# Does Text Matter?

## Extending CLIP with OCR and NLP for Image Classification and Retrieval

**Jordan Sassoon**[1]

**Supervisor(s): Lydia Y. Chen, Zilong Zhao**

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Jordan Sasssoon
Final project course: CSE3000 Research Project
Thesis committee: Lydia Y. Chen, Zilong Zhao, Anna Lukina

## Abstract

Contrastive Language-Image Pretraining (CLIP) has gained vast interest due to its impressive performance on a variety of computer vision tasks: image classification, image retrieval, action recognition, feature extraction, and more. The model learns to associate images with their descriptions, a powerful method which allows it to perform well on unseen domains. Often, the descriptions fail to capture text which is contained within the image, a source of information which could prove useful for a handful of computer vision tasks. This limitation requires finetuning in domains where contained text is important. In fact, CLIP has mixed performance on Optical Character Recognition (OCR). This paper proposes a novel architecture: OSBC (OCR Sentence BERT CLIP), which combines CLIP and a custom text extraction pipeline, composed of an OCR model, and a Natural Language Processing (NLP) model. OSBC uses the text contained within images as an additional feature when performing image classification and retrieval. We tested the model on multiple datasets for each task, occasionally outperforming CLIP when images contained text, while maintaining finetunability, and improving the model's robustness. In addition, OSBC was designed to be generalizable, meaning it is expected to perform well on unseen domains without finetuning, though this was not achieved in practice.


## 1 Introduction

With ever increasing compute power and dataset sizes, zero-shot learning has become an interesting and viable alternative to supervised learning. Zero-shot models, such as the Contrastive Language-Image Pretraining (CLIP) model [1] or the Generative Pretrained Transformer (GPT) family [2, 3], train on massive datasets, with millions or billions of data points [4, 5]. The purpose behind this expensive compute is to teach the model to recognize many different classes, so that their feature extraction is applicable in various contexts. This allows zero-shot models to perform well on numerous tasks, without needing to be finetuned. This is what is referred to as "generalizability". However, a broader approach to problem-solving fails to confront problems with highly specific domains. These fine-grained tasks often require knowledge which is not available to zero-shot models at training time.

Amongst the fields that have been challenged by zero-shot models, computer vision is one of the most popular ones. Tasks such as image classification, image retrieval, action recognition, and many more, were explored by Radford et al. with their deeply interesting model: CLIP [1]. Now well known, this algorithm was the first to prove that learning the association between images and their descriptions was a valid strategy for zero-shot learning. In fact, CLIP was trained on the Web Image Text dataset, composed of 400 million image-text pairs. Various versions of the model have been released through the years with larger training set sizes and architectures [6], and it has proven to be customizable and extendable to many use-specific cases [7, 8, 9, 10].

While CLIP comes with multiple benefits: generalizability, state-of-the-art performance, finetunability, it still suffers from the same issues as other zero-shot, general models. Fine-grained classification, abstract concepts such as counting or measuring distances between objects, and Optical Character Recognition (OCR), prove difficult for the model. CLIP is also highly susceptible to prompt engineering: the descriptions of the images have to be formatted to resemble natural language, though often the images only come with one or few word descriptors. These bare descriptions need to be engineered into full sentences, so that CLIP can make proper use of them [Fig. 1].

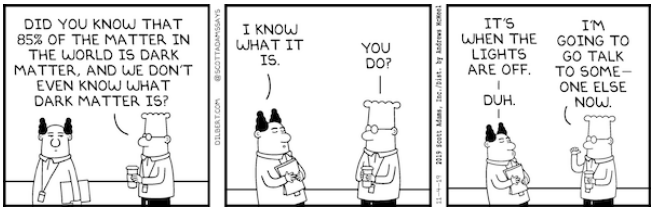



Figure 1: An image of the Dilbert [11] comic from the Laion5B dataset [5]. The image has the original caption "Dilbert" which can be formatted into "A comic strip of Dilbert".

It is therefore unlikely that the image descriptions express the whole text contained within the images, preventing CLIP from learning to use this additional feature in training and inference. Therefore, we aimed to create an architecture which supports CLIP by explicitly extracting inner text with a novel pipeline and using it for image classification and retrieval [Fig. 2].

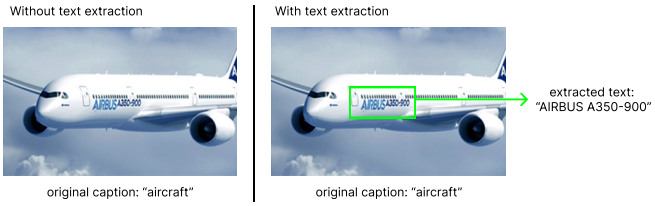



Figure 2: An image of an aircraft from the Laion5B dataset [5]. The image has the original caption "aircraft", and fails to capture the text of the aircraft model. This text would be useful, for example, for aircraft classification.

Firstly, the text extraction pipeline needs an OCR model, and thankfully there are many models which have been designed with this sole purpose in mind. Openly available tools such as Google's Tesseract OCR [12], Cloud Vision API [13], EasyOCR [14], TrOCR [15] specialize in extracting raw natural language text from images.

Secondly, since OCR models have a limited understanding of language, we leveraged Natural Language Processing

(NLP) models to encode the extracted text. Bidirectional Encoder Representation from Transformer (BERT) [16] models compose a family of models that tackle many complex NLP tasks. BERT models excel at comprehending language in broader contexts and resolving text ambiguities. They are also extremely useful when comparing two texts and evaluating their similarity. In this regard, Sentence BERT (SBERT) [17] is a model which retains the family's attributes and specializes in calculating the similarity between sentences.

With this, we have all the parts needed to build and introduce the new architecture: OSBC (OCR Sentence BERT CLIP). The model combines the strengths from all the models with the hope to achieve better accuracy on computer vision tasks, while maintaining all the desirable properties of the components. The OCR and SBERT sections form a pipeline (OCR-SBERT) which focuses on text extraction and comparison, while CLIP focuses on understanding the relationship between images and their descriptions. The resulting search space is therefore composed of a triplet of features: image, description, and inner text.

We focus on image classification and retrieval for this paper, though we emphasize that OSBC is a model intended to work with any task involving images or videos that contain text. In order to rigorously evaluate the proposed architecture, OSBC, the following research questions need to be answered:

- Does OSBC outperform CLIP and the OCR-SBERT pipeline on image retrieval and classification?
- Does OSBC maintain zero-shot generalizability over tasks and datasets?
- Does OSBC maintain finetunability?
- Do the results hold with larger, newer CLIP versions?

OSBC is expected to prove better performance than the two pipelines it is composed of. In theory, the model combines the useful parts of the text extraction pipeline and CLIP, meaning that it should perform at least as well as the best model in the architecture. OSBC was created with the aim to maintain the properties of generalizability and finetunability, both of which have to be tested. Furthermore, it is possible that as CLIP evolves and trains on larger datasets, it could learn to perform OCR at a high level. This could remove the need for explicitly extracting the contained text using the proposed pipeline.

The following sections aim to explain in further detail all the models we chose to create OSBC, and how they work together. OSBC is thoroughly evaluated, and the results are discussed in hopes of answering the above stated research questions. There is also a section regarding the reproducibility and ethical aspects of our research, followed by a conclusion.

## 2 Related Work

### CLIP and Zero-Shot Learning

This research relies on a combination of multiple machine learning models and ideas. A lot of inspiration was drawn upon reading the CLIP paper from Radford et al. [1]. Contrastive Language-Image Pretraining (CLIP) stands in the idea of learning the similarity between images and their captions. This allows the model to leverage natural language as a descriptor for computer vision tasks, and understand broad concepts. The original paper trains two encoders (one for the image, one for the description) and minimizes the cosine distance between the linear projection of the returned embeddings. The proposed image encoder is either a ResNet [18] model or a Vision Transformers (ViT) [19] model. As ViTs proved much more effective and efficient, we will focus on these models in this paper. The text encoder is a standard transformer [20] based model.

The concept of zero-shot learning was already developed a decade prior, under the name of dataless classification [21]. In the following years the term zero-shot learning was coined [22], and gained traction thanks to Lampert et al. [23]. Most notably, the GPT family [2, 3] is an example of zero-shot learning applied to text prediction. Various works laid the basis for CLIP to be developed, stating zero-shot learning to be a viable method for transfer learning in computer vision [24, 25, 26, 27, 28]. Radford et al. brought zero-shot learning into the state-of-the-art bracket of computer vision and proved this is a highly interesting alternative to supervised learning. Moreover, CLIP has been extended into audio tasks [7], multilingual tasks [8], and other use-specific scenarios [9, 10]. This helped us understand how malleable the model is. Though we will not follow this, Wortsman et al. [29] show that CLIP can be finetuned without losing out-of-domain performance, and Song et al. [30] state that CLIP is a strong few-shot learner.

### OCR, TrOCR and PyTesseract

With regards to text extraction, or OCR, Google is one of the leaders in the industry OCR, releasing Tesseract [12] and Cloud Vision [13] as their most popular engines. Other tools were developed based on different underlying architecture and languages. EasyOCR [14] was developed by Jaded AI with the popular machine learning python library PyTorch [31]. Ocropus [32] uses the Long-Short Term Memory (LSTM) [33] network as its neural basis. Attention-ocr [34] implements attention [20] mechanisms in its architecture. Fitting in with the huggingface [35] environment, TrOCR [15] offers a simple yet powerful implementation based on a transformers architecture. Often, OCR models excel at extraction, but require NLP post-processing to improve their output, since NLP models can adjust the OCR text based on their understanding of language. TrOCR was developed with the aim of removing the post-processing step. To achieve this, Li et al. use a pretrained image transformer encoder (such as a DeiT [36]) and a pretrained text decoder (RoBERTa [37]). This allows TrOCR to train the interaction between an image-tailored transformer mechanism, and a text-tailored transformer mechanism, so that the output is already optimized for natural language correctness. Just as TrOCR, PyTesseract also post-processes the extracted text to refine the results. PyTesseract [38] is a python wrapper for Google's Tesseract OCR, and is one of the most commonly used OCR models. This model uses complex text detection techniques such as component analysis or stroke width transform to identify text regions. A large pretrained LSTM [33] network then classifies the recognized characters, and another component post-processes the output.

### NLP and SBERT

The BERT [16] family has become a staple amongst NLP models. As opposed to previous, unidirectional works, bidirectional encoding makes use of all the context surrounding a word. In this way, each token is not restricted to only having information about what precedes it. Though it achieved many state-of-the-art performances, BERT proved to be undertrained, which RoBERTa [37] aimed to tackle by robustly improving the model. Longer training times, larger batch sizes, removing objectives, and more adjustments elevated BERT to match the models published after it. However, both BERT and RoBERTa come with significant computational overhead on the task of semantic text similarity. To establish sentence similarity, both sentences have to be fed to the model, an inference limitation which scales quadratically. Instead, Sentence BERT (SBERT) [17] adds a pooling layer to the BERT or RoBERTa output, which enables it to generate fixed-size embeddings. These can be then compared with a trained classifier or with cosine similarity. Inference is therefore run on each sentence, not each sentence pair, which drastically reduces the compute.

In conclusion, many previous works laid the foundation for OSBC to be developed. However, we believe there is no previous work that uses inner text as an additional, explicit feature in zero-shot computer vision tasks.

## 3 Methodology

The proposed model, OSBC (OCR Sentence BERT CLIP), combines CLIP's pipeline with a novel OCR-SBERT pipeline for text extraction. The goal is to create an embedding space able to represent three features: images, descriptions, and the text contained within the image. This section explains in detail the OSBC model, with different sections based on the task at hand. Again, there are two tasks this model has been extended to for now: image classification and image retrieval, although the model could be extended to any other task involving text within images, or videos.

### OSBC Architecture

OSBC is composed of three models: an OCR model (either TrOCR or PyTesseract), SBERT, and CLIP. The OCR and SBERT models form the OCR-SBERT pipeline, and are responsible for text extraction and embedding. TrOCR or PyTesseract extract the text within the image in raw natural language, which is then encoded by the SBERT model. With this, inner text can be rapidly compared with a query, based on the task OSBC is applied to. CLIP instead focuses on encoding images and descriptions. Depending on the query, CLIP can either act as an image-to-text classifier, or a text-to-image retriever. Regardless of the task, both the OCR-SBERT pipeline and the CLIP pipeline output a similarity vector between the query, and the pre-populated search space [Fig 3]. For example, in image classification OSBC pre-computes the embeddings of the possible classes. Once the two pipelines return their similarity vectors, they are L1 normalized and added together. The following is the formula for L1 normalization, given a vector $v$ of length $n$.

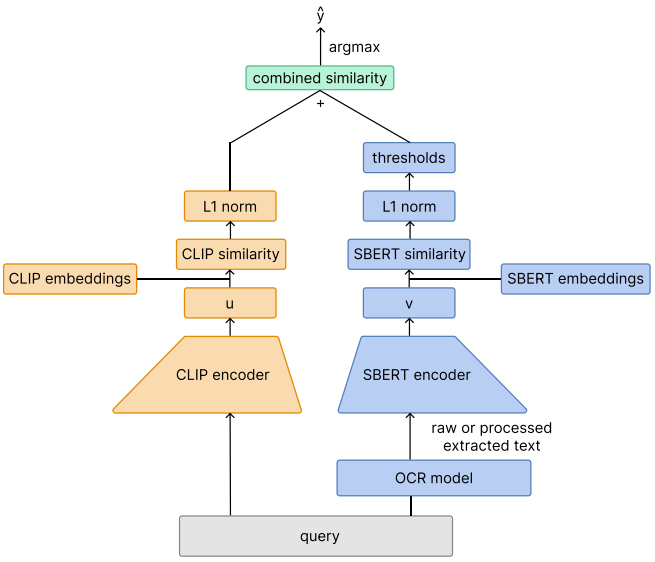



Figure 3: A representation of OSBC during inference. In blue the OCR-SBERT pipeline, in orange the CLIP pipeline. After encoding the query, similarity scores with previous embedding are normalized, post-processed in OCR-SBERT, and added together.

$$v_{normalized} = \frac{v}{\sum_{i=1}^{n} |v_i|}$$

In the case the OCR-SBERT pipeline does not extract any text, its predictions are ignored. Furthermore, if the similarity between the query and the embeddings computed by SBERT is lower than 70%, then those scores also do not affect OSBC's output. We set this arbitrary threshold with the aim to ignore misleading predictions, only taking the pipeline in consideration if its confidence is high. In practice, some misdirection is still present, a limitation which can be explored deeper in future works. The threshold value could be a trainable parameter, though we kept it fixed at 70% as this value performed well in our experiments.

The CLIP pipeline of OSBC does not modify the original architecture proposed by Radford et al. [1]. We used three pre-trained models offered by OpenAI's huggingface repository, namely "openai/clip-vit-base-patch16" (ViT-B/16 image encoder architecture), "openai/clip-vit-base-patch32" (ViT-B/32 image encoder architecture), and "openai/clip-vit-large-patch14" (ViT-L/14 image encoder architecture). The different versions come with increasing complexity and model sizes.

The OCR-SBERT pipeline [Fig. 4] is the innovative section of the architecture, as it is responsible for text extraction and embedding. We opted to use the TrOCR and PyTesseract models for OCR as they both have desirable advantages, and complement each other's performance in practice.

The TrOCR model is useful for extracting raw text since it already post-processes the output, which reduces the complexity of the pipeline. In addition, the model has been implemented in the huggingface environment, which greatly improved the speed of inference (about twice as

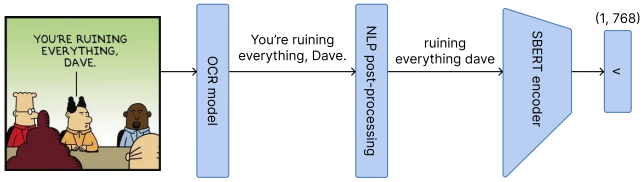



Figure 4: An example embedding for the OCR-SBERT pipeline on a Dilbert panel. The text "You're runing everything, Dave." is extracted by either TrOCR or PyTesseract. It is then post-processed in "runining everything dave", and encoded in a (1,768) sized vector by SBERT.

fast as PyTesseract), as it can be run with parallel batches and optimized data structures. We evaluate two versions of the TrOCR model: "microsoft/trocr-base-printed" and "microsoft/trocr-base-handwritten". The former is pretrained on printed text, the latter on handwritten text.

The PyTesseract model is a standard for OCR. Though it comes with a significantly lower compute time, we found it occasionally outperformed TrOCR in text extraction. This model was configured with the '–oem 3 psm 6' flags for sentence extraction, and the '–oem 3 psm 10' flags for single character extraction. PyTesseract advises using rigorous image preprocessing techniques to improve accuracy, though these are beyond the scope of this research, therefore we did not implement them.

Although both TrOCR and PyTesseract post-process their output for refined prediction, in image retrieval we implemented a secondary post-processing step based on common NLP techniques. Using the SpaCy [39] and NLTK [40] libraries, we ensured three actions would happen: the text would be converted to lowercase, stopwords would be removed, and numbers would be removed. Punctuation removal was a byproduct of the previous steps. This was done to maximize the extraction of significant information, so that text comparison could already ignore irrelevant words or numbers. This step does not apply to image classification, as the extracted text and the labels should be equivalent, therefore post-processing both is redundant or disruptive.

The extracted text is then fed into a Sentence BERT model, with the aim to calculate pair similarity. Also implemented in the huggingface environment, it comes with multiple versions. We use the "all-mpnet-base-v2" model which is currently the best performing SBERT version.

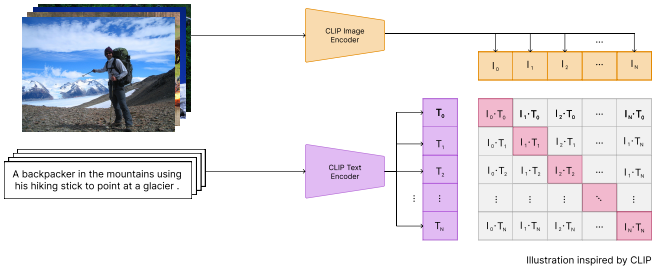



Figure 5: CLIP Contrastive Language-Image training pipeline, where the image and the text are encoded in the same space. Image and caption belong to the Filckr8k [41] dataset.

Regarding finetuning, we additionally trained the selected CLIP models [Fig 5]. As they were all integrated in the huggingface environment, we adapted the VisionTextDualEncoder class documentation [42] to train our pipelines. We leveraged the huggingface Trainer to carry out the finetuning, which resulted in quick computes. To interactively track the learning process, we used Weights and Biases [43], which logged all the relevant information (loss, model configuration, runtime, system information). This step was useful for reproducibility. As both the TrOCR and SBERT model are also trainable in a similar fashion, we imagine OSBC's performance could be improved even further. Refer to Table 10 for more information regarding the training configurations.

## Image Retrieval

Image retrieval was the task that sparked the idea for the model. Initially, we thought that a search-engine-like problem could benefit from comparing inner text and queries. OSBC ended up being applicable to many other scenarios. Retrieval is the task of selecting the most similar image given a query text.

As the prompt is already formatted in a natural language sentence, the prompt engineering step is unnecessary, and we can immediately feed the query in the CLIP model. All images must be embedded before retrieving, since the similarity vector will be calculated between each embedded image and the embedded query. For CLIP, the similarity vector is the dot product of the embedded query and the embedded images.

Not only the images must be processed before, but the text contained within them must also be extracted and encoded. This way, SBERT can compute the cosine similarity between the embedded query and the extracted text belonging to each image. To note that before we calculate pair similarity, the text and query are turned into lower case, and numbers and stopwords are removed.

After treating text-less images, normalizing, and applying the threshold, the two similarity vectors are added together. The image with the highest aggregate score is the final prediction from OSBC.

As an example use case, in the Dilbert [11] dataset extracting dialogue could be valuable information when querying [Fig. 6]. We envisioned that with this additional information, the user can retrieve specific images based on quotes or contained sentences with higher accuracy.

## Image Classification

In image classification, the aim is to select the correct class between a selection of choices, given the image [Fig. 7]. The datasets we worked with offered labels in natural language, so with some prompt engineering, the label was extended in a sentence. For example, the Modified National Institute of Standards and Technology (MNIST [44]) dataset contains images of handwritten digits. The numbers 0 through 9 are the labels, which are turned into the sentences "an image of the digit: 0". This was necessary since we needed to create prompts for the CLIP model to work correctly. Both the prompt and the image are embedded in the same space, enabling the model to calculate the dot product similarity between them.

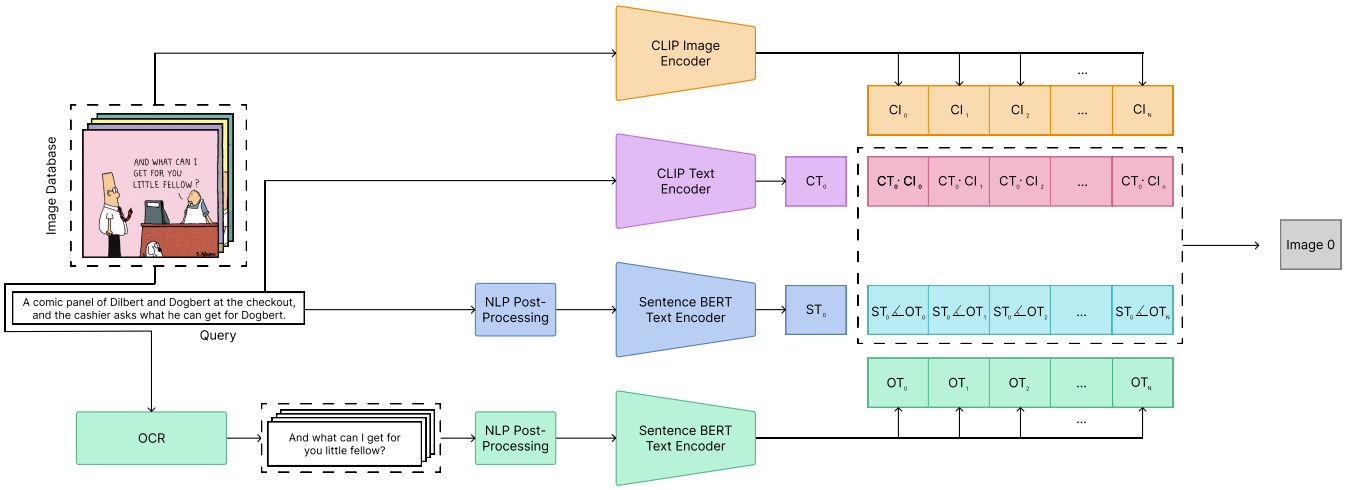



Figure 6: An example overview of the model for image retrieval on Dilbert. The images and the text contained within them are pre-encoded and ready for inference. The query is then encoded by CLIP and SBERT, and the similarity scores are aggregated.

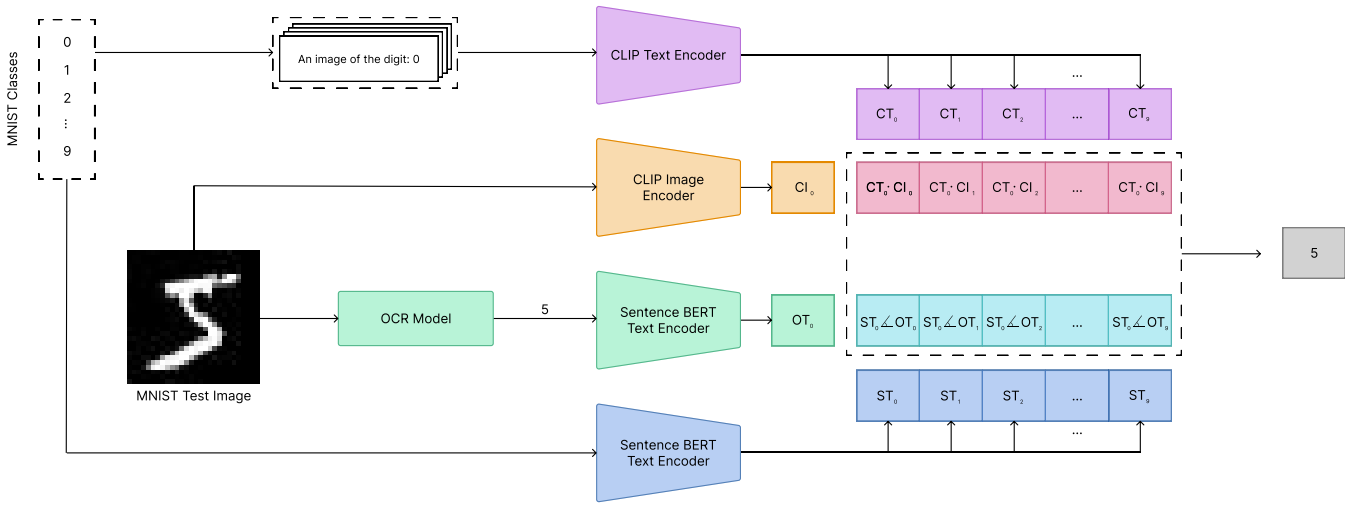



Figure 7: An example overview of the model for image classification on MNIST. The classes are formatted and pre-encoded ready for inference. The image of a 5 is encoded by CLIP, and the text "5" is extracted and embedded by SBERT. The similarity scores are aggregated.

In the case of the MNIST digits, the contained text is the label itself. The extracted inner text is embedded by SBERT and cosine similarity establishes the correlation between the text and each class. For the OCR-SBERT pipeline the classes are not formatted into sentences, as this adds unnecessary natural language information.

Once the similarity is calculated from both CLIP and SBERT for each class, and after post-processing, the class with the highest combined score is the selected one.

We recognize that the classification datasets we evaluate our model on are quite simple. They serve as a proof of concept for OSBC, as we believe the model can be applied to more complicated domains, specifically image classification tasks that require visual and textual context. An example could be classification of vehicles, as both the image of the vehicle (visual context) and the car brand or model (textual context) give essential clues.

## 4 Experimental Setup and Results

The experiments were designed to compare the OSBC architecture with two baselines: CLIP and the OCR-SBERT pipeline. The aim is to show that combining the two components is beneficial, instead of using them individually. For this reason, if either component generates disruptive information, their predictions will be disregarded. In theory, OSBC performs at least as well as its best component. In practice, we implemented a threshold which removes similarity scores if they are too low, meaning that the model could still have disruptive artefacts when predicting.

We tested the model on image classification datasets (MNIST [44], Characters from the standard OCR dataset [45], CIFAR-10 [46]) and image retrieval datasets (Flickr8k [41], Dilbert) [Fig 8]. The evaluation setup compared the CLIP and OCR-SBERT pipelines individually and OSBC (the two pipelines combined), and ran for multiple combina-


Image of a 0 from the Modified National Institute of Tecnhology (MNIST) dataset

Image of an A from the Characters dataset

Image of a car from the CIFAR-10 dataset

"A dog stands on his hind feet and catches a stream of water" from the Flickr8k dataset

"A picture of Dogbert wearing glasses, as Dilbert asks what he thinks" from the Dilbert dataset

Figure 8: Example images for each dataset

tions given the various versions of them. To answer the first research question positively, we compare OSBC to CLIP's performance on CIFAR-10 and Flickr8k, as they are datasets without inner text in the images, and expect to outperform both baselines on the rest. Notably, we only ran evaluations on a test subset of the data, as we will compare the models to their respective finetuned versions. Refer to Table 8 and Table 9 to see how the datasets were split.

## Datasets

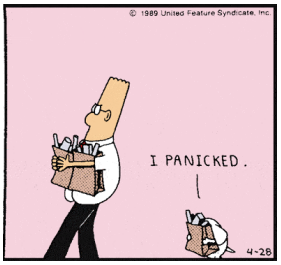
The three chosen image classification datasets are MNIST [44], Characters, and CIFAR-10 [46]. MNIST is a Modified version of the NIST (National Institute of Standards of Technology [47]) datasets, which is a collection of images of handwritten digits. Characters is a subset of the standard OCR dataset [45], consisting of a collection of letters from the English alphabet. CIFAR-10, as opposed to the other two datasets, is a collection of text-less images. The classes are the following 10: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. For all experiments, the evaluation pipeline compares the models on their classification accuracy, calculated as shown below.

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} * 100$$

Flickr8k [41] is a small scale image dataset, which offers captions for each image. This dataset does not contain text within the images, therefore we use it as a benchmark against traditional CLIP. We used this dataset for image retrieval.

The most interesting dataset is the Dilbert comics set. This was a custom image retrieval dataset we created with the help of another research project by Jordi van Setten et al. The images are drawn by comic illustrations, and split up in the panels contained within them. The description of the image was then manually labeled, with the intent of not only capturing a general explanation, but also the dialogue between characters [Fig. 9]. This is the dataset which the model was thought for.

All image retrieval tasks were evaluated using the Top-1 accuracy metric, meaning that when searching for an image with its related query, the correct image should be returned. This is because the query and the image are expected to have the highest similarity score. This metric is equivalent to the classification accuracy metric.




Caption 1: "I panicked"

Caption 2: "dogbert tells dilbert he panicked, while holding grocery bags"

Figure 9: Example image and corresponding captions for the Dilbert dataset

## Results

The results were gathered by running three zero-shot classification experiments, two zero-shot image retrieval experiments, and one image classification finetuning experiment. One PyTesseract (configuration dependent on the task) and two TrOCR models were tested, along with three CLIP models with varying image encoders. Tables 1 through 6 elicit the results, where cells intersecting both an OCR and ViT model indicate an OSBC aggregation model. The results are color coded: OSBC models that underperfom their contained CLIP pipeline are in red, OSBC models that outperform CLIP but underperform their contained OCR-SBERT are in blue, and OSBC models that outperform both are in green.

### Zero-Shot Image Classification

| | ViT-B/16 | ViT-B/32 | ViT-L/14 | - |
|---|---|---|---|---|
| TrOCR Printed | 77.55 | 75.84 | **79.92** | 78.86 |
| TrOCR Handwritten | 54.89 | 54.45 | 70.16 | 49.97 |
| PyTesseract 'psm 10' | 36.45 | 31.87 | 66.06 | 27.92 |
| - | 29.51 | 24.37 | 71.02 | - |

Table 1: Zero-shot image classification accuracy on MNIST

| | ViT-B/16 | ViT-B/32 | ViT-L/14 | - |
|---|---|---|---|---|
| TrOCR Printed | 94.642 | 94.642 | 94.505 | 90.521 |
| TrOCR Handwritten | 82.554 | 82.417 | 82.829 | 70.604 |
| PyTesseract 'psm 10' | 58.379 | 58.379 | 60.164 | 32.829 |
| - | 91.620 | 93.269 | **94.780** | - |

Table 2: Zero-shot image classification accuracy on Characters

| | ViT-B/16 | ViT-B/32 | ViT-L/14 | - |
|---|---|---|---|---|
| TrOCR Printed | 90.44 | 89.11 | 95.43 | 9.18 |
| TrOCR Handwritten | 87.29 | 86.00 | 92.05 | 9.44 |
| PyTesseract 'psm 6' | 90.13 | 88.82 | 95.11 | 8.71 |
| - | 90.45 | 89.12 | **95.44** | - |

Table 3: Zero-shot image classification accuracy on CIFAR-10

Tables 1, 2, and 3 display the accuracy of the pipelines on image classification for the MNIST, Characters, and CIFAR-10 datasets.

For MNIST, the most drastic discrepancy is between the ViT-B/16 and ViT-B/32 based CLIP models, and the TrOCR Printed based OCR-SBERT pipeline. Combining the two results in an OSBC configuration that boosts these CLIP models from 29.51% and 24.37% accuracy to 77.55% and 75.84% accuracy respectively. Two OSBC configurations damage CLIP, while the remaining five outperform both composing pipelines, one of them achieving the highest accuracy of 79.92%.

The Characters dataset also shows improvement for the two simpler CLIP versions when combined with the more accurate OCR-SBERT pipeline. However, both the PyTesseract and the TrOCR Handwritten models highly affect OSBC's performance and disrupt CLIP's high accuracy. The highest score is achieved by the ViT-L/14 image encoder based CLIP model.

As a textless dataset, we did not expect OSBC to outperform CLIP on CIFAR-10. Scores of 9.18%, 9.44%, and 8.71% show that the OCR-SBERT pipelines underperform random guessing (accuracy of 10%, as there are 10 classes equally represented). The TrOCR Printed and PyTesseract based OSBC models perform slightly worse than their CLIP counterpart, while the Handwritten model is quite impactful (drops of 3% in accuracy).

**Zero-Shot Image Retrieval**

| | ViT-B/16 | ViT-B/32 | ViT-L/14 | - |
|---|---|---|---|---|
| TrOCR Printed | 42.306 | 37.658 | 48.833 | 0.002 |
| TrOCR Handwritten | 40.704 | 36.412 | 48.229 | 0.007 |
| PyTesseract 'psm 6' | 42.365 | 37.717 | 48.793 | 1.087 |
| - | 42.494 | 37.865 | **48.872** | - |

Table 4: Zero-shot image retrieval accuracy on Flickr8k

| | ViT-B/16 | ViT-B/32 | ViT-L/14 | - |
|---|---|---|---|---|
| TrOCR Printed | 55.940 | 47.029 | **78.217** | 4.950 |
| TrOCR Handwritten | 56.930 | 47.524 | **78.217** | 0.0 |
| PyTesseract 'psm 6' | 66.831 | 62.871 | 75.247 | 65.346 |
| - | 57.425 | 48.019 | **78.217** | - |

Table 5: Zero-shot image retrieval accuracy on Dilbert

Tables 4 and 5 show the results for image retrieval on the Flickr8k and Dilbert datasets.

On the Flickr8k dataset, OSBC consistently underperforms CLIP. Due to the dataset being textless, both TrOCR-SBERT pipelines have an accuracy close to 0%, and PyTesseract-SBERT performs slightly better, a little above 1%. While both the PyTesseract and the TrOCR Printed based OSBC models are very close to the compared CLIP model, the Handwritten based OSBC model has more disruptive artefacts.

The Dilbert dataset instead provides text in its images. Though both the TrOCR based OCR-SBERT pipelines struggle on this data (scores of 4.95% and 0%), the PyTesseract based models are more effective. In fact, PyTesseract based OSBC outperforms the two simpler CLIP models it is composed of. Still, the largest CLIP model achieves the highest accuracy of 78.217%, which is matched by the TrOCR based OSBC models as the OCR-SBERT pipelines do not provide valuable information, and are therefore ignored.

**Finetuning CLIP on Characters**

To test finetuning, we trained the three CLIP versions on the Characters training data. Table 6 illustrates the results. The finetuned models achieve staggering accuracies of 99.175% (ViT-B/16 CLIP model), 99.862% (ViT-B/32 CLIP model), and 100% (ViT-L/14 CLIP model). The TrOCR Printed based OCR-SBERT pipeline was not finetuned, and proved disruptive when combined through OSBC. However, the overall accuracy of OSBC still improved by a significant margin (on average a 2.61% increase). Though 100% accuracy is somewhat strange, we believe that a human could easily achieve a perfect score due to the simplicity of the task.

## 5 Discussion

This section attempts to satisfyingly answer the previously posed research questions, in light of the results obtained.

### On Zero-Shot Performance

It is clear that in any scenario involving textless data, OSBC is only bound to match CLIP at best. In the majority of experiments, the OCR-SBERT pipeline seems to be combined in a way that is disruptive to the whole architecture. We assume that highly misguided text extraction could generate very confident similarity scores that are not removed by the threshold, and skew CLIP's predictions.

The three datasets containing text within images instead have varied performances. The OCR-SBERT pipeline combined with CLIP through OSBC is beneficial in 11 cases, disruptive in 14, and inconsequential in 2. OSBC seems to be a viable choice when both CLIP and the text extraction pipeline have similar accuracies, or if the text extraction pipeline vastly outperforms CLIP. However, we cannot generally conclude that OSBC outperforms CLIP on images containing text. OSBC does often outperform OCR-SBERT as, unfortunately, this often is the bottleneck of the model. However, the results indicate that with a more accurate OCR-SBERT pipeline, or a less naive aggregation method, OSBC could consistently outperform CLIP.

### On Generalizability

The OSBC model does show some resilience to domain shifts, although the correct OCR model has to be chosen in advance, which is not a trivial task. The model was applied to two tasks, and five datasets, proving beneficial in three of them. Although OSBC is theoretically applicable

| | ViT-B/16 | ViT-B/16 - Finetuned | ViT-B/32 | ViT-B/32 - Finetuned | ViT-L/14 | ViT-L/14 - Finetuned | - |
|---|---|---|---|---|---|---|---|
| TrOCR Printed | 94.642 | 96.565 | 94.642 | 97.527 | 94.505 | 97.527 | 90.521 |
| - | 91.620 | 99.175 | 93.269 | 99.862 | 94.780 | **100.00** | **-** |

Table 6: Image classification accuracy on Characters, comparing zero-shot and finetuned versions of CLIP and derived OSBC models.

to many computer vision problems, we cannot conclude that this model is general as the performance highly depends on the choice of OCR model.

### On Finetunability

Through a proof-of-concept finetuning test, OSBC shows that it can be finetuned in parts. The CLIP component of the architecture is successfully tuned, and the overall accuracy of the model increases. We believe the OCR-SBERT pipeline to be finetunable as well, further improving OSBC's scores, and carrying this step out could help bridge the gap to CLIP's finetuned accuracy.

### On Large Architectures

The larger CLIP version clearly outperforms its simpler counterparts. The difference in performance between them is occasionally quite drastic (for example on the MNIST or Dilbert datasets). There is only one combination where OSBC outperforms the ViT-L/14 image encoder based CLIP model, which could indicate that larger CLIP architectures inherently improve in OCR. The OCR-SBERT pipeline is often less accurate than the best CLIP model, meaning that our implementation of the OSBC model is far from ideal. Again, it would be interesting to trial other OCR techniques, though they are out of the scope of this paper.

### On Prompt Robustness

This is an added discussion point, which does not answer a research question, though we believe is still interesting. Throughout our implementation, we found that CLIP heavily relies on correct prompt engineering. The results published here are the best scores we could push CLIP to. However, there were often times where a slight change in the prompt caused CLIP to heavily underperform, which on datasets such as Characters re-establish the usefulness of OSBC [Fig. 7]. Here, the OCR-SBERT pipeline makes the overall architecture more robust to prompt engineering changes.

| | CLIP (ViT-L/14) | OSBC (ViT-L/14, TrOCR Printed) |
|---|---|---|
| "an image of the letter _" | 94.780 | 94.505 |
| "an image of the letter: _" | 66.346 | 87.912 |

Table 7: CLIP and OSBC's performance on Characters if the prompt engineering step formatted the classes with an extra ":". CLIP drops by 28% in accuracy, while OSBC drops by 7%. The OCR-SBERT pipeline is not affected by this change.

## 6 Responsible Research

This research project was carried out with reproducibility as a priority. To evaluate the transparency of this work, we firstly interpret the project data through the FAIR (Findable, Accessible, Interoperable, and Reusable) principles. Subsequently, the code, documentation and long-term reproducibility are evaluated. To conclude, we interpret OSBC in a larger ethical context.

### Is This FAIR?

The FAIR principles are guidelines to assure the data is findable, accessible, interoperable, and reusable, so that external users can more easily reproduce the paper.

The MNIST, Characters, CIFAR-10, and Flickr8k datasets are all publicly available, ensuring their findability and accessibility. The Dilbert dataset, on the other hand, is not yet easily findable, as this was a custom dataset we built during this research. To aid this, we will provide this dataset upon request.

The integration of the data within the workflow has been explained in the sections above. In addition, we provide in-depth descriptions on how to gather and store the datasets in our repository [48]. Here, we also render available our code, which also explains the preprocessing steps applied before each training and testing pipeline.

### Code, Documentation, and Long-Term Reproducibility

All the code behind the experiments is available online [48]. In this repository not only do we describe how to gather the data, but we also document all the steps in the architecture. Information on the training pipelines is also available online [49], where we provide code screenshots, model configurations, evaluation metrics, and information on the system that ran the experiment (operating system, GPU, CPU, and other hardware descriptions). Though we are occasionally running computationally large models, we focused on optimizing the code to keep the required compute low. This should also help make the experiments reproducible on less powerful machines. Still, we believe this implementation of OSBC is far from perfect, both in accuracy, and in efficiency. Further work needs to be invested to optimize the architecture, for example starting with a more efficient OCR method.

Though we can attest to how our code will change in the future, we do not have control over the models we used for our implementation. Huggingface versions of CLIP, TrOCR, and SBERT may be modified in the coming years, and require changes in the OSBC model. We also provide a snapshot of all the used libraries, so that any user can recreate our environment. We expect these libraries to develop over time, possibly rendering our model's requirements obsolete.

### Ethical Implications

This paper relies on using a combination of models offered online. Though we assume their integrity, they are still black-box components, which renders them vulnerable to ethical

misguidance. It is hard to tell to what extent we can trust these systems when dealing with sensitive topics such as social biases and fairness. Radford et al. offer a detailed analysis of their model by evaluating it on the FairFaces [50] dataset, which deals with race, gender, and age classification.

Open-source models are also targetable by malicious users, which could pose a cyber security threat, by means of direct manipulation or by calculated modification of the data they process.

Machine learning hold great power in today's society, and we realize that without transparent models it is complicated to guide their development correctly. As such, we see OSBC as a black-box, vulnerable model, which must be used with care and only in specific settings.

## 7 Conclusion and Future Works

This paper introduced a new machine learning model: OSBC, which aims to extract text within images as an added feature in computer vision tasks. The model aggregates the similarity vectors returned by CLIP and by a novel pipeline: OCR-SBERT. This pipeline is composed of either a TrOCR or PyTesseract model for text extraction, and an SBERT model for text embedding. We compare the model to various versions of CLIP on image classification and retrieval datasets. OSBC occasionally vastly outperformed CLIP, especially smaller CLIP architectures, but more often slightly underperformed it. The tests showed that OSBC is highly dependent on the OCR model selection, resulting in a loss of generalizability. On the other hand, OSBC was successfully partly finetuned, and showed far more resilience than CLIP on prompt engineering. We believe that given a more general OCR method, and a more stable overall architecture, OSBC could consistently outperform CLIP on images containing text, and match CLIP when the data is textless.

Future work includes evaluating an OSBC model based on a different OCR method (which could also reduce its runtime overhead), evaluating OSBC on a more complex dataset which better fits the intricacies of the architecture, replacing the SBERT model with another text encoder (for example, CLIP's text encoder), or exploring another aggregation technique to ensemble the two similarity vectors. We believe zero-shot models can be supported in context specific domains, if the model supporting them is sufficiently informative.

## References


[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[3] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[4] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[5] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[6] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773. If you use this software, please cite it as below.

[7] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2022.

[8] An Yang, Junshu Pan, Junyang Lin, Rui Men, Yichang Zhang, Jingren Zhou, and Chang Zhou. Chinese clip: Contrastive vision-language pretraining in chinese. *arXiv preprint arXiv:2211.01335*, 2022.

[9] Xiang An, Jiankang Deng, Kaicheng Yang, Jaiwei Li, Ziyong Feng, Jia Guo, Jing Yang, and Tongliang Liu. Unicom: Universal and compact representation learning for image retrieval. *arXiv preprint arXiv:2304.05884*, 2023.

[10] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Conditioned and composed image retrieval combining and partially fine-tuning clip-based features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4959–4968, 2022.

[11] Wikipedia. Dilbert — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Dilbert&oldid=1159915438, 2023. [Online; accessed 25-June-2023].

[12] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[13] Google. Vision ai. https://cloud.google.com/vision, 2023.

[14] Jaded AI. Easyocr. https://github.com/JaidedAI/EasyOCR, 2023.

[15] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[21] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In *Aaai*, volume 2, pages 830–835, 2008.

[22] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. *Advances in neural information processing systems*, 22, 2009.

[23] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE conference on computer vision and pattern recognition*, pages 951–958. IEEE, 2009.

[24] Yasuhide Mori, Hironobu Takahashi, and Ryuichi Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *First international workshop on multimedia intelligent storage and retrieval management*, pages 1–9. Citeseer, 1999.

[25] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Learning visual representations using images with captions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[26] Ang Li, Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Learning visual n-grams from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4183–4192, 2017.

[27] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11162–11173, 2021.

[28] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D Manning, and Curtis P Langlotz. Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pages 2–25. PMLR, 2022.

[29] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

[30] Haoyu Song, Li Dong, Wei-Nan Zhang, Ting Liu, and Furu Wei. Clip models are few-shot learners: Empirical studies on vqa and visual entailment. *arXiv preprint arXiv:2203.07190*, 2022.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[32] Thomas M Breuel. The ocropus open source ocr system. In *Document recognition and retrieval XV*, volume 6815, pages 120–134. SPIE, 2008.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Zbigniew Wojna, Alexander N Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. Attention-based extraction of structured information from street view imagery. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 844–850. IEEE, 2017.

[35] Hugging Face. Hugging face. https://huggingface.co/, 2023.

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[37] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] Matthias Lee Lars Kistner Ryan Mitchell Emilio Cecchini John Hagen Darius Morawiec Eddie Bedada Uğurcan Akyüz Samuel Hoffstaetter, Juarez Bochi. Pytesseract. https://github.com/madmaze/pytesseract, 2023.

[39] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[40] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with*

*the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[41] Micah Hodosh, Peter Young, and Julia Hockenmaier. Flickr8k dataset.

[42] huggingface. Visiontextdualencoder. https://github.com/huggingface/transformers/tree/main/examples/pytorch/contrastive-image-text, 2023.

[43] Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/,. Software available from wandb.com.

[44] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[45] Abhishek Jaiswal. standard ocr dataset. https://www.kaggle.com/datasets/preatcher/standard-ocr-dataset, 2021.

[46] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[47] National Institute of Standards and Technology. Security requirements for cryptographic modules. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 140-2, Change Notice 2 December 03, 2002, U.S. Department of Commerce, Washington, D.C., 2001.

[48] Jordan Sassoon. Osbc github repository. https://github.com/jordisassoon/OSBC, 2023.

[49] Jordan Sassoon. Osbc weights and biases logs. https://wandb.ai/jordisassoon/huggingface?workspace=user-jordisassoon, 2023.

[50] Kimmo Kärkkäinen and Jungseock Joo. Fairface: Face attribute dataset for balanced race, gender, and age. *arXiv preprint arXiv:1908.04913*, 2019.

# Appendix

## Datasets

| | Trainset Size | Testset Size | Number of Classes |
|---|---|---|---|
| MNIST | 50’000 | 10’000 | 10 |
| Characters | 14’898 | 728 | 26 |
| CIFAR-10 | 50’000 | 10’000 | 10 |

Table 8: Image classification dataset sizes.

| | Trainset Queries | Trainset Images | Testset Queries | Testset Images |
|---|---|---|---|---|
| Dilbert | - | - | 202 | 101 |
| Flickr8k | 30’336 | 6066 | 10’112 | 2022 |

Table 9: Image retrieval dataset sizes.

## Finetuning

| | Number of Trainable Parameters | Learning Rate | Batch Size | Epochs |
|---|---|---|---|---|
| CLIP(openai/clip-vit-base-patch16) | 149,620,737 | 3e-5 | 32 | 3 |
| CLIP(openai/clip-vit-base-patch32) | 151,277,313 | 5e-5 | 64 | 3 |
| CLIP(openai/clip-vit-large-patch14) | 427,616,513 | 5e-6 | 4 | 3 |

Table 10: Finetuning parameters and model information on Characters. For a full training suite overview, see [43].

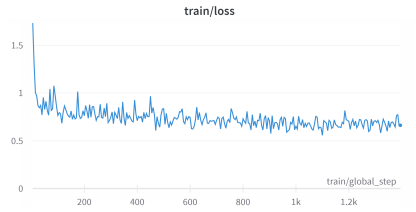



Figure 10: CLIP ViT-B/16 learning curve on Characters


Figure 11: CLIP ViT-B/32 learning curve on Characters

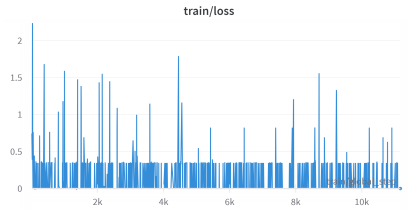



Figure 12: CLIP ViT-L/14 learning curve on Characters