learning rates for different blocks/layers when finetuning on ImageNet-1K. Cursive layer numbers indicate Transformer blocks. The learning rates are calculated using a base learning rate of 1e-3 and a layer decay of 0.81.

cls token embedding. We follow the approach of Data2Vec [Bae+23, Bae+22] and BEiTv2 [Pen+22], and take the mean over all patch embeddings as the output of the student model, which is then passed to the layer normalization and linear classifier (cls token embedding is ignored). For all three tasks we perform full finetuning, i.e. we finetune all layers of the student model on the downstream task, and linear evaluation, we only train the added layer norm and linear classifier on top of the student model. For pytorch pseudocode of linear evaluation and full finetuning see Code 1.

For data augmentation during finetuning we use RandAugment [Cub+20], mixup [Zha+18] and cutmix [Yun+19] augmentation, and random erasing [Zho+20]. The hyperparameters for these augmentations are provided in Table 33, and have been selected based on the values used in BEiTv2 [Pen+22], Data2Vec [Bae+22], and Data2Vec2 [Bae+23]. We refrain from explaining the augmentation techniques in detail here, as they are well documented in the respective papers.

For finetuning on ImageNet-1K we use a base learning rate of 1e-3 in combination with layer decay. Layer decay is a technique to reduce the base learning rate for each layer of the model by a certain factor [Bae+23, Bao+22]. The goal is to have lower learning rates for layers closer to the input, and higher learning rates for layers closer to the output. This ensures that low-level features learned during pretraining or distillation are not destroyed during finetuning. We use a decay factor of 0.81, which is derived by scaling the layer decay used in Data2Vec2 [Bae+23], from which we extract layers for the student model, by the square root. We use scaling instead of the value used in Data2Vec2, which is 0.65 ($\sqrt{0.65} \approx 0.81$), as we only have half the number of layers in the student model, and can therefore afford larger learning rates for the lower layers. The actual learning rate for a layer l is then calculated by:

$$\texttt{lr}_l = \texttt{lr}_{\text{base}} * d^{L_s+1-l} \tag{42}$$

$\texttt{lr}_{\text{base}}$ denotes the base learning rate, which is 1e-3, L_s is the number of layers in the student model (6), and d is the decay factor, which is 0.81. The resulting learning rates can be seen in Table 5.

We show the learning rates for 8 layers in total, even though the student model only has 6 Transformer blocks. This is because we count all weights before the first Transformer block as layer 0, which includes the weights used for projecting patches to embeddings, the cls token, and the positional encodings. Correspondingly, layer 7 includes the weights for the

Method	Finetune	Linear eval
Data2Vec2	84.5	-
BEiTv2	85.0	80.1
ResNet-101	80.1	-
DistilData2Vec2 (ours)	75.0	56.2

Table 6: Comparison of finetuning and linear evaluation results with SOTA self-supervised models on ImageNet-1K.

Dataset	Approach	Accuracy
CIFAR-10	Finetune	97.0
	Linear Probe	68.4
CIFAR-100	Finetune	85.1
	Linear Probe	46.2

Table 7: Results of finetuning and linear evaluation of our distilled Data2Vec2 image model on CIFAR-10 and CIFAR-100.

layer norm and linear classifier on top of the student model, which are initialized randomly and can be assigned a higher learning rate than the other layers.

For all hyperparameters used on the downstream tasks see Table 33.

The results, displayed in Table 6 and Table 7, show that while the student model is not able to outperform the teacher model (Data2Vec2), as well as all other models we compare to, it is able to achieve acceptable performance on all 6 evaluations considering both BEiTv2 and Data2Vec2 are based on the ViT-B/16 architecture [Bae+23, Pen+22], which is twice as large as the student model. We also compare to the ResNet-101 model from the original ResNet paper [He+16], which has 44.5M parameters, and is therefore comparable in size to the student model, but has been trained only supervised.

5.1.2 Language

5.1.2.1 Method

For knowledge distillation of a language model, we decide against the intuitive choice of distilling the corresponding Data2Vec2 language model, and opt for distilling a BERT model instead. We do this for two reasons. First, we will also use BERT as the text encoder in our multimodal model, so for consistency we will use BERT for the unimodal distillation as well. Second, as mentioned before, there already exists a distilled version of BERT, DistilBERT [San+19], to which we can directly compare our results.

We use the same approach as for the image model, and distill a smaller version of BERT from a pretrained BERT model. As with our image model, we take the first 6 Transformer blocks

of the teacher model and organize them into a smaller model together with the embedding layer and positional encodings. The student model is therefore, again, initialized with a subset of the teacher's weights.

The distillation loss is defined analogously to the distilled image model, and is defined in Equation 41. We do not need to introduce changes, as the loss is applicable to any Transformer, regardless of the modality, making it a universal loss function for feature-based knowledge distillation. This follows the philosophy of Data2Vec2, which uses the same loss for all modalities [Bae+23].

5.1.2.2 Importance of Sequence Length

Setting the maximum sequence length as long as possible is crucial when regressing the mean activation of all teacher layers for each token. Self-attention mechanisms enable each token to encode not only its own information but also contextual information from other tokens in the sequence. This phenomenon, referred to as "contextualized targets" by the authors of Data2Vec [Bae+23, Bae+22], becomes more pronounced with longer sequences. A larger number of tokens in the sequence increases the opportunities for tokens to attend to one another, thereby enriching contextual representations and enhancing the model's ability to capture complex token relationships.

For instance, in the short phrase "a dog", each token can attend to only one other token, which limits the model's capacity to understand the broader context and relationships. In contrast, the sentence "a dog is playing in the garden with a tennis ball" provides a richer context with more tokens. This expanded context allows the model to capture a wider variety of relationships between tokens, offering the student model more opportunities to learn from the teacher model's outputs. A visualization of the attention between the different tokens of both sentences is provided in Figure 20. Here, we can see that a longer sequence allows tokens to attend to a high variety of other tokens, leading to relationships that represent real-world scenarios. The token "dog" attends mostly to tokens "playing", "garden", "tennis" and "ball", which combined represent the context of the sentence and a real-world scenario. The representation of the token "dog" therefore encodes information about a dog and what it is doing, which can be learned by the student model when regressing the teacher's outputs. In contrast, the token "dog" in the short sentence "a dog" can only attend to the token "a" (and the special tokens [CLS] and [SEP]), which does not provide any context about the dog. However, the disadvantage of longer sequences is that they require more memory and computational resources, as the self-attention mechanism has a quadratic complexity with respect to the sequence length.

A variable sequence length is less relevant for the image model in Section 5.1.1, as the size of an image is fixed to 224×244 pixels, so there is no variable sequence length.

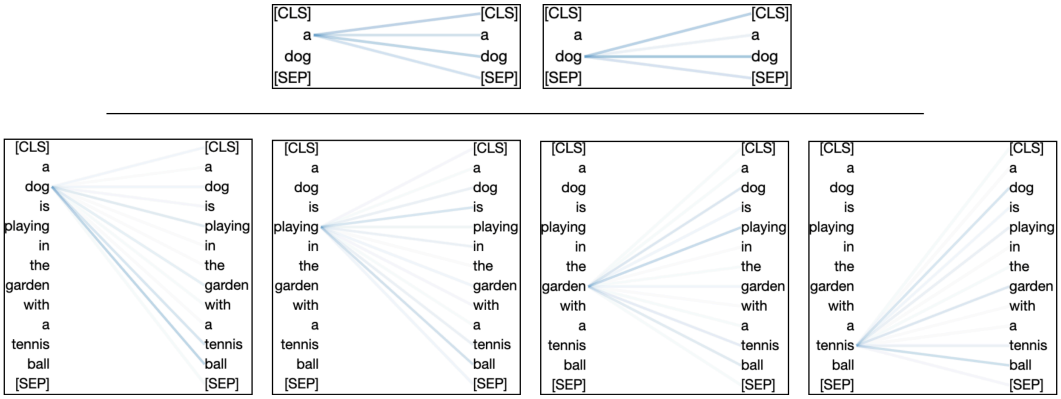

Figure 20: Visualization of the attention between tokens in a short and longer sentence. Longer sequences allow for more complex relationships between tokens that represent the real-world context of the sentence. Attention scores have been taken from the first attention head of the first Transformer layer of the trained uncased BERT model used for distillation.

5.1.2.3 Distillation

During training, we set the maximum sequence length to 256, which is the maximum sequence length that can fit on a single GPU with 24GB of memory (RTX 4090).

The BERT model is distilled on a subset of the OpenWebText dataset [GC19], introduced in Section 4.1, of which the text is tokenized into subwords using the BERT tokenizer. We use the uncased variant, meaning all tokens are first converted to lowercase: To the model, there is no difference between “Dog” and “dog”.

We validate the student model on the dedicated validation dataset of OWT we introduced in Section 4.1.

The model is trained for only one epoch, as we found the model to converge quickly, and because the text data we collect yields almost 1M batches with a sequence length of 256 per sample. For comparison, the 1.2M images of ImageNet-1K [Rus+15] yield 5004 batches of size 256 for a single epoch.

Other hyperparameters used, including the learning rate, are similar to that of the image model, and are provided in Table 34.

Before the loss (Equation 47) is applied, we perform a similar preprocessing as for the image model. However, following Data2Vec2 [Bae+23], we do not apply layer normalization on the averaged activations, and the loss is only calculated for non-padding tokens. This was not relevant in the image model, as there is no padding in the image data.

$$\begin{aligned} \boldsymbol{H'}^{t}_{w,l} &= \mathrm{IN}\left(\boldsymbol{H}^{t}_{w,l}\right), l \in \{1, 2, ..., L_t\} \\ \boldsymbol{H'}^{s}_{w,l} &= \mathrm{IN}\left(\boldsymbol{H}^{s}_{w,l}\right), l \in \{1, 2, ..., L_s\} \end{aligned} \tag{43}$$

$$\begin{aligned}\widehat{\boldsymbol{H}}_w^t &= \frac{1}{L_t}\sum_{l=1}^{L_t} \boldsymbol{H'}_{w,l}^t \\ \widehat{\boldsymbol{H}}_w^s &= \frac{1}{L_s}\sum_{l=1}^{L_s} \boldsymbol{H'}_{w,l}^s\end{aligned} \tag{44}$$

$$\begin{aligned}\boldsymbol{Y} &= \left[\boldsymbol{y}_{[\texttt{T\_CLS}]}, \boldsymbol{y}_1, ..., \boldsymbol{y}_M, \boldsymbol{y}_{[\texttt{SEP}]}\right] = \widehat{\boldsymbol{H}}_w^t \\ \widehat{\boldsymbol{Y}} &= \left[\hat{\boldsymbol{y}}_{[\texttt{T\_CLS}]}, \hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_M, \hat{\boldsymbol{y}}_{[\texttt{SEP}]}\right] = \widehat{\boldsymbol{H}}_w^s\end{aligned} \tag{45}$$

$$\mathcal{L}'_{\text{MSE}}(\boldsymbol{y}_m, \hat{\boldsymbol{y}}_m) \coloneqq \begin{cases} 0 & , \text{if } \boldsymbol{e}_m^w = \boldsymbol{t}_{[\texttt{PAD}]} \\ \mathcal{L}_{\text{MSE}}(\boldsymbol{y}_m, \hat{\boldsymbol{y}}_m), & \text{else} \end{cases} \tag{46}$$

$$\begin{aligned}\mathcal{L}_{\text{KD}}\left(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}\right) = \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|_2^2 = \\ \frac{1}{M+2}\left(\sum_{m=1}^{M} \mathcal{L}'_{\text{MSE}}(\boldsymbol{y}_m, \hat{\boldsymbol{y}}_m) + \mathcal{L}_{\text{MSE}}\left(\boldsymbol{y}_{[\texttt{T\_CLS}]}, \hat{\boldsymbol{y}}_{[\texttt{T\_CLS}]}\right) + \mathcal{L}_{\text{MSE}}\left(\boldsymbol{y}_{[\texttt{SEP}]}, \hat{\boldsymbol{y}}_{[\texttt{SEP}]}\right)\right)\end{aligned} \tag{47}$$

Here, $\boldsymbol{e}_m^w$ denotes the embedding of token m in the sequence, and the loss is ignored, i.e. 0, if the token is the padding token, denoted $\boldsymbol{t}_{[\texttt{PAD}]}$.

We refrain from showing and analyzing the training and validation loss, as it is difficult to judge the quality of the student model just by looking at the distillation loss. Even though a low distillation loss is a good sign, there are no additional metrics to express how well the language understanding of our student is (we actually observe a similar behavior of the loss as in the image KD, see Figure 19). We therefore directly advance to finetuning the distilled student model on downstream tasks.

5.1.2.4 Finetuning

To get a sense of the language understanding capabilities of the trained/distilled student model, we finetune it on all GLUE benchmark tasks [Wan+18], which are described in Section 4.1, and visualized in Table 38.

Model Setup

To perform finetuning, we load the weights of the trained student model. After an example (e.g. sentence pair) is passed through the Transformer layer, the representation $\boldsymbol{h}_{w,L_s,[\texttt{T\_CLS}]}$ of the [T_CLS] token is extracted, and passed through the BERT pooler [Dev+19], which is a linear layer followed by a Tanh activation function. The linear layer retrains the input dimensionality of the [CLS] token, which is 768. The weights of the pooler come directly from the BERT model, and are also further trained as part of our finetuning. The pooler is followed by a dropout layer, for which the dropout probability is set to $p = 0.1$ for all tasks (shown in Table 35), and is followed by a linear classification layer, which maps the representation to

the number of classes of the downstream task. If the task is regression, for example sentence similarity in [0, 5] for STS-B [Cer+17], the second linear layer returns a scalar value. The classification layer is initialized randomly, and trained from scratch. Pytorch pseudocode for the forward pass is provided in Code 2.

Details on Tokens

There are two important things to consider when tokenizing the input for the GLUE tasks.

First, we set the maximum sequence length to 256, which is the same as for the distillation process. In theory, it is possible to set the maximum sequence length to 512, which is the maximum sequence length BERT can handle [Dev+19] without interpolation. However, this would (1) cause problems with memory, as those sequences are too long for a 24GB GPU. Furthermore, (2) the positional encoding of the student model comes directly from the teacher model, which is BERT. The positional encoding $\boldsymbol{T}_w^{\text{pos}}$ of BERT is trainable, and has one positional encoding $\boldsymbol{t}_{\text{pos}}^w$ for each position in the sequence. Since we only distill with a sequence length of 256, only the first 256 positional encodings are actually further trained during distillation, meaning that the positional encodings for positions 257 to 512 are not trained further, and therefore still the same as in the normal BERT model. That means they are not "used" to a BERT model that has only 6, instead of 12, Transformer blocks, and therefore might not be optimal for the student model.

In general, a sequence length of 256 is acceptable, as most of the GLUE tasks rarely have examples that exceed this length. If an example is longer than 256 tokens, it is truncated to the first 256 tokens. If an example consists of a sentence pair, both sentences are tuncated equally, so that the total length of the sequence is 256.

Second, if the task consists of sentence pairs, we add a special token-type embedding to each token before the positional encoding is added. This, together with the [T_SEP] between both sentences, helps the model to better differentiate between the two sentences in the sentence pair. The first token-type embedding $\boldsymbol{t}_{\texttt{[TYP\_1]}}^w$ is added to each token of the first sentence, and the second token-type embedding $\boldsymbol{t}_{\texttt{[TYP\_2]}}^w$ is added to each token of the second sentence:

$$\begin{aligned}\boldsymbol{E}_w = \Big[&\boldsymbol{e}_{\texttt{[T\_CLS]}}^w, \boldsymbol{e}_1^w, \boldsymbol{e}_2^w, ..., \boldsymbol{e}_{M_1}^w, \boldsymbol{e}_{\texttt{[T\_SEP]}}^w, \\ &\boldsymbol{e}_{M_1+1}^w, \boldsymbol{e}_{M_1+2}^w, ..., \boldsymbol{e}_{M_1+M_2}^w, \boldsymbol{e}_{\texttt{[T\_SEP]}}^w\Big] \in \mathbb{R}^{(M_1+M_2+3)\times D}\end{aligned} \tag{48}$$

$$\begin{aligned}\boldsymbol{E}'_w = \Big[&\boldsymbol{t}_{\texttt{[TYP\_1]}}^w + \boldsymbol{e}_{\texttt{[T\_CLS]}}^w, \boldsymbol{t}_{\texttt{[TYP\_1]}}^w + \boldsymbol{e}_1^w, \\ &\boldsymbol{t}_{\texttt{[TYP\_1]}}^w + \boldsymbol{e}_2^w, ..., \boldsymbol{t}_{\texttt{[TYP\_1]}}^w + \boldsymbol{e}_{M_1}^w, \\ &\boldsymbol{t}_{\texttt{[TYP\_1]}}^w + \boldsymbol{e}_{\texttt{[T\_SEP]}}^w, \boldsymbol{t}_{\texttt{[TYP\_2]}}^w + \boldsymbol{e}_{M_1+1}^w, \\ &\boldsymbol{t}_{\texttt{[TYP\_2]}}^w + \boldsymbol{e}_{M_1+2}^w, ..., \boldsymbol{t}_{\texttt{[TYP\_2]}}^w + \boldsymbol{e}_{M_1+M_2}^w, \\ &\boldsymbol{t}_{\texttt{[TYP\_2]}}^w + \boldsymbol{e}_{\texttt{[T\_SEP]}}^w\Big]\end{aligned} \tag{49}$$

$$\boldsymbol{H}_{w,0} = \left[\boldsymbol{h}_{w,0,\texttt{[T\_CLS]}}, \boldsymbol{h}_{w,0,1}, ..., \boldsymbol{h}_{w,0,M_1+M_2}, \boldsymbol{h}_{w,0,\texttt{[T\_SEP]}}\right] = \boldsymbol{E}'_w + \boldsymbol{T}^{\text{pos}}_w \tag{50}$$

The representations $\boldsymbol{t}^w_{\texttt{[TYP\_1]}}$ and $\boldsymbol{t}^w_{\texttt{[TYP\_2]}}$ of the token-type embeddings are part of the BERT model [Dev+19], but are not used during distillation, as there are no sentence pairs during distillation. However, during finetuning on sentence pairs, they are required, and we take the pretrained token-type embeddings from the BERT model and also train them during finetuning.

For examples of sentence pairs see Table 38.

Hyperparameters

Since for most tasks the amount of training data is marginal, e.g. CoLA [WSB19] has only 8.5k training samples, we do not use the layer decay technique as for the image model, and directly select a very low learning rate. Most of the hyperparameters are inspired by BERT [Dev+19], and Data2Vec [Bae+23, Bae+22], and are provided in Table 35. We increase the number of epochs and lower the batch size if the dataset, like CoLA, is very small. This ensures that we have a sufficient number of updates for the model to learn from the data.

Results

The results of finetuning our distilled BERT model, which we denote as F-DistilBERT (for feature-based distilled BERT), are shown in Table 8, and we observe a similar performance to DistilBERT [San+19]. All scores are based on the dev sets of the respective tasks, as the test datasets, if available, usually do not provide labels.

We are able to retain 96.7% of the performance of BERT, which is almost the same as the 96.8% of DistilBERT. Noteably, we outperform DistilBERT on the RTE [DGM05] task by more than 7 percentage points, and even record the best score of all methods we compare to on WNLI [LDM12]. The latter is most likely due to the fact that WNLI is a very small dataset, with 635 training and only 71 dev samples. Both DistilBERT and F-DistilBERT have considerable less parameters than BERT, which makes them less prone to overfitting on small datasets, and the performance on 71 samples will be prone to noise. F-DistilBERT is also able to achieve a higher performance on CoLA [WSB19] and SST [Soc+13], compared to DistilBERT, which is a sign that the knowledge transfer through feature-based distillation was successful.

We do not investigate possible improvements, as the focus of this work lies on multimodal models. Nonetheless, the results of unimodal distillation provide a good foundation on which we can build in the following sections, and we will now proceed to multimodal distillation.

	MNLI	QNLI	RTE	MRPC	QQP	STS-B	CoLA	SST	WNLI	**Score**
ELMo [Pet+18]	68.6	71.1	53.4	76.7	86.2	70.4	44.1	91.5	56.3	68.7
BERT [Dev+19]	**86.7**	**91.8**	**69.3**	**88.6**	**89.6**	**89.0**	**56.3**	**92.7**	53.5	**79.5**
DistilBERT [San+19]	82.2	89.2	59.9	87.5	88.5	86.9	51.3	91.3	56.3	77.0
F-DistilBERT (ours)	81.2	88.0	67.64	85.0	86.5	81.0	55.1	91.4	**56.4**	76.9

Table 8: Comparison of finetuning results on the GLUE benchmark tasks [Wan+18] with other models. Results for BERT [Dev+19], DistilBERT [San+19], and ELMo [Pet+18] are taken from the DistilBERT paper [San+19]. Bold scores indicate the best score for the respective task, while underlined scores indicate the best score for distilled models. The metrics, and therefore the scores shown, to evaluate the models are task-specific, and shown in Table 35.

5.2 Multimodal Knowledge Distillation

5.2.1 Transformer SHRe

Before we develop an end-to-end self-supervised approach to multimodal knowledge distillation, we first follow the approach of SHRe [AVT17] and train a multimodal model with a probability distribution over the classes as the prediction target. This allows us to closely observe the impact of switching from a supervised to a self-supervised teacher model on the student model's performance. Moreover, it allows us to gradually increase the complexity of our approach and build on our previous advancements.

5.2.1.1 Method

5.2.1.1.1 Architecture

What makes our approach different from SHRe is that we use a language Transformer as the text encoder, and a vision Transformer as the image encoder, bringing us closer to a unified architecture. In contrast, SHRe uses 2D convolutions and 1D convolutions for the image and text, respectively. For now, the shared encoder remains a 3-layer MLP as in SHRe [AVT17]. Since the output of the image and text encoder is a sequence of features, but we use a normal MLP as the shared encoder, we have to reduce the sequence of features to a single represenation. This is done by taking the representation of the [I_CLS] and [T_CLS] token from the image and text encoder, respectively, and aligns with our first implementation of a multimodal model shown in Figure 10. This represenation, namely $\boldsymbol{h}_{v,L_s,\texttt{[I\_CLS]}}$ and $\boldsymbol{h}_{w,L_s,\texttt{[T\_CLS]}}$, is then passed to the shared encoder, which produces the final multimodal representations. Here, L_s denotes the number of Transformer layers in the image and text encoder, respectively, which is defined as $L_s = 6$.

To keep the model size manageable, we will resort the the same approach as in our unimodal experiment, and use 6 Transformer layers for the image and text encoder, respectively. This also gives us the opportunity to directly compare the performance of the image and text encoder from our multimodal model to that of the unimodal models (Section 5.1). This can be done by evaluating for instance the image encoder of the multimodal model on image classification tasks, and then comparing its performance to the results observed for the unimodal image KD model of Section 5.1.1 on the same tasks. A performance similar to that of the strictly unimodal models would indicate that multimodal pretraining yields strong unimodal encoders as a byproduct. As argued by the authors of FLAVA [Sin+21], the aforementioned is in fact even a requirement for multimodal models. This can be attributed to the fact that an alignment in the shared encoder is only possible if the unimodal encoders generate features from which the shared encoder can extract modality-invariant information, like the semantic content of an image or text. If the unimodal encoders are not able to extract high-level features, then neither will the shared encoder be able to extract modality-invariant information, nor will the features be useful in the corresponding unimodal downstream tasks.

As done in our unimodal experiments, we initialize the image and text encoder using the embeddings, positional encodings, and the first 6 Transformer layers from Data2Vec2 [Bae+23] and BERT [Dev+19], respectively. The shared encoder will be initialized randomly. For an illustration of the architecture we refer to the same depiction used for SHRe, shown in Figure 15. The only difference is that we only use image and text encoders, which are now Transformers instead of CNNs.

The shared encoder, which is a 3-layer MLP, is implemented by using the same 2-layer MLP module as present in Transformer layers, and adding an additional LayerNorm and linear layer on top of it. Given the output from the text encoder, the forward pass of the shared encoder is then given in the following, and changes for the image encoder accordingly:

$$\begin{aligned} \boldsymbol{h}'_{w,K} &= \boldsymbol{h}_{w,L_s,[\texttt{T\_CLS}]}\boldsymbol{W}_1^T + \boldsymbol{b}_1 \\ \boldsymbol{h}''_{w,K} &= \mathrm{LN}\big(\mathrm{GELU}\big(\boldsymbol{h}'_{w,K}\big)\big)\boldsymbol{W}_2^T + \boldsymbol{b}_2 \end{aligned} \tag{51}$$

$$\boldsymbol{h}'''_{w,K} = \mathrm{LN}\big(\boldsymbol{h}''_{w,K}\big)\boldsymbol{W}_3^T + \boldsymbol{b}_3 \tag{52}$$

We consider the 3-layer MLP as a single layer stacked on the image and text encoder, and therefore denote the layer number as $K = L_s + 1 = 7$. The operation done in Equation 51 is analogous to the definition of the MLP layers in a Transformer layer, defined in Equation 15. We choose a different notation here to also capture the result $\boldsymbol{h}'_{w,K}$, or $\boldsymbol{h}'_{v,K}$ for an image, of the first linear layer (the latter is represented by parameters $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$), which is important when defining the loss in the next section.

The whole model has a total of around 114.2M parameters, which is broken down in Table 9.

Component	# Params
Image Encoder	42.3M
Text Encoder	66.4M
Shared Encoder	5.5M
Total	114.2M

Table 9: A summary of the number of parameters of the Transformer SHRe model.

The 24M parameters the text encoder has more than the image encoder can be attributed to the embedding matrix of the text encoder, which alone has 23.4M parameters. Since we use parts of a pretrained BERT model, we also have to resort to using the BERT tokenizer and vocabulary. The vocabulary consists of 30522 (sub)words, and the embedding matrix has a dimensionality of 768 ($30522 * 768 = 23.4M$). The remaining parameters are related to the BERT-specific implementation of positional encodings.

While 115M parameters can be considered as quite large, considering we strive to build efficient(/smaller) models, it is still significantly smaller than the vision-language models we compare to. For example, VLMo [Bao+22] has 562M[13], CLIP [Rad+21] has more than 400M[14], and BEiT-3 [Wan+23] has 1.9B parameters [Wan+23].

5.2.1.1.2 Training Objective

Since we start with using a supervised teacher, the loss for knowledge distillation will remain KL-Divergence. As the application of the KL-Divergence is two-fold, once for the prediction based on the image and once for the prediction based on the caption, we provide a refined version of the loss function. As a preliminary step, we define the softmax normalization of a vector $\boldsymbol{u}$ as follows:

$$\begin{aligned} \pi(\boldsymbol{u}) = \boldsymbol{z} &= [z_0, z_2, ..., z_{C-1}] \in \mathbb{R}^C \\ z_i &= \frac{\exp(u_i)}{\sum_{j=0}^{C-1} \exp(u_j)} \end{aligned} \tag{53}$$

This allows us to generate a probability distribution over the classes for the logits generated by the teacher for the image, and for the logits generated by the student for image and caption. The loss for knowledge distillation is then given by:

[13] https://github.com/microsoft/unilm/tree/master/vlmo
[14] https://huggingface.co/openai/clip-vit-large-patch14

$$
\begin{aligned}
\mathcal{L}_{\mathrm{KD}} = \\
\frac{1}{2} * \mathcal{L}_{\mathrm{KD}}^{v} + \frac{1}{2} * \mathcal{L}_{\mathrm{KD}}^{w} = \\
\frac{1}{2} * D_{\mathrm{KL}}\big(\pi(\boldsymbol{p}), \pi\big(\boldsymbol{h}_{v,K}'''\big)\big) + \frac{1}{2} * D_{\mathrm{KL}}\big(\pi(\boldsymbol{p}), \pi\big(\boldsymbol{h}_{w,K}'''\big)\big)
\end{aligned}
\tag{54}
$$

$\boldsymbol{p}$ denotes the logits generated by the teacher for the image, $\boldsymbol{h}_{v,K}'''$ the logits generated by the student for the image, and $\boldsymbol{h}_{w,K}'''$ the logits generated by the student for the caption. All are in $\mathbb{R}^{1000}$ for the 1000 classes of ImageNet.

We decide to replace the ranking loss of SHRe with the contrastive loss introduced by CLIP [Rad+21], and explained in Section 2.5.2.2. We justify this decision with the fact that vision-language contrast has become the de-facto standard for multimodal self-supervised learning, and has lead models like CLIP [Rad+21], VLMo [Bao+22], and CoCa [Yu+22] to reach state-of-the-art results in image-text retrieval.

We apply this loss on the outputs of all three MLP layers of the shared encoder, as we want to enforce the shared encoder to generate aligned representations in all layers. The refined contrastive loss is then given by:

$$
\begin{aligned}
\mathcal{L}_{\mathrm{CL}} = \\
\frac{1}{3} * (\mathcal{L}_{\mathrm{CL}'} + \mathcal{L}_{\mathrm{CL}''} + \mathcal{L}_{\mathrm{CL}'''}) = \\
\frac{1}{6}\mathcal{L}_{\mathrm{CL}'}^{\mathrm{i2t}} + \frac{1}{6}\mathcal{L}_{\mathrm{CL}'}^{\mathrm{t2i}} + \\
\frac{1}{6}\mathcal{L}_{\mathrm{CL}''}^{\mathrm{i2t}} + \frac{1}{6}\mathcal{L}_{\mathrm{CL}''}^{\mathrm{t2i}} + \\
\frac{1}{6}\mathcal{L}_{\mathrm{CL}'''}^{\mathrm{i2t}} + \frac{1}{6}\mathcal{L}_{\mathrm{CL}'''}^{\mathrm{t2i}}
\end{aligned}
\tag{55}
$$

Let's break this down: The prime ($'$) symbol defines on which outputs from the shared encoder (Equation 51 and Equation 52) the contrastive loss is applied. Since we have three linear layers in our shared encoder, and we want to enforce alignment in the whole shared encoder, we apply the contrastive loss on all three layers, but seperately. The superscripts $\mathrm{i2t}$ and $\mathrm{t2i}$ denote if we apply the contrastive loss from image to text or from text to image, and should already be known from when we introduced vision-language contrast (Section 2.5.2.2). To sum up, we apply the contrastive loss on all three linear layers of the shared encoder, and we weight the loss equally for each layer. The contrastive loss itself weights image-to-text and text-to-image equally, which is why each component of the contrastive loss is weighted with $\frac{1}{6}$. For each contrastive loss $\mathcal{L}_{\mathrm{CL}'}$, $\mathcal{L}_{\mathrm{CL}''}$, and $\mathcal{L}_{\mathrm{CL}'''}$ we generate the matrix $\boldsymbol{L}$ from Section 2.5.2.2 once.

Since we use the contrastive loss from CLIP, we also have to define a temperature for the contrastive loss on each layer. In total, we have three temperature parameters, one for each linear layer of the shared encoder. As done in CLIP, we optimize them in log space and initialize them with 0.07 [Rad+21].

The total training objective is then given by:

$$\min \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{CL}} \tag{56}$$

5.2.1.1.3 Training

For the teacher model we select an improved variant of the ResNet-50 [He+16], called ResNet-50-A1 [WTJ21], which has 25.6M parameters but runs in inference mode, so no gradients are computed. The model was trained on ImageNet-1K [Rus+15] and is available on HuggingFace[15].

We train the student model on all 3.3M image-text pairs we collected (Table 4) for 7 epochs, using a batch size of 256. We do not train for longer, as (1) we want to keep the training time manageable, and (2) we use a lot of pretrained components, which need less training time to converge. As done in prior experiments, we use the AdamW optimizer [LH19] and a learning rate of 1e-4. After every epoch, we validate the model on CLIP's zero-shot image classification approach, introduced in Section 2.7.2.2, and select the best model based on the achieved accuracy. The representations for the zero-shot classification are generated by the shared encoder of the student model, which we define as $\boldsymbol{h}'''_{v,K}$ and $\boldsymbol{h}'''_{w,K}$. The classification is performed on the validation set of ImageNet-1K [Rus+15]. At this point it is important to note that the accuracy we report with CLIP zero-shot classification is actually not zero-shot. This is because the teacher model is trained supervised on ImageNet-1K, and the student model is trained using the teacher's probability distribution over the ImageNet-1K classes. Our student model therefore learns the ImageNet-1K classes even though we do not train on ImageNet-1K directly. However, the accuracy we achieve still gives us a good indication of the quality of the student model's representations.

As mentioned before, we tokenize the text using the uncased BERT tokenizer. Again, uncased means that all text is lowercased before tokenization. Inspired by BEiT-3, we set the maximum text length, the length of the captions, to 64 tokens [Wan+23], truncate longer captions and pad shorter captions. This reduces the time required for a forward pass of the text encoder, and the captions of the data we collect are on average not larger than 15 tokens anyway (see Table 4).

For image augmentation, we use the same techniques as in the unimodal image KD experiment (Section 5.1.1.2). However, we make one important distinction in the size of the random crop: As seen in Table 32, the range of the random crop size is between 0.08 and 1.0 of the original image size. The lower bound is quite small, but because student and teacher receive

[15] https://huggingface.co/timm/resnet50.a1_in1k

the same crop, even if it is very small, the student can still learn the teacher's representation for a small part of the image. However, this gets problematic with image text pairs. If the crop is very small, then important semantic information of the image might be lost, which is still present in the text. Therefore, the resulting probability distribution of the teacher for that small crop might not be representative of the image's high-level content, which is described by the text. This could result in the student predicting a probability distribution that is correct with respect to the whole image, but not with respect to the small crop. To avoid this, we set the lower bound of the random crop size to 0.9, which is also the value used by VLMo [Bao+22]. This ensures that the crop is large enough to capture the high-level content of the image. A visualization of too small crops is shown in Figure 35, and a visualization of a minimum crop size of 0.9 is shown in Figure 36.

All hyperparameters are summarized in Table 36, pytorch pseudocode for the forward pass can be found in Code 3.

5.2.1.1.4 Results

We report the results of image-text retrieval on the test sets of COCO [Lin+14] and Flickr30k [You+14], which are shown in Table 10. Note that we do not report results on unimodal downstream tasks like image classification or text classification. This is because finetuning is expensive, which is why will refrain from doing so until we reach our final approach.

While the results on image-text retrieval are significantly worse than the state-of-the-art, we can still observe that we are not far off from the performance of FLAVA [Sin+21]. Considering that FLAVA was developed by a team of researchers from Meta AI, and that this is our first iteration of a multimodal model, the results are promising. From CLIP, VLMo, and BEiT-3 we are still far off, but this can at least partly be attributed to the fact that those model are significantly larger than ours, and that they have been trained on much more data. The latter is shown in Table 39. The increased performance on Flickr30K compared to COCO can be attributed to the fact that the Flickr30K dataset is smaller. For a given image, there are 5 correct captions in only 5k possible captions (25k for COCO), and for a given caption there are only 1k possible images (5k for COCO). We use the representations $\boldsymbol{h}'''_{v,K}$ and $\boldsymbol{h}'''_{w,K}$ for retrieval.

We also proof our statement of Section 2.4 that using two unrelated unimodal models, one image and one text model, for image-text retrieval fails, as the representations produced by the two models are not aligned. We do this by using the pretrained image and text variant from Data2Vec2 [Bae+23] as the image and text encoder, respectively. Each image is encoded by the image encoder, and each caption is encoded by the text encoder, after which the representation of the [I_CLS] and [T_CLS] token is extracted and used for retrieval according to our formulation from Section 2.5.2.2. This approach does not reach a score of more than 0.2% on any metric, and is therefore inappropriate for image-text retrieval. Consequently, we can conclude that training a multimodal model is essential for the alignment of modalities.

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Data2Vec2 [Bae+23]	0.02	0.08	0.19	0.01	0.10	0.19	0.02	0.12	0.18	0.02	0.06	0.12
FLAVA [Sin+21]	42.74	76.76	-	38.38	67.47	-	67.7	94.0	-	65.22	89.38	-
CLIP [Rad+21]	58.4	81.5	88.1	37.8	62.4	72.2	88.0	98.7	99.4	68.7	90.6	95.2
VLMo [Bao+22]	83.1	96.0	98.2	65.2	86.5	92.2	96.8	100	100	88.1	98.4	99.3
BEiT-3 [Wan+23]	**84.8**	**96.5**	**98.3**	**67.2**	**87.7**	**92.8**	**98**	**100**	**100**	**90.3**	**98.7**	**99.5**
SHRe_T (ours)	37.06	67.74	79.7	25.3	54.73	68.19	49.9	78.5	88.5	37.04	67.34	77.38

Table 10: Results of image-text retrieval on the COCO and Flickr30K test sets for Transformer SHRe. The results are compared to FLAVA [Sin+21], CLIP [Rad+21], VLMo [Bao+22], and BEiT-3 [Wan+23].

Transformer SHRe reaches 40.26% accuracy on the (zero-shot) ImageNet classification task, which is significantly worse that the 76.2% reached by CLIP [Rad+21], with CLIP being an actual zero-shot application (see previous section).

Since we are using the same approach as SHRe, we also provide a comparison of the average median rank on the COCO test set. How SHRe reports the average median rank is described in Section 2.7.1. Unfortunately, the authors do not provide the exact pairs used to calculate the average median rank. We therefore opt for an approach that should closely resemble that of SHRe: We select all 5k images from the COCO test set, and split them into 5 distinct sets of 1k images each. For each set we do the following: For each image one caption, out of the 5 captions available, is selected. We then have 5 splits of 1k image-caption pairs each. In each split, for one candidate there is only one correct target and 999 incorrect targets. We then perform retrieval on each split, and calculate the median rank of the correct target. The average median rank is then the average of the median ranks over all 5 splits. This procedure itself is repeated 5 times, so that each image is paired with each of its 5 correct captions exactly once. The result is 5 average median ranks, which are then averaged to get the metric reported in Table 11. Our approach significantly improves the baseline of SHRe, and the results indicate that the correct pair for a query is in most cases either the first or second retrieved item, though the results do not account for outliers, since the median rank is used. The reason for the almost perfect results, considering the minimum possible value is 1, can be attributed to the advances in deep learning research since SHRe was published, which was in 2017 [AVT17]. Some of the advances include the Transformer architecture, and the use of contrastive learning. Furthermore, we can assume that the quality of the teacher model, which is a ResNet-50-A1 [WTJ21], is also higher than the teacher model used in SHRe, which they do not specify explicitly, but they mention AlexNet as an example teacher model. Lastly, our approach is a vision-language model, while SHRe is a vision-language-audio model, which might make the task of alignment more difficult.

Model	MSCOCO	
	Image → Text	Text → Image
Random	500	500
SHRe [AVT17]	5.8	6.0
SHRe_T (ours)	**1.5**	**2.0**

Table 11: Comparison of the average median rank over 5 1k image-caption splits on the COCO test set. Our approach outperforms SHRe by a large margin. The best possible value is 1.

As mentioned in the introduction of SHRe [AVT17], the idea of not only predicting the probability distribution over the ImageNet-1K classes for a given image, but also for its caption, works because the ImageNet-1K classes can also be used to describe the content of a caption. The main reason for this is that the ImageNet-1K classes describe real-world objects, which are independent of the image modality, and can therefore also be used to describe the content of a text. A visualization of that is shown on image-text pairs from COCO, which is part of the data we use for training, in Figure 37.

5.2.1.2 Larger Batch Sizes with DDP

As mentioned in the introduction of contrastive learning (Section 2.5.2.2), a large batch size is crucial for the success of contrastive learning. Larger batch sizes allow for more negative samples, which makes the task of finding the correct pair among those negatives more difficult. Unfortunately, we are not able to exceed a batch size of 256, as we run out of memory with our current GPU setup (1 $\times$ NVIDIA RTX 4090).

To overcome this limitation, we utilize Distributed Data Parallel (DDP) [Li+20], which allows us to train our model on multiple GPUs. Each GPU has its own replica (copy) of the model, and for a single forward pass each GPU processes a different batch of the data. The forward pass and backward pass are then computed on each GPU seperately, based on the mini-batch of data it received. The resulting gradients for each replica are then aggregated across all GPUs/replicas, and each replica updates its weights based on the aggregated gradients. We are therefore able to increase the batch size by the number of GPUs we use, and update the weights of the model based on gradients that have been computed on a larger batch size than a single model received in a forward pass [Li+20]. An illustration of DDP with 2 GPUs is shown in Figure 38.

A side effect of DDP is that after the forward pass of each replica, there now exists not just a single batch of image-text representations, but as many batches as there are replicas/GPUs. Those representations can, similar to the gradient accumulation, be communicated across all replicas. If we use 2 GPUs with a batch size of 256, then all representations on the first GPU are communicated to the second GPU, and vice versa. Consequently, we now have 512

image-text pairs on which the contrastive loss is computed, which is equivalent to using a batch size of 512 on a single GPU.

This not only increases the effectiveness of the contrastive loss, but also reduces the time required to train the model. This is because when using DDP, the whole dataset is split across all GPUs (see Figure 38), and each GPU always only processes its own part of the dataset. Since the forward pass on each GPU is done in parallel, and the number of steps per epoch is reduced by a factor equal to the number of GPUs used, the training time is reduced. Even though the time required for training the model will not exactly be reduced by the factor of the number of GPUs used, as the distributed communication between the GPUs introduces overhead, it is still significantly faster than training on a single GPU. From a cost perspective, multiple GPUs are obviously more expensive than a single GPU, but the time saved by using DDP outweighs the increased cost to some extent. This is especially important considering that we can traing more models in a shorter amount of time, which allows us to iterate faster.

While it is tempting, also from a technical perspective, to use as many GPUs as possible, we refrain from doing so. This is because we do not want to make the success of our approach too dependent on the hardware we use. After all, the goal is to develop an approach that is feasible even for researchers with limited resources. We therefore limit the number of GPUs to 2, effectively increasing the batch size to 512. The architecture, loss, and training procedure remain the same as described in the previous sections.

As shown in Table 12, introducing DDP with a second GPU lead to an absolute gain of more than 4 percentage points on the COCO and Flickr30K retrieval tasks. This underlines the importance of a large batch size for contrastive learning. However, we observe no improvement on the (zero-shot) ImageNet classification task. We hypothesize that the classification task is more dependent on the quality of the text-based class prototypes, which are generated from predefined textual descriptions of the class labels (see Section 2.7.2.2). The quality of these prototypes, which are fixed representations of each class, is not directly influenced by the batch size used during training. In contrast, retrieval tasks rely on contrastive learning, where having a higher diversity of negative samples during training helps the model better distinguish between relevant and irrelevant image-text pairs. As a result, an increase in batch size translates more directly into better retrieval performance, while its effect on zero-shot classification is less pronounced.

Further, the average median rank (AMR) for image-to-text and text-to-image retrieval on COCO decreases to almost perfect scores, seen in Table 13.

The training time, displayed in Table 14 behaves as expected: The wall clock time per batch (the time needed for one full training step, including weight updates) is higher when using DDP, but the total training duration is significantly lower. Again, the latter is due to the fact that the number of steps per epoch is reduced by a factor equal to the number of GPUs used, which reduces the total training time.

DDP	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
×	**40.26**	55.45	66.44
✓	40.0	**59.59**	**70.78**

Table 12: DDP improves the performance of the model on the COCO and Flickr30K retrieval tasks.

Model	**MSCOCO**	
	Image → Text	Text → Image
SHRe [AVT17]	5.8	6.0
SHRe_T (ours)	1.5	2.0
SHRe_T + DDP (ours)	**1.0**	**1.04**

Table 13: Performance of the average median rank over 5 1k image-caption splits on the COCO test set. Increasing the negative samples by using DDP improves the performance to nearly perfect scores (1.0).

Approach	**Wall Clock Time per Batch (ms)**	**Training duration (h)**
Default	**331**	8.2
DDP	387	**4.8**

Table 14: Comparison of the wall clock time per batch and the total training duration for the default and DDP approach. The training duration is calculated for 7 epochs on the 3.3M image-text pairs we collected. While the wall clock time per batch is higher for the DDP approach, the total training duration is significantly lower.

5.2.1.3 Shared Transformer Encoder

We prepend the name of our approach with "Transformer" to indicate that we use a Transformer architecture for both the image and text encoder. However, this does not hold for the shared encoder, which is, like the original approach, a 3-layer MLP [AVT17]. We now experiment with also replacing the shared encoder with a Transformer, leading to a fully Transformer-based model. We motivate this decision with the architecture of VLMo and BEiT-3, which both exclusively consist of Transformer layers [Bao+22, Wan+23].

We hypothesize that learning the alignment using a Transformer for the shared encoder might be more challenging than learning it with an MLP. This is because the shared Transformer would, unlike the shared 3-layer MLP, receive the whole image/text sequence from the respective encoder, and not just the `[I_CLS]`/`[I_CLS]` token. Since image and text inputs are inherently different, the self-attention in the Transformer (layers) would not only have to perform self-attention, but also infer the modality of the input and adjust the attention accordingly, adding a layer of complexity.

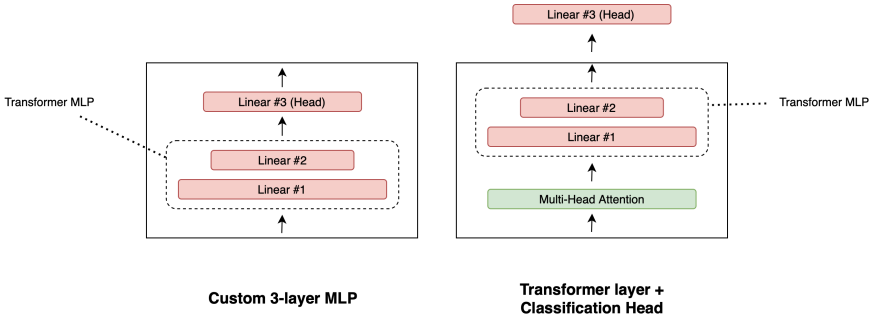

Figure 21: Switching from our implementation of the shared encoder as a 3-layer MLP to a Transformer layer only corresponds to adding a multi-head self-attention layer before the linear layers. The dimensionality of the linear layers remains the same. On the right, layer normalization and residual connections are omitted for simplicity.

Nevertheless, VLMo demonstrates that this modality-specific complexity may not be an issue. In VLMo, masked language modeling is performed using a Transformer whose self-attention weights are frozen and from a pretrained image model, yet the model still effectively handles text-only pretraining [Bao+22].

Given that VLMo can successfully leverage frozen image self-attention weights for text tasks, we believe that our approach, where the self-attention weights are explicitly trained to work with both image and text inputs, should also be effective, if not more so.

For the shared Transformer encoder, we decide to only use a single Transformer layer, as we want to keep the model size manageable. For comarison, VLMo and BEiT-3 use two Transformer layers for the shared encoder [Bao+22, Wan+23].

The change in the architecture does not imply a change in the training procedure, loss, or hyperparameters. In fact, the only thing we actually add is one multi-head self-attention layer to the shared encoder (plus the two residual connections). Recall that we implemented the 3-layer MLP as a 2-layer MLP network as used in Transformer layers, and added an additional LayerNorm and linear layer on top of it (see Section 5.2.1.1.1). Replacing this with a single Transformer layer means we still have the 2-layer MLP network, but now also add a ,ulti-head self-attention layer before it. Since we are still predicting a probability distribution over the ImageNet classes (the one returned by the teacher), which can be seen as a type of classification task, we still need a classification head on top of the Transformer layer to output logits for the 1000 classes. Fittingly, this is exactly the task of the third MLP layer we had in the shared 3-layer MLP. The number of neurons for each linear layer also remains the same: The first linear layer expands the hidden dimension of the Transformer from 768 to 3072, and the second linear layer reduces it back to 768. The third linear layer then expands it to 1000, the number of classes in ImageNet. In Figure 21, where we illustrate our previous explanation, we indicate the difference in dimensionality between the linear layers by a different breadth.

Since the shared encoder is now can actual Transformer layer plus the classification head, we redefine our previous forward pass of the shared encoder from Equation 51 and Equation 52 to the following (also on the example of the text modality):

$$\begin{aligned} \boldsymbol{H}'_{w,K} &= \mathrm{MHA}\Big(\mathrm{LN}\Big(\boldsymbol{H}_{w,L_s}\Big)\Big) + \boldsymbol{H}_{w,L_s} \\ \boldsymbol{H}_{w,K} &= \mathrm{FFN}\big(\mathrm{LN}\big(\boldsymbol{H}'_{w,K}\big)\big) + \boldsymbol{H}'_{w,K} \end{aligned} \tag{57}$$

$$\boldsymbol{h}_{w,[\texttt{T\_CLS}]} = \boldsymbol{h}_{w,K,[\texttt{T\_CLS}]}\boldsymbol{W}_3^T + \boldsymbol{b}_3 \tag{58}$$

It holds that $K = L_s + 1 = 7$, since the shared encoder is one additional Transformer layer. Equation 57 is the same operation of one Transformer layer known from the ViT architecture, and is therefore also shown in Equation 21 of the section on Vision Transformers (Section 2.3.2).

The operation in Equation 58 now resembles that of an actual classification head from the vision Transformer architecture, and is only applied on the [I_CLS] and [T_CLS] token, respectively. Different to what was proposed by the authors of SHRe [AVT17], we now do **not** apply the contrastive loss on the output of all three linear layers (the linear layers are shown in Figure 21). Instead, we only apply it on the output of the shared Transformer layer, meaning the representations $\boldsymbol{h}_{v,K,[\texttt{I\_CLS}]}$ and $\boldsymbol{h}_{w,K,[\texttt{T\_CLS}]}$. We do this for the following reasons:

We remove the contrastive loss from the classification head, as the task of the classification head is to leverage the knowledge learned from the teacher to provide the student with guidance. It does this by predicting a probability distribution over the ImageNet-1K classes once for a given image, and once for the image's caption. Both probability distributions, for the image and the caption, should be the same as the teacher's probability distribution for the same image. Based on this intuition, the classification head is not meant to output representations of the image or text, but only to predict the ImageNet-1K classes. It outputs logits for each class, and not a representation as used in retrieval tasks. Therefore, it is not meant to be used for alignment and retrieval tasks. Because of this, the classification head can be discared after training, as it is not needed for retrieval or any other downstream task anymore. It is only used during training to provide the student with guidance from the teacher.

We remove the contrastive loss from the intermediate linear layer (linear #1) of the shared Transformer layer, as VLMo and BEiT-3 also exclusively use the *final* representation of the shared encoder for retrieval tasks. Removing the requirement for alignment in the intermediate layer reduces the complexity of the task for the model, and gives it more freedom. Moreover, the point where the alignment is actually crucial is where the image and text representations are used for retrieval. Since we use the final representation of the shared Transformer layer for retrieval, we do not need to enforce alignment in the intermediate layer.

Loss	Modality	SHRe	Transformer SHRe
$\mathcal{L}_{\text{KD}}$	Image	$\boldsymbol{h}'''_{v,K}$	$\boldsymbol{h}'''_{v,K,[\texttt{I\_CLS}]}$
	Text	$\boldsymbol{h}'''_{w,K}$	$\boldsymbol{h}'''_{w,K,[\texttt{T\_CLS}]}$
$\mathcal{L}_{\text{CL}}$	Image	$\boldsymbol{h}'_{v,K}$	$\boldsymbol{h}_{v,K,[\texttt{I\_CLS}]}$
	Text	$\boldsymbol{h}'_{w,K}$	$\boldsymbol{h}_{w,K,[\texttt{T\_CLS}]}$
$\mathcal{L}_{\text{CL}''}$	Image	$\boldsymbol{h}''_{v,K}$	-
	Text	$\boldsymbol{h}''_{w,K}$	-
$\mathcal{L}_{\text{CL}'''}$	Image	$\boldsymbol{h}'''_{v,K}$	-
	Text	$\boldsymbol{h}'''_{w,K}$	-

Table 15: A comparison between the tokens used in the loss functions for the approach of SHRe [AVT17] (with a shared 3-layer MLP), and our Transformer SHRe. For Transformer SHRe we do not use the contrastive loss components $\mathcal{L}_{\text{CL}''}$ and $\mathcal{L}_{\text{CL}'''}$.

An overview which tokens are used in which part of the training objective is shown in Table 15.

The full contrastive loss is now just:

$$\mathcal{L}_{\text{CL}} = \frac{1}{2}\mathcal{L}_{\text{CL}}^{\text{i2t}} + \frac{1}{2}\mathcal{L}_{\text{CL}}^{\text{t2i}} \tag{59}$$

Both loss components work on the representations $\boldsymbol{h}_{v,K,[\texttt{I\_CLS}]}$ and $\boldsymbol{h}_{w,K,[\texttt{T\_CLS}]}$ from the final output of the shared Transformer layer $\boldsymbol{H}_{v,K}$ and $\boldsymbol{H}_{w,K}$.

5.2.1.4 Results

The influence on retrieval when adding a shared Transformer layer, and changing the representations used for retrieval, can be seen in Table 16. This change not only outperforms our first two experiments in all metrics, but also FLAVA [Sin+21] in R@1 text retrieval on COCO. For other metrics on COCO we are also surprisingly close to FLAVA. On Flickr30K, we still lack behind FLAVA, but the gap is significantly smaller than before.

We are also able to increase the performance on CLIP-like ImageNet-1K classification from 40.26% to 44.57%, and reach a perfect average median rank for both text and image retrieval on the COCO test set, which is 1.0. As a reference: The previous scores were 1.0 for text retrieval, and 1.04 for image retrieval (see Table 12). While this is indeed the perfect score for the average median rank, it is important to note that we use the *median* rank, and a value of 1.0 therefore means that in at least half of all retrievals (so ≥ 500, since we always do retrieval on 1k subsets) the correct pair is the first retrieved item. This does not account for outliers, and is only applied on a small subset of 1k image-caption pairs. Nevertheless, this is the benchmark provided by SHRe [AVT17], so a perfect score is still worth mentioning. Unless the average median rank gets worse (meaning it increases), we will refrain from re-

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
FLAVA [Sin+21]	42.74	**76.76**	-	**38.38**	**67.47**	-	**67.7**	**94.0**	-	**65.22**	**89.38**	-
Baseline	37.06	67.74	79.7	25.3	54.73	68.19	49.9	78.5	88.5	37.04	67.34	77.38
+ DDP (↑ 4.23)	41.88	71.86	83.06	29.4	59.3	72.06	56.1	83.5	90.7	41.80	71.36	81.14
+ Shared T (↑ 6.67)	**44.6**	75.3	85.64	31.69	62.1	74.48	58.8	85.6	92.4	43.92	74.06	82.8

Table 16: The overview of retrieval results on the COCO and Flickr30K test sets show significant improvements when using DDP and a shared Transformer encoder. We achieve an average absolute gain of almost 7 percentage points across all metrics compared to our first approach, which we denote as "Baseline". We add FLAVA [Sin+21] as a reference.

porting it in the future, as it is not a very informative metric and we have already reached the best possible score. A visualization of the retrieved samples for candidate captions (image retrieval) and images (text retrieval) is shown in Figure 39 and Figure 40, respectively.

5.2.2 Self-Supervised Teacher

5.2.2.1 Challenges of Self-Supervision

With the architecture and the first benchmarks in place, we can now focus on the main goal of this research: Applying the idea of SHRe [AVT17] using a self-supervised teacher model. A consequence of a self-supervised teacher is that the teacher's prediction for a given image is not a probability distribution over a set of classes, but merely a representation of the input sample. This raises the question which training objective to use when using a self-supervised teacher, as we cannot use the KL-Divergence loss anymore. This only works on probability distributions, and not on representations.

In our experiments on unimodal knowledge distillation (image and text) a self-supervised teacher did not pose a problem, as both the teacher and student received the same input, and the latter was able to regress all time steps of the teacher model. However, this was only possible because the teacher and student received exactly the same input: The patch/text token the teacher and student received at a time step i was the same for both models, allowing the student to learn the teacher's representation for each patch or text token, respectively. Since we predict the output of an image teacher model, in which ever form it may be, the aforementioned still holds true when our multimodal student receives an image as the input. The output will be a representation for each image patch, which is also the prediction of the teacher model, allowing for the same approach used in unimodal knowledge distillation (see Section 5.1.1).

When the multimodal model receives a text (an image's caption) as the input however, the teacher's prediction is still a representation of the image. This poses the following problems:

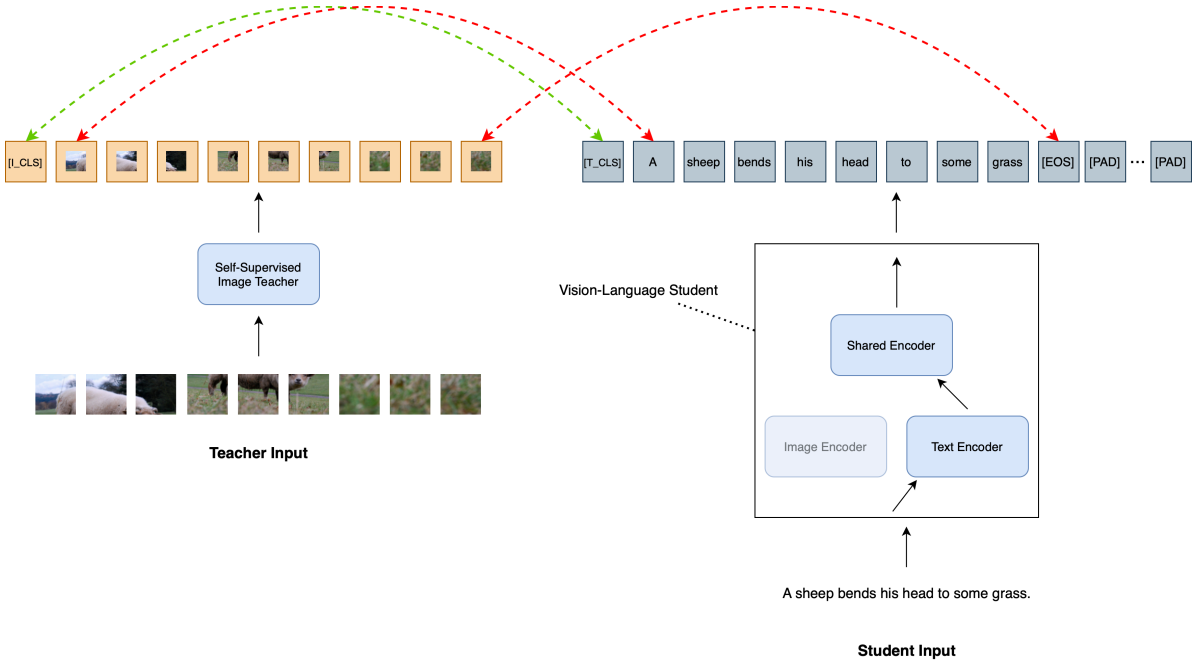

Figure 22: The meaning/content of time steps across modalities is not aligned, and the number of time steps will differ between modalities. This makes alignment on the level of individual time steps impossible. The cls token aggregates global information independent of time steps, and captures the meaning and interpretation of the respective input, making alignment possible. However, this requires the teacher cls token to not contain any modality-specific (in this case image) information. Image-Text example is taken from the COCO train set [Lin+14].

1. The content of the time steps is aligned: The text token at a time step i does not necessarily describe the content of the image patch at the same time step i. Moreover, text naturally contains fill words such as “the”, “a”, “is”, which do not have any meaning with respect to the content of an image. Another example are padding tokens, which do not contain any information at all, and are merely used for batching a set of texts.
2. The number of time steps is not aligned: The number of text tokens in a caption does not match the number of image patches. Consequently, there is not a 1:1 mapping between the time steps of the image and the text.

Both problems are illustrated in Figure 22. It is therefore not possible to regress the representation of individual patches when the multimodal student model receives a text as the input. That means we have to resort to regressing the global representation of the image, which is the [I_CLS] token. This choice solves both of the aforementioned problems, as we do not rely on the content and placement of individual time steps. To clarify, the concept of the forward pass remains the same as in SHRe [AVT17] and our Transformer variant of the previous chapter (Section 5.2.1): For a single image-text pair, the image can be passed to the teacher, and the image and its caption can be passed to the student seperately. When we focus on the global representation (the cls token), the teacher will always return a repre-

sentation of the [I_CLS] token, aggregating global information of the image. The same holds true for the student, producing a representation of the [I_CLS] token. As a training objective we can then require:

$$\min_{\boldsymbol{h}^s_{v,[\texttt{I\_CLS}]}} \|\boldsymbol{h}^s_{v,[\texttt{I\_CLS}]} - \boldsymbol{h}^t_{v,L_t,[\texttt{I\_CLS}]}\| \tag{60}$$

We denote a representation generated by the student with the superscript s, and the teacher with the superscript t. The objective forces the student to push its representation of the [I_CLS] token as close as possible to the teachers representation of the same token.

Most importantly, the student can also be trained to push the representation of the caption, given by the [T_CLS] token, as close as possible to the teacher's representation of the image:

$$\min_{\boldsymbol{h}^s_{w,[\texttt{T\_CLS}]}} \|\boldsymbol{h}^s_{w,[\texttt{T\_CLS}]} - \boldsymbol{h}^t_{v,L_t,[\texttt{I\_CLS}]}\| \tag{61}$$

The combined training objective when regressing only global information is then:

$$\min_{\boldsymbol{h}^s_{v,[\texttt{I\_CLS}]},\boldsymbol{h}^s_{w,[\texttt{T\_CLS}]}} \|\boldsymbol{h}^s_{v,[\texttt{I\_CLS}]} - \boldsymbol{h}^t_{v,L_t,[\texttt{I\_CLS}]}\| + \|\boldsymbol{h}^s_{w,[\texttt{T\_CLS}]} - \boldsymbol{h}^t_{v,L_t,[\texttt{I\_CLS}]}\| \tag{62}$$

While this objective in theory forces the student to output the same global representation for an image and its caption as the teacher, it requires the teacher to produce a representation of the [I_CLS] token that is abstract enough to also describe the content of the caption. Conretely, the representation of the [I_CLS] token should **not** contain any image-specific information, as this would make it impossible for the student to align the representation of the caption with that of the image: It is not possible to extract any image-specific information, like the exact position of an object in the image, from the caption. Consequently, whether the representation of the [I_CLS] token produced by the teacher is abstract enough to also describe the content of the caption remains to be seen in the following experiments.

The challenges that come with the choice of an **unimodal** self-supervised teacher raises the question why we do not directly use a **multimodal** model as the teacher. This reason behind this choice can be attributed to the fact that the goal of this research is to train a multimodal model without using any existing, especially pretrained, *multimodal* components. Instead, we aim to extract knowledge from purely unimodal models and learn to generate modality-invariant features from it (1), and to not rely on labeled data in the end-to-end training of the multimodal model (2). This includes the teacher model, which should not be trained on labeled data, but only self-supervised.

5.2.2.2 Feature-based Knowledge Distillation

Based on the previous discussion, we propose an adjustment to the loss used to train the multimodal student. We reformulate the knowledge distillation loss to a feature-based loss, which implements the training objective discussed in the previous section:

$$\begin{aligned} \mathcal{L}_{\text{KD}} = \\ \frac{1}{2} * \mathcal{L}^{v}_{\text{KD}} + \frac{1}{2} * \mathcal{L}^{w}_{\text{KD}} = \\ \frac{1}{2} * \|\boldsymbol{h}^{s}_{v,[\texttt{I\_CLS}]} - \boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]}\|^2_2 + \frac{1}{2} * \|\boldsymbol{h}^{s}_{w,[\texttt{T\_CLS}]} - \boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]}\|^2_2 \end{aligned} \tag{63}$$

The loss is the average of the mean squared error between the student's and teacher's representation of the [I_CLS] token, and between the student's representation of the [T_CLS] token and the teacher's representation of the [I_CLS] token.

The target representation $\boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]} \in \mathbb{R}^{768}$ stems from a new, now self-supervised, teacher model, which is BEiTv2 [Pen+22]. Since BEiTv2 is a Transformer-based self-supervised image model, it also returns a sequence of patch representations, and we extract the representation $\boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]} \in \mathbb{R}^{768}$ of the [I_CLS] token from the output sequence. The teacher has $L_t = 12$ Transformer layers, and is based on the ViT-B/16 [Dos+21] architecture. It therefore is with almost 86M parameters significantly larger than the previous ResNet-50-A1 [WTJ21] teacher model, which had 25M parameters.

The architecture of the student remains largely the same as before, the prediction of the student for the [I_CLS] token of the teacher, which is the global image represenation, is generated by the student's classification head. A consequence of this is that the classification head now also outputs 768-dimensional representations, instead of the 1000-dimensional output used previously for the ImageNet-1K classes. Therefore: $\boldsymbol{h}^{s}_{v,[\texttt{I\_CLS}]} \in \mathbb{R}^{768}$ and $\boldsymbol{h}^{s}_{w,[\texttt{T\_CLS}]} \in \mathbb{R}^{768}$. As a side note, the student's classification head is technically not a classification head anymore, as it does not predict any classes but rather a representation of the teacher. We will refer to it as the "regression head" from now on.

5.2.2.3 Results

Apart from the change in the teacher and the impact on the loss, the training setup remains the same, and no hyperparameters are changed. As we now do not follow the approach of SHRe, that is, predicting a probability distribution over ImageNet-1K classes [AVT17], we name the new approach "**S**elf-**S**upervised **M**ultimodal **K**nowledge **E**xtraction" (S-SMKE).

Surprisingly, the results show a significant improvement in retrieval performance compared to the previous benchmarks, as shown in Table 17. We achieve competetive performance with FLAVA [Sin+21] and CLIP [Rad+21] on COCO, and even outperform CLIP on COCO R@10 image retrieval by more than 6 percentage points.

Less surprising is that we observe a drop in ImageNet-1K classification performance of more than 14 percentage points compared to the supervised teacher of SHRe [AVT17] (see left in Table 18). We consider this as not surprising, because we previously used representations $\boldsymbol{h}'''^{s}_{v,K,[\texttt{I\_CLS}]}$ and $\boldsymbol{h}'''^{s}_{w,K,[\texttt{T\_CLS}]}$ for the CLIP-like Imagenet-1K classification, which were also the representations used for predicting the ImageNet-1K classes as given by a supervised

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
FLAVA [Sin+21]	42.74	76.76	-	**38.38**	**67.47**	-	67.7	94.0	-	65.22	89.38	-
CLIP [Rad+21]	**58.4**	**81.5**	88.1	37.8	62.4	72.2	**88.0**	**98.7**	**99.4**	**68.7**	**90.6**	**95.2**
SHRe_{T}	44.6	75.3	85.64	31.69	62.1	74.48	58.8	85.6	92.4	43.92	74.06	82.8
S-SMKE (↑ 4.38)	51.66	79.9	**88.66**	36.17	66.55	**78.28**	64.5	88.4	93.0	51.78	78.54	86.46

Table 17: An updated contrastive loss with a self-supervised teacher, which we denote as "S-SMKE" leads to an average gain of 4.38% compared to our previous approach SHRe_{T}. On Flickr30K, we reach substantial improvements while still being behind FLAVA [Sin+21] and CLIP [Rad+21].

Approach	Accuracy
Visual N-Grams [Li+17] †	11.5
CLIP [Rad+21] †	**76.2**
SHRe_{T} (ours)	44.57
S-SMKE (ours)	30.4

Approach	Training time (h)
SHRE_{DDP}	**4.8**
SHRe_{T}	5.27
S-SMKE	6.9

Table 18: (Left) Accuracy of S-SMKE on the validation set of ImageNet-1k using CLIP-like zero-shot classification. We compare to our supervised (SHRe [AVT17]) approach, CLIP [Rad+21], and Visual N-Grams [Li+17]. The latter was developed as a first proof-of-concept of zero-shot image classification in 2017 [Li+17, Rad+21]. We denote a full zero-shot approach with †. (Right) Development of the training time for the different approaches. Adding additional parameters through self-attention (SHRe_{T}) and a large self-supervised teacher (BEITv2 [Pen+22] in S-SMKE) increases the training time by more than two hours.

teacher trained on ImageNet-1K. Therefore, those representations were optimized for predicting ImageNet-1K classes. Now, we regress the teacher's representation of the [I_CLS] token, and since the teacher was not trained on the ImageNet-1K classes, the student's representations do not contain any information about them. This leads to a drop in classification performance.

While we now train the student without the ImageNet-1K classes, the CLIP-like zero-shot classification still cannot be considered as zero-shot. This can be attributed to the fact that, although trained without the classes/labels, our teacher model BEiTv2 [Pen+22] has been trained on ImageNet-1K. The knowledge contained in the teacher's representations of the [I_CLS] token could therefore leak information about the images of ImageNet-1K to the student. In order for it to be a true zero-shot approach the teacher would have to be trained on a dataset disjoint from ImageNet-1K.

Speaking of the teacher, its increased size leads to a longer training time, as shown in Table 18 (right), which would be even more pronounced if we wouldn't use DDP with 2 GPUs.

5.2.3 Token-Type Embeddings

In order for the shared Transformer approach to work, the linear layers and self-attention mechanism in the shared Transformer layer need to be able to somewhat differentiate between both modalities (image and text). Even though we desire an aligned representation to form in the shared block, especially the self-attention mechanism still needs to be able to differentiate between the two modalities. This can be explained by the fact that for images the model needs to find 2-dimensional spatial relationships, while for text only 1-dimensional relationships are required. Even though we learned from the previous experiments that the shared Transformer block is able to learn a good representation without any modality-specific information, we still want to investigate how the performance of the model changes when we explicitly provide the shared Transformer layer with the information which modality it is currently processing. This nessessitates a special embedding for both image and text, which is added to each token of the respective modality, even the special tokens. This is called a token-type embedding (TTE) [Bao+22], and is also used in multimodal models such as VLMo [Bao+22].

The intuitive approach to implement this would be to directly follow VLMo and add the token type embeddings after the positional encoding, before the input is fed into the Transformer blocks [Bao+22]. Our definition of the Transformer input changes as follows for text:

$$\begin{aligned} \boldsymbol{H}_{w,0} &= \left[\boldsymbol{h}_{w,0,\texttt{[T\_CLS]}}, \boldsymbol{h}_{w,0,1}, ..., \boldsymbol{h}_{w,0,M}, \boldsymbol{h}_{w,0,\texttt{[T\_SEP]}}\right] = \boldsymbol{E}_w + \boldsymbol{T}_w^{\text{pos}} + \boldsymbol{T}_w^{\text{type}} \\ \boldsymbol{T}_w^{\text{type}} &= \left[\boldsymbol{t}_{\text{type}}^w, \boldsymbol{t}_{\text{type}}^w, ..., \boldsymbol{t}_{\text{type}}^w, \boldsymbol{t}_{\text{type}}^w\right] \end{aligned} \tag{64}$$

Similar holds for images:

$$\begin{aligned} \boldsymbol{H}_{v,0} &= \left[\boldsymbol{h}_{v,0,\texttt{[I\_CLS]}}, \boldsymbol{h}_{v,0,1}, ..., \boldsymbol{h}_{v,0,N}\right] = \boldsymbol{E}_v + \boldsymbol{T}_v^{\text{pos}} + \boldsymbol{T}_v^{\text{type}} \\ \boldsymbol{T}_v^{\text{type}} &= \left[\boldsymbol{t}_{\text{type}}^v, \boldsymbol{t}_{\text{type}}^v, ..., \boldsymbol{t}_{\text{type}}^v\right] \end{aligned} \tag{65}$$

The token type embedding is the same for every token in the sequence. This follows VLMo [Bao+22] and is because each token/patch of a text/image input sequence is of the same modality. The parameters added to the model are negligible, as they only include two additional embeddings of size D, with $D = 768$ this accounts for just $768 * 2 = 1536$ parameters.

We present the results in Table 19, and show that the variant with TTE does not improve the performance of the model.

We suspect that this is due to two reasons. First, the TTE is added before the modality-specific encoders, so the image and text encoder. Their task is to extract features from the input, independent of the **other** respective modality. Consequently, the TTE is of no use to them, as the same embedding is added to every token/patch, and both encoders do not need to differentiate between image and text in their input: The image encoder will always

TTE	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
×	**30.4**	**66.87**	**77.1**
✓	29.6	66.0	76.5

Table 19: Introducing token-type embeddings (TTE) after the positional encoding degrades performance slightly. We compare to the previous model, which does not use TTE.

receive an image, and the text encoder always a text. Second, even worse, we use pretrained layers for the image and text encoder, which already extract rich features from the input. Adding the token-type embedding, which is initialized randomly, to their input will destroy the patch (token) embeddings the image (text) encoder receives as its input, and thus the feature extraction will be less effective. Even though we are also training the image and text encoder it will take time until the encoders learn to adapt to the TTE, and until the TTE has been trained in itself.

We therefore opt for the following change: We add the TTE after the modality-specific encoders, meaning the token type embedding is added to the output of the image and text encoder. This way, the TTE will not disturb the feature extraction of the image and text encoder. However, what will inevitably happen is that adding the TTE after the modality-specific encoders will destroy the features extracted by the encoders, which is exactly the second problem mentioned before, just at a different stage in the model.

A possible solution can be found in the Transformer block of extremely deep and large models, such as BEiT-3 [Wan+23]. Here, each embedding dimension of the output generated by the self-attention and MLP in a Transformer layer is multiplied with a seperate, learnable, scalar. What makes this approach special is that the scalars are all initialized with values close to zero. As this multiplication is done before the residual connection, i.e. the addition of the input to the output of self-attention and MLP respectively, the contribution of the self-attention and MLP to the output of the Transformer block is very small at the beginning of training. What follows is that the initial input to the model is carried very far through the model, and the actual contribution of the self-attention and other parameters is added gradually as the model lears to extract meaningful features from the input [Tou+21].

We will use LayerScale not for the purpose it was intended for, that is, to allow training of extremely deep models [Tou+21], but to allow the TTE to be added gradually after the modality-specific encoders, without destroying the features extracted by the encoders. Before the TTE is added to the output sequence of the image and text encoder, we multiply each dimension of the TTE with its own learnable scalar, as given by LayerScale. We initialize the weights of the LayerScale to 1e-5, so that each dimension of the TTE is multiplied with a very small scalar, making the contribution of the TTE, when adding it to the output of the image and text encoder, very small initially. This way, the shared Transformer block will receive the almost unaltered features from the encoders, and since the scale is learned, the

TTE	LayerScale	ImageNet-1K	Retrieval	
			MSCOCO	Flickr30K
×	×	**30.4**	66.87	77.1
✓	×	30.3	**67.2**	78.08
✓	✓	30.3	**67.2**	**78.29**

Table 20: Comparison of S-SMKE with different TTE and LayerScale settings **after** the modality-specific encoders. We observe that adding TTE and LayerScale especially improves performance on the retrieval tasks, with Flick30K showing an absolute gain of 1.19%.

TTE will be added as the models sees fit - as much as it helps the model to learn. Implementing LayerScale is straightforward, the representation of each TTE dimension is multiplied element-wise ($\odot$) with the LayerScale embedding $\boldsymbol{t}_{\text{scale}}$ before the TTE is added to the output of the image and text encoder:

$$\begin{gathered}\boldsymbol{H}'_{w,L_s} = \boldsymbol{H}_{w,L_s} + \boldsymbol{t}_{\text{scale}} * \boldsymbol{T}^{w}_{\text{type}} = \\ \boldsymbol{H}_{w,L_s} + \left[\boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{w}_{\text{type}}, \boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{w}_{\text{type}}, ..., \boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{w}_{\text{type}}, \boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{w}_{\text{type}}\right]\end{gathered} \tag{66}$$

$$\begin{gathered}\boldsymbol{H}'_{v,L_s} = \boldsymbol{H}_{v,L_s} + \boldsymbol{t}_{\text{scale}} * \boldsymbol{T}^{v}_{\text{type}} = \\ \boldsymbol{H}_{v,L_s} + \left[\boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{v}_{\text{type}}, \boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{v}_{\text{type}}, ..., \boldsymbol{t}_{\text{scale}} \odot \boldsymbol{t}^{v}_{\text{type}}\right]\end{gathered} \tag{67}$$

We use the same LayerScale $\boldsymbol{t}_{\text{scale}} \in \mathbb{R}^{768}$ for both image and text type embeddings, as the contribution of the type embeddings should be the same for both modalities. We do not want to bias the model towards one modality.

Table 20 shows that this approach is able to slightly improve the performance of the model, and we achieve a gain of at least 0.3% in all tasks. While this is not a significant improvement, it shows that the TTE [Bao+22], together with LayerScale [Tou+21], can be beneficial, which is why we keep this approach.

After training, to validate whether the TTE is actually utilized, we check the average value of the LayerScale weights. If they show a value lower or equal to the initial value of 1e-5, then we can conclude that even though the model performance improves, the relative importance of TTE is low. We measure a mean of 2e-4 and observe the maximum weight for a channel to be 0.37. This shows that the model utilizes the TTE to some extend, and we keep the TTE and LayerScale in the model, as they increase the performance slightly. Only few parameters are added: 1,536 for both token type embeddings, and 768 for the learnable scaling weights of LayerNorm.

5.2.4 Increasing Negative Examples for ITC

Our current approach utilizes Distributed Data Parallel (DDP) [Li+20] with a batch size of 256 per GPU. With two GPUs, the combined batch size is 512 and image/text features are gathered from all devices to increase the number of negative examples, as described in VLMo [Bao+22]. This method, as demonstrated in Section 5.2.1.2, improves performance.

However, implementations of image-text models that leverage contrastive learning typically use much larger batch sizes, and therefore have much more negative examples available. For instance, CLIP uses a batch size of 32,768 [Rad+21]. Achieving such large batch sizes would require more GPUs, as batch sizes exceeding 256 (per device) lead to out-of-memory (OOM) errors on the GPU (NVIDIA RTX 4090).

One intuitive approach to increase the number of negative examples without a larger batch size is to use a queue-based memory bank. The queue stores image and text embeddings, i.e. tokens `[I_CLS]` and `[T_CLS]`, from previous batches. With each new batch, the memory bank is updated by adding the current batch embeddings and discarding those of the oldest batch. These stored embeddings, combined with the embeddings of the current batch, are then used as negative examples.

Given a batch size of B and a memory bank size of U, we can achieve $B + U - 1$ negative samples. For instance, with a batch size of 256 and a memory bank size of 768, we can attain 1023 negative samples. This configuration effectively simulates a contrastive loss with a batch size of 1024, and is comparable to four GPUs with DDP and a batch size of 256 per GPU. However, it is important to note that this "simulation" of larger batch sizes with a memory bank only applies to the number of negative examples: The actual gradients are still computed using the effective batch size of 512 (… because we still use the double GPU setup with a batch size of 256 per GPU).

Implementing this approach actually requires to maintain two distinct memory banks. This is because we need to store old image representations (1), and old text representations (2).

Illustrated in Table 21, we observed a significant drop in performance, and the resulting model is not usable. The accuracy on ImageNet-1K is 0.001%, and therefore corresponds to a random classification (1000 classes). This suggests that the model did not learn anything useful, so simply increasing the number of negative examples via a memory bank does not help in learning richer represenations.

We suspect this drop in performance originates because of the following reasons: When using an actual batch size of 512 without a memory bank all negative examples are generated during the same step, and therefore by the same model with the same weights. This is the case even with DDP, as all model replicas are synchronized, i.e. share the same weights. Therefore, the representations share the same latent space and are consistent with each other. A similarity measure, in our case the cosine similarity, can then provide the model with a meaning of distance between these, in our case image and text, representations.

Memory Bank	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
×	**30.3**	**67.2**	**78.29**
✓	0.001	0.12	0.58

Table 21: Using a memory bank to store negative examples leads to unusable results. Especially noticeable is an accuracy of 0.001% on ImageNet-1K, which corresponds to a random classification.

However, when using a memory bank most negative examples come from previous batches that were stored in the memory bank. As the model's weights constantly change, especially at the beginning of training, there is a continous shift in the representation space. This shift is so pronounced that even representations from the immediate previous steps differ significantly from the current representations, and a similarity measure will not provide meaningful information to the model. To demonstrate, say we generate a representation $\boldsymbol{h}_{v,K,[\texttt{I\_CLS}]}$ for an image, and store it in the memory bank. If we generate a new representation for the same image in the next step, then both representations will not be the same, because the weights with which the representations have been generated were not the same. Consequently, the cosine similarity will not be at its maximum value of 1, even though it is the same image. It follows that the representations stored in the memory bank are not consistent with the representations generated in the current training step, and comparing them with cosine similarity does not provide a meaningful measure of similarity to the model.

This can be thought of as a less extreme case of comparing the representations of an image-only and text-only model, which are not associated with each other. In the beginning of our experiments, we tested image-text retrieval with the Data2Vec2 image and text model (see Table 10), and observed that this approach is ineffective for image-text retrieval. The representations produced by both models do not have any relationship with each other, and therefore the cosine similarity does not provide any meaningful information to the model.

While this effect is less pronounced with a memory bank, as the representations are still generated by the same model, the shift in the model's weights is still significant enough to make the representations inconsistent with each other.

5.2.4.1.1 Relation to Initial Memory Bank Approach

The memory bank was initially introduced by [Wu+18] as a mapping of the complete training dataset: The embedding of each sample in the dataset is stored in the memory bank (illustrated in Figure 23). For each batch/step, K samples are randomly drawn from the memory bank to be used as negative examples. The representations of samples in the current batch are then updated in the memory bank [Wu+18]. This approach is similar to ours, but faces the same problem: The representations come from an older variant of the model with different weights. Even worse, the representation of an example in the dataset is updated only

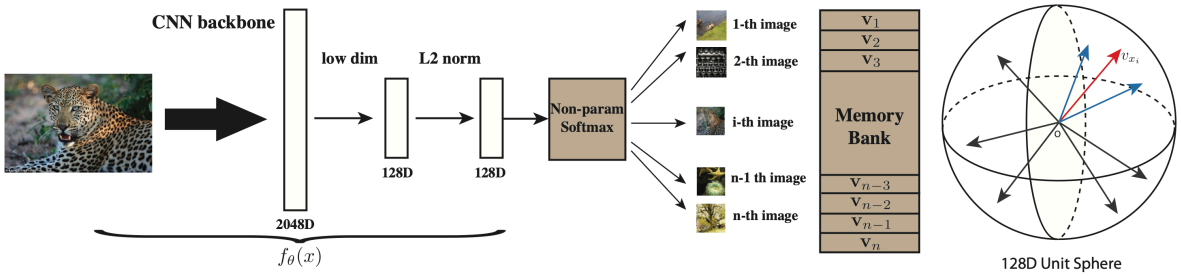

Figure 23: The memory bank, originally developed for self-supervised image representation learning, stores a 128-dimensional embedding of each training example. Contrary to what can be observed, instead of the whole dataset, just K samples are drawn from the memory bank in each iteration to be used as negative examples [Wu+18].

when it was last seen in a batch, which can be a long time ago for large datasets. However, the authors mitigate this issue by using proximal regularization, as shown in Equation 68.

$$-\log h(i, \boldsymbol{v}_i^{t-1}) + \lambda * \|\boldsymbol{v}_i^t - \boldsymbol{v}_i^{t-1}\|_2^2 \tag{68}$$

While the term $-\log h(i, \boldsymbol{v}_i^{t-1})$ can be ignored for now, as it just denotes a form of contrastive loss, the other term $\lambda * \|\boldsymbol{v}_i^t - \boldsymbol{v}_i^{t-1}\|_2^2$ serves as the proximal regularization. It describes the mean squared error between the representation $\boldsymbol{v}_i^t$ of a training example i in the current batch, e.g. an image, and the representation of the same training example $\boldsymbol{v}_i^{t-1}$ stored in the memory bank, which was updated the last time the image was in the current batch. This time is denoted as time step $t-1$.

The goal is to minimize Equation 68, and therefore to also minimize the proximal regularization, which is the mean squared error. The mean squared error is only at its minimum when both inputs are the same. Therefore, the proximal regularization enforces the representation of a training example to not change too rapidly between updates, and allows for more stable/consistent negative examples, depending on the value of weight λ. The authors report improved results with a value of $\lambda = 30$ [Wu+18], meaning that the proximal regularization term is 30 times more important than the contrastive loss term.

This forces the model to keep the representations of the training examples in the memory bank consistent, so that a similarity measure can provide meaningful learning signals to the model. Our approach does not take this into account.

5.2.4.1.2 Momentum Encoder

Inconsistent representations in a memory bank is a problem also identified by the authors of MoCo [He+20], which employ a different approach to address this issue. They also use a queue-based memory bank, which is, similar to our approach, much smaller than the training dataset. However, in MoCo, the negative examples in the memory bank are not updated by the model that is being trained, but instead by a momentum encoder. The momentum en-

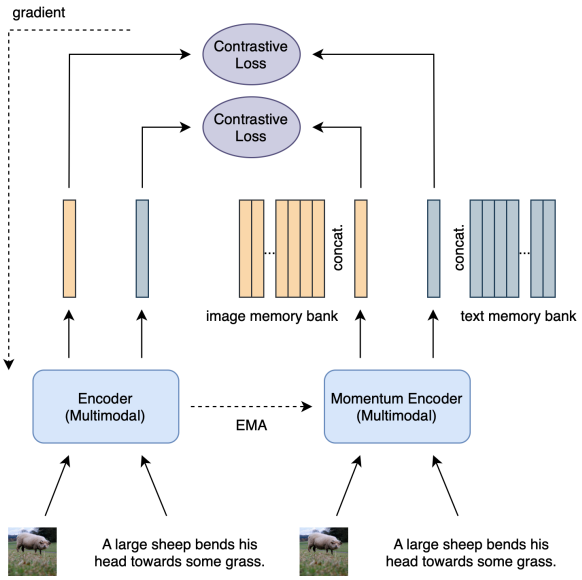

Figure 24: A momentum encoder generates negative examples for ITC, which are stored in a memory bank that discards the representations of the oldest batch when new ones are added after every step. Figure inspired by MoCo v2 [Che+20], image and text taken from COCO [Lin+14].

coder is a copy of the model, but its weights are an exponential moving average of the actual model weights, defined in Equation 69 [He+20]. Consequently, the momentum encoder's weights are not updated by gradient descent, and the negative examples do not come from the model that is trained, but only from the momentum encoder.

$$\theta_k = m * \theta_k + (1 - m) * \theta_q \tag{69}$$

With θ_k being the momentum encoder weights, θ_q the actual model weights, and m the momentum coefficient, the momentum encoder can be updated very slowly. Typical values for m are usually between $m = 0.99$ and $m = 0.999$ [CXH21, Che+20, He+20, Li+24].

The results are weights that change very slowly, which will also hold for the representations the momentum encoder produces. This approach can be seen as maintaining consistency in the model that produces the negative examples, rather than making the negative examples consistent themselves, as is done through the regularization term in Equation 68. An illustration of this method for our image-text contrastive learning can be seen in Figure 24.

Even though no gradients are needed to update the momentum encoder, it still requires additional GPU memory to keep it in memory, which is the disadvantage of this variant.

5.2.4.1.3 Resolution

We can't use the memory bank style of [Wu+18], since we have 3,264,868 training examples (see Table 4). Each embedding has a size of 768, and storing them at full precision (float32)

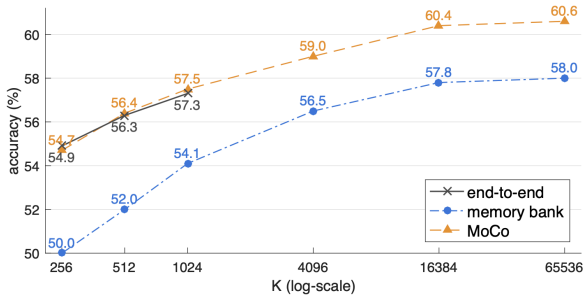

Figure 25: Experiments done by the authors of MoCo show that increasing the memory bank size up to 65k is beneficial to performance [He+20].

would require $3,264,868 * 768 * 4$ bytes ≈ 10 GB of additional GPU memory. However, with our current setting we only have around 1.2 GB of GPU memory remaining.

We suspect that using a proximal regularization term, as in Equation 68, could also stabilize our memory bank approach. However, we cannot apply it, since the term is based on the difference (MSE) between the representation of an individual training example when it was last updated in the memory bank, and its current representation in the batch. This requires the exact approach of [Wu+18], which we just deemed as infeasible.

In conclusion:

1. We cannot use a FIFO queue-based memory bank, as representations between samples are inconsistent.
2. We cannot use a memory bank that stores all training examples, as it would require too much additional GPU memory.
3. Proximal regularization is not applicable to a memory bank that is smaller than the training dataset.

The only alternative is to use a momentum encoder as in MoCo [He+20], which is why we opt for this approach in the next experiment. Our experimental setup remains the same, but we add a momentum encoder, which is a copy of our student model (as shown in Figure 24). Oriented on ALBEF [Li+24], we use a memory bank of size 65,536 and a momentum factor of $m = 0.995$. Both hyperparameters also lead to good results in MoCo, where this approach was first introduced [He+20].

However, we encounter an OOM error, which is not surprising considering the large memory bank size of 65k, and that we need to maintain two of these in total (see Figure 24). Considering that the size of the memory bank is crucial for performance (illustrated by MoCo [He+20] in Figure 25), we would like to keep it as large as possible. Based on this goal, we apply an optimization to reduce the GPU memory consumption:

Usually, the forward pass of the momentum encoder is done after the forward pass of the model that is trained, which is an approach MoCo [He+20] and ALBEF [Li+24] perform. It follows that during the forward pass of the momentum encoder GPU memory is already al-

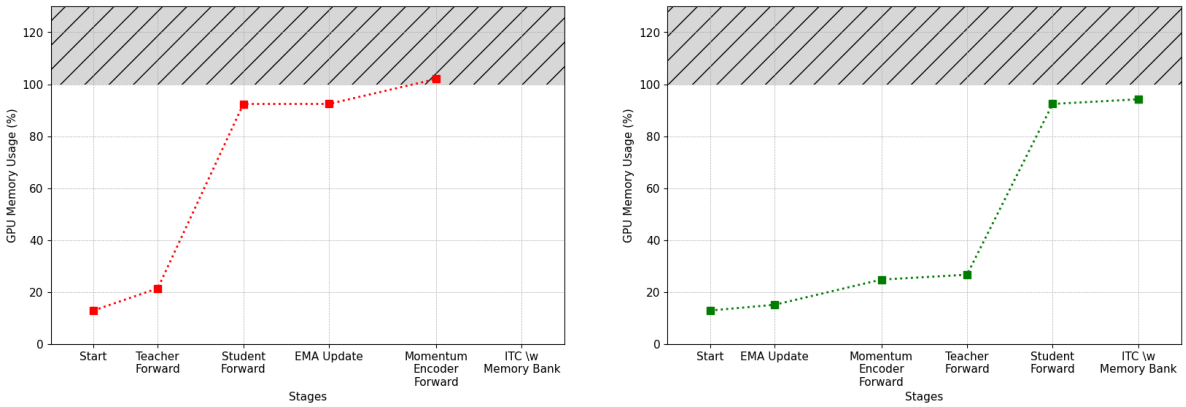

Figure 26: Doing the forward pass of the momentum encoder after the forward pass of the model, when the model's activations are already stored on the GPU memory (left), leads to a cuda OOM error. This can be avoided by reordering the operations, so that the momentum encoder (EMA) update and forward pass are performed before the forward pass of the model (right). Results are based on NVIDIA RTX 4090 with 24 GB of GPU memory.

located to store all activations of the actual model, as they are needed for the backward pass later. Because we did not encounter an OOM error before using the momentum encoder, and we observed that the GPU memory in previous experiments was almost fully utilized (up to 98%) without a momentum encoder, we suspect that the forward pass of the momentum encoder is the reason for the overflow.

This can be remedied by performing the update and forward pass of the momentum encoder in each training step before any other work is done (this includes the forward pass of the teacher and student). This way, the activations of intermediate layers of the momentum encoder are freed before the forward pass of the actual model. The result is the same performance, as the work done per step remains the same. Merely the operations in a step are reordered to avoid the memory overflow. We illustrate this in Figure 26.

The result is shown in Table 22. The performance does not exceed that of the the standard gathering from all devices with just 511 negative examples (effective batch size of 512). However, the experiment seems more promising to achieve a better retrieval performance with more epochs, as it does not appear to saturate towards the end of training (see Figure 27).

We consider efficiency and simplicity as a key aspect of our work. Since adding a momentum encoder

1. increases the complexity of our approach,
2. increases the training time from approx. 7 hours to 10.3 hours, and
3. does not lead to a significant improvement in performance,

we decide to abandon this approach. As a side note: The increase training time can be attributed to the additional forward pass of the momentum encoder, and the large matrix multiplication resulting from the large memory bank.

Momentum Encoder	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
×	**30.3**	**67.2**	**78.29**
✓	30.0	64.28	76.17

Table 22: Comparison of the Standard ITC approach with a momentum encoder and memory bank of size 65,536. The momentum encoder approach does not exceed the performance of the standard ITC approach.

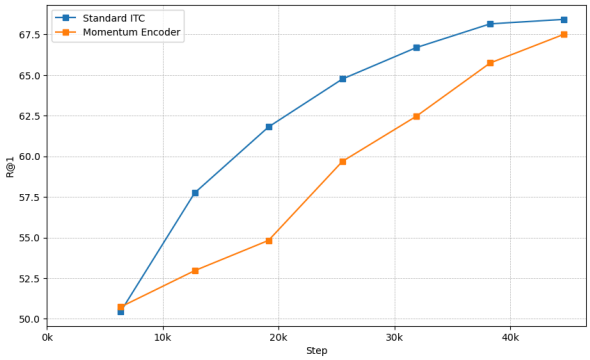

Figure 27: Comparison of the average R@1 score for image and text retrieval on the MSCOCO val set throughout training. A momentum encoder shows a promising trend to achieve better retrieval performance towards the end of training.

5.2.5 Target Cross-Modal Late Interaction

5.2.5.1 Cross-Modal Late Interaction

Until now, we used the global text and image representations, given by [T_CLS] and [I_CLS], for contrastive learning. This has the disadvantage that only global information is utilized, and fine-grained, token/patch-specific, information is not considered. This can make retrieval difficult, especially if real-world concepts described by an image and a text differ in small, yet important details. An example of this can be seen in Figure 43, where multiple retrievals are incorrect, even though they are semantically very similar to the query. The differences between the query and the retrieved samples are often so small that they are not captured by the global representations. To address the issue of fine-grained alignment, the authors of FILIP [Yao+21] introduced Cross-Modal Late Interaction (CMLI) for contrastive learning, which led to improvements in retrieval performance.

As shown in Figure 28, no cosine similarity between [T_CLS] and [I_CLS] is computed, but instead the cosine similarity between all image patches $\left[\boldsymbol{h}_{v,l,k}\right]_{1\leq k\leq N}$ and text tokens $\left[\boldsymbol{h}_{w,l,j}\right]_{1\leq j\leq M}$, with N being the number of image patches, and M being the number of text tokens. Specifically, N and M denote the number of patches/tokens in a sequence that are not the cls token ([I_CLS]/[T_CLS]) or padding token ([PAD]) [Yao+21]. The choice to exclude padding tokens is obvious, as they do not carry any semantic information. The cls token is

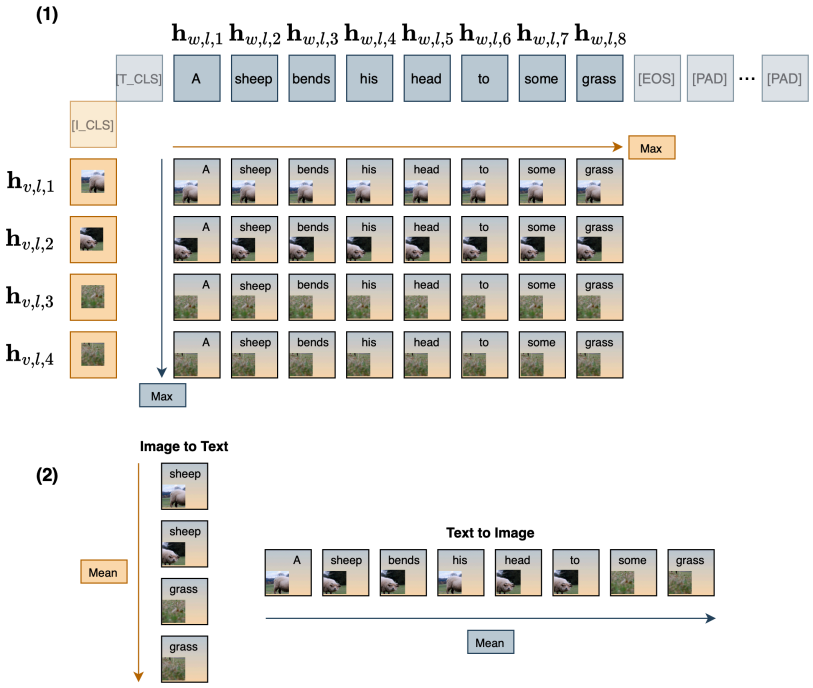

Figure 28: For a token/patch, CMLI finds the timestep with the highest semantic match from the other modality. This enables the model to associate small details of image and text with each other. Notice how through the max-operation patches containing grass are always associated with the word "grass", and the words "sheep" and "head" are matched with the head of the sheep (associations created through max are shown in (2)). The cosine similarity is then the average of all associations between an image-text pair. Figure inspired and adapted from [Yao+21].

excluded, as it is not specific to any token/patch, and is used to represent global information. We additionally exclude the end-of-sequence token ([EOS]), as it is not specific to any token/patch either. The result is the cosine similarity between all image patches and text tokens of an image-text pair.

The next step is to find for each image patch k the text token with the maximum cosine similarity to this image patch.

$$m_k^{\text{i2t}} = \underset{1 \leq j \leq M}{\operatorname{argmax}} \left[\boldsymbol{h}_{v,l,k}\right] \left[\boldsymbol{h}_{w,l,j}\right]^T \tag{70}$$

Likewise, for each text token j, we get the image patch with the maximum cosine similarity to this text token

$$m_j^{\text{t2i}} = \underset{1 \leq k \leq N}{\operatorname{argmax}} \left[\boldsymbol{h}_{v,l,k}\right] \left[\boldsymbol{h}_{w,l,j}\right]^T \tag{71}$$

This has an interesting effect: For each image patch the semantically most similar text token is found, and vice versa for each text token - the result of this operation can be seen in (2) of

Figure 28. Consequently, the model will be able to associate small details of an image with individual text tokens, and vise versa. The actual cosine similarity between an image-text pair is then the average of all associations between an image patch and a text token.

$$s^{\text{i2t}}_{\boldsymbol{H}_{v,l},\boldsymbol{H}_{w,l}} = \frac{1}{N}\sum_{k=1}^{N}\left[\boldsymbol{h}_{v,l,k}\right]\left[\boldsymbol{h}_{w,l,m^{\text{i2t}}_k}\right]^T \tag{72}$$

$$s^{\text{t2i}}_{\boldsymbol{H}_{v,l},\boldsymbol{H}_{w,l}} = \frac{1}{M}\sum_{j=1}^{M}\left[\boldsymbol{h}_{v,l,m^{\text{t2i}}_j}\right]\left[\boldsymbol{h}_{w,l,j}\right]^T \tag{73}$$

Here, for one image-text pair, m^{i2t}_k denotes the index of the text token with the highest cosine similarity to image patch k, and m^{t2i}_j the index of the image patch with the highest cosine similarity to text token j. $s^{\text{i2t}}_{\boldsymbol{H}_{v,l},\boldsymbol{H}_{w,l}}$ denotes the the similarity score between an image representation $\boldsymbol{H}_{v,l}$ and text representation $\boldsymbol{H}_{w,l}$. Vice versa, $s^{\text{t2i}}_{\boldsymbol{H}_{v,l},\boldsymbol{H}_{w,l}}$ denotes the similarity score between a text representation $\boldsymbol{H}_{w,l}$ and an image representation $\boldsymbol{H}_{v,l}$. l can denote any layer of the model, but is usually the last layer, as the representations are most meaningful there.

In contrast to the standard contrastive learning, this similarity measure is not necessarily symmetric, as e.g. a text token might have a maximum cosine similarity to another image patch than the image patch that has its maximum similarity to the text token [Yao+21]. The process in illustrated in Figure 28.

While this approach allows for a fine-grained alignment of image and text, its practical implementation is very computationally and memory intensive. For standard constrastive learning is is sufficient to compute the cosine similarity of the global representation (cls token) between every possible image-text pair in a batch. If negative examples are gathered from all devices, then the number of dot products to compute is defined as $(B * P)^2$, with B being the batch size per device, and P being the number of devices (in our case GPUs). As we use a batch size of $B = 256$ per device, and use $P = 2$ GPUs, the number of dot products to compute is $(256 * 2)^2 = 262,144$. Considering that we perform this efficiently using matrix multiplication, and the embedding size is 768 with float32 precision, we already need $262,144 * 768 * 4 \text{ bytes} = 805.31 \text{ MB}$ of GPU memory, which is still manageable, since we have around 2 GB of GPU memory remaining for a step.

However, with CMLI, we need to compute the similarity between all possible image-text pairs, where the similarity for one pair requires the computation of the cosine similarity between all image patches and text tokens of the image-text pair. With a maximum text sequence length of 64 tokens [Wan+23] and an image sequence length of 197, the number of dot products to compute for just one image-text pair is $197 * 64 = 12,608$. With a batch size of 256 per device, and 2 GPUs, the number of dot products increases from $262,144$ to $256 * 2 * 12,608 = 6,455,296$. Even if the embedding dimension is reduced to 256, which

is a simplification done in FILIP [Yao+21], we need $6,455,296 * 256 * 4$ bytes $= 6.61$ GB of additional GPU memory just to store the result. This is not feasible in our setup.

5.2.5.2 Method

What is feasible though, is to apply CMLI to a setting where the computation is more lightweight. As the driving factor behind the memory requirements in contrastive learning is that the similarity between all possible image-text pairs in a batch is computed, this is removed when just the similarity between positive pairs is computed, which is what we call Target-CMLI.

Target-CMLI is not used for contrastive learning, but rather to alleviate the problem of regressing patch-level information of the teacher model. Recall that in the current setting it is merely possible to regress the global representations of the teacher model, and not the patch-level information. This is because the teacher model outputs patch-level predictions for the image modality, but not for the text modality. Consequently, the student model can replicate the output of the teacher model for the image modality, but not for the text modality, as it is not possible to assign a text token to a specific image patch. This is illustrated in Figure 22, and discussed in Section 5.2.2.1.

However, as seen in Figure 28, when computing the cosine similarity between all image patches and text tokens of an image-text pair, and selecting the argmax over all image patches with respect to a text token, then a corresponding, or at least similar, image patch can be found for each text token ((2) of Figure 28). This means that the student model can replicate the output of the teacher model for the text modality by first selecting the most similar image patch for each text token, and then minimizing the Mean Squared Error (MSE) between the teacher's representation of the selected image patch and the representation of the text token.

For the patch-level image representation of the student model that means that we can now also regress the patch-level information of the teacher, and not only the global information. This was also possible in all previous experiments, as the order of the image patches does not change between student and teacher image representations. However, this would heavily bias the parameters of the shared Transformer block towards the image modality.

The definition of the loss changes as follows:

$$\mathcal{L}_{\mathrm{KD}}^{\mathrm{i2i}} = \mathcal{L}_{\mathrm{MSE}}\Big(\boldsymbol{H}_{v,K}^{s}, \boldsymbol{H}_{v,L_t}^{t}\Big) = \sum_{n=1}^{N} \|\boldsymbol{h}_{v,K,n}^{s} - \boldsymbol{h}_{v,L_t,n}^{t}\|_2^2 \tag{74}$$

For a given text representation we first need to find m_j^{t2i} for each text token j, and then define the loss as:

$$\mathcal{L}_{\mathrm{KD}}^{\mathrm{t2i}} = \mathcal{L}_{\mathrm{MSE}}\Big(\boldsymbol{H}_{w,K}^{s}, \boldsymbol{H}_{v,L_t}^{t}\Big) = \sum_{z=1}^{M} \|g\Big(\boldsymbol{h}_{w,K,z}^{s}\Big) - g\Big(\boldsymbol{h}_{v,L_t,m_z^{\mathrm{t2i}}}^{t}\Big)\|_2^2 \tag{75}$$

T-CMLI	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
×	**30.3**	**67.2**	**78.29**
✓	26.7	65.68	76.2

Table 23: Comparison of adjusting the loss function $\mathcal{L}_{\text{KD}}$ from regressing the [I_CLS] of the teacher model to regressing the patch-level information of the teacher model using Target-CMLI. We observe a decrease in all metrics.

We denote $g(\cdot)$ as a linear projection to reduce the dimensionality of the image representation from the teacher, and the text representation from the student, to 32. This is done to (1) reduce memory consumption when computing the similarity between all image patches and text tokens for each image-text pair, and (2) to reduce the information that can be expressed by $g(\boldsymbol{x})$. We motivate (2) by the fact that matching individual image patches to text tokens is a very fine-grained task, and we want to avoid having pixel-level information in the patch representations of the teacher model, which would, despite the matching via argmax, cause problems with predicting those representations, as we can't extract those pixel-level information from the text tokens. The total knowledge distillation loss remains the same, and is the mean of the two losses:

$$\mathcal{L}_{\text{KD}} = \frac{1}{2} * (\mathcal{L}_{\text{KD}}^{\text{i2t}} + \mathcal{L}_{\text{KD}}^{\text{t2i}}) \tag{76}$$

We use the mean instead of the sum, as we also use the contrastive loss, and we want both losses to have the same weight. We find m_k^{t2i} using an embedding dimension of 32, which is achieved through the projection $g(\cdot)$. The hidden size of the model remains at 768.

What makes the implementation of Target-CMLI feasible is that we only need to find the most similar image patch for each text token in a positive pair, and not all possible image-text pairs in a batch. For a per-device batch size of 256, Target-CMLI requires $12,608 * 256 = 3,227,648$ dot products to compute. With an embedding dimension of 32 just $3,211,264 * 32 * 4$ bytes $= 411$ MB of additional GPU memory is required. This is feasible in our setup.

5.2.5.3 Results

The results, as seen in Table 23, can be considered as disappointing. We lose more than 2 percentage points in average retrieval on both MSCOCO and Flickr30K, while the performance on ImageNet-1K decreases by nearly 4 percentage points.

A look at a visualization (Figure 29) of matches between text tokens and their top-5 most similar image patches under cosine similarity reveals that while there are some cases where the token has the highest similarity to an actual image patch belonging the the correct object, the matching is far from perfect, and matches are often either completely unrelated, or scattered across almost random regions. A matching that would be expected from a successful approach is illustrated by the authors of FILIP [Yao+21], and can be seen in Figure 41. While

the exaples of FILIP are based on CMLI for contrastive learning, and not our Target-CMLI for knowledge distillation, the principle of matching text tokens to image patches remains the same, and the quality of the matches is expected to be similar.

In general, we observe that the patch with the highest similarity for a text token is often a patch completely unrelated to the token. While it is true that, thanks to self-attention, the representation of an image patch not only contains information about the patch itself, but also about patches it considers important, the self-attention map of the image patch with the highest similarity to a text token, which is the matched image patch, does not show any clear signs that it contains aggregated information from patches that are part of the object the text token describes. If that were the case, then the self-attention map of the matched image patch would show a clear focus on the object the text token describes/represents. This is illustrated in Figure 29, where for each example, the left image shows the top 5 most similar image patches for a text token under cosine similarity, and the right image shows the self-attention map of the image patch with the highest similarity to the text token, i.e. the image patch m_j^{t2i} for a text token j. For the aforementioned, we observe a very inconsistent matching between text tokens and image patches. Especially examples "planes" and "birds" show that a mismatch is not rare.

As a reference, we provide the self-attention map, w.r.t. the [I_CLS] token, of the final Transformer layer from the self-supervised image model DINO [Car+21] in Figure 42 (Appendix). We would expect the self-attention map of the matched image patch m_j^{t2i} to the example text token j (Figure 29) to be similar to that of DINO. However, this is not always the case.

Moreover, the top 5 matched image patches are often scattered across the image, and not focused on a specific region, underlining that the approach does not work as intended. Instead, a result similar to that of FILIP in Figure 41 is expected, where the matched image patches are focused on the object the text token describes, and lie next to each other. Our results are much more similar to that of CLIP, also illustrated by the authors of FILIP (Figure 41), leading us to believe that a missing cross-modal attention might be the reason.

We suspect this, because FILIP [Yao+21], like VLMo [Bao+22] and BEiT-3 [Wan+23], uses cross-attention to align text and image. This allows individual text tokens to attend to specific image patches, and vice versa, leading to e.g. text tokens to be able to "locate" the objects they describe in an image. Cross-Modal Late Interaction then allows to apply this "locating of the matched image patch" to contrastive learning and retrieval.

CLIP however, like our model, does not use cross-attention, and only uses global representations for contrastive learning. Therefore, CLIP is not able to find relationships between individual text tokens and image patches, leading to a scattered matching, as seen in Figure 41.

While we do not use CMLI for contrastive learning, but for knowledge distillation, the same principles apply. Even worse, since the image patches that we try to match to text tokens are not even the result of the student model, but of the teacher model, the model does not have

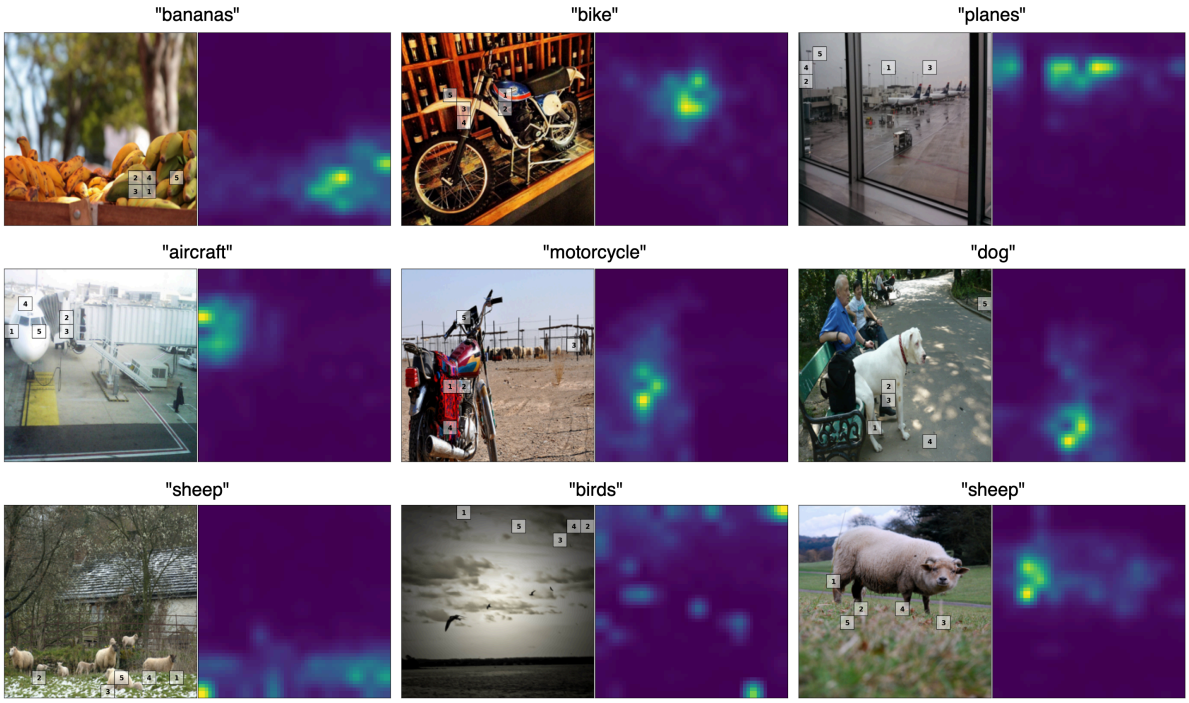

Figure 29: Visualization of selected text tokens from the image's caption and the tokens top 5 most similar image patches under cosine similarity (left). The self-attention map (right) of the image patch with the highest similarity to the text token is shown next to the original image. We observe that matched image patches are often scattered across the image, are not part of the object the text token represents, and their self-attention map, indicating the composition of the token, does not always show a clear focus on the object the text token describes. The title shows a text token j, and the self-attention map with respect to image patch m_j^{t2i} (i.e. the matched one). Image-text pairs taken from COCO test set [Lin+14].

the possibility to somehow learn patch-level representations that are suitable for matching to text tokens. Consequently, there really is no guidance for the model to learn matching its text tokens to the right image patches of the teacher model.

Lastly, Figure 29 shows the problem based on text tokens that have a real-world meaning, and therefore a counterpart in images: The text token "plane" can also be present in an image as an actual plane. However, text tokens like "a", "the", and even a full stop ".", which is a valid token, are also matched to image patches. They are merely fill words or grammatic nuances in a sentence, and do not carry any semantic information that can be mapped to an image patch. Because CMLI works over all text tokens, and only few text tokens actually have an object-level counterpart in images, like "plane", most of the matchings between text tokens and image patches are not meaningful to begin with. Recall that this was one of the reasons why we decided to only regress the teacher's `[I_CLS]` token (see Section 5.2.2.1).

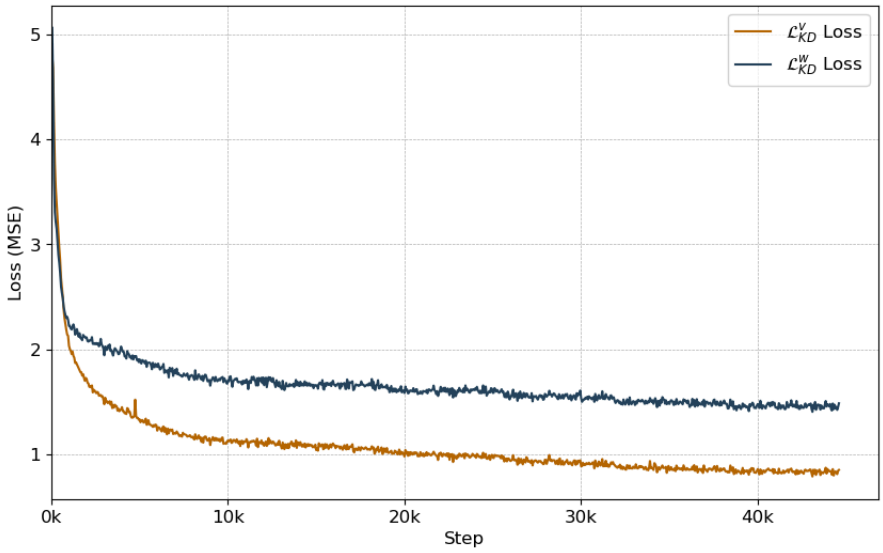

Figure 30: Training loss for the image and text component of the $\mathcal{L}_{\text{KD}}$ loss. The image component $\mathcal{L}_{\text{KD}}^{v}$ shows a significantly lower loss compared to the text component $\mathcal{L}_{\text{KD}}^{t}$, indicating that the [I_CLS] of the teacher still contains image-specific information.

To come to a conclusion, even though Target-CMLI seems to be a promising approach to alleviate the mismatch between text tokens and image patches, especially considering that some examples in Figure 29 show a self-attention map that is focused on the object the text token describes, the results are far from consistent, and are constrained to only a few text tokens that have a real-world counterpart in images, and even this are not reliable.

5.2.6 Contrastive Distillation

Throughout the previous experiments, we have seen that the misalignment between image patches and text tokens leads to problems when regressing the image features of the teacher. While we are already exceeding the performance of the supervised teacher by predicting the [I_CLS] of the teacher, there is still room for improvement. A glance at the loss $\mathcal{L}_{\text{KD}}$, which is now again the previous one (defined again in Equation 77 for ease of access), since Target-CMLI (Section 5.2.5.2) proved to be ineffective, shows that the loss for the image features is clearly lower than that for the text features. This indicates that the [I_CLS] of the teacher still contains image-specific information. If that were not the case, the loss for both components, i.e. $\mathcal{L}_{\text{KD}}^{v}$ and $\mathcal{L}_{\text{KD}}^{w}$, would be similar. Consequently, we aim to introduce a modality-invariant target loss that is less affected by the image-specific information in the [I_CLS] of the teacher.

$$\begin{aligned} \mathcal{L}_{\text{KD}} = \\ \frac{1}{2} * \mathcal{L}_{\text{KD}}^{v} + \frac{1}{2} * \mathcal{L}_{\text{KD}}^{w} = \\ \frac{1}{2} * \|\boldsymbol{h'''}^{s}_{v,K,[\texttt{I\_CLS}]} - \boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]}\|_2^2 + \frac{1}{2} * \|\boldsymbol{h'''}^{s}_{w,K,[\texttt{T\_CLS}]} - \boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]}\|_2^2 \end{aligned} \tag{77}$$

5.2.6.1 Contrastive Target Loss

We identify the MSE loss, used as a criterion for knowledge distillation, as unfavourable, as it enforces the student representation of the caption $\boldsymbol{h'''}^{s}_{w,K,[\texttt{T\_CLS}]}$ to be the same as the teacher representation of the image $\boldsymbol{h}^{t}_{v,L_t,[\texttt{I\_CLS}]}$. The loss only becomes zero if both are identical. This is less of a problem for $\boldsymbol{h'''}^{s}_{v,K,[\texttt{I\_CLS}]}$, as this is the image representation of the student, which can contain the image-specific information. However, the student is not able to extract the image-specific information from the caption.

To address this issue, we propose a contrastive target loss, which is inspired by contrastive learning. This loss collects all teacher representations from the current batch $\left\{\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),b}\right\}_{b=1}^{B}$, and computes the cosine similarity between those representations and the student representation of a candidate caption $\boldsymbol{h'''}^{s}_{w,K,[\texttt{T\_CLS}]}$. Similar to the contrastive loss, we aim to maximize the similarity between the student representation of caption $\boldsymbol{h'''}^{s}_{(v,K,[\texttt{I\_CLS}]),i}$ and the teacher representations of image $\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),i}$ in the batch, while minimizing the similarity between the student representation of caption $\boldsymbol{h'''}^{s}_{(v,K,[\texttt{I\_CLS}]),i}$ and the teacher representations of all other images $\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),j}, j \neq i$ in the batch.

This has the advantage that we are now focusing less on making $\boldsymbol{h'''}^{s}_{(v,K,[\texttt{I\_CLS}]),i}$ and $\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),i}$ identical, but rather on what representations match, and which do not This puts more emphasis on the relative similarity with respect to the similarity to other images in the batch. Maximizing the cosine similarity is further a less strict criterion than minimizing the MSE, as the cosine similarity only requires both representations to be in the same direction, but not necessarily to be identical.

Like in the contrastive loss, if the representations are good enough $\boldsymbol{h'''}^{s}_{(v,K,[\texttt{I\_CLS}]),i}$ and $\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),i}$ will have more in common than $\boldsymbol{h'''}^{s}_{(v,K,[\texttt{I\_CLS}]),i}$ and $\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),j}$ for $in = j$, so the student representation of the caption i is more likely match the teacher representations of the image i.

5.2.6.2 Implementation

The implementation closely follows that of the image-text contrast of Section 2.5.2.2. We concatenate all teacher representations of the image $[\texttt{I\_CLS}]$ in the batch to a tensor, and all student representations of the caption $[\texttt{T\_CLS}]$ to another tensor:

$$\begin{aligned}\boldsymbol{I}_t &= \left[\boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),1}, \boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),2}, ..., \boldsymbol{h}^{t}_{(v,L_t,[\texttt{I\_CLS}]),B'}\right] \in \mathbb{R}^{B' \times D} \\ \boldsymbol{T}_s &= \left[\boldsymbol{h'''}^{s}_{(w,K,[\texttt{T\_CLS}]),1}, \boldsymbol{h'''}^{s}_{(w,K,[\texttt{T\_CLS}]),2}, ..., \boldsymbol{h'''}^{s}_{(w,K,[\texttt{T\_CLS}]),B'}\right] \in \mathbb{R}^{B' \times D}\end{aligned} \tag{78}$$

Additionally, we also collect all image representations of the student $[\texttt{I\_CLS}]$ in the batch to a tensor:

$$\boldsymbol{I}_s = \left[\boldsymbol{h}^{s}_{(v,K,[\texttt{I\_CLS}]),1}, \boldsymbol{h}^{s}_{(v,K,[\texttt{I\_CLS}]),2}, ..., \boldsymbol{h}^{s}_{(v,K,[\texttt{I\_CLS}]),B'}\right] \in \mathbb{R}^{B' \times D} \tag{79}$$

We define B' as the combined batch size over all devices, as we can gather all representations from all devices (see Section 5.2.1.2). It therefore holds that $B' = B * P$, where P is the number of devices. In our case we use $P = 2$ devices and $B = 256$ samples per device, resulting in $B' = 2 * 256 = 512$.

The cosine similarity can again be computed efficiently using matrix multiplication of the normalized representations:

$$\begin{aligned} \boldsymbol{L}^w &= \delta(\boldsymbol{T}_s)\delta(\boldsymbol{I}_t)^T \in \mathbb{R}^{B' \times B'} \\ \boldsymbol{L}^v &= \delta(\boldsymbol{I}_s)\delta(\boldsymbol{I}_t)^T \in \mathbb{R}^{B' \times B'} \end{aligned} \tag{80}$$

Here, δ denotes the normalization:

$$\begin{aligned} \delta(\boldsymbol{X}) &= [\delta(\boldsymbol{x}_1), \delta(\boldsymbol{x}_2), ..., \delta(\boldsymbol{x}_{B'})] \in \mathbb{R}^{B' \times D} \\ \delta(\boldsymbol{x}) &= \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2} \in \mathbb{R}^D \end{aligned} \tag{81}$$

Notice how the contrastive target loss is computed both between the student representation of the caption and the teacher representations of the image, which is the important part we covered in the previous section, and between the student representations of the image and the teacher representations of the image. The latter is the image-to-image part of the knowledge distillation loss, which does not suffer from image-specific information in the teacher representation of the image, but we also adapt it to the contrastive loss for consistency.

Analogous to image-text contrast, we define the loss, which is just cross-entropy, as follows:

$$\begin{aligned} \mathcal{L}_{\text{KD}}^{\text{t2i}} &= \frac{1}{B'} \sum_{i=1}^{B'} -\log \frac{\exp\left(L_{i,i}^w\right)}{\sum_{k=1}^{B'} \exp\left(L_{i,k}^w\right)} \\ \mathcal{L}_{\text{KD}}^{\text{i2i}} &= \frac{1}{B'} \sum_{i=1}^{B'} -\log \frac{\exp\left(L_{i,i}^v\right)}{\sum_{k=1}^{B'} \exp\left(L_{i,k}^v\right)} \\ \mathcal{L}_{\text{KD}} &= \frac{1}{2} * \mathcal{L}_{\text{KD}}^{\text{t2i}} + \frac{1}{2} * \mathcal{L}_{\text{KD}}^{\text{i2i}} \end{aligned} \tag{82}$$

We rename the components of the loss from $\mathcal{L}_{\text{KD}}^w$ to $\mathcal{L}_{\text{KD}}^{\text{t2i}}$, and $\mathcal{L}_{\text{KD}}^v$ to $\mathcal{L}_{\text{KD}}^{\text{i2i}}$, in order to reflect the text-to-image and image-to-image parts of contrastive learning, respectively.

When comparing our current configuration to our previous best, which was reached when introducing a token-type embedding after the modality specific encoders (Section 5.2.3), we find that while performance on MSCOCO and Flickr30K retrieval remains unchanged, performance on ImageNet-1K improves by nearly 3 percentage points. This improvement on ImageNet-1K is likely due to our use of CLIP zero-shot classification, which involves retrieving the most similar class prototype using a contrastive loss (see Section 2.7.2.2). Since we now employ a contrastive loss for knowledge distillation, and the weights of the teacher

KD Loss	ImageNet-1K	Retrieval	
		MSCOCO	Flickr30K
MSE	30.3	**67.2**	78.29
Contrastive	**33.0**	67.15	**78.3**

Table 24: Replacing the MSE loss with the contrastive target loss improves the performance on ImageNet-1K by almost 3 percentage points, while matching the performance on MSCOCO and Flickr30K retrieval.

model — responsible for generating one component of the contrastive target loss (the teacher representations $\boldsymbol{I}_t$ of the images) — were originally trained on ImageNet-1K, the student model's learning method aligns more closely with CLIP zero-shot classification. This alignment is likely to enhance performance on ImageNet-1K.

5.2.6.3 Memory Bank

In Section 5.2.4, we evaluated the possibility of storing representations, produced by the student, from previous batches in a memory bank. The idea was that since the contrastive loss requires a large number of negative samples to be effective, we could use representations from previous batches as additional negative samples. However, we found that using representations from previous batches leads to a significant performance drop, as the representations were inconsistent.

Fortunately, the contrastive target loss does not suffer from outdated representations. This is because the representations we compare the student representation to are from the teacher. The teacher's weights are frozen during training, meaning that the teacher's representations are consistent over the entire training process. Therefore, all negative examples that are used in the contrastive target loss are consistent with each other, meaning we can savely use a simple memory bank to store the teacher representations from previous batches. Workarounds like a momentum encoder (Section 5.2.4.1.2) or proximal regularization of the features (Section 5.2.4.1.1) are not necessary.

The formulation of the loss does not change, but only the concatenated teacher representations.

$$\boldsymbol{I}_t' = \boldsymbol{I}_t \| \boldsymbol{V} = \left[\boldsymbol{h}^t_{(v,L_t,[\texttt{I\_CLS}]),1}, ..., \boldsymbol{h}^t_{(v,L_t,[\texttt{I\_CLS}]),B'}, \boldsymbol{v}^t_{(v,L_t,[\texttt{I\_CLS}]),1}, ..., \boldsymbol{v}^t_{(v,L_t,[\texttt{I\_CLS}]),G}\right] \in \mathbb{R}^{(B'+G)\times D} \tag{83}$$

$$\boldsymbol{L}^w = \delta(\boldsymbol{T}_s)\delta(\boldsymbol{I}_t')^T \in \mathbb{R}^{B'\times G} \tag{84}$$

We denote $\boldsymbol{v}^t_{(v,L_t,[\texttt{I\_CLS}]),i}$ as the teacher representation of the image i from the memory bank, so from a previous batch, and G as the number of representations stored in the memory bank, i.e. the size. We set $G = 65536$ in our experiments, which we orientate on the ideal size found by MoCo [He+20] for contrastive learning (see Figure 25).

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
FLAVA [Sin+21]	42.74	76.76	-	**38.38**	**67.47**	-	67.7	94.0	-	65.22	89.38	-
CLIP [Rad+21]	**58.4**	**81.5**	88.1	37.8	62.4	72.2	**88.0**	**98.7**	**99.4**	**68.7**	**90.6**	**95.2**
S-SMKE	51.66	79.9	88.66	36.17	66.55	**78.28**	64.5	88.4	93.0	51.78	78.54	86.46
S-SMKE$_{\text{CTL}}$	52.68	80.56	88.3	36.5	66.58	**78.28**	69.0	89.2	94.3	51.48	79.18	86.66
S-SMKE$_{\text{CTL_MB}}$	53.54	81.1	**89.52**	35.65	66.0	77.77	70.9	92.1	96.0	52.72	80.2	87.46

Table 25: A contrastive target loss with memory bank especially improves text retrieval, while the performance on COCO image retrieval degrades slightly. S-SMKE$_{\text{CTL}}$ denotes the contrastive target loss without memory bank, and S-SMKE$_{\text{CTL_MB}}$ denotes the contrastive target loss with memory bank.

A detailed look on retrieval performance in Table 25 shows a gain especially on text retrieval tasks, and we are able to increase text retrieval on Flickr30K by approx. 2-3 percentage points in each metric. This is surprising, as we would expect an increase in image retrieval performance, because the contrastive target loss uses text-image retrieval, expressed through the loss $\mathcal{L}_{\text{KD}}^{\text{t2i}}$ in Equation 82. This should have a positive effect on text-image retrieval in COCO and Flickr30K, but we instead observe a slight decrease in performance. Simultaneously, the performance on image-text retrieval increases consistently, even though the contrastive target loss does not use image-text retrieval. We would expect the opposite.

The performance on ImageNet-1K is increasd by an impressive 4 percentage points to 37% (33% before). Considering the increase in image-text retrieval on Flickr30K and COCO, this is actually to be expected: In CLIP zero-shot classification, a candidate image is compared to all class prototypes, which have been created from text descriptions of the classes. The class prototype, which is a text representation, with the highest similarity to the image is then chosen as the predicted class, which is essentially image-text retrieval. Therefore, if the performance on image-text retrieval increases, we would expect an increase in performance on ImageNet-1K.

5.3 Teacher Ablation Studies

In this section we will investigate how sensitive our approach is to teachers different from BEiTv2 [Pen+22], and we will test if a teacher is needed at all. To avoid repretition, we will will first introduce the individual ablations, and then present the results at the end of this section.

5.3.1 Different Teachers

First, we will train the same model as in the previous experiments, but this time using two different self-supervised image teachers: Data2Vec2 [Bae+23] and DINO [Car+21]. The choice of Data2Vec2 [Bae+23] is motivated by the fact that we already used it in the unimodal image distillation in Section 5.1.1, and because we initialize the image encoder of the multimodal model with the weights of the first 6 layers of Data2Vec2. DINO [Car+21] has shown to be a self-supervised model with an excellent understanding of image features, and its attention mechanism is able to detect high-level semantics in images, examples of which are illustrated in Figure 42. We therefore test if this deep image understanding can be used to better align image and text, or if the features are too image-specific to be useful for the multimodal task.

In both cases, we will use the contrastive target loss with a memory bank for the knowledge distillation process. As the output representation of the teacher we keep using the representation of the [I_CLS] token, $\boldsymbol{h}_{v,L_s,\texttt{[I\_CLS]}}$.

Furthermore, we employ BEiTv2 finetuned on ImageNet-1K as a teacher model. Essentially, this is the same teacher used in all our experiments for S-SMKE, but with additional finetuning on ImageNet-1K after pretraining. Since the teacher is supervised in this case, we will utilize the KL-Divergence loss, as in SHRe [AVT17] and our experiments with Transformer SHRe (see Section 5.2.1). We take this path for two main reasons. First, the teacher used in Transformer SHRe was a ResNet-50 [He+16, WTJ21] model with only 25 million parameters, compared to the 86 million parameters of BEiTv2. Therefore, comparing our end-to-end self-supervised approach with the results of Transformer SHRe is not unbiased due to the difference in teacher model sizes. Second, we aim to investigate the direct impact of using a supervised teacher with S-SMKE. By utilizing BEiTv2 as a self-supervised teacher in S-SMKE and now employing the same model finetuned on ImageNet-1K, we can directly assess the difference that additional supervised training of the teacher makes.

All teachers are based on the same ViT-B/16 [Bae+23, Car+21, Dos+21] architecture, which is the same as used by BEiTv2 [Pen+22], meaning all teachers are comparable in terms of model size and complexity.

5.3.2 Removing Distillation

Throughout the previous sections we repeatedly attempted to improve our approach by reducing the gap between the text-to-image and image-to-text knowledge distillation losses. Unfortunately, only one of them, the contrastive target loss, was successful, but only increased the performance marginally. That is why we will now investigate whether the knowledge distillation process is beneficial at all. To test this, we will train a version without any knowledge distillation, and only use the contrastive loss:

Teacher	ImageNet-1K	MSCOCO		Flickr30K	
		I2T	T2I	I2T	T2I
BEiTv2 [Pen+22]	37.0	**74.72**	59.81	**86.33**	**73.46**
BEiTv2 FT [Pen+22]	34.6	72.5	**61.16**	82.23	72.07
Data2Vec2 [Bae+23]	24.5	66.72	53.96	74.83	64.09
DINO [Car+21]	**37.5**	71.88	57.63	82.07	69.45
–	25.8	69.97	58.05	79.7	68.57

Table 26: Comparison of the retrieval performance when using different teachers for knowledge distillation. "I2T" and "T2I" denote text retrieval from image and image retrieval from text, respectively. We denote a BEiTv2 finetuned on ImageNet-1K as "BEiTv2 FT".

$$\mathcal{L}_{\text{S\_SMKE}} = \mathcal{L}_{\text{CL}} \tag{85}$$

Since we now only focus on the alignment of the visual and textual features, we remove the linear layer that produced the representations $\boldsymbol{h}'''_{v,K,[\texttt{I\_CLS}]}$ and $\boldsymbol{h}'''_{w,K,[\texttt{T\_CLS}]}$. They were used to predict the teacher representation $\boldsymbol{h}_{v,L_s,[\texttt{I\_CLS}]}$ using the mean squared error loss in earlier experiments, and using the contrastive target loss in the latest successful experiments. However, without knowledge distillation, they are no longer needed.

Everything else about the model architecture and training process remains the same, both the image and text encoder are still initialized with layers from Data2Vec2 [Bae+23] and BERT [Dev+19], respectively.

5.3.3 Results

We show a comparison of the retrieval performance between different teacher, and the performance on ImageNet-1K, in Table 26. We observe that the choice of the teacher is significant for the performance of the student, especially in the unimodal case. We observe a significant reduction across all benchmarks when using Data2Vec2 [Bae+23] as the teacher, and a slight reduction when using DINO [Car+21]. For Data2Vec2, we lose up to 12 percentage points in the retrieval metrics, and 12.5 percentage points on ImageNet-1K CLIP-like classification, while for DINO, we lose up to 4 percentage points in the retrieval metrics, but gain half a percentage point on ImageNet-1K.

Self-Supervised Teachers

We assume that the decrease in performance can be attributed to the different strategies used to train BEiTv2, Data2Vec2, and DINO. In contrast to Data2Vec2, the architecture and loss of BEiTv2 forces the model to aggregate as much (global) information as possible in the $[\texttt{I\_CLS}]$ token [Pen+22], which is the token we predict with the student model. Aggregating global information in one token leads to it being less sensitive to smaller regions of the image, which is crucial when we predict the representation of the $[\texttt{I\_CLS}]$ token using the caption of the image ($\mathcal{L}_{\text{KD}}^{\text{t2i}}$ loss). Data2Vec2, on the other hand, only includes a small term in the loss

that forces the model to aggregate global information in the [I_CLS] token. However, this term has a weight of only 1% w.r.t the total loss [Bae+23].

For DINO, aggregating global information in the [I_CLS] token is, similar to BEiTv2, forced by the architecture and the loss function. The loss function only operates on the representation of the [I_CLS] token, forcing the model to push all information to this token. This is why we observe a better performance when using DINO compared to Data2Vec2. We assume the performance with DINO is worse than with BEiTv2, because DINO only reaches 78.2% accuracy on ImageNet-1K when performing linear evaluation [Car+21], compared to the 80.1% of BEiTv2 [Pen+22]. So the quality of the representation for the [I_CLS] token seems to be higher for BEiTv2 than for DINO.

Supervised Teacher
Using a BEiTv2 finetuned on ImageNet-1K also generally leads to a drop in performance, although it is less pronounced than with Data2Vec2. However, we observe an increase in image retrieval from text on MSCOCO by around 1.2 percentage points. This comparison between self-supervised and supervised teacher, which is now unbiased due to the same model size, reflects our previous findings that using representations from self-supervised models are more beneficial that probability distributions from supervised models. Consequently, our approach not only does not rely on labeled data in the end-to-end training process, but also leads to better performance, compared to the method proposed by the authors of SHRe [AVT17].

S-SMKE without Distillation
We further observe that using no knowledge distillation at all, leads, as with Data2Vec2, to a significant drop in performance. This shows that guiding the alignment with knowledge distillation is crucial for the performance of S-SMKE. However, as we can see when comparing this performance with the Data2Vec2 teacher, the advantage of using knowledge distillation can only be exploited when the teacher's representation of the [I_CLS] aggregates as much global information as possible.

5.4 Evaluation on Unimodal Benchmarks

What is often lost sight of in multimodal models is the performance on unimodal downstream tasks. While the main goal of multimodal models is to learn a joint representation of text and images, a multimodal model should also excel at unimodal tasks. In our case: image classification and text classification. When it comes to unimodal downstream tasks, most papers on vision-language models, with the exception of FLAVA [Sin+21], exclusively focus on image classification and segmentation tasks, and do not evaluate the performance on text classification tasks (e.g. BEiT-3 [Wan+23], VLMo [Bao+22], and CoCa [Yu+22]). This is surprising, as an adequate language understanding is crucial for any multimodal model, especially when it comes to vision-lanuage reasoning like in NLVR2 [Suh+19] or VQAv2

Method	ImageNet-1K	
	Lin eval	Finetune
Data2Vec2 [Bae+23]	-	84.5
BEiTv2 [Pen+22]	**80.1**	**85.0**
ResNet-101 [He+16]	-	80.1
FLAVA† [Sin+21]	75.54	-
DistilData2Vec2	56.2	75.0
S-SMKE†	65.0	75.5
C-DistilData2Vec2	<u>71.1</u>	<u>76.1</u>

Table 27: Using the image encoder of S-SMKE for image classification tasks leads to an increase in performance over DistilData2Vec2, but a decrease in performance when making the models directly comparable using C-DistilData2Vec2. † indicates the usage of an image encoder from vision-language models, and <u>...</u> indicates the best performance among distilled models, and bold values indicate the best performance overall.

[Ant+15]. Moreover, it is quite simple to test the language understanding of a model by evaluating it on the GLUE benchmark [Wan+18], which is what we already did once in the language distillation experiments of Section 5.1.2. We therefore evaluate our best multimodal models on both image and text classification tasks.

5.4.1 Vision

For image classification, we take the image encoder of S-SMKE and finetune it using the same strategy as done for the image-only distilled model: The output $\boldsymbol{H}_{v,L_s}$ of the image encoder is pooled by taking the mean of all patch representations, with the exception of the [I_CLS] token. This pooled representation is then passed through a layer normalization and a linear classification layer. The pytorch pseudocode is the same as for the image-only distilled model, and can be found in Code 1. We do not use the shared encoder at the top of our multimodal models for the image classification task, as a shared representation is not desired for image-specific tasks. The image encoder returns a representation specific to the image modality, which is what we want to use for image classification. The hyperparameters (see Table 33), and all other settings, are the same as for the image-only distilled model DistilData2Vec2, and we refer to Section 5.1.1.3 for more details.

The results on ImageNet-1K [Rus+15] and CIFAR-10/100 [Kri12] are shown in Table 27 and Table 28, respectively. Incredibly, we observe an increase in performance, compared to DistilData2Vec2, on all finetuning tasks, and a significant increase for linear evaluation over all datasets.

This is unusual, as DistilData2Vec2 is an image-only model, and S-SMKE is a multimodal model. Single modality models usually perform better on their respective modality-specific tasks, as they only focus on one modality. Even though the image encoder of S-SMKE is used

Method	CIFAR-10		CIFAR-100	
	Lin eval	Finetune	Lin eval	Finetune
BEiTv2 [Pen+22]	**94.4**	**98.8**	**78.5**	**91.1**
FLAVA† [Sin+21]	93.44	-	78.37	-
DistilData2Vec2	68.4	97.0	46.2	85.1
S-SMKE†	89.7	97.6	71.3	85.2
C-DistilData2Vec2	93.2	97.7	77.3	87.2

Table 28: Results of performing linear evaluation and full finetuning of the image encoder from S-SMKE on CIFAR-10/100 [Kri12]. Note that the authors of BEiTv2 did not publish results on CIFAR-10 and CIFAR-100, so we perform finetuning and linear evaluation on the models ourselves. The procedure for linear evaluation and finetuning is the same as for our own models, with the exception that we use a batch size of 128 for finetuning, instead of 256. This is required to avoid out-of-memory errors.

Approach	Teacher	Encoder init	Loss
DistilData2Vec2	Data2Vec2 [Bae+23]	Data2Vec2 layer 1-6	Data2Vec loss
Image S-SMKE	BEiTv2 [Pen+22]	Data2Vec2 layer 1-6	Contrastive Target loss

Table 29: DistilData2Vec2 and S-SMKE are not directly comparable on visual downstream tasks, as they use different teachers and loss functions. “Image S-SMKE” refers to the image encoder of the trained S-SMKE model.

for finetuning, which also technically only focuses on the image modality, its representations are optimized for the alignment of text and images in the shared encoder, and should therefore not be as beneficial for image-specific tasks as those of DistilData2Vec2.

However, both results of DistilData2Vec2 and S-SMKE are not directly comparable. In order to compare the two models directly, everything about the DistilData2Vec2 and the image encoder of S-SMKE should be the same, however, looking at Table 29, we see that S-SMKE uses a different teacher and loss function for the distillation.

To make both approaches directly comparable, so that we can directly see the impact of using an image encoder of a vision-language model on an image-only task, we conduct the following experiment: We distill DistilData2Vec2 again, but now train the student model with the contrastive target loss (with memory bank), and change the teacher to BEiTv2. We call this approach C-DistilData2Vec2. The only thing that differentiates C-DistilData2Vec2 from the image encoder of S-SMKE is that the image encoder of S-SMKE is trained for the alignment of text and images, while C-DistilData2Vec2 is only trained to replicate image representations. The hyperparameters and settings are the same as for DistilData2Vec2, and we refer to Section 5.1.1.3 for more details.

The result of finetuning C-DistilData2Vec2 after the distillation are also shown in Table 27 and Table 28. Compared to DistilData2Vec2, we observe an even more pronounced increase than through the image encoder of S-SMKE. In linear evaluation, C-DistilData2Vec2 in-

creases the performance by at least 15 percentage points over all benchamrks, and we even record an increase of over 25 percentage points on CIFAR-100 linear evaluation.

This experiment shows that having a unimodal image model that focuses only on image representations is better for image-specific tasks than using an image encoder of a vision-language model. This is exactly what we expected earlier.

The most important benchmark to consider is the performance on ImageNet-1K, where S-SMKE achieves the lowest performance among all models. Again, this is not surprising, as models like BEiTv2 [Pen+22] and Data2Vec2 [Bae+23] are specific to images. More interesting is the comparison with FLAVA [Sin+21], which is a multimodal model that also uses its image encoder for downstream image classification tasks. We can see that FLAVA outperforms S-SMKE on all tasks, but since the image encoder of FLAVA is a full ViT-B/16 [Dos+21, Sin+21] model with 12 layers, and the image encoder of S-SMKE has only 6 layers, FLAVA's results are based on a model twice as large as S-SMKE's image encoder. We therefore consider the performance of S-SMKE on ImageNet-1K as acceptable, even though we do not consider it as the strength of our model, which lies more in image-text retrieval.

5.4.2 Language

Analogue to image classification, we extract the text encoder of S-SMKE and finetune it on the GLUE benchmark [Wan+18]. We follow the same strategy as for the text-only distilled model: From the output $\boldsymbol{H}_{w,L_s}$ of the text encoder we take the representation $\boldsymbol{h}_{w,L_s,[\texttt{T\_CLS}]}$ of the $[\texttt{T\_CLS}]$ token, pass it through a linear pooling layer, the weights of which come from a pretrained BERT model, through a dropout layer with $p = 0.1$, and finally through a linear classification layer. The pytorch pseudocode is the same as for the text-only distilled model F-DistilBERT, and can be found in Code 2. Again, all hyperparameters and settings are the same as for F-DistilBERT, and we refer to Section 5.1.2.4 for more details, and to Table 35 for the hyperparameters.

The results, displayed in Table 30, show that we lose more than 16 percentage points in the overall GLUE score compared to our F-DistilBERT, and even more compared to the original BERT model. While the reason for this decrease lies again in the fact that the text encoder of S-SMKE is optimized for the alignment of text and images, and not for text-specific tasks, it certainly does not explain the performance on CoLA [WSB19], which is the lowest among all models (14.2). While we continously perform worse than FLAVA [Sin+21], which again uses a text encoder with twice the number of layers of our text encoder, S-SMKE outperforms CLIP [Rad+21] in all tasks, with the exception of CoLA. A visually more appealing version of Table 30 is shown when discussing all results in Section 5.6.

	MNLI	QNLI	RTE	MRPC	QQP	STS-B	CoLA	SST	WNLI	**Score**
Unimodal										
ELMo [Pet+18]	68.6	71.1	53.4	76.7	86.2	70.4	44.1	91.5	56.3†	68.7
BERT [Dev+19]	**86.7**†	**91.8**†	**69.3**†	**88.6**†	**89.6**†	**89.0**†	**56.3**†	**92.7**†	53.5	**79.5**†
Unimodal Distilled										
DistilBERT [San+19]	82.2	89.2	59.9	87.5	88.5	86.9	51.3	91.3	56.3	77.0
F-DistilBERT (ours)	81.2	88.0	67.64	85.0	86.5	81.0	55.1	91.4	**56.4**	76.9
Multimodal										
CLIP [Rad+21]	33.5	50.5	55.2	65.0	53.9	16.0	25.4	88.2	-	(48.5)
FLAVA [Sin+21]	80.3‡	87.3‡	57.8‡	86.9‡	87.2‡	85.7‡	50.7‡	90.9‡	-	(78.4)
S-SMKE (ours)	73.88	78.71	51.6	79.9	81.2	57.5	14.2	83.5	45.04‡	61.3

Table 30: Results of finetuning the text encoder of S-SMKE on the GLUE benchmark [Wan+18]. We compare to unimodal, unimodal distilled, and multimodal models, where the latter use their text encoder for finetuning. † indicates the best performance among unimodal models, underlined values indicate the best performance among unimodal distilled models, and ‡ indicates the best performance among multimodal models. Bold values indicate the best performance overall. Note that the score for FLAVA and CLIP are not directly comparible with others, as both works do not publish results on WNLI [LDM12]. CLIP generally does not publish results on GLUE directly, so we take the results reported by th authors of FLAVA [Sin+21].

5.4.3 Retrieval

After pretraining, papers like VLMo [Bao+22] perform finetuning on MSCOCO [Lin+14] and Flickr30K [You+14] as an additional step. Here, only the contrastive loss is used:

$$\mathcal{L}_{\text{S-SMKE}} = \mathcal{L}_{\text{CL}} \tag{86}$$

This essentially means that the model is finetuned once *exclusively* on the MSCOCO train dataset, and then evaluated on image-text retrieval with the MSCOCO test dataset, and the same for Flickr30K. This will strengthen the alignment of text and images on the respective datasets, and is a common practice in vision-language models [Bao+22, Wan+23].

We follow this strategy and finetune S-SMKE on MSCOCO and Flickr30K. We finetune the whole model, and train for only 5 epochs, as we found it to be sufficient for the model to converge. Since the quality of the results is highly dependent on the batch size, i.e. the number of negative samples, we increase the batch size to 1024. During pretraining, we used a batch size of 256 per device, which resulted in a contrastive loss with 511 negative samples. For finetuning, we only use one GPU instead of two, but switch from the RTX 4090 24GB to the A100 80GB. Even though this GPU is more expensive, it allows us to increase the batch size to 1024, resulting in 1023 negative samples. Since COCO and Flickr30K are smaller datasets, and we only finetune for 5 epochs, the increased cost per GPU hour is acceptable. Naturally, we also use a smaller peak learning rate of 3e-5 and warmup for 10% of the total steps. All

Model	MSCOCO (5K test set)						Flickr30K (1K test set)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
FLAVA [Sin+21]	42.74	76.76	-	38.38	67.47	-	67.7	94.0	-	65.22	89.38	-
CLIP [Rad+21]	**58.4**	81.5	88.1	37.8	62.4	72.2	**88.0**	**98.7**	**99.4**	**68.7**	**90.6**	**95.2**
S-SMKE$_{\text{CTL_MB}}$	53.54	81.1	89.52	35.65	66.0	77.77	70.9	92.1	96.0	52.72	80.2	87.46
S-SMKE$_{\text{CTL_MB}}$ † (4.8 ↑)	56.2	**83.3**	**91.1**	**39.8**	**69.2**	**79.8**	82.0	95.4	98.0	64.6	87.5	93.1

Table 31: Image-text retrieval results of finetuning S-SMKE on MSCOCO and Flickr30K. We compare to FLAVA [Sin+21] and CLIP [Rad+21]. † indicates the finetuned variant of our model.

hyperparameters are shown in Table 37. The results of finetuning S-SMKE on MSCOCO and Flickr30K are shown in Table 31.

Finetuning S-SMKE on MSCOCO and Flickr30K increases the performance on all metrics compared to the pretrained model, we gain on average 4.8 percentage points over all metrics. Especially the performance on Flickr30K is significantly increased, with image retrieval increasing by almost 12 percentage points on the R@1 metric. On MSCOCO, the performance is also increased, but not as much as on Flickr30K. This can be explained by the fact that the COCO train dataset is part of the data that we use for pretraining, so the model has already seen the data of COCO. Consequently, there is less room for improvement since the data is not completely new to the model. This is different for Flickr30K, where the model has not seen the data during pretraining, so there is much more to gain from finetuning on this dataset. Overall, we outperform CLIP [Rad+21] and FLAVA [Sin+21] on all metrics in COCO, except for text retrieval on the R@1 metric (CLIP). On Flickr30K, CLIP still outperforms us on all metrics.

It has to be noted that the retrieval results of both CLIP and FLAVA are not the result of finetuning on the respective datasets, but the direct application of the pretrained model on the test set [Rad+21, Sin+21]. We did the same when we reported results on retrieval before, which is shown by S-SMKE$_{\text{CTL_MB}}$ without †. If one would finetune CLIP or FLAVA on the respective datasets, then both models would likely outperform S-SMKE due to their size. The goal of finetuning is **not** to claim that S-SMKE is better than CLIP or FLAVA, but to show that finetuning on retrieval tasks after pretraining is **beneficial** for the alignment of text and images in our model.

5.5 Limitations and Insights

While the proposed method is an efficient way to train comparatively small multimodal models, and can easily be adapted to other modalities, e.g. audio, it has its limitations.

5.5.1 Performance on Unimodal Downstream Tasks

When finetuning S-SMKE on unimodal downstream tasks in Section 5.4, we found that the performance is generally worse compared to the performance of the unimodal distilled models. While this is nothing unusual, it is still a limitation considering that the goal of multimodal models is to excel at both unimodal and multimodal tasks.

Fortunately, this can be solved by not only training the multimodal model on aligning the modalities, but simultaneously training the modality-specific encoders, in our case the image and text encoder, on modality-specific pretraining tasks. A good example of this is the approach of BEiT-3 [Wan+23], which trains the whole model on alignment of image and text, but also trains the image encoder to reconstruct masked images, and the text encoder to predict masked tokens. This way, both encoders are encouraged to learn representations that are still specific to their modality, while providing representations that can also be used for multimodal tasks and therefore alignment. The result is that BEiT-3 reaches a finetuning accuracy of 85.4%[16] on ImageNet-1K using its image encoder. This is better than the performance of BEiTv2, which is an image-only model and reaches a finetuning accuracy of 85.0% [Pen+22]. Since both the image encoder of BEiT-3 and BEiTv2 are based on the same ViT-B/16 [Dos+21] architecture, the results are directly comparable and show that modality-specific tasks can strengthen the performance of multimodal models on unimodal tasks.

5.5.2 Modality-Specific Bias

Our method relies on knowledge distillation of a self-supervised **unimodal** image model as the teacher. The fact that there has been no incentive for the teacher to learn a representation that is independent of the image modality makes it difficult for the teacher to provide guidance to the student model on how to align the modalities because the teacher representations are not aligned and modality-agnostic themselves. This has repeatedly been shown when comparing the loss between image-to-image and text-to-image distillation, where the former is consistently lower. Interestingly, we were still able to outperform the approach of a supervised teacher, showing that even though the ImageNet-1K classes, which we predict using KL-Divergence (see Section 5.2.1), are real-world concepts independent of the image modality, they might not capture the content of an image's caption better than a image-specific representation (used with the self-supervised teacher), which is what we first assumed.

A glance at the comparison between the components of the knowledge distillation loss (KL-Divergence) when using a supervised teacher also shows that this approach suffers from the same problem as when using a self-supervised teacher (see Figure 31). Here, the KL-Divergence for the image-to-image loss $\mathcal{L}_{\mathrm{KD}}^{v}$ is also consistently lower than for the text-to-image

[16] This score was not published in the original paper, but can be found in the official BEiT-3 repository: https://github.com/microsoft/unilm/tree/master/beit3.

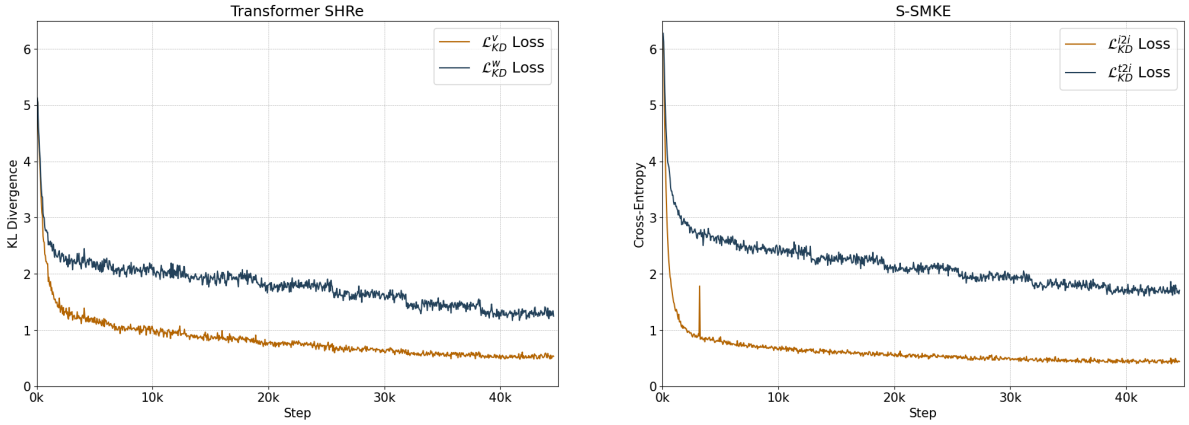

Figure 31: Both approaches, the supervised as well as the self-supervised (ours), show that the image component is consistently lower than the text component of the knowledge distillation loss. Moreover, the loss behavior troughout the training process is very similar between both approaches, although the loss functions are different.

loss $\mathcal{L}_{\text{KD}}^{w}$, and the loss components between both approaches (Transformer SHRe and S-SMKE) generally perform very similar.

Consequently, a bias towards the teacher modality is not specific to S-SMKE, but generally a problem when using an unimodal teacher for distilling knowledge to a multimodal model.

5.5.3 Fine-Grained Alignment

S-SMKE (and SHRe) processes image and text seperately through a forward pass for the image and a forward pass for the text. This is similar to CLIP [Rad+21] (see Section 2.7.2). Because there is no attention mechanism between individual image patches and text tokens, both approaches miss a fine-grained alignment between the modalities. Even though our model performs quite well on the retrieval task, even outperforming well-established research papers on some metrics, there is still room for improvement. However, we believe that there is not much more performance to gain for our approach. Wrong retrievals, mostly wrong image retrievals, are often still semantically similar to the query, and only differ in little (token-level) details. Since our representations are based on the global content of the image and text, those details are not captured in most cases, leading to a "false" retrieval. Examples of this can be seen with example retrievals on the full MSCOCO test set in Figure 43, which is the exact dataset we publish our results on (previous visualizations are based on 1k subsets as in SHRe [AVT17], and are therefore a simpler task since there are less possible retrieval candidates).

At this point it has to be noted that when it comes to a production-grade retrieval system, e.g. a system for searching images by text, the retrievals of our model can still be considered as correct, or "good enough", as most of them, even though they are flagged as "false" under the evaluation metrics, are still semantically related to the query.

Still, the problem remains, and is actually relevant when it comes to applications where the fine-grained details are crucial. This includes tasks like visual question answering and visual reasoning, benchmarked by the datasets VQAv2 [Ant+15] and NLVR2 [Suh+19], respectively.

For instance, in NLVR2 the task is to decide whether a given sentence about an image is either true or false. The sentence, so the statement about the image, often focuses on fine-grained details for which there is no guarantee that they are captured by just global representations. Further, even if statements are false, then the differences between the image and the statement are often so subtle that the statement, on a high level, can still be considered as a caption for the image. If e.g. the cosine similarity between the global representations of image and statement would be used as the binary classifier, then the model would likely fail on such tasks. Examples of NLVR2 can be seen in Figure 44.

That simply aligning global representations is not enough to excel in such tasks was also shown by the authors of ViLT [KSK21], who were one of the first to propose a model that aligns image and text on a fine-grained level. They found that when finetuning CLIP [Rad+21] on the NLVR2 dataset the model achieved an accuracy of only 51% [KSK21]. Considering that the task is a binary classification, and both classes are equally distributed, this is only slightly better than random guessing and therefore unusable for practical applications. Since our model works on a similar level of alignment, the same will likely apply with our approach.

5.6 Discussion of Results

In this section, we briefly discuss the performance on all the tasks that we use to evaluate the proposed method. We compare the results on natural language processing, computer vision, and vision-language tasks in Figure 32.

In most cases, the proposed method does not outperform the methods to which we compare. However, recall that our approach was designed as a **proof-of-concept** to demonstrate the feasibility of creating cost-effective and efficient vision-language models. Since we achieve reasonable performance across all tasks, and even outperforming well-known baselines on tasks like WNLI and COCO image retrieval, we consider our approach to be successful.

Furthermore, it is not realistic to expect the proposed method to outperform the state-of-the-art methods, as they are larger in every aspect: parameters, data, and compute. A good impression on where S-SMKE ranks among the vision-language models we repeatedly compare to can be obtained from Figure 33.

Our most noteable achievements are:

- **WNLI Finetuning:** We outperform BERT [Dev+19] and DistilBERT [San+19] (by just 0.1 percentage point) on the WNLI task, which is likely due to the fact that it consists of just

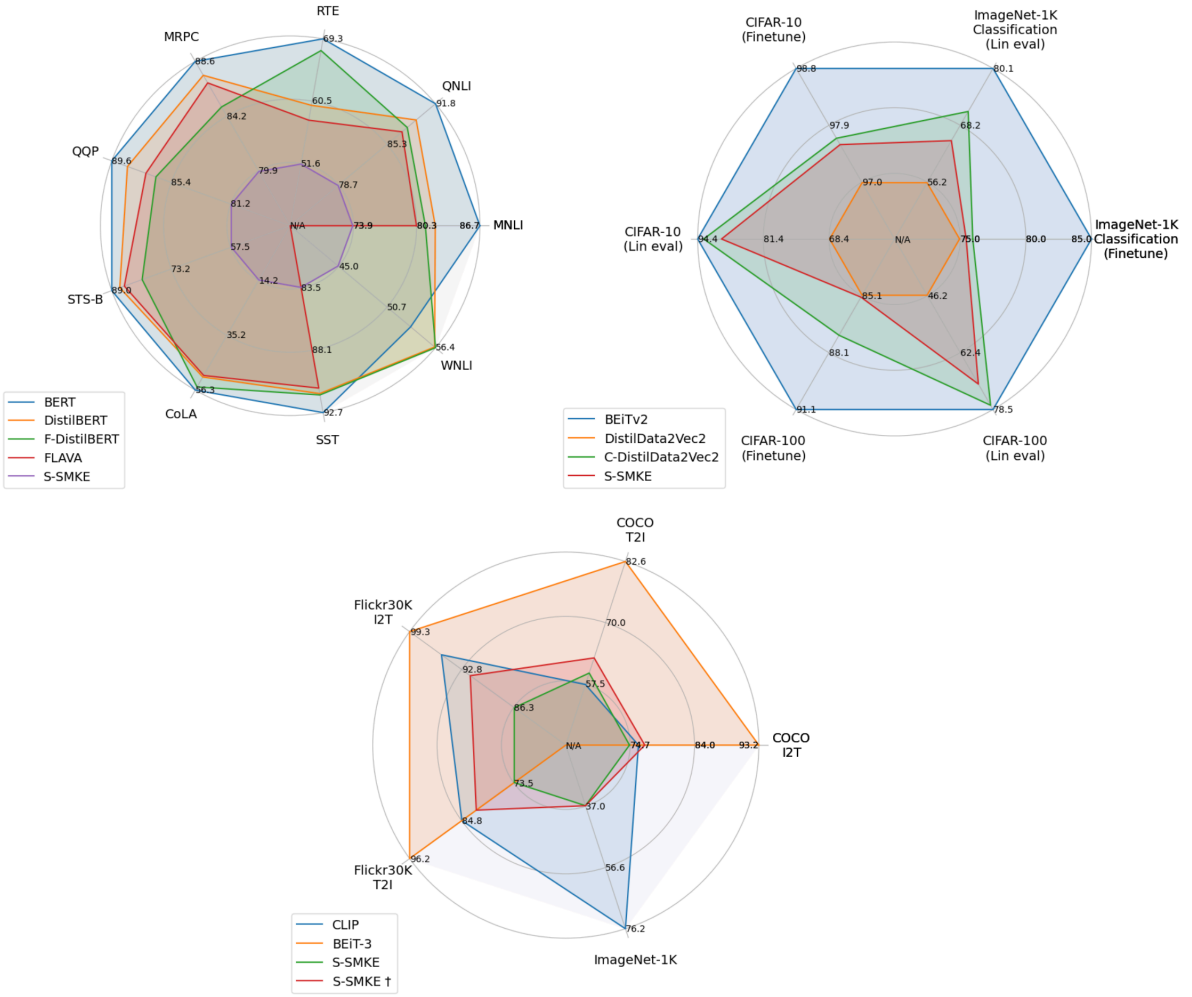

Figure 32: Overview of all benchmark results across natural language processing (top left), computer vision (top right), and vision-language tasks (bottom).

635 training and 71 validation examples [LDM12]. Our text-only model is around half the size of BERT, making it less prone to overfitting on such a small dataset.

- **CIFAR-10 and CIFAR-100 Linear Evaluation:** Our specialized variant of the image-only distillation, C-DistilData2Vec2, is almost on par with BEiTv2 [Pen+22] on CIFAR-10 and CIFAR-100 [Kri12] linear evaluation. This is remarkable, as BEiTv2 is also twice as large as our model. We consider the results on both benchmarks (CIFAR-10 and CIFAR-100) as reliable, as we performed the linear evaluation of BEiTv2 ourselves, using the exact same setup as for C-DistilData2Vec2.
- **COCO Image Retrieval:** S-SMKE outperforms CLIP [Rad+21] on COCO image retrieval, and the finetuned variant of S-SMKE outperforms CLIP on all COCO retrieval metrics. It has to be noted that CLIP was neither pretrained nor finetuned on the COCO dataset, so S-SMKE has an advantage here. However, considering that CLIP is a much larger model trained with over 121× the data, the result is still remarkable.

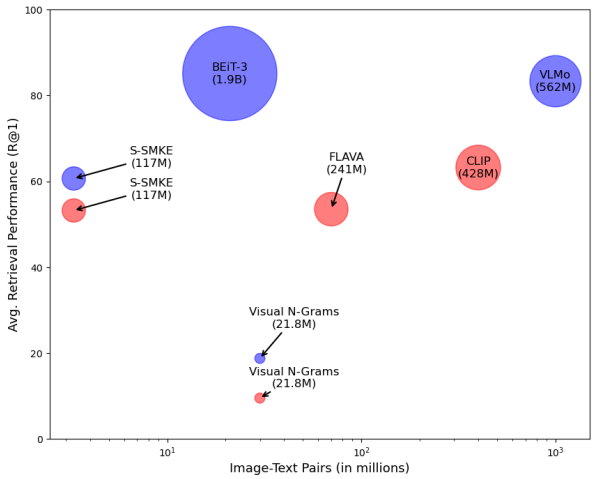

Figure 33: Overview of vision-language model landscape. Bubble sizes represent the number of parameters, also shown in parentheses next to the model name. The number of image-text pairs used is in log scale, and the retrieval performance is the average of the R@1 scores on COCO [Lin+14] and Flickr30k [You+14]. We denote red bubbles as the direct application of a pretrained model on image-text retrieval, and blue bubbles as models that were finetuned on the retrieval task after pretraining. Visual N-Grams [Li+17] was developed in 2016 as a first proof-of-concept for vision-language models.

On unimodal tasks, S-SMKE does not perform as well as our unimodal distilled models and other baselines. As mentioned in the limitations of our method (Section 5.5.1), the representations of our image and text encoder are optimized for alignment, not modality-specific tasks. A loss in performance is therefore to be expected but not necessarily unavoidable, since FLAVA [Sin+21] shows that including text-specific objectives during pretraining can greatly improve performance when finetuning on text tasks. The same observation holds BEiT-3 [Wan+23] on vision tasks (see Section 5.5.1).

Considering the above, it is still striking that S-SMKE only reaches 14.2% accuracy on CoLA [WSB19], while our text-only model F-DistilBERT reaches 55.1%, which is just 1.2 percentage points below BERT's [Dev+19] 56.3%. Unfortunately, we do not have an explanation for this discrepancy. One possible factor is that we pretrain S-SMKE on subsets of Conceptual Captions 3M and 12M [Cha+21, Sha+18]. Since these datasets are largely uncurated and scraped from the web, the captions passed through the text encoder might not be grammatically correct. As CoLA is a task to assess the grammatical understanding of a model [WSB19], finetuning on a dataset with just 8551 samples might not be sufficient to correct the model's understanding of grammar. An example of poor captioning in Conceptual Captions 12M can be seen in Figure 18.

6 Conclusion

6.1 Summary of Contributions and Research

In this thesis, we presented an efficient end-to-end self-supervised approach to vision-language learning that is significantly cheaper to train and smaller in size compared to existing multimodal models. While previous approaches have relied on large-scale models, extensive datasets, and significant compute, our method offers an alternative that aligns with the needs of smaller research groups, thus filling a critical gap in the current literature.

Our approach is characterized by the use of pretrained unimodal encoders, which generate representations of their input that are aligned by a randomly initialized shared encoder. We employ a contrastive loss function to enforce alignment between image and text representations and utilize a self-supervised image model for simultaneous knowledge distillation. This helps guide the alignment through high-level image representations predicted using both an image and its caption.

Building upon the method of SHRe [AVT17], we adapted it to the Transformer architecture and employed a self-supervised teacher instead of a supervised one. Our experiments demonstrate that this adaptation leads to increased performance across all benchmarks. Furthermore, we showed that our approach achieves competitive performance with vision-language models such as CLIP [Rad+21] and FLAVA [Sin+21] on retrieval tasks, and even outperforms them on certain scoring metrics, all while using only 0.75% of the data used by CLIP and 4.3% of the data used by FLAVA.

The overall framework of our method is presented in Figure 45.

6.2 Future Work

Positional Encoding
We initialized our multimodal model with components from pretrained unimodal models, which were Data2Vec2 [Bae+23] for the image encoder and BERT [Dev+19] for the text encoder. The representations of these models were they passed seperately through a shared Transformer encoder. However, we only briefly considered what implications this has for the shared Transformer encoder: We introduced a learnable token-type embedding, which

is used to distinguish between both modalities (see Section 5.2.3), which is especially important for the self-attention mechanism of the Transformer as the sequence of representations has an inherently different meaning for each modality: 1D for text and 2D for images.

A part of these sequences are the positional encodings of the image and text encoder, respectively. These positional encodings are used to provide the Transformer with information about the position of each token in the sequence. However, the positional encodings of the image and text encoder are of a different type. While the positional encodings of the text encoder (BERT) are learnable representations for each position, so a learnable absolute positional encoding, the positional encoding of the image encoder (Data2Vec2) is a fixed sinusoidal positional encoding. Therefore, the shared Transformer encoder not only has to account for the difference between the modalities, but also for the difference in positional encodings. In contrast, other vision-langaue models like BEiT-3 [Wan+23] or VLMo [Bao+22] use the same positional encoding for both modalities, so it would also be worth investigating the impact of using the same positional encoding for both modalities.

A caveat of using the same positional encoding type for both modalities is that it greatly restricts the flexibility of the choice of the pretrained unimodal models we use to initialize the modality-specific encoders. One would have to find pretrained unimodal models that all use the same positional encoding type, which becomes increasingly difficult with more modalities.

Strengthening Unimodal Encoders
Finetuning S-SMKE on unimodal downstream tasks showed that our performance is generally worse on those taks compared to the performance of unimodal models, including our unimodal distilled models. Since this is a limitation of our approach, it is worth investigating how the performance of our model can be improved on unimodal tasks. BEiT-3 [Wan+23] solves this problem by including modality-specific pretraining tasks during the training of the multimodal model, which we find as worth investigating for our approach as well. It would be relatively easy to include unimodal tasks such as masked language modeling for the text encoder and masked patch prediction for the image encoder.

Fine-Grained Alignment
As already mentioned in the limitations of S-SMKE (see Section 5.5), the model processes image and text seperately. Even though this makes alignment on the level of global representations possible, it does not allow for fine-grained alignment of image and text on the level of individual image patches and text tokens. This is a limitation that is shared with approaches like CLIP [Rad+21]. This **limits** the actual application of S-SMKE to image-text retrieval, as other vision language tasks like image captioning, visial question answering, or visual reasoning require individual image patches to attend to individual text tokens, and vice versa. This is only possible through cross-modal attention, which is not part of our approach. To ensure a wider applicability of our approach, it would be necessary to include such a cross-modal attention mechanism by e.g. concatenating the image and text representations and

passing them through the shared Transformer layer(s). For details on cross-modal attention through concatenation we refer to BEiT-3 [Wan+23].

6.3 Outlook

Towards a General Framework
While this work introduces efficient multimodal learning on the example of vision-language models, the question arises whether this approach can be extended to other modalities, such as audio or video. While the methodology presented is not restricted to vision and language, as it can be adapted by using a teacher and pretrained models (for initialization) from other modalities, the success of this approach still needs to be practically demonstrated. Even though SHRe [AVT17] has shown that the approach allows the alignment of vision, language, and audio, the question remains whether this can also be achieved through an end-to-end self-supervised learning approach.

To really ensure the general applicability of this approach, it would be necessary to demonstrate the presented learning paradigm with a variety of modality combinations.

Application on Many-to-Many Alignment
As of September 2024, there exists a variety of vision-language models, each of them presenting different approaches to aligning vision and language. However, with the exception of SHRe [AVT17] and AudioCLIP [Guz+22], to our knowledge, there are no multimodal models that align more than two modalities.

Based on our intuition that the same concept, expressed in different modalities, should be aligned, this would be a promising direction for future research and the next logical step in multimodal learning. While there has been extensive research on vision-language models since 2020, with over 20 papers published, there is still a lack of research on models that align more than two modalities.

One possible reason could be that with each additional modality, the complexity of the model increases, as the model needs to learn to align more modalities, and each modality requires additional parameters that are responsible for extracting low-level features. Furthermore, while vision-language models only require image-text pairs, and potentially unimodal image and text data (see e.g. VLMo [Bao+22] and BEiT-3 [Wan+23]), for training, vision-language-audio models would either require image-text-audio triplets, or at least two datasets that align two of the three modalities and form a transitive alignment between the three modalities. One example would be the strategy of SHRe [AVT17], which aligns image, text, and audio by aligning image-text and image-audio pairs. The authors show that this approach naturally leads to an alignment of audio and text, as they are both aligned with the image modality [AVT17]. However, even though the approach is successful, the alignment of audio and text does not work as well as the alignment of image and text (see Table 2 in the introduction of SHRe of Section 2.7.1).

Consequently to actually ensure a many-to-many alignment, it would be necessary to either collect a single dataset where all modalities form a pair (e.g. image-text-audio triplets), or to collect multiple datasets with each aligning two of the modalities. The latter approach has the disadvantage that for the alignment of n modalities 2^n multimodal datasets, and potentially n unimodal datasets, would be required. For the former approach, with an increased number of modalities, finding or constructing such, especially large, datasets becomes increasingly difficult.

This is where the presented approach could be beneficial, as there are usually pretrained models available for each modality, and the approach generally requires less data. Furthermore, as previously mentioned, the approach, as least with a supervised teacher, has shown to be able to align multiple modalities with a transitive method (even though this leads to suboptimal performance).

We therefore deem it as critical to explore the alignment of more than two modalities with an end-to-end self-supervised learning approach so that our philosophy of general modality-invariant representations can be fully realized.

6.4 Broader Impact

This work focuses on reducing the computational cost of training vision-language models by leveraging pretrained unimodal models. While our approach does not achieve state-of-the-art performance, which is not surprising given that we compare against models developed by large organizations such as OpenAI, Meta, and Microsoft, it demonstrates the potential of utilizing existing components to generate new model paradigms like multimodal models. We hope that this proof-of-concept will inspire other researchers to explore efficient methods for training multimodal models and models in general, thereby making the technology more accessible.

The current trend in deep learning emphasizes scaling models, data, and computational resources, as illustrated by OpenAI's neural scaling laws (see Figure 3). However, we believe that democratizing AI requires the development of efficient methods that enable smaller research groups and individual researchers to contribute to the field without the need for extensive computational resources. By lowering the barriers to entry, which our approach is characterized by, we can generate a more diverse set of approaches, which may lead to new breakthroughs in the field.

In recent years, advances in AI have been driven largely by major organizations and have become increasingly closed-source. A popular example is OpenAI's GPT series, where, starting with GPT-3, less detail has been released about the methodologies used to create these models. This shift towards reduced transparency contrasts with the original ideals of open collaboration in AI research and makes it more challenging for researchers to build upon and understand the work of others. Our work, for instance, relies on codebases published by the

authors of models such as FLAVA, Data2Vec, BERT, and BEiT-3. We believe that transparency is essential for advancing AI, which is why we, like others, publish our entire codebase. By doing so, we aim to help others advance their research and make AI more accessible.

Furthermore, our work towards a general and efficient appraoch to multimodal learning, which has to be investigated more in future work, aims to advocate for the adoption of a philosophy centered on **general modality-invariant representations**, that is unless explicitly undesired for certain tasks, such as unimodal applications or cases where modality-specific details are crucial (e.g., image super-resolution). This philosophy conveys that representations learned by a model should be independent of the modality on which they were trained. We believe that the ultimate goal should be to generate representations that are both detailed, capturing all relevant information about the input, and general, allowing for versatile use across different modalities. Such modality-agnostic/invariant representations would not only resemble human perception, where our understanding of real-world concepts is not bound to a single modality (the concept of a "dog" is not dependent on images), but would also be beneficial in a variety of applications.

One potential application is universal modality translation, where a model generates a representation of an input that can be used to translate the concept into another modality. Currently, popular examples include image-to-text (image captioning) and text-to-image (image generation from text) models. However, embracing the philosophy of general modality-invariant representations could extend this capability to any combination of modalities, enabling the development of more general and versatile AI systems.

A Appendix

AA Hyperparameters

Type	Hyperparameters	Values
Model	Layers	6
	Hidden size	768
	FFN inner hidden size	3072
	Attention Heads	12
	Patch size	16×16
	Input resolution	224×224
Training	Epochs	10
	Total steps	50040
	Batch size	256
	Optimizer	AdamW
	AdamW ε	1e-06
	AdamW β	(0.9,0.98)
	Weight decay	0.01
	Base learning rate	1e-4
	Learning rate schedule	Cosine
	Warmup steps	5004 (10% of total steps)
	Hardware	1 × RTX 4090 24GB
Augmentations	Horizontal flipping prob.	0.5
	RandomResizeCrop range	[0.08, 1.0]

Table 32: Hyperparameters used for distilling a Data2Vec2 image model.

Type	Hyperparameters	ImageNet		CIFAR10		CIFAR100	
		Finetune	Linear probe	Finetune	Linear probe	Finetune	Linear probe
Training	Num classes	1k	1k	10	10	100	100
	Epochs	15					
	Batch size	256					
	Optimizer	AdamW					
	AdamW ε	1e-8					
	AdamW β	(0.9, 0.999)					
	Weight decay	0.01					
	Base learning rate	1e-3					
	Layer Decay	0.81	-	0.75	-	0.75	-
	Learning rate schedule	Cosine					
	Warmup steps	10% of total steps					
	Hardware	1 × RTX 4090 24GB					
Mixup [Zha+18]/ **Cutmix** [Yun+19]	Mixup prob.	0.8					
	Cutmix prob.	1.0					
	Prob.	0.9					
	Switch prob.	0.5					
	Label smooting	0.1					
RandAugment [Cub+20]	Magintude	9					
	Magnitude std.	0.5					
	Magnitude inc.	1					
	# ops	2					
RandomErase [Zho+20]	Prob.	0.25					
	Mode	pixel					
	# erase	1					

Table 33: Hyperparameters used for the ImageNet-1K [Rus+15], CIFAR10 [Kri12], and CIFAR100 [Kri12] of the distilled Data2Vec2 image model. We refer to the respective papers for details on the augmentation techniques [Cub+20, Yun+19, Zha+18, Zho+20].

Type	Hyperparameters	Values
Model	Layers	6
	Hidden size	768
	FFN inner hidden size	3072
	Attention Heads	12
	Max sequence length	256
	Vocabulary size	30522
Training	Epochs	1
	Total steps	1M
	Batch size	256
	Optimizer	AdamW
	AdamW ε	1e-06
	AdamW β	(0.9,0.98)
	Weight decay	0.01
	Base learning rate	5e-4
	Learning rate schedule	Cosine
	Warmup steps	10k (1% of total steps)
	Hardware	1 × RTX 4090 24GB

Table 34: Hyperparameters used for distilling a BERT model.

Type	Hyperparameters	MNLI	QNLI	RTE	MRPC	QQP	STS-B	CoLA	SST	WNLI
Training	Num classes	3	2	2	3	2	-	2	2	2
	Head Dropout prob.	0.1								
	Epochs	10	10	20	20	10	20	20	10	20
	Total steps	61360	32733	3113	5095	31563	7187	10689	21047	1588
	Batch size	64	32	16	16	128	16	16	32	8
	Optimizer	AdamW								
	AdamW ε	1e-6								
	AdamW β	(0.9, 0.98)								
	Weight decay	0.1								
	Base learning rate	2e-5	2e-5	2e-5	2e-5	2e-5	4e-5	1e-5	2e-5	1e-5
	Learning rate schedule	Polynomial decay								
	Warmup steps	10% of total steps								
	Metric	Accuracy	Accuracy	Accuracy	F1	F1	Spearman	Accuracy	Accuracy	Accuracy
	Hardware	1 × RTX 4090 24GB								

Table 35: Hyperparameters for the GLUE [Wan+18] benchmark tasks of the distilled BERT model. STS-B [Cer+17] is a regression task, and therefore does not have any classes. Here, the head returns a scalar.

Type	Hyperparameters	Values
Model	Image/Text layers	6 (Transformer)
	Shared layers	3 (MLP)
	Hidden size	768
	FFN inner hidden size	3072
	Attention Heads	12
	Patch size	16×16
	Input resolution	224×224
	Max. caption length	64
Training	Epochs	7
	Total steps	89273
	Batch size	256
	Optimizer	AdamW
	AdamW ε	1e-06
	AdamW β	(0.9,0.98)
	Weight decay	0.01
	Base learning rate	1e-4
	Learning rate schedule	Cosine
	Warmup steps	8927 (10% of total steps)
	Hardware	1 × RTX 4090 24GB
Augmentations	Horizontal flipping prob.	0.5
	RandomResizeCrop range	[0.9, 1.0]

Table 36: Hyperparameters used for training the Transformer SHRe model.

Type	Hyperparameters	Values
Training	Epochs	5
	Total steps	2770/710
	Batch size	1024
	Optimizer	AdamW
	AdamW ε	1e-06
	AdamW β	(0.9,0.98)
	Weight decay	0.01
	Base learning rate	3e-5
	Learning rate schedule	Cosine
	Warmup steps	277/71 (10% of total steps)
	Hardware	1 $\times$ A100 80GB
Augmentations	Horizontal flipping prob.	0.5
	RandomResizeCrop range	[0.5, 1.0]

Table 37: Hyperparameters used for finetuning S-SMKE on MSCOCO [Lin+14] and Flickr30K [You+14] image-text retrieval. The number of total steps and warmup steps are different for the two datasets, as they are different in size. The steps are displayed as [MSCOCO steps/ Flickr30K steps].

AB Pseudocode

```
# model: pretrained (e.g. distilled) model
# layer_norm: layer normalization layer
# cls_head: linear classifier -> nn.Linear(D, C)
# x: batch of images (B, 3, H, W)
# target: batch of labels (B, )
# linear_probe: boolean flag for linear evaluation
def image_downstream_forward(model, layer_norm, cls_head, x,
  target, linear_probe):

    if linear_probe:
      with torch.no_grad():
        x = model(x) # (B, T, D)
    else:
      x = model(x) # (B, T, D)

    x = x[:, 1:] # remove cls token (B, T-1, D)
    x = x.mean(dim=1) # mean over all patches (B, D)
    x = layer_norm(x)
    x = cls_head(x) # (B, C)

    loss = cross_entropy(x, target)
    pred = x.argmax(dim=-1) # (B, )
    return loss, pred
```

Code 1: Pytorch pseudocode for the forward pass during finetuning or linear evaluation of a pretrained model on an image classification tasks. “pred” contains the predicted class index for each image in the batch.

```
# model: pretrained (e.g. distilled) model
# bert_pooler: pretrained BERT pooler layer with tanh activation
# dropout: dropout layer (p=0.1)
# cls_head: roberta classification head
# x: batch of text tokens (B, M+2) -> M: max sequence length, 2: cls and sep
token
# target: batch of labels (B, )
# regression: boolean flag for regression tasks
# metric: either Accuracy, F1, or Spearman correlation
def glue_forward(model, bert_pooler, dropout, cls_head,
                 x, target, regression, metric):

    x = model(x) # (B, M+2, D)

    x = x[:, 0] # take cls token (B, D)
    x = dropout(bert_pooler(x)) # (B, D)
    x = cls_head(x) # (B, C) -> C: number of classes

    if regression:
      x = x.squeeze() # (B, ) -> remove the last dimension, as C=1
      pred = x
      loss = mse_loss(x, target)
    else:
      pred = x.argmax(dim=-1) # (B, )
      loss = cross_entropy(x, target)

    score = metric(pred, target)

    return loss, pred, score
```

Code 2: Pytorch pseudocode for the forward pass during finetuning on GLUE benchmark tasks.

```
# teacher_model: ResNet-50-A1 model
# image_encoder: Image encoder of the multimodal student model
# text_encoder: Text encoder of the multimodal student model
# shared_encoder: Shared encoder of the multimodal student model
# imgs: batch of images (B, 3, H, W)
# captions: batch of image captions (B, 64)
# kl_div: KL-Divergence
# clip_loss: Contrastive loss used in CLIP
def forward(teacher_model, image_encoder, text_encoder,
            shared_encoder, imgs, captions):

    with torch.no_grad():
      target = teacher_model(imgs) # (B, 1000)

    img_layer_res = shared_encoder(image_encoder(imgs)[:, 0])
    # [(B, 3072), (B, 768), (B, 1000)]

    text_layer_res = shared_encoder(text_encoder(captions)[:, 0])
    # [(B, 3072), (B, 768), (B, 1000)]

    kl_loss = 1/2*kl_div(target, img_layer_res[2]) +
              1/2*kl_div(target, text_layer_res[2])

    itc_loss = 1/3*clip_loss(img_layer_res[0], text_layer_res[0]) +
               1/3*clip_loss(img_layer_res[1], text_layer_res[1]) +
               1/3*clip_loss(img_layer_res[2], text_layer_res[2])

    loss = kl_loss + itc_loss

    return loss
```

Code 3: Abstract code used in the forward pass for distilling the multimodal Transformer SHRe from a pretrained ResNet-50-A1 model. The shared encoder returns a list of representations. Each element corresponds to the activations of one linear layer in the shared encoder.

AC Figures and Visualizations

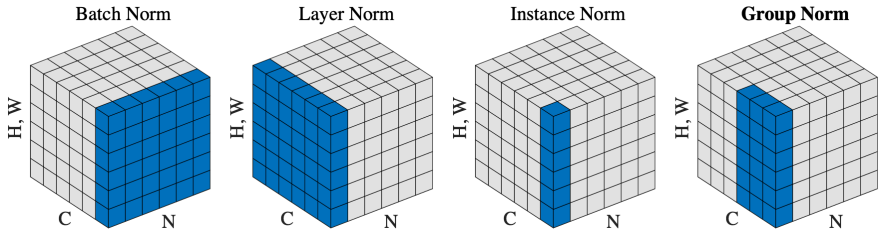

Figure 34: Comparison of different normalization operations on the example of images. Dimension "H, W" refers to the height and width of the input, "C" to the number of channels or embedding dimensions, and "N" to the number of samples, i.e. the batch dimension. Since we are working with sequences of embeddings, the height and width dimension correspond to the time steps ("H, W" -> "T"). The normalization operations work correspondingly on text sequences, where we also have time steps, so dimension "H, W" can also be replaced by "T" [WH18]. Please note that group norm, even though it is displayed in bold , is not used in this work. The figure merely was introduced in the paper of Group Norm [WH18].

Dataset	Example	Label
CoLA	[CLS] Our friends won't buy this analysis, let alone the next one we propose. [SEP]	1
SST-2	[CLS] hide new secretions from the parental units [SEP]	0
MRPC	[CLS] Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence. [SEP] Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence. [SEP]	1
STS-B	[CLS] A plane is taking off. [SEP] An air plane is taking off. [SEP]	5.0
QQP	[CLS] How is the life of a math student? Could you describe your own experiences? [SEP] Which level of preparation is enough for the exam "jlpt5"? [SEP]	0
MNLI	[CLS] Conceptually cream skimming has two basic dimensions - product and geography. [SEP] Product and geography are what make cream skimming work. [SEP]	1
QNLI	[CLS] When did the third Digimon series begin? [SEP] Unlike the two seasons before it and most of the seasons that followed, Digimon Tamers takes a darker and more realistic approach to its story featuring Digimon who do not reincarnate after their deaths and more complex character development in the original Japanese. [SEP]	1
RTE	[CLS] No Weapons of Mass Destruction Found in Iraq Yet. [SEP] Weapons of Mass Destruction Found in Iraq. [SEP]	1
WNLI	[CLS] I stuck a pin through a carrot. When I pulled the pin out, it had a hole. [SEP] The carrot had a hole. [SEP]	1

Table 38: Training examples of the GLUE benchmark tasks (one example per task). Examples are taking from the GLUE dataset card on Hugging Face[17].

[17] https://huggingface.co/datasets/nyu-mll/glue

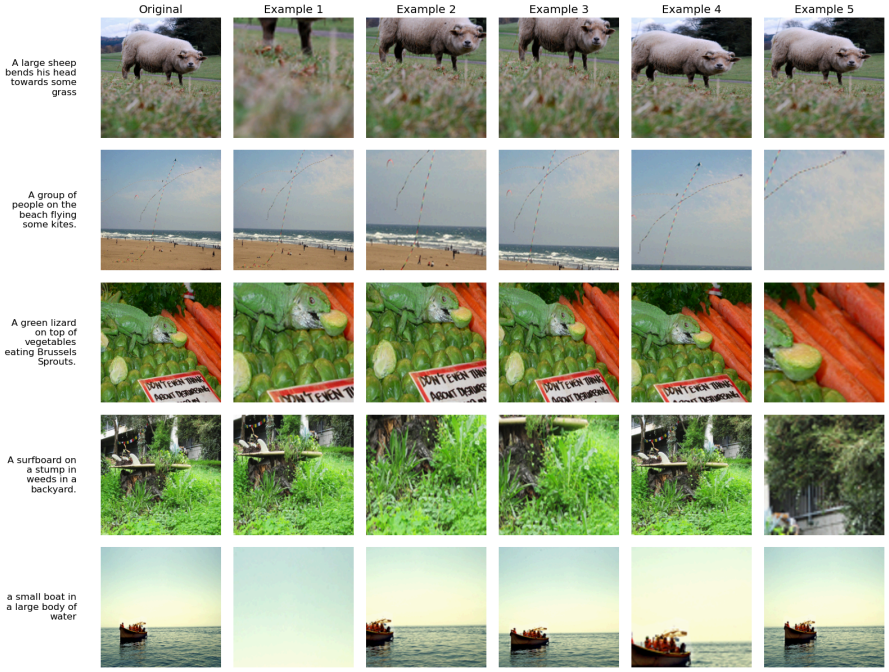

Figure 35: A small lower bound (8%) for a random crop erases high-level semantic features which are important for aligning text and image. Image-text pairs have been taken from the COCO train set [Lin+14].

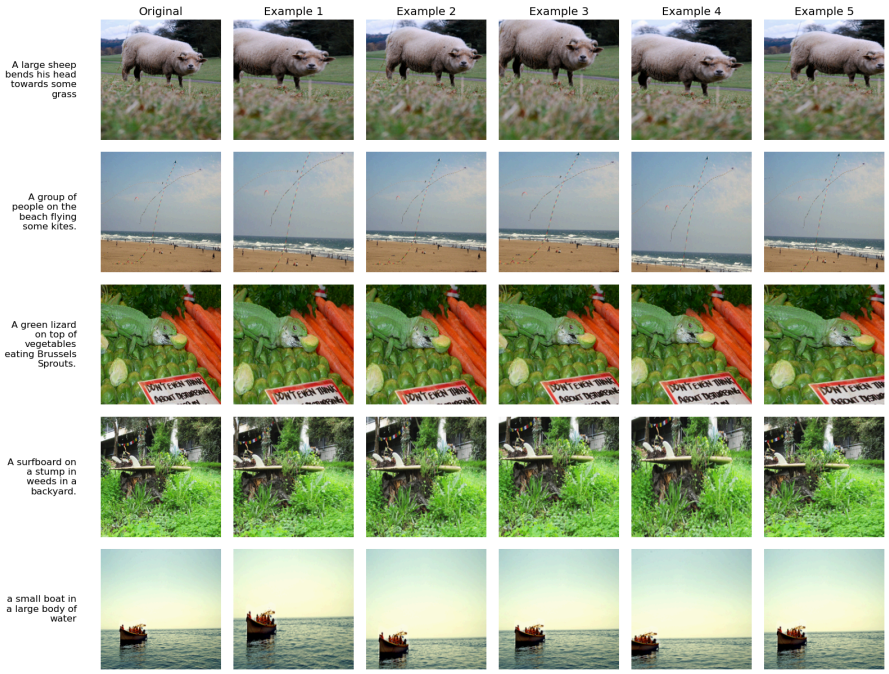

Figure 36: A larger lower bound (90%) for a random crop retains high-level semantic features. Notice how this is not the case for very low values, as shown in Figure 35. Image-text pairs have been taken from the COCO train set [Lin+14].

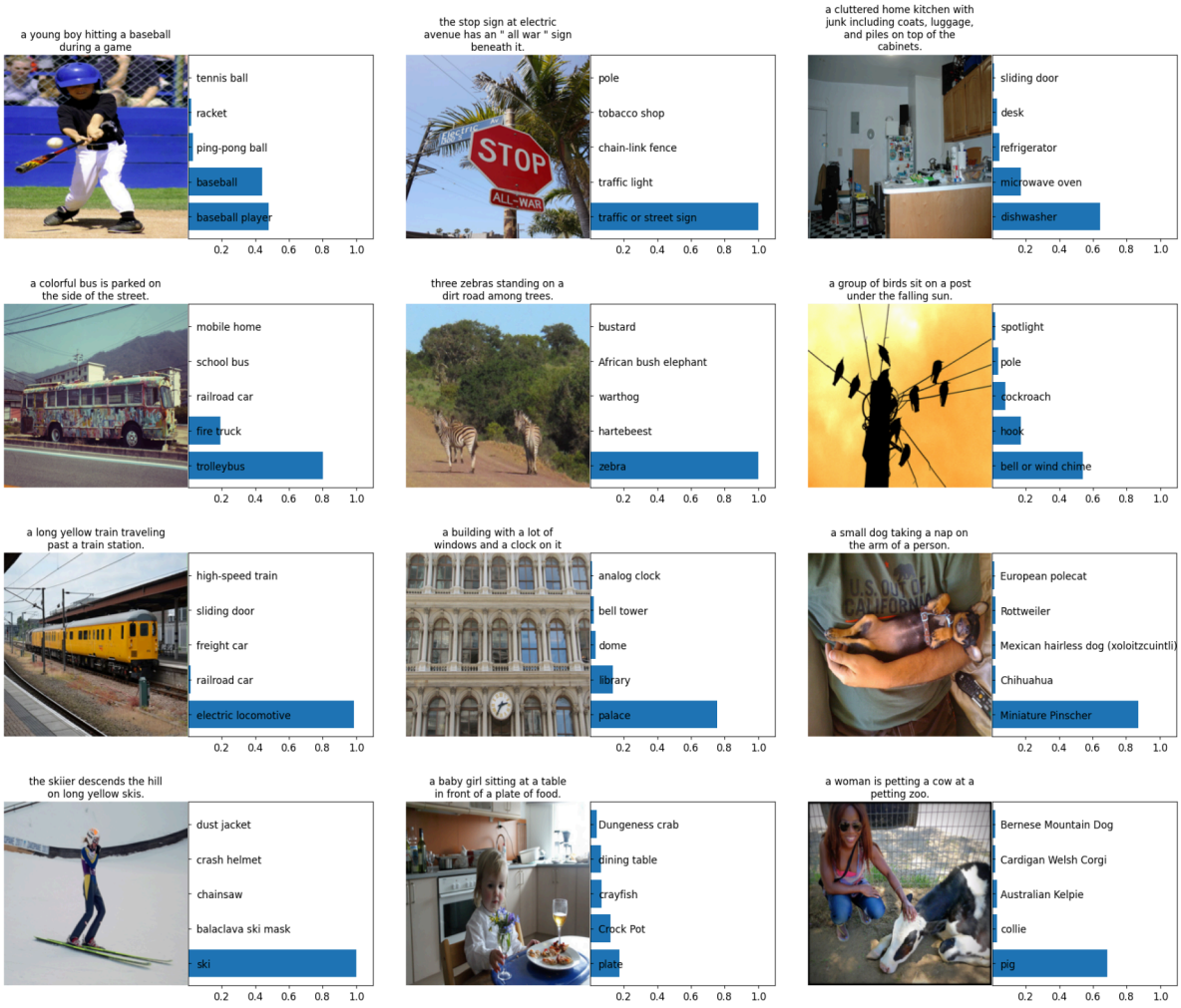

Figure 37: Visualization of the predicted probabilities for the top-5 ImageNet-1K [Rus+15] classes on image-text pairs from the COCO train set [Lin+14]. While the predicted classes are not always correct, e.g. bottom right, they are able to capture to some extend the semantic content of the image, and even the text. The latter is crucial for the approach of SHRe [AVT17]. Note: The figure does not stem from the SHRe paper [AVT17], but is a custom visualization of the concept. However, it is inspired by the CLIP paper [Rad+21]. The ResNet-50-A1 [WTJ21] model is used for the predictions.

Approach	# Image-Text pairs
FLAVA [Sin+21]	70M
CLIP [Rad+21]	400M
VLMo [Bao+22]	4M/1B
CoCa [Yu+22]	>3B
BEiT-3 [Wan+23]	21M
This work	3.3M

Table 39: A comparison of the number of image-text pairs used for pretraining in different approaches. We use significantly fewer pairs compared to other approaches. We compare the 1B variant of VLMo [Bao+22] to our models, and compare the cost of training our final model to the cost of VLMo with 4M pairs. This is because the authors of VLMo did not publish the compute used for their 1B variant.

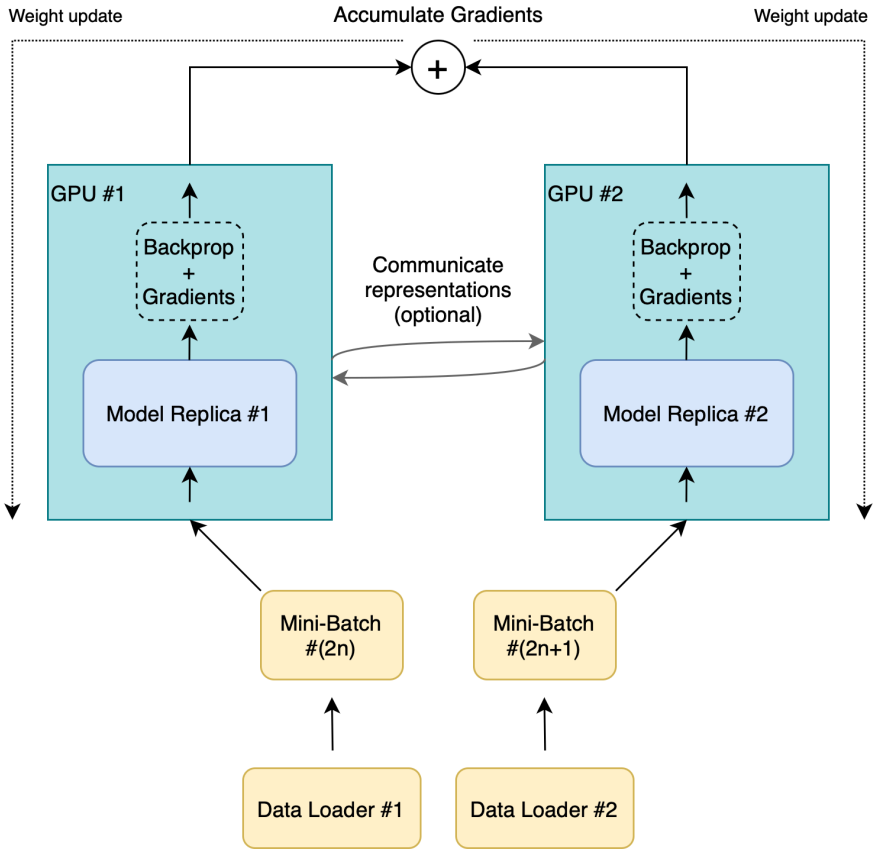

Figure 38: Distributed Data Parallel allows training a model with larger batch sizes than possible on a single GPU. The actual batch size per GPU does not change, but gradient updates are synchronized across GPUs, leading to weight updates that are equivalent to a larger batch size [Li+20].

Figure 39: Results of image retrieval on a 1k subset of the COCO test set [Lin+14], as described in [AVT17].

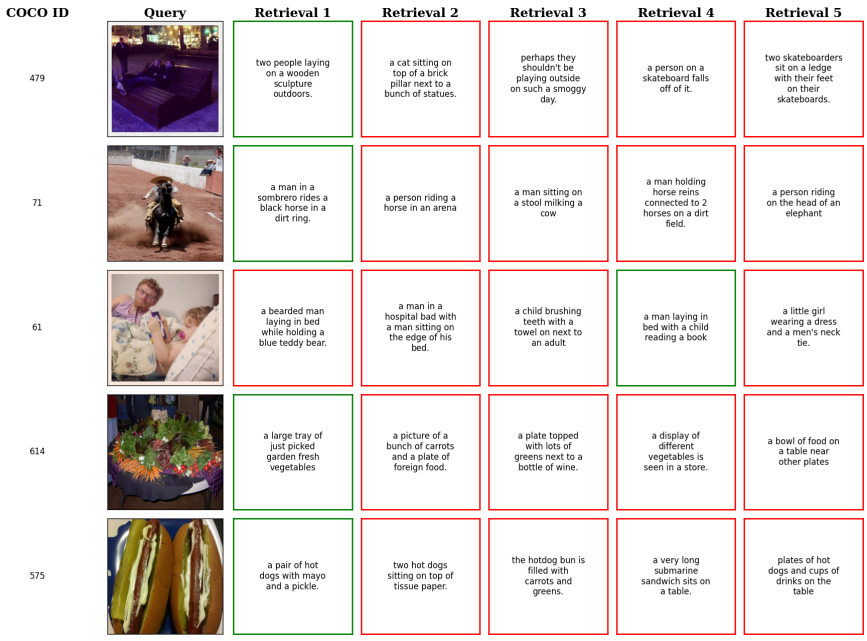

Figure 40: Results of text/caption retrieval on a 1k subset of the COCO test set [Lin+14], as described in [AVT17].

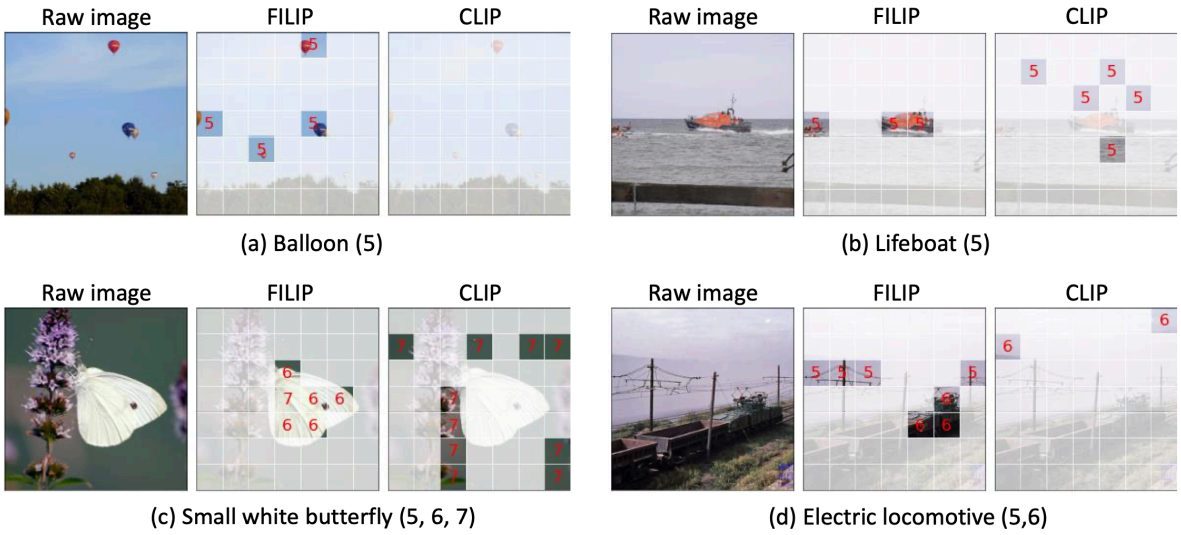

Figure 41: Cross-Model Late Interaction (CMLI) on FILIP [Yao+21] and CLIP [Rad+21]. Numbers in parentheses describe the index of the text token in the text sequence, and numbers in patches to which text token a patch is matched. While FILIP is able to achieve a localization of the correct image patches, CLIP fails to do so. The figure is directly taken from FILIP [Yao+21].

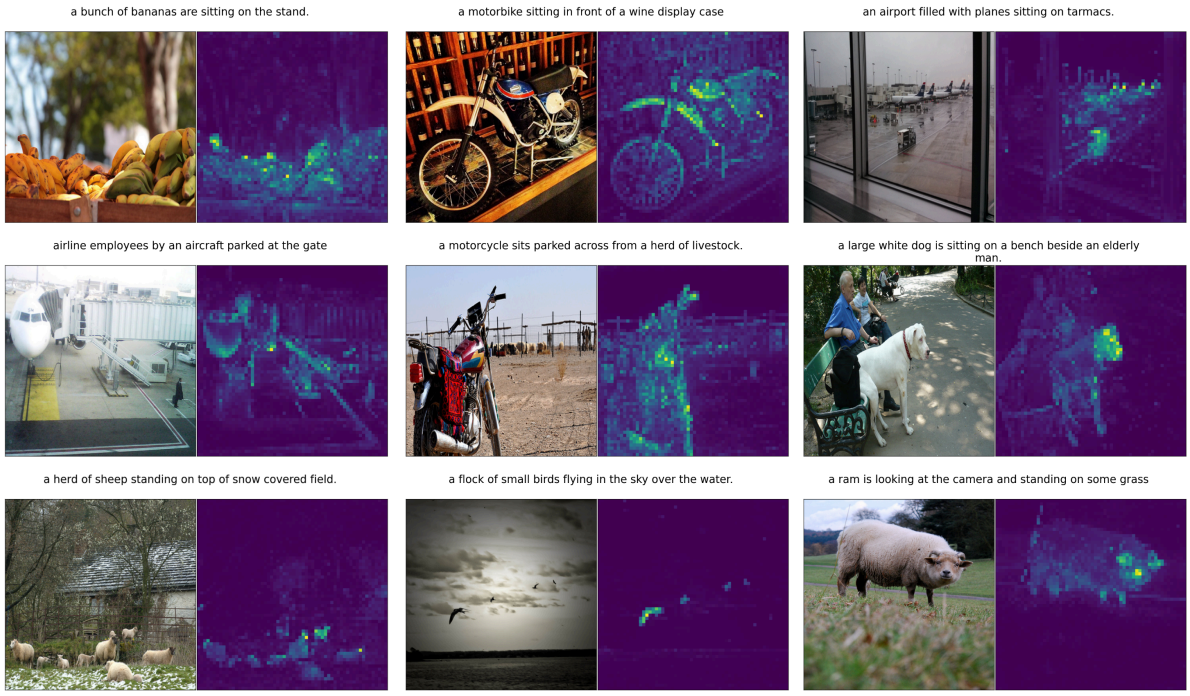

Figure 42: Fine-grained DINO [Car+21] self-attention map of the [I_CLS] token on image-text pairs of the COCO test set [Lin+14]. Heatmap is average over all attention heads.

COCO ID	Query	Retrieval 1	Retrieval 2	Retrieval 3	Retrieval 4	Retrieval 5
255165		small office workspace with multiple computer monitors and office equipment.	a desk filled with some paperwork, a laptop and a computer with two monitors	a desk with keyboards and a laptop and a pair of monitors	several computers, including a laptop, sit on a desk.	a computer desk with a laptop computer and a desktop computer.
517610		a photo taken inside a green house with many plants.	an open with plants sitting in front of it on a ledge.	long pot of flowers sitting underneath an open window.	a case of flowers sitting on top of the window seal.	some window plants near an open window pane
246597		a young boy determinedly rides a skate board	black and white photograph of a man with a skateboard.	a boy that is standing on a skateboard.	a black and white photograph of a baseball player about to catch a ball.	a young man in a sweat suit preparing to kick a soccer ball.
475923		a blue door with a white and black dog in front of it.	a dog sleeping on the front poor of a building with bright blue doors.	walls are painted blue and white while an open doorway reveals another room painted yellow.	a black and white dog sleeps in front of a blue door.	the doorway and antique mirror is in the room with blue sky and cloud painted walls.
425151		a view of a living room area decorated with tiles and wood flooring.	a couch, a mirror and some cabinets in an open room.	a room that has some very colorful carpeting.	a yellow wall living room with a large and bright white window.	a living room with wooden floors and furniture

Figure 43: Example text retrievals on the full COCO test set [Lin+14] with 25k images. Green and red boxes indicate correct and incorrect retrievals, respectively. Since each image has multiple captions, there are multiple correct retrievals for each image.

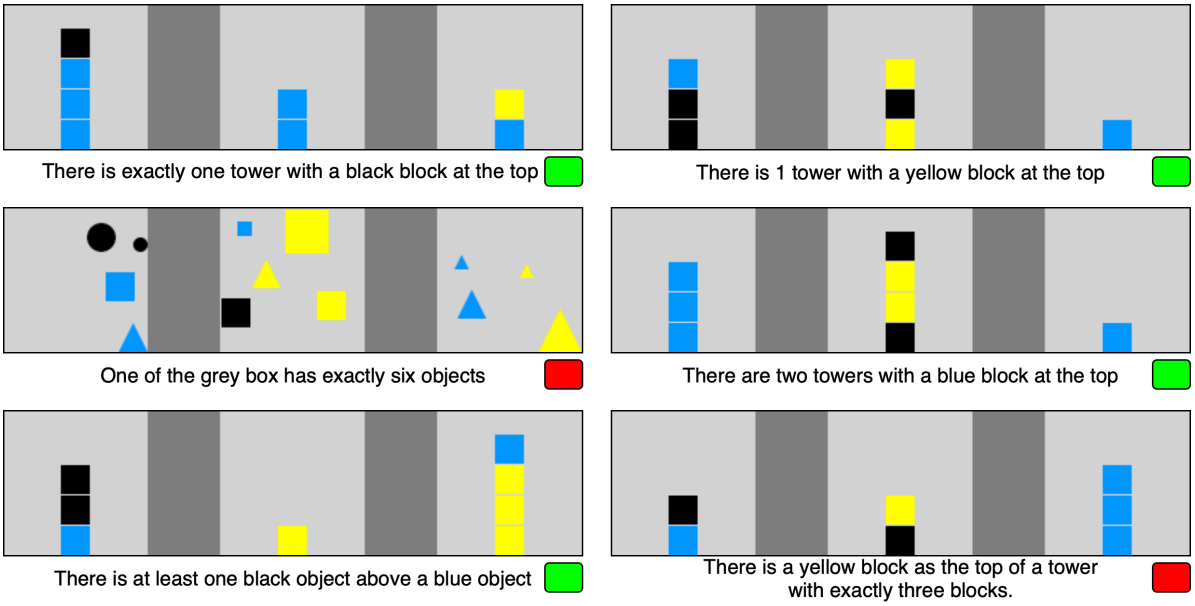

Figure 44: Example images with corresponding statements from the NLVR2 dataset [Suh+19]. Green and red boxes at the bottom right of each example indicate a correct and incorrect statement, respectively. The examples are taken from the official NLVR2 website[18].

[18] https://lil.nlp.cornell.edu/nlvr/nlvr.html

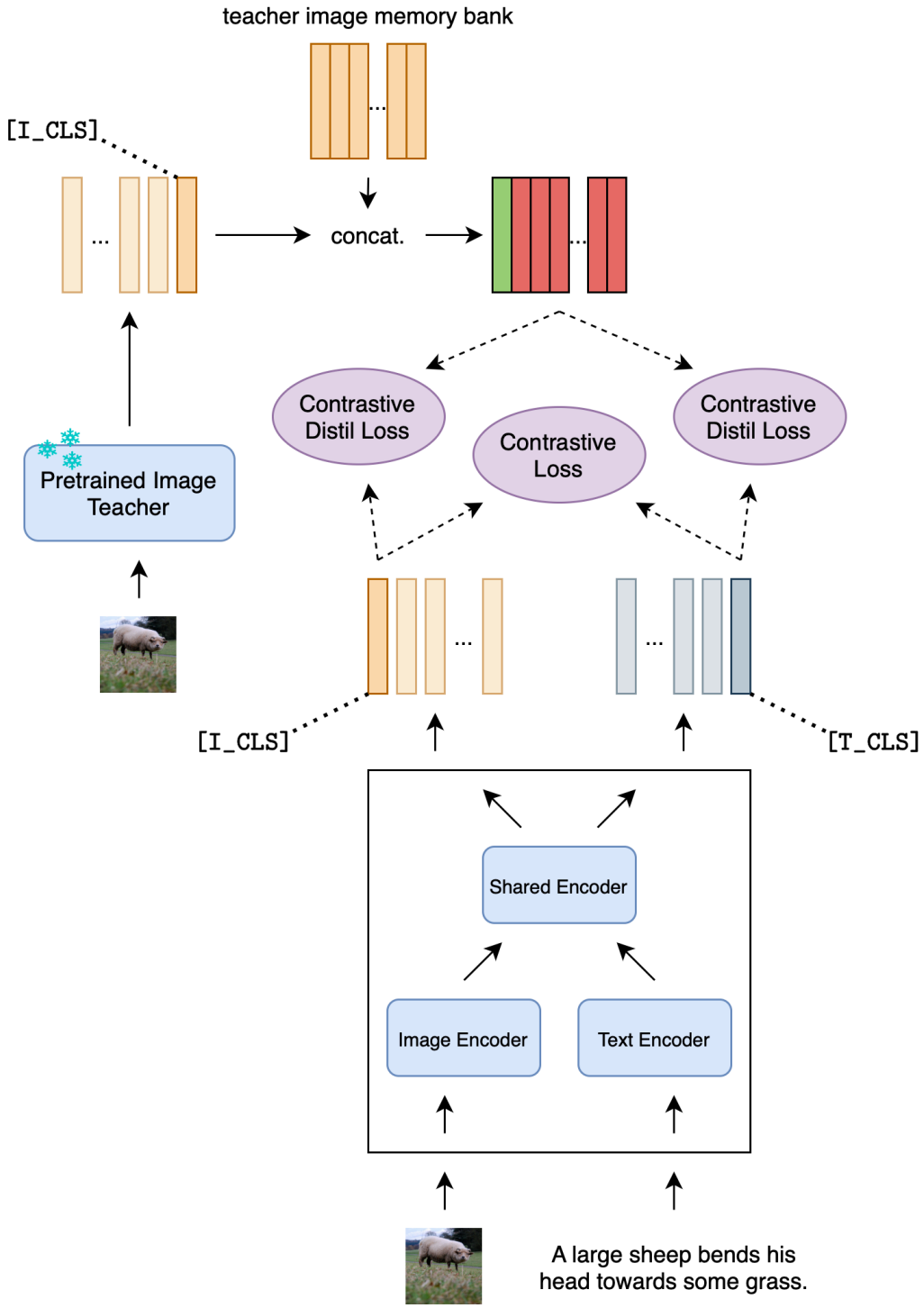

Figure 45: Visualization of the overall framework of S-SMKE. The teacher remains frozen during training, indicated by the snowflakes.

AD Technical Details

ADA Software

All implementations in this research were conducted using PyTorch [Ans+24] and PyTorch Lightning [al.19]. PyTorch Lightning is a high-level library built on top of PyTorch that provides functionalities such as checkpointing, logging, and distributed training. These features are particularly important for the distributed data parallel (DDP) training we perform (see Section Section 5.2.1.2). By leveraging PyTorch Lightning, we avoid the need to manually implement these functionalities in PyTorch, which, although it offers a high-level API, can be more prone to errors. This approach allows us to save time and focus on the actual implementation of our models, as errors in vanilla PyTorch are likely and can be difficult to debug, especially in the context of distributed training.

All experiments were performed using Python 3.10, with PyTorch 2.2.0, PyTorch Lightning 2.4.0, and CUDA 12.1.1, on an Ubuntu 22.04 machine.

Given that research inevitably involves a significant amount of experimentation and iterative development, we utilized the experiment tracking tool Weights & Biases[19] to keep track of all experiments. The code is available at https://github.com/TimCares/EV-LP.

ADB Hardware

To train the models and store the data, we used the GPU cloud platform runpod.io[20] . This platform provides access to a wide range of GPU types, including the NVIDIA RTX 4090 24GB, which we use for training almost all of our models. The reason for choosing this platform over traditional cloud providers like AWS or GCP is that the price per GPU hour is significantly lower, which allows us to train more models for the same budget. For example, as of September 2024, the price per GPU hour for an NVIDIA RTX 4090 24GB on runpod.io is $0.69. A comparable GPU on AWS, e.g. the NVIDIA V100 16GB, costs $3.06[21] per GPU hour. This price difference is despite the fact that the V100 was released in 2017, while the RTX 4090 was released in 2022. The RTX 4090 is also faster than the V100, with a higher memory bandwidth and more CUDA cores, so training on the RTX 4090 is more cost-effective by a margin. This evaluation is significant, as there is no external funding for this work.

To store all of our data, which is a total of >900 GB, we use a network volume provided by runpod.io. This volume is mounted on a virtual machine on start-up, allowing to access the data for training on the GPUs and provides high flexibility.

The RTX 4090 instances we used have 61 GB of memory and 8 virtual CPUs for one GPU, and around 132 GB of memory with 16 virtual CPUs if using two GPUs, which is similar to that of AWS.

[19] https://wandb.ai

[20] https://www.runpod.io

[21] Price is taken from the official AWS pricing page (https://aws.amazon.com/de/ec2/instance-types/p3/) and based on the on-demand price for the region us-east (North-Virginia).

runpod.io also provides on-demand vm instances, as well as spot instances. The latter can be automatically terminated if demand is high, but the price is significantly lower. On-demand instances are more expensive but are non-interruptible, which is, even though we create a model checkpoint after each training epoch, important for long-running experiments, which is the case for most training runs in this work. As of September 2024, the price for an on-demand instance with a single RTX 4090 is $0.69 per GPU hour, while the price for a spot instance is just $0.35 per GPU hour. The price for an instance increases proportionally with the number of GPUs used, so for us, a two-GPU instance costs $1.38 per GPU hour.

ADC Cost Breakdown

We present a summary of the costs for training individual models in Table 40 and the cumulative costs for pretraining, fine-tuning, data transfer, and data storage in Table 41. The total expenses for this thesis amount to $1,744, which, thanks to the cost-effective GPU instances on runpod.io, is significantly lower than the estimated costs on AWS, which would be more than $6,195.

Model	Parameters	Data (Samples)	Hardware	Compute Time (hrs)	Compute Cost ($)	Estim. AWS Cost ($)[22]
DistilData2Vec2	43M	1.28M	1× RTX 4090	6.9	4.8	21.1
C-DistilData2Vec2	43M	1.28M	1× RTX 4090	6.9	4.8	21.1
F-DistilBERT	66M	13M	1× RTX 4090	27	18.6	82.6
SHRe	115M	3.3M	1× RTX 4090	13.1	9.1	40.1
Transformer SHRe	117M	3.3M	2× RTX 4090	7.7	10.6	47.1
S-SMKE	117M	3.3M	2× RTX 4090	11.1	15.3	67.9
S-SMKE$_{\text{CTL_MB}}$	117M	3.3M	2× RTX 4090	11.2	15.5	68.5

Table 40: Cost breakdown of all major models trained in this work.

	Amount	Total Cost ($)	Estim. AWS Cost ($)
Pretraining	1,399h	1,373	4,281[22]
Finetuning	145h	142	444[22]
Data Transfer	1TB	39	-
Data Storage	1TB	190	1,470[23]
Total	-	1,744	>6,195

Table 41: Total costs for pretraining, finetuning, data transfer, and data storage on runpod.io[20]. We compare to the estimated costs on AWS for similar services.

[22]Cost was calculated based on the on-demand price of one NVIDIA V100 32GB AWS in the region US East (Northern Virginia), which is, as of September 2024, $3.06 per hour: https://aws.amazon.com/ec2/instance-types/p3/. The GPU is slightly slower than the RTX 4090, so the actual cost is likely higher.

[23]Cost was calculated based on the price for 1TB of storage on AWS S3 in the region (Europe) Frankfurt, which is, as of September 2024, $0.0245 per GB per month: https://aws.amazon.com/s3/pricing/

Although S-SMKE, with more than 100 million parameters, is still a large model, utilizing knowledge distillation and pretrained unimodal models greatly reduces the costs compared to training a model from scratch. Training our final model, S-SMKE$_{\text{CTL_MB}}$, costs \$15.5, which is negligible compared to the cost of training CLIP [Rad+21], estimated at approximately \$77,414[24] , and the large variant of VLMo [Bao+22], trained on 4 million image-text pairs, which costs around \$9,676[25].

The authors of VLMo did not publish the compute used for their variant trained on 1 billion image-text pairs, which is the one we compare to, so the true cost will be significantly higher. Nonetheless, our model is approximately **99.84%** cheaper than VLMo and an astounding **99.98%** cheaper to train than CLIP. Other papers, such as BEiT-3 [Wan+23] and FLAVA [Sin+21], unfortunately do not provide any information on the GPUs used or the duration of training, which is how we estimated the costs for VLMo and CLIP.

While the end-to-end process of training our model is technically more expensive when accounting for the costs of training the unimodal models and the teacher used for distillation, these models already existed and were originally trained for other purposes, which is why we do not consider them in our calculation. Moreover, VLMo [Bao+22] also utilizes pretrained components in their model.

It is important to note that the cost reduction does not account for the size of the datasets and the number of parameters of the models. Our dataset is smaller than those used by VLMo and significantly smaller than that used by CLIP (see Table 39), which is a major factor in the cost reduction. Additionally, we acknowledge that VLMo achieves significantly better results, and we only outperform CLIP on retrieval tasks. Therefore, whether the comparison can be considered fair is debatable, and it would be more significant if we outperformed, or were on par with, VLMo and CLIP on all tasks. When it comes to raw performance (and a multimodal model already exists for the use case), we encourage the use of pretrained multimodal models like VLMo, CLIP, or BEiT. Nonetheless, when it comes to building cheap and efficient multimodal models from scratch—that is, without any pretrained **multimodal** components—our approach excels.

[24]CLIP uses 256 NVIDIA V100 32GB GPUs for 12 days of training. Although the exact cost per GPU hour is unknown, we estimate it to be around \$1.05 per hour, which is the price for a V100 16GB on AWS with a 3-year reserved instance. Since OpenAI trains models on a daily basis, we consider using the reserved instance price as realistic. The on-demand price for a V100 32GB is \$3.06 per hour.

[25]VLMo uses 128 NVIDIA V100 32GB GPUs for 3 days of training. Since this model was developed by researchers at Microsoft, we assume the same price as for CLIP[24].

Bibliography

[LM21] Y. LeCun and I. Misra, "Self-supervised learning: The dark matter of intelligence." [Online]. Available: https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/

[Lin+14] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, Springer International Publishing, 2014, pp. 740–755.

[Rad+21] A. Radford *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, M. Meila and T. Zhang, Eds., in Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 8748–8763.

[Sin+21] A. Singh *et al.*, "FLAVA: A Foundational Language And Vision Alignment Model," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15617–15629, Jun. 2021.

[Wan+23] W. Wang *et al.*, "Image as a Foreign Language: BEIT Pretraining for Vision and Vision-Language Tasks," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 19175–19186. doi: 10.1109/CVPR52729.2023.01838.

[Kap+20] J. Kaplan *et al.*, "Scaling Laws for Neural Language Models," *CoRR*, 2020, [Online]. Available: https://arxiv.org/abs/2001.08361

[AVT17] Y. Aytar, C. Vondrick, and A. Torralba, "See, Hear, and Read: Deep Aligned Representations," *ArXiv*, 2017.

[Rus+15] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[Yu+22] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "CoCa: Contrastive Captioners are Image-Text Foundation Models," *Transactions on Machine Learning Research*, 2022, [Online]. Available: https://openreview.net/forum?id=Ee277P3AYC

[Gou+21] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[San+19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in *NeurIPS EMC^2 Workshop*, 2019.

[HVD15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network." 2015.

[Bao+22] H. Bao *et al.*, "VLMo: Unified Vision-Language Pre-Training with Mixture-of-Modality-Experts," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Curran Associates, Inc., 2022, pp. 32897–32912.

[Bae+23] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, "Efficient Self-supervised Learning with Contextualized Target Representations for Vision, Speech and Language," in *International Conference on Machine Learning, ICML*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., in Proceedings of Machine Learning Research, vol. 202. Honolulu, Hawaii, USA: PMLR, Jul. 2023, pp. 1416–1429.

[Bae+22] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language," in *International Conference on Machine Learning, ICML*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., in Proceedings of Machine Learning Research, vol. 162. Baltimore, Maryland, USA: PMLR, Jul. 2022, pp. 1298–1312.

[Vas+17] A. Vaswani *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[Bao+22] H. Bao, L. Dong, S. Piao, and F. Wei, "BEiT: BERT Pre-Training of Image Transformers," in *The Tenth International Conference on Learning Representations, ICLR*, OpenReview.net, Apr. 2022.

[Dev+19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *1st International Conference on Learning Repre-*

sentations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2013.

[BKH16] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *CoRR*, 2016, [Online]. Available: http://arxiv.org/abs/1607.06450

[He+16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778.

[Rad+19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019.

[Dos+21] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *9th International Conference on Learning Representations*, Austria: OpenReview.net, May 2021.

[He+20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 9726–9735.

[Che+20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, in ICML'20. Journal of Machine Learning Research, 2020.

[CH21] X. Chen and K. He, "Exploring Simple Siamese Representation Learning," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15745–15753.

[CH20] T. Chen and G. Hinton, "Advancing Self-Supervised and Semi-Supervised Learning with SimCLR." [Online]. Available: https://research.google/blog/advancing-self-supervised-and-semi-supervised-learning-with-simclr/

[Yao+21] L. Yao *et al.*, "FILIP: Fine-grained Interactive Language-Image Pre-Training," *CoRR*, 2021.

[You+14] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, Feb. 2014.

[SF08] A. Sorokin and D. Forsyth, "Utility data annotation with Amazon Mechanical Turk," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, Anchorage, AK, USA: IEEE Computer Society, Jun. 2008, pp. 1–8.

[GC19] A. Gokaslan and V. Cohen, "OpenWebText Corpus." 2019.

[Wan+18] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, T. Linzen, G. Chrupała, and A. Alishahi, Eds., Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355.

[Soc+13] R. Socher *et al.*, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, Eds., Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.

[WSB19] A. Warstadt, A. Singh, and S. R. Bowman, "Neural Network Acceptability Judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, 2019.

[Cer+17] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, S. Bethard, M. Carpuat, M. Apidianaki, S. M. Mohammad, D. Cer, and D. Jurgens, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14.

[DB05] B. Dolan and C. Brockett, "Automatically Constructing a Corpus of Sentential Paraphrases," in *Third International Workshop on Paraphrasing (IWP2005)*, Asia Federation of Natural Language Processing, Jan. 2005.

[Raj+16] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392.

[DGM05] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, in MLCW'05. Southampton, UK: Springer-Verlag, 2005, pp. 177–190.

[WNB18] A. Williams, N. Nangia, and S. Bowman, "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H.

Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122.

[LDM12] H. J. Levesque, E. Davis, and L. Morgenstern, "The Winograd schema challenge," in *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, in KR'12. Rome, Italy: AAAI Press, 2012, pp. 552–561.

[Kri+17] R. Krishna *et al.*, "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations," *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, May 2017.

[Sha+18] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2556–2565.

[Cha+21] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut, "Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Vision Foundation, Jun. 2021, pp. 3557–3567.

[OKB11] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2Text: describing images using 1 million captioned photographs," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, in NIPS'11. Granada, Spain: Curran Associates Inc., 2011, pp. 1143–1151.

[UVL17] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization." [Online]. Available: https://arxiv.org/abs/1607.08022

[LH19] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA: OpenReview.net, May 2019.

[Kri12] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *University of Toronto*, p. , 2012.

[Pen+22] Z. Peng, L. Dong, H. Bao, Q. Ye, and F. Wei, "BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers," *CoRR*, 2022.

[Cub+20] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le, "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., Curran Associates, Inc., 2020, pp. 18613–18624.

[Zha+18] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada: OpenReview.net, Apr. 2018.

[Yun+19] S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2019, pp. 6022–6031.

[Zho+20] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 13001–13008, Apr. 2020.

[Pet+18] M. E. Peters *et al.*, "Deep Contextualized Word Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237.

[WTJ21] R. Wightman, H. Touvron, and H. Jégou, "ResNet strikes back: An improved training procedure in timm." 2021.

[Li+20] S. Li *et al.*, "PyTorch distributed: experiences on accelerating data parallel training," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3005–3018, Aug. 2020, doi: 10.14778/3415478.3415530.

[Li+17] A. Li, A. Jabri, A. Joulin, and L. van der Maaten, "Learning Visual N-Grams from Web Data," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017.

[Tou+21] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jegou, "Going deeper with Image Transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 32–42.

[Wu+18] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised Feature Learning via Non-parametric Instance Discrimination," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 3733–3742.

[CXH21] X. Chen, S. Xie, and K. He, "An Empirical Study of Training Self-Supervised Vision Transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 9620–9629.

[Che+20] X. Chen, H. Fan, R. Girshick, and K. He, "Improved Baselines with Momentum Contrastive Learning," *arXiv preprint arXiv:2003.04297*, 2020.

[Li+24] J. Li, R. R. Selvaraju, A. D. Gotmare, S. Joty, C. Xiong, and S. C. Hoi, "Align before fuse: vision and language representation learning with momentum distillation," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, in NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2024.

[Car+21] M. Caron *et al.*, "Emerging Properties in Self-Supervised Vision Transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9630–9640.

[Suh+19] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi, "A Corpus for Reasoning about Natural Language Grounded in Photographs," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6418–6428.

[Ant+15] S. Antol *et al.*, "VQA: Visual Question Answering," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, pp. 2425–2433.

[KSK21] W. Kim, B. Son, and I. Kim, "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, M. Meila and T. Zhang, Eds., in Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 5583–5594.

[Guz+22] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending Clip to Image, Text and Audio," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, May 2022, pp. 976–980.

[WH18] Y. Wu and K. He, "Group Normalization," in *Computer Vision -– ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, Munich, Germany: Springer-Verlag, 2018, pp. 3–19.

[Ans+24] J. Ansel *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, in ASPLOS '24. La Jolla, CA, USA: Association for Computing Machinery, 2024, pp. 929–947.

[al.19] W. F. et al., "PyTorch Lightning." [Online]. Available: https://www.pytorchlightning.ai/