CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Multi-Modal Learning for Threat Analysis

A study on fine-tuning CLIP for a specific domain and how to use it in a classification setup

Master's thesis in Computer science and engineering

Kajsa Andreasson, Ria Dass Raj

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Multi-Modal Learning for Threat Analysis

A study on fine-tuning CLIP for a specific domain and how to use it in a classification setup

Kajsa Andreasson, Ria Dass Raj

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Multi-Modal Learning for Threat Analysis
A study on fine-tuning CLIP for a specific domain and how to use it in a classification setup
Kajsa Andreasson, Ria Dass Raj

Supervisor: Tobias Norlund, *Computer Science and Engineering*
Advisors: Aron Lagerberg, Mats Kvarnström, *Recorded Future*
Examiner: Richard Johansson, *Computer Science and Engineering*

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg

Typeset in LaTeX
Gothenburg, Sweden 2022

Multi-Modal Learning for Threat Analysis
A study on fine-tuning CLIP for a specific domain and how to use it in a classification setup

Kajsa Andreasson, Ria Dass Raj
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In recent years, the area of multi-modality has gained immense interest in computer vision, where it has showed to be powerful for the purpose of letting models learn visual concepts from raw text instead of from manual annotations. One specific model using this concept is CLIP [1], which has shown state-of-the art performance on general zero-shot image classification tasks. However, few works have explored how competitive CLIP is in specialized tasks. To fill this gap, this report explores whether a CLIP model can be successfully adapted to the domain of security intelligence using threat associated data collected from social media, while using the same training task as in the original article. In addition, we explore how CLIP's Image Text Alignment abilities can be used for multi-modal event classification. We present a novel approach to using CLIP's zero-shot capabilities for event classification, in addition to a traditional, supervised approach where CLIP is used for feature extraction. Our fine-tuned model and the pre-trained CLIP model are used side-by-side for both approaches to compare performance.

Our results show that CLIP can be successfully fine-tuned on social media data where its zero-shot image-caption matching abilities are improved with 2%. We furthermore show that our novel approach achieves an AUC-score of 22% and the traditional approach 74%, which leads to the conclusion that using CLIP's innate zero shot capabilities for event classification requires far more work to be competitive compared to a traditional approach. Finally, we conclude that our fine-tuning does not affect the performance in the event classification setup.

Keywords: Multimodality, ITA-Models, CLIP, Event Detection, Fine-tuning, Continued Training, Classification

Acknowledgements

First of all, we would like to give a big thanks to our academic supervisor Tobias Norlund who always, with an enthusiastic and encouraging outlook, has given constructive feedback and thorough help. Aron Lagerberg and Mats Kvarnström, thank you both for your insightful discussions and always providing new ideas we could explore. Anders Hansson, thank you for your willingness to contribute with your time, energy and feedback. A big thanks to Recorded Future, and everyone therein, for being the ones providing us with the possibility to conduct this research. We would also like to thank our examiner Richard Johansson for constructive feedback, quick responses to all of our queries and of course for taking on this project to examine.

Thank you to OpenAI for providing open source code, and a well documented and maintained code, and to AI Sweden for providing computational resources that enabled the heavy computations in this project.

Of course, without the support from our friends and families, we would never have been where we are, so thank you for always being there.

Finally, we would like to thank Chalmers University of Technology for everything these five years have had to offer. Thank you for giving us the knowledge we have been able to use to carry out this research investigation.

Kajsa and Ria, Gothenburg, July 2022

Contents

List of Figures

List of Tables

1

Introduction

For many years, Machine Learning modelling focused on using a single data type to create systems that could learn to understand patterns in the data and then use that understanding for various tasks, such as classification. There are many examples of models that use only one data type, also known as modality – image classification, numerical regression models, Generative Adversarial Neural Networks (GANs), text analysis models such as BERT [4] and audio classification models to mention a few. More recently, the task of creating AI systems with a deeper understanding of its surroundings and the world has gained interest – essentially with the purpose of replicating human intelligence captured within a computer. As we all know, but might be more or less aware of, our outlook of the world does not only consider one type of input at a time, e.g. sounds, visions, emotions etc, but instead considers all of the available inputs at once. One step towards replicating human intelligence in computers is to mathematically recreate the human mind's complex way of correlating everything it perceives, which has opened up a new field of so called *Multi-Modal* ML modelling.

Multi-modality refers to the interplay between different representational modes, for instance, between images and written/spoken word. One example of how humans subconsciously use the multi-modal interplay is when understanding the meaning of a sarcastic meme that has the text *"Look at all the people who love you"* along with an image of an empty desert. The way the post is perceived and understood changes completely when the image is added to the otherwise seemingly sweet text. When both the text and image are perceived together, they display a message of sarcasm or even bullying. This sarcastic message would be impossible to understand without seeing the image. The idea behind Multi-Modal modelling is hence to try to give models this type of deeper understanding by relating different modalities to each other.

Multiple works have introduced ways of using Multi-Modality to build models that can process and relate information from separate modalities [5], such as [6], who create shared representations between image, text and sound shown to be useful for several tasks such as cross-modal retrieval or transferring classifiers between modalities. Another example is [7] who learns words from pairs of short video-clips and sentences, which could be used for e.g. automatically generating descriptions of new videos.

Much of the effort spent in the multi-modal field has been directed towards exploring how to use one of the modalities as ground truth of a single-modal problem. Traditional, single-modal systems, especially in the field of computer vision, are trained to predict a fixed set of predetermined classes from annotations – a procedure known as supervised learning. This type of training requires large amounts of annotated data to successfully learn to generalize on unseen data. Advancements of Graphical Processing Units (GPUs) and computational resources in recent years has allowed for the development of larger and deeper artificial neural networks, where research has shown that they in many cases outperform their smaller precursors. However, with a larger network, a larger amount of data is also required. In combination with the increased use of supervised learning in the community, the need for manually annotated data has skyrocketed, leading to a strong interest in how to work around the need for annotations. In addition, this traditional approach to supervision is restrictive in nature since it limits usability, which occurs due to its need for additional labeled data in order to learn any other concept, i.e. class, than the ones originally used for training.

Learning about images directly from natural language (NL) has shown to leverage a much broader source of supervision [1], and eliminates the need for manually created annotations. The state-of-the-art Contrastive Language-Image Pretraining (CLIP) model, presented by [1], has shown leading performance when learning joint representations of images along with text, also known as multi-modal representations. At the core of their approach they adopt the idea of learning image perception through supervision from natural language.

CLIP is a so called *pre-trained* Image Text Alignment (ITA) model, which means that it has been trained to align correlated text and image pairs in a single coordinate system. The dataset of text and image pairs that have been used for training has been collected from various sources on the internet, and hence has a large variation of language and visual concepts – in contrast to domain specific datasets used for example in tumor detection or dog breed classification. The purpose of training with a highly varied dataset is to learn a wide range of concepts, providing general knowledge rather than domain specific knowledge.

Due to the wide range of data CLIP has been trained on, there are almost endless possible use cases. The authors of [1] have showed that CLIP performs exceptionally well on high level, general tasks, such as image classification on a wide range of images. However, its performance on complex and specialized tasks in specific domains, such as the medical domain, is quite weak [1]. One common way of using pre-trained models is to leverage their general knowledge in a fine-tuning process. This means that the pre-trained model is used as a starting point in an additional task-specific training, with the purpose of adding domain specific knowledge to the model's base knowledge – a procedure known as *transfer learning*. Hence, an exploration of how CLIP can be affected from domain-specific learning is of interest to actors that are in need of specialized high performing, yet easily accessible, Machine Learning (ML) models. Recognizing this aspect, this project's exploration embarks in investigating the extent to which CLIP is affected by further learning

from text-image pairs collected from social media that are associated to the threat domain.

One example of an industrial actor that has an interest in applying CLIP's general knowledge to a more specific domain is Recorded Future, which is a company that works with cyber and threat intelligence by collecting, processing, analysing and communicating threat and intelligence information. A model like CLIP can be used in many different ways in an industrial setting, for example for the purpose of detecting information about threat related events on the internet such as protests or military activity. For companies within the threat intelligence domain, like Recorded Future, it can be valuable to learn whether CLIP's pre-trained state is useful as it is for tasks relevant to the domain, or if a fine tuning on threat associated data improves its performance.

As mentioned, one specific task in the threat intelligence industry is to detect events on social media as soon as possible or even before they have taken place, either for monitoring purposes or to immediately take the needed actions to avert or reduce the threat. However, the research in the past few years on event extraction and classification has been conducted on images and text *separately*. Because of the natural way that image and text complement each other with information, it gives rise to the question of whether using the modalities together, instead of separately, is valuable. Seeing how a text and image pair often gives a human-reader more context to a situation compared to what can be understood from only the text or only the image, makes us wonder if this might have the same affect on a ML model. As of today, Recorded Future uses a language model to detect threat activity. With the aim to add a new dimension to previous text based event detection, this project will explore whether joint image-text representations can be used to identify events on social media. More specifically, the event detection in this project will be based on text and image representations created by CLIP. This exploration will be conducted using both the pre-trained CLIP as well as a fine-tuned CLIP-model.

1.1 Related Work

As described in the introduction, a lot of research has been conducted on multi-modal models in the last few years. The most groundbreaking studies have used multi-modality for the purpose of creating a self-supervised learning process, where language normally is the modality that acts as supervision. Apart from OpenAI's CLIP, other similar works have been presented that learn about visual concepts from natural language with equally good or even better results, such as [8], Microsoft's T-Bletchley [9] and Google's ALIGN [10]. The common goal for all of these is to show the model a wide enough range of concepts that it can learn to understand additional concepts that haven't been included in the training. CLIP stands out from the rest of the mentioned models since OpenAI has made their pre-trained model publicly available. All three works are described to use separate language and image encoders that are trained to align semantically similar image and text inputs by encoding the in a common feature space. At the time of publi-

cation, T-Bletchley showed state of the art performance on multi-lingual image-text alignment with the ability to understand 94 different languages. When it comes to domain specific image classification, the creators of CLIP show results on domain specific image classification tasks only without fine-tuning the model for that specific dataset. In contrast, the authors of ALIGN reports results only after fine-tuning on specific datasets such as the Oxford Flowers-102 [11]. The authors of T-Bletchley don't report any results at all for domain specific data. The finetuning of ALIGN shows promising results, which indicates that a fine-tuning of CLIP should yield an improvement as well, since they share the same conceptual architecture of one text encoder and one image encoder. The authors of [10] explain that their finetuning process consists of finetuning the encoders on a chosen dataset along with finetuning a single classification layer, similar to the fine tuning carried out in this project. In addition, ALIGN showcases how multi-modal queries can be used for image retrieval – such as inputting an image of a pair of grey shoes, and adding the word "red" to the query. This experiment indeed shows that ALIGN manages to retrieve an image of the same pair of shoes but in red. The merge of the modalities in the query is described to be done with elementwise addition of the two created feature vectors. These interesting results suggests that ITA models can interpret the meaning of two concepts at once when the feature vectors for the two concepts are merged using elementwise addition.

Multiple works have studied single-modal approaches to event detection [12], [13], [14], [15], [16]. A common approach to event detection has, in the past, been based on extracting semantics or trigger words, which is used by [13],[15], [16] and [14]. Similarly, studies have been focused on event detection by extracting concepts from images only [12]. However, all these single-modal approaches fail to make use of the context for which the text or image appears in, since images and text often appear together in news articles or social media posts.

Other works have presented multi-modal approaches to event detection, for example [17] which uses a text and graph-based approach for event detection, or [18] which presents an approach for detection of disasters in text and images in tweets. The approach presented in [18] is based on creating feature embeddings for the text and image using both a vanilla Supervised Multimodal Bitransformer and a ViLBERT model, that have both been finetuned to model the correlation between the text and image by aligning them in a common feature space. They furthermore use the created embeddings by adding the two vectors with elementwise addition, i.e. the same approach as in [10], before sending them through a linear classification layer. [19] presents a similar approach to do multi-modal event detection in news articles using text and images. They propose a new model, Dual Recurrent Multimodal Model (DRMM), that uses BERT for text encoding and ResNet for image encoding. The competetive results presented in these two works suggests that using combined text and image embeddings is a good approach for event detection. Based on these works, our study aims to adopt the same approach as [18] for threat related event detection, but using CLIP as the text and image encoder.

One study that has used CLIP for encoding purposes unrelated to event detection

is [20], which studies multi-modal sentiment analysis on text and images. Their experiments show that CLIP was the superior encoder for images specifically, which is yet another argument to use CLIP for feature generation in an event detection pipeline.

1.2 Aim

The purpose of this project is firstly to assess how fine-tuning a pre-trained CLIP model on data that is associated to the threat domain affects its performance. One could argue that this process should be called *continued pre-training*, since we aim to train CLIP using the same task as in the pre-training, but instead of using general data, which is normally the case for pre-training, we will use domain-specific data. *Fine-tuning* on the other hand normally means that the training is done for a specific task, on domain specific data, with an additional classification layer. Since the training we refer to in this project lies somewhere in between continued pre-training and fine-tuning, we have chosen to use the phrase fine-tuning to emphasise the use of domain-specific data. The performance of the pre-trained, and later on the fine-tuned model, will be evaluated using the same methods as in the original article [1], i.e. to evaluate how well the model can identify the actual matching text and image pairs out of a large set of such. This assessment will be conducted on data sourced from one social media outlet only[1], which differs to the data used in the pre-training. Due to the frequent use of sarcasm, memes and in other ways un-related text to images on social media, the nature of the dataset is challenging. This fact gives rise to the question of whether CLIP's way of learning allows for it to understand complex concepts such as sarcasm, and in turn deepen its understanding, or if it will simply confuse CLIP.

Up until now, there has been very little research done on the topic of adapting and fine-tuning CLIP for a specific domain. We aim to fill that gap by exploring if and how CLIP can be adapted to the threat domain on social media. A successful adaptation could open up for many businesses to create specialized, well-performing models by taking advantage of CLIP's general knowledge and fine-tuning it in a chosen domain – which would be a relatively efficient and cheap process compared to training an equally good model from scratch. Considering the fact that CLIP has showed to perform inadequately on specialist tasks [1], the idea of fine-tuning to improve the performance in such tasks seems even more purposeful.

Lastly, this project will investigate different ways of using the image and text feature vectors created by CLIP for event detection, as a multi-label classification problem. The first approach will use CLIP's built-in ability to find similarities between the text and image pair and the event type labels and make the classification based on the calculated similarities. The second approach will use an added classifier that takes the embedded text and image as input – which essentially is a type of feature learning. Comparing these two approaches to each other, but also to

[1]Because of confidentiality and legal matters, we're not able to explicitly write out the name of the social media outlet.

existing single modal event detection models, can indicate whether CLIP and its feature embeddings can be a good alternative for event detection. Since very few works, if any at all, has previously explored using CLIP's embeddings for event detection, we see a good opportunity to contribute with novel investigations to the community.

In short, the research questions that are to be investigated throughout the project are:

1. Is it possible to successfully fine-tune a general ITA-model within a more specific domain?

2. How can text and image representations from an ITA-model be used for multi-label classification, more specifically for threat classification, and how is the performance affected by the previously mentioned, domain-specific, fine-tuning?

1.3 Limitations

The main limitation for this project has been the computational resources. Many existing ITA models, such as CLIP, consist of millions of trainable parameters, and hence need extreme amounts of computing power to train and test. The authors of CLIP reported that their training took up to 18 days on 592 V100 GPUs [1]. The training in this project will be carried out on 5 NVIDIA A100-SXM4-40GB GPUs. The limitation on computational power has limited our work to only work with smaller datasets compared to the pre-training of CLIP, as well as the number of different trainings we have been able to run to find the optimal training scheme.

1.4 Contribution

This project has shown that CLIP can be fine-tuned towards a specific domain. Furthermore, CLIP has most probably used social media data in their pre-training, making this an inadequate dataset to use when fine-tuning the model. Since the authors of CLIP do not specify what data they use in their pre-training, this project shows the extent to which the model's general knowledge ranges giving indications on what could be a good dataset when fine-tuning the model.

Furthermore, the project has suggested a novel approach in which one may use the numerical representations CLIP generate to conduct domain-specific tasks. The project showed that a traditional approach, in which one fine-tunes a fully connected classification layer, performs better. However, due to the quality of the dataset being questionable, this novel approach is worth to explore further. The code developed and used to conduct this study can be found at our github repo[2] and is open source.

[2]https://github.com/riiaraj/multimodal-learning-for-threat-analysis

2

Theory

This chapter aims to provide the reader with the required information that is needed to understand the concepts mentioned and used in the following chapters. It starts by giving a general introduction to the field of *Natural Language Processing (NLP)* (Section 2.1) and *NLP methods* (Section 2.2) and continues with an explanation of *transformers* and *encoder-decoder models* (Section 2.3), which is some of the technology used in CLIP that has acted the backbone model in this project. The chapter continues with an explanation of the technique used to train CLIP, namely *Constrastive representation learning* (Section 2.4), followed by a more detailed presentation of CLIP to give an understanding of how it works under the hood (Section 2.5). Finally, we end the chapter with presenting the metrics used for evaluation in this project (Section 2.6).

2.1 Natural Language in technology

NLP is a subfield of computer science, artifical intelligence and linguistics. As the name suggests, NLP is concerned with the interactions between computers and human natural language (NL). NL differs from other types of languages, such as computer language, by including only such language that would be written or spoken by a human. The goal for the NLP field is in short to give computers the ability to understand and respond to natural language in the same way as humans [21]. Examples of applications of NLP are programs that do language translation from one language to another such as the famous Google Translate functionality, virtual agents or chat bots such as Apple's Siri and voice recognition such as voice operated GPS systems - just to mention a few. All these NLP use cases can be divided into separate NLP tasks, such as *named entity recognition*, *speech recognition*, *natural language generation*, *sentiment analysis* or *semantic analysis*. This project's NL scope lies mainly within the semantic analysis task, which is an area that can be summarized as identifying the meaning of language and understanding the topic of a text by looking at the context. The semantic analysis carried out in this project is related to both using the contextual information from text to learn visual concepts, as well as using the context extracted from both text and images to detect information about events.

2.2 Natural Language Processing – methods and preprocessing

There is a standard way to approach processing natural written text, which allows the computer to make sense of natural language. The first step is normally to parse the written string by removing parts of the string that won't add any value to the model. This could be emojis, url:s or other special characters. The next step is normally to translate the text into the machine's own vocabulary, called *tokenization*. This means that the text is broken down into smaller pieces that the machine can process, which can be sentences if the text is a long paragraph, or words, parts of words or even individual characters. A common practice is to divide the text by white space, such that each word becomes an individual token. In some cases, the tokenized string has to be *truncated* at a certain length, i.e. cut off at a certain number of tokens, due to some language models having a constraint on how many tokens it can process at a time. It is also common to add one special token at the beginning of each tokenized string, called a *Start-Of-String* (SOS) token, and one *End-Of-String* (EOS) token at the end of the tokenized string. These two tokens allow the language model to better understand the beginning and end of a sentece or a longer text. The tokenized string can then be further processed to create textual features, using various vectorization methods, such as the *term frequency–inverse document frequency* (TF-IDF) or bag-of-words. These methods are not used in this project and will hence not be further explained. The set of unique tokens that are remaining after tokenization make up the so called model *vocabulary*. In practice this means that the model can normally only recognize tokens that are included in its vocabulary.

2.3 Transformers

The transformer architecture was proposed by [2] in 2017. This architecture revolutionized the way in which inputs strings are mapped to output strings i.e. transduction, where machine translation is one common application. Before its coming about, most state-of-the-art sequence transduction models based their architecture on a complex two-stage RNNs or CNNs in a encoder-decoder structure.

Encoder-decoder models can be summarised as a family of models that learn to map data-points from one input domain to an output domain. A classical encoder-decoder model is a two-stage network, which consist of firstly an encoder that compresses, i.e. encodes, the input to a latent feature space representation – hence on be referred to as *embedding*. The network's second part consists of a decoder which aims to predict the output by decompressing, i.e. decoding, the embedding.

The Transformer also has an encoder-decoder structure, however its revolutionizing aspect is that it eschews recurrence and entirely relies on a so called *attention* mechanism, which is a technique used to provide an additional focus on a specific component. This concept is further explained in section 2.3.1. However, what is

worth to notice for now is how this approach creates global dependencies between input and output, while also allowing for significantly more parallellisation as compared to the RNNs and CNNs. As presented in [2], Figure 2.1 shows the architecture of the Transformer. One component of the transformer is the *multi-head attention* sub-layer. This is essentially multiple self-attention (See Section 2.3.1) modules stacked on top of each other. Another component is point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 2.1, respectively.

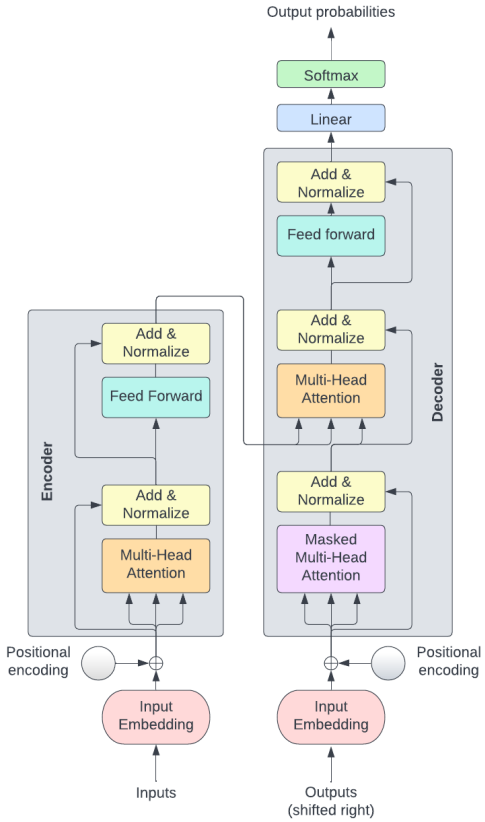


Figure 2.1: A visualization of one of the six layers in the Transformer Architecture, as presented in [2]. The left half represents the encoder and the right half the decoder.

The encoder of the Transformer is composed of a stack of six identical layers, each layer consisting of two sub-layers [2]. As explained in the article, the first sub-layer is a multi-head self-attention mechanism and the second a position-wise fully connected feed-forward network, each with a residual connection, followed by a layer normalization. Meaning, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where Sublayer(x) refers to the function implemented by the specific sub-layer. To facilitate the residual connections, each output carries the same dimension of $d = 512$.

The decoder of the Transformer is also composed of a stack of six identical layers. In addition to the two sub-layers in each encoder layer, the decoder begins with a third sub-layer performing multi-head attention over the output of the encoder

stack. The motivation to this structure is to ensure that the predictions for position i can depend only on the known outputs at positions less than i.

To make sure the model makes use of the order of the sequence, even though it does not contain recurrence or convolution, the transformer adds *positional encodings* to the input embedding at the bottom of the encoder and decoder stacks through a simple summation.

2.3.1 Attention

Attention in neural networks is a technique, mimicking cognitive attention, that enhances some parts of the input data while diminishing other parts [22]. The thought behind it is that the network should devote more focus to important parts of the data similar to e.g. the visual attention mechanism that the human brain uses to focus on aspects of an image with higher resolution and subsequently viewing the surrounding areas with a lower resolution. The attention mechanism evaluates inputs to identify the most important components, and assigns each of them with a weight. Deciding which part of the data is more important than others depends on the context.

There are different types of attention mechanisms, with one of the most common one being self-attention. [2] used self attention, which is the attention mechanism that relates different positions *within* a single sequence, in order to compute a representation of the whole sequence. Self-attention has successfully been used in a variety of tasks, including reading comprehension, textual entailment and learning task-independent sentence representations [2].

2.3.2 Vision Transformer

The Vision Transformer (ViT) architecture was proposed by [3] in 2021. After the Transformer architecture grew to become the de-facto standard for NLP tasks, ViT was created as a result of wanting to apply the Transformer architecture to CV applications with the fewest possible modifications.

Figure 2.2 shows an overview of ViTs architecture. In order to preserve the main structure of the Transformer, ViT preprocesses the input-images to a format compatible with the Transformer's encoder. In practice, ViT splits the input-images into a sequence of patches (allowing for the images to be treated as tokens in an NLP application) that are linearly embedded, and then inputs this sequence of embeddings to the Transformer.

As seen in Figure 2.2, ViT prepends a learnable [class] embedding (much like BERT's [class] token that is used to represent the entire sentence) to the sequence of embedded patches, which serves as the image representation. It also prepends positional embeddings to retain positional information. The positional information in this context refers to one element's (of the whole patch-sequence) position. The resulting sequence of embedding vectors then serves as input to the encoder that shares its

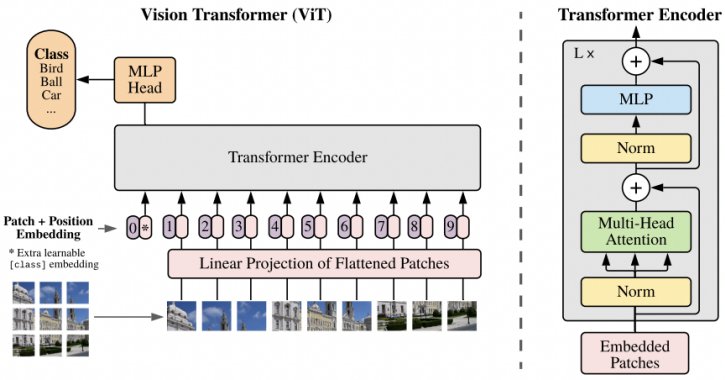


Figure 2.2: Vision Transformer Architecture, as presented in [3][1]. The left half represents the encoder and the right half the decoder.

structure with the original Transformer presented in [2].

2.4 Contrastive representation learning

Representation is an important part of machine learning. To explain *contrastive* representation learning, we first have to explain the concept of representation learning itself. Representation learning consists of extracting relevant features or information from raw data, that allows for later use on downstream tasks such as classification. One straightforward example of representation learning is feature extraction from e-mails to detect spam. The raw e-mail text can be tokenized (see Section 2.2) and then turned into features using a vectorization method, such as TF-IDF or bag-of-words. The vectorization methods result in feature vectors, often based on the frequency of each token, transformed with a given mathematical formula depending on which vectorization method that is used. As an example, the bag-of-words vectorization creates a feature vector where each element is an integer representing the number of occurrences of each token in a given sentence. The vector representation can then be fed into a linear classifier to make a spam/not spam prediction.

Contrastive representation learning is a type of representaion learning with a slightly different objective. Whereas the regular contrastive learning focuses on only extracting relevant information from raw data, the contrastive representation learning aims to extract the relevant data by learning the optimal way to embed samples in a feature space, in which similar sample pairs stay close to each other while dissimilar ones are far apart. Although we find contrastive learning to be applied to both supervised and unsupervised settings, it is one of the most powerful approaches in self-supervised learning.

The contrastive learning loss is defined using *cosine similarity*, which is a measure of similarity between two sequences of numbers. The sequences are viewed as vectors

[1]This image is re-used from the original article [3] with the permission from the authors.

$\vec{A}$ and $\vec{B}$ in a feature space and the cosine similarity is defined as the cosine of the angle between them. That is, the dot product of their vectors divided by the product of their lengths 2.1, meaning the similarity depends not on the magnitudes of the vectors, but on the angle between them.

$$\text{cosine similarity} = \frac{\vec{A} \cdot \vec{B}}{\|A\|\|B\|} = \frac{\sum_{i=0}^{n} A_i B_i}{\sqrt{\sum_{i=0}^{n} A_i^2}\sqrt{\sum_{i=0}^{n} B_i^2}} \tag{2.1}$$

2.5 Contrastive Language–Image Pre-training (CLIP)

The Contrastive Language–Image Pre-training (CLIP) model presented by [1] is a state-of-the art multi-modal model. When the cross-modal retrieval is based in images and text, i.e. retrieving an image given a text query, or captions that successfully label an image, the models are usually referred to as Image-Text Alignment (ITA) models [23]. At the core, CLIP implements an approach to learning perception through representation learning with supervision from NL.

Many state-of-the-art computer vision systems use a predictive approach to learn visual concepts, such as the Noisy Student EfficientNet-L2 that predicts 1000 ImageNet classes [1]. However, CLIP required a whole 33 TPUv3 core-years to be trained. Considering the amounts of computational power this model required during training only to learn a limited amount of visual concepts, training a model to learn an open set of visual concepts from NL requires a refined approach. CLIP overcomes the limitation by swapping the predictive objective for a contrastive objective. This simplifies the task in that, that it predicts only which text *as a whole* is paired with which image, rather than the exact words of that text.

To attain this, CLIP learns a multi-modal embedding space, which essentially is a feature space that is common for the images and texts. Through firstly embedding the image and text of the (image, text) pair in an image and text encoder respectively and secondly jointly train the encoders, this embedding space is created. The training aims to maximise the cosine similarity between the embeddings of the image and text pairs that belong together and minimise the cosine similarity between the pairs that don't belong together, which is done through optimising a symmetric cross entropy loss over the similarity scores.

This means, for a batch of N (image, text) pairs, CLIP is trained to predict which of the NxN possible (image, text) pairings across a batch actually occurred. And to do that, they maximise the cosine similarity of the N real pairs in the batch, and minimise the cosine similarity of the $N^2 - N$ false pairs. A summary of the approach is visualized in 2.3.

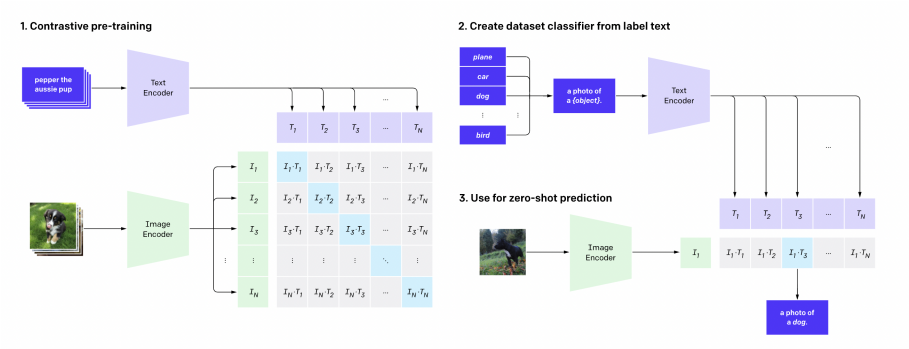


Figure 2.3: A summary of CLIP's approach as described by the original article.

2.5.1 Encoders

CLIP's image encoding component is a closely followed ViT architecture as described in Section 2.3.2, however with an additional layer normalization to the combined patch, and a different initialization scheme.

The text encoding component is the Transformer, with architecture modifications as described in [24]. The base size is a 63M-parameter, 12-layer 512-wide model with 8 attention heads. The text is represented as lower-case byte encodings (BPE), where the max sequence length is capped at 76 (for computational efficiency). The total vocabulary size is 49,152.

2.5.2 Loss

In traditional binary and multi-class classification problems, cross entropy is the most commonly used loss function, which measures the difference between two probability distributions for a given random variable. CLIP however uses a *contrastive loss* function whose goal is to pull together text and images, that belong together (positive-pair), in the feature space, while simultaneously pushing apart (or contrast) text and images that don't belong together (negative-pair). The contrastive loss function is introduced in Equation 2.2,

$$l_{I_i,T_j} = -\log\left(\frac{\exp[\frac{sim(f(I_i),g(T_j))}{\tau}]}{\exp[\frac{sim(f(I_i),g(T_j))}{\tau}] + \sum_{k=1}^{N}{}_{[k\neq j]}\exp[\frac{sim(f(I_i),g(T_k))}{\tau}]}\right) \tag{2.2}$$

where the similarity function sim is the cosine similarity (see Equation 2.1) between the image embedding of image i ($f(I_i)$) and text embedding of text j ($g(T_j)$). τ is a normalization factor also called the temperature parameter.

To minimise the loss the numerator must be maximized while the denominator must be minimized, which is done by ensuring that the cosine similarity for negative-pairs is as small as possible and that of positive-pairs is as large as possible.

Since we know which text and images that belong together, the contrastive learning

carried out when training CLIP is in practice supervised. This allows for implementing the contrastive loss using loss functions that are normally used for supervised problems, such as the Cross Entropy Loss. The remainder of this section will hence consist of a technical description of how contrastive loss can be implemented using Cross Entropy Loss.

The formula for cross entropy looks as follows:

$$H(t,p) = -\frac{1}{N}\sum_{k=1}^{N} t_k log(p_k) \tag{2.3}$$

where t_k = target for sample k, p_k = prediction for sample k and N = batch size.

The classification task used for training CLIP is a multi-class problem. In such multi-class cases, the cross-entropy calculations of the torch library expects a logits-matrix (prediction) as input (showed in equation 2.4), paired with an array of class-indices (target). The logits-matrix in this case is a symmetric matrix where each element contains a logit, in our case *similarity*, for each possible text and image pair in the batch. High similarity naturally means that the text and image are likely to belong together, while low similarity means that the text and images are likely to not belong together. Using both the logits-matrix and the target vector, it's possible to calculate the cross-entropy-loss over *each row* in the logits-matrix (see Equation 2.4), with the correct target class index. The correct target class index in our case is each diagonal element (since this element represents the image-text pair), visualized in figure 2.4. Therefore it is sufficient to represent the correct target for each row using a range vector $[0, 1, 2, ..., N-1]$, where N = batch size.

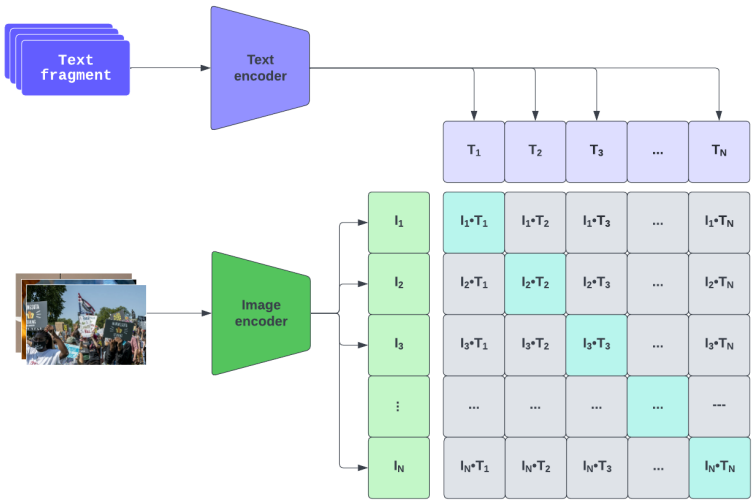


Figure 2.4: A visualization of how the prediction matrix, or logits matrix, is calculated in the training step of CLIP.

$$\vec{a} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & ... & a_{0,N} \\ a_{1,0} & a_{1,1} & a_{1,2} & ... & a_{1,N} \\ ... & & ... & & ... \\ ... & & ... & & ... \\ a_{N,0} & a_{N,1} & a_{N,2} & ... & a_{N,N} \end{bmatrix} \tag{2.4}$$

Equation 2.4 shows the logits array where

$$a_{i,j} = \frac{f(I_i) \cdot g(T_j)}{\|f(I_i)\| \|g(T_j)\|} \tag{2.5}$$

The logits matrix corresponds to the predictions in terms of how likely it is that each image and text pair correctly belong together. In torch, the predictions are first translated to a probability, row-by-row, using the softmax function as presented in Equation 2.6 (which shows an example when calculating the softmax for row number 0). This gives a "prediction-matrix", as shown in Equation 2.7

$$\vec{p}_{0,i} = \frac{\Big[\exp[a_{0,0}], \exp[a_{0,1}], \exp[a_{0,2}], ..., \exp[a_{0,N}]\Big]}{\sum_{j=1}^{N} \exp[a_{0,j}]} \tag{2.6}$$

where $a_{0,0}$ to $a_{N,N}$ are entries in a row in the logits-matrix and exp is the exponential function.

$$\vec{p} = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & ... & p_{0,N} \\ p_{1,0} & p_{1,1} & p_{1,2} & ... & p_{1,N} \\ ... & & ... & & ... \\ ... & & ... & & ... \\ p_{N,0} & p_{N,1} & p_{N,2} & ... & p_{N,N} \end{bmatrix} \tag{2.7}$$

When the predictions are found, the cross-entropy is calculated. Recall that the objective of our loss calculation is to minimise the loss over the diagonal elements (the true pairs, meaning the targets), hence these are the relevant elements to be included in the calculations from each row. In practice, this means that the loss is calculated only for the elements on the diagonal, since all other elements have a target value 0 and are hence not included due to the multiplication of the target value in equation 2.3. That is, when inserting all the elements in Equation 2.3 the resulting loss calculations contain only terms for the diagonal elements and look as follows:

$$
\begin{aligned}
H(t,p) &= -\frac{1}{N}\sum_{k=1}^{N} t_k log(p_k) \\
&= -\frac{1}{N}\left(\log(p_{0,0}) + \log(p_{1,1}) + ... + log(p_{N,N})\right) \\
&= -\frac{1}{N}\left(\log\left(\frac{\exp[a_{0,0}]}{\sum_{k=1}^{N}\exp[a_{0,k}]}\right) + ... + \log\left(\frac{\exp[a_{N,N}]}{\sum_{k=1}^{N}\exp[a_{N,k}]}\right)\right)
\end{aligned} \tag{2.8}
$$

And so we see that Equation 2.8 is in the same form as that of Equation 2.2, showing how CLIP incorporates contrastive learning in their training. To minimize the loss, we have to maximize the fractions in the log expressions in equation 2.8. To do so, the numerator for each term containing the similarity values for the diagonal elements in the prediction matrix have to be maximized while the denominators containing the sum of all the similarity values have to be minimized. This means that the other entries of the matrix, signifying the incorrect pairs, are implicitly addressed for the loss calculations through minimizing the log of the sum. In practice, this means that the similarities for the correct pairs on the diagonal are trained to be as high as possible while the similarities for all the other pairs are trained to be as low as possible.

2.6 Metrics

This section will describe the metrics used to evaluate the different models in this project.

2.6.1 Zeroshot classification

Most regular classification tasks consist of predicting one or several classes for a certain sample out of a number of classes that the model has been trained on. The sample could be a text, image, numerical features etc – for example an image of a dog with the class "dog". The model has previously to the prediction been "showed" (trained on) images of dogs and their class-labels so that it has learned to classify images of dogs with the class "dog". Zero shot classification on the other hand, works by predicting a class (or several classes) for a sample without having seen that particular class before. An example of this could be that a model trained on cats and dogs is given an image of a lion and is then supposed to predict the correct class "lion" out of a few alternative classes "lion", "jaguar", "tiger" for the image. Naturally, this increases the complexity of the classification task and the model itself has to be adjusted to be able to handle zero shot classification.

In practice, the zero shot classification works by taking an input sample and a list of alternative classes that are unfamiliar to a given model. By using it's previous knowledge of classes that are familiar to the model, it can make a prediction of which

classes that are the most probable for the given sample, often based on a similarity value. The zero-shot classification using CLIP in [1] is based on the cosine-similarity values between the sample and all alternative classes.

In practice, the zero-shot classification in CLIP means that we view each text sample as its own class and based on a given input image, CLIP finds the correctly associated "class", i.e. text sample with the highest similarity value, for that image. The zero-shot classification pipeline naturally also works the other way around, where each image is treated as its own class, and the correct "class" (image) is to be predicted for a given text fragment based on the highest similarity value.

2.6.2 Mean Reciprocal Rank

The first evaluating metric used in this work is the Mean Reciprocal Rank (MRR), which is commonly used in ranking tasks. In the context of ranking, the terms *query* and *documents* are commonly used. The *query* is the input to the model, and the *documents* are all the alternatives that should be ranked based on a chosen method with regards to the *query*. For zero-shot image classification, an input image is the *query*. The *documents* are all the available classes that should be ranked, i.e. all text strings the model can "choose" from.

The MRR evaluates the performance by considering the highest rank of one of the relevant documents for each query. The mathematical formula for MRR is

$$\text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\text{rank}_q} \tag{2.9}$$

where Q is the number of queries and rank_i is the highest rank of any of the relevant documents in the predicted ranking list. The MRR will get a final value between 0 and 1, where a higher value means that the model is better at ranking the relevant documents at top positions in the ranking list. In other words, a perfect ranking for each sample would give a MRR of 1.

2.6.3 Mean Average Precision

The second evaluating metric used is the mean Average Precision (mAP) which measures the performance by looking at how many relevant documents are predicted with a high ranking. The more relevant documents at the top of the ranking list, the higher the average precision (AP) value. This metric is similar to the precision-at-k, but instead of a given number k, all documents are considered. The AP is calculated for each query, followed by taking the mean over all queries. The mathematical formula for MAP is:

$$\text{MAP} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \text{AP}(q) \tag{2.10}$$

where Q is the number of queries, i.e. inputted images/fragments (depending on the direction), and AP(q) is the average precision for a query q. The average precision itself is calculated according to the following formula:

$$\mathrm{AP}(q) = \frac{\sum_{k=1}^{N} \mathrm{NP}(k) \times \mathrm{rel}(k)}{\text{number of relevant fragments}} \tag{2.11}$$

where NP(k) is the precision at cut-off k in the list, rel(k) is an indicator function which equals 1 if the item at rank k is a relevant document, and zero otherwise. Similarily as for the MRR, the mAP will get a final value between 0 and 1, where a higher value means that the model is better at ranking the relevant documents at top positions in the ranking list. A perfect ranking for each sample would give a mAP of 1.

3

Method

This project investigates CLIP's zero-shot classification abilities within the threat intelligence domain along with if and how CLIP's feature embeddings can be used for event detection. As earlier stated, the research questions that are to be investigated in this project are:

1. Is it possible to successfully fine-tune a general ITA-model within a more specific domain?

2. How can text and image representations from an ITA-model be used for multi-label classification, more specifically for threat classification, and how is the performance affected by the previously mentioned, domain-specific, fine-tuning?

The research to investigate these questions has been divided into two parts: the first one called the *fine-tuning*, where we explore how to successfully conduct a fine-tuning of CLIP on domain specific data and how such a fine-tuning affect the performance. The second part is called *multi-label event classification*, where we set up two different pipelines to make use of the text and image embeddings created by CLIP. For these two parts, two separate datasets have been collected.

In this chapter the research through which these investigations took place are described, along with an examination of the datasets used, pre-processing methods and a detailed explanation of the experiments conducted.

3.1 Dataset

Two main datasets have been used for this project:

- Social media sourced text- and image pairs, denoted as *RFSM dataset*.

- Text and image pairs, labeled with 3 event types through Google Vision, denoted as *Google Vision dataset*.

3.1.1 RFSM dataset

The first dataset consists of 54 million text and image pairs originating from social media posts. Because of confidentiality and legal matters, we're not able to explicitly write out the name of the social media outlet. The dataset has been provided by Recorded Future, and they have sourced data relevant to their domain of threat intelligence giving its threat association. The data was collected continuously starting from September 1st 2021 throughout December 18th 2021. Due to the nature of social media and how images often are re-posted, the same image can occur in many different posts. This is true also for our dataset, where one image can occur in multiple samples with different fragments.

Those who are familiar with modern social media outlets know that there is a high frequency of so called *memes* as well as screenshots with computer typed text. It is also quite common that the text is contradictory or unrelated to the image, due to for example irony or sarcasm. With this in mind, there is a possibility that CLIP might perform worse on our own data due to the possible lack of correlation between the text and images. To give an idea of what the data looks like in this dataset, a few samples are presented in Table 3.1.

Image	Fragment
	"Vienna Protest Against Tyrannical Vaccine Passports and Mandatory Vaccinations."
	"This is your Classic VICTIM mentality, which has become very popular in our current society and judicial system. I'm SMH as someone who believes in consequences that match the behavior/crime..."
	"#syria #raqqah Al-Khabour: Two members of the regime forces were killed in an armed attack targeting a military vehicle on the road between the cities of Maskana and Debsi Afnan, southwest of Raqqah."

Table 3.1: Three examples of what the data looks like in the RFSM dataset.

3.1.2 Google Vision dataset

The second dataset has been collected by Recorded Future from all over the web and consists of 363 096 text and image pairs. The images in these pairs are annotated using Google Vision and the text fragments are annotated with Recorded Future's internal rule-based event annotator. Each sample can have 0 or more labels. These labels are of different nature but those that are relevant in the case of this project relate to threat oriented activity. Hence, the labels from this dataset that are used are **Manmade Disaster, Protest, Military Event, Tank**. We make the assumption that the text and image in each pair are related to each other and hence describe the same concept. In other words, in the case of where only the image has a label, we consider that label to be valid for the text as well. The same goes for the opposite case, when only the text has a label. For the case of when the text and image have different labels, we assign both labels to the sample. To give an idea of what the data looks like in this dataset, a few samples are presented in Table 3.2.

Class	Image	Fragment
Manmade disaster		"The Ukrainian emergency service said it had put out 24 fires in and around Kharkiv caused by shelling, and it had disabled 69 explosive devices."
Military		"A Ukrainian soldier directs a Russian tank that Ukrainians captured after fighting with Russian troops, as Russia's attack on Ukraine continues, outside Brovary, near Kyiv, Ukraine, March 10, 2022."
Protest		"GENEVA - The U.N. human rights office called Tuesday for the release of all peaceful protesters who were arrested after taking part in Russia in demonstrations protesting the war in Ukraine."

Table 3.2: One example of an image and text pair per event category from the Google Vision dataset.

A so called *Manmade Disaster* is an event where a disaster has occured as a consequence of human actions, such as non-natural fires or explosions, and the events

following after such. *Military* on the other hand, is any event that includes military vehichles, such as tanks or military aircrafts, or military persons. Any text related to *invasions* or other war related vocabulary will also fall into this category. *Protest* is a bit more straightforward, and will include text and images of both protest and demonstrations, usually showing crowds with poster and flags.

After pre-processing the data, 55833 relevant samples are retrieved, out of which 45220 are negative samples. The distribution of the labels is reflected in Table 3.3. Note that each sample can have more than 1 label, hence will the percentages add up to a value larger than 100.

	Number of samples	%
Manmade disaster	5745	10.29
Military	12761	22.86
Protest	6187	11.08
No label	32 694	58.56

Table 3.3: Data distribution of Google Vision dataset for each event label. Each sample can have more than one label.

	Number of multi-label samples
Manmade disaster and Military	830
Manmade disaster and Protest	206
Military and Protest	524
Manmade disaster, Military and Protest	6

Table 3.4: Number of samples having multiple labels.

The dataset was also divided in a training, validation, and test set consisting of 70, 10 and 20 percent of the data respectively, where the label distribution reflects its distribution as in the whole dataset.

3.2 Data pre-processing

A few pre-processing steps were taken to prepare the data for being used to train and evaluate the CLIP-model.

For the RFSM dataset, the first step of pre-processing the textual data was to remove user tags in the post text fragments, for example @*username*, since the user names themselves aren't part of the natural language. Secondly, all url:s were removed from the text, for example *https://www.examplewebsite.com*. The text fragments that, after this step, had empty text strings were also removed from the dataset. Like most NLP architectures, CLIP can't send raw text through its text encoder: first each text fragment has to be truncated and tokenized, which is done using the tokenizer developed along with CLIP by [1]. The tokenization process truncates all

text fragments at 77 tokens so that the remaining tokens are removed, and then finally returns a tensor of 77 elements with the tokenized representation of each inputted fragment.

The same steps were taken even for the Google Vision dataset, however with an additional step included. Since this dataset was used with the aim to evaluate CLIP's event detection abilities, the labels had to reflect events. As described in Section 3.1.2, the labels from the dataset that were used were *Manmade Disaster, Protest, Military Event* and *Tank*. However, it is only the three first labels that reflect occurrence of an event. Seeing that tanks are almost exclusively found in military contexts, the *Tank*-labels were mapped into the *Military Event* category.

The image pre-processing was conducted in the exact same way for both datasets. This was done with CLIP's own image preprocess method, which is a torchvision transformer that converts an image into a tensor. Simply put, the pre-processing resizes, crops and normalizes the image into a $3 \times 224 \times 224$ tensor (where the first dimension is the RGB-scale).

3.3 Experiments

The backbone of this project was, not only the architecture used by [1], but the model itself along with its pretrained weights. To answer the research questions (see Section 1.2), the project set sail in **two** streams. The **first** one was to use CLIP for zero-shot transfer and then fine-tuning CLIP (a continued training of the original pre-trained CLIP)[1] to see how it is affected from domain-specific learning. For this stream, the RFSM dataset was used for training and evaluation, see section 3.1.1. The training was conducted with different learning rates along with different learning rate schedules. To evaluate the performance before and after fine-tuning, ranking metrics such as mAP and mRR were used.

The **second** stream of the project regarded seeing how and if CLIP's image-text representations can be used for multi-label classifications. This was done by comparing two approaches. The first approach evaluated CLIP itself as a multi-label classifier in a similarity based classification. The second approach evaluates CLIP extended with a fine-tuned linear classification layer. This chapter explains in detail how these experiments were conducted.

[1]To highlight our reasoning behind the naming of this stream, we give further explaintion here. Due to the fast advancements in this field of pre-trained models, terminology is lacking a bit behind. To note is that this experiment is not fine-tuning in its traditional sense, where one add a classification layer to a model, and *fine-tunes* it for a specific task. This combined with the fact that our training objective is the same as the original one, one could argue that the experiment is one of simply a *continued training* of CLIP. But, since the data used for training is domain specific, it is not completely true to call this experiment a continued training either. Hence, note that our experiment is a middle way approach to traditional fine-tuning and continued pretraining, but to underline our aim we have chosen to denote this experiment as *fine-tuning CLIP*, which is how we refer to it throughout the report.

3.3.1 Using CLIP for Zero-shot transfer

As earlier explained, CLIP has been trained to find matching text-image pairs from a large batch of alternatives, and it could be argued that it carries an innate zero-shot classification set up. To see how well CLIP performed on zero-shot classification on threat associated social media-sourced data, an evaluation pipeline was built. The evaluation was done for the two use cases explained above, meaning 1) Zero-shot image to text pairing 2) Zero-shot text to image pairing.

3.3.

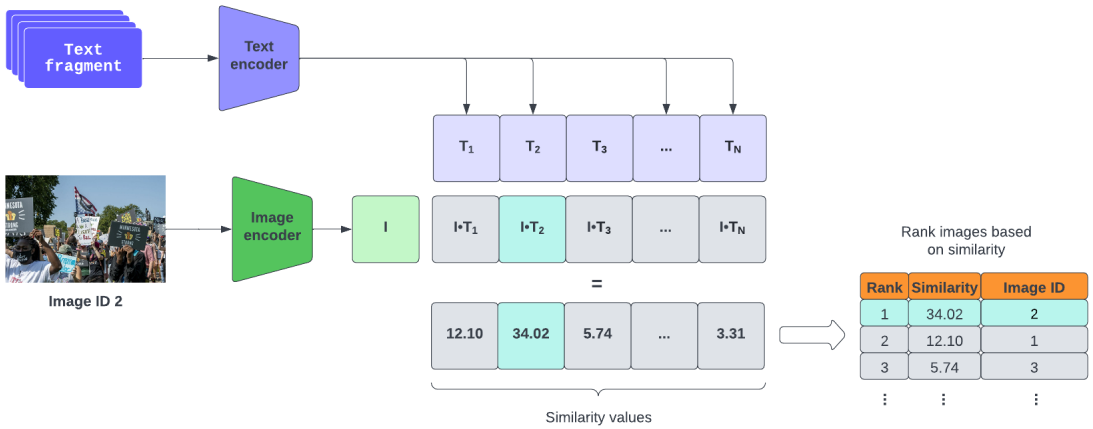


Figure 3.1: A descriptive image of the pipeline used to perform zero-shot image to text pairing.

1) Zero-shot image to text pairing In this first evaluation task, CLIP was used to compute a feature embedding for an inputted image (using the image encoder) as well as for the set of *all* possible texts in the dataset (using the text encoder). These embeddings were then used to calculate the cosine similarity between the inputted image and each of the text alternatives in the dataset. These similarity values were then normalized using softmax. Finally, using the similarity values, the text alternatives were ranked based on descending similarity values, for the specific image and a Average Precision (AP), Reciprocal Rank (RR) and Top-1 accuracy could be calculated. After conducting these scores for each of the images in the dataset, a mean AP (mAP) and mean RR (mRR) was calculated. Given that one image can have several corresponding fragments, meaning one or more "classes" per image, these ranking metrics are the way in which performance was evaluated.

2) Zero-shot text to image pairing The second use case uses the same pipeline but in the **opposite direction**, i.e. it takes a fragment as input and ranks all images

in a batch according to their similarity to the specific fragment. This is, in similar fashion as in case 1), done for each fragment to allow for calculation of evaluating metrics. 3.3.

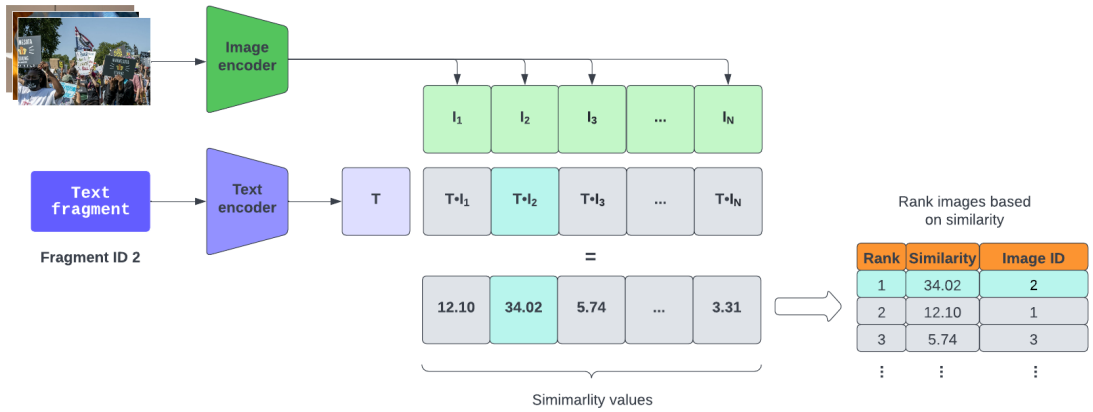


Figure 3.2: A descriptive image of the pipeline used to perform zero-shot text to image pairing.

3.3.1.1 Fine-tuning CLIP

The training of CLIP on the RFSM-dataset in this project has been set up following the same principle as in the pretraining carried out in [1], using PyTorch Lightning which provides boilerplate code to easily train neural networks using multiple GPU's. The training was carried out based on most of the configurations used in the original pre-training in [1]. The Adam optimizer [25] was used with a weight decay of 0.2 and beta values of $(0.9, 0.98)$, apart from the inherited parameters from the pre-trained CLIP model. Our best performing models were trained with a 1cycle learning rate scheduler initially described in [26], instead of a cosine annealing scheduler which was used in [1]. The 1cycle learning rate scheduler anneals the learning rate from a small initial learning rate up to a maximum learning rate for a given percentage of the total training steps, and from that maximum learning rate down to a learning rate 10^{-4} times lower than the initial rate during the remaining training steps. We call our final, best performing model Social Media-CLIP, SM-CLIP, and the pre-trained model Base-CLIP, B-CLIP, to simplify the notation from now on. The learning rate scheduler used to train SM-CLIP used a initial learning rate of $9 \cdot 10^{-10}$ and a maximum learning rate of $2 \cdot 10^{-8}$.

Several experiments were carried out with regards to the learning rate scheduler to find a schedule that resulted in a successful training, where both a flat learning rate was evaluated as well as a cosine annealing learning rate scheduler [27], in addition to the 1cycle scheduler. Different values for the minimum and maximum learning

rates were also experimented with for both the 1cycle scheduler as well as the cosine annealing scheduler.

CLIP was trained using a batch size of 32 768 and trained for 32 epochs [1]. The trainings in this project were carried out on 5 NVIDIA A100-SXM4-40GB GPUs for 18 epochs in most cases, which required a total computational time of 24 hours. The effective batch size used in this project was 33 280. The actual batch size used was 512 due to memory constraints, but by accumulating the gradients for 13 batches before taking one optimizer step, the training could be carried out with an effective batch size of $512 \times 13 \times 5 = 33280$, where 5 is the number of GPU's. The validation was carried out on 148 666 samples.

The contrastive loss calculations were carried out using cross entropy loss, as explained in Section 2.5.2. Throughout the training, mRR and mAP was also calculated in each validation step to allow for monitoring the progress according to our own metrics. These metrics were calculated in both directions, i.e. ranking images based on a given text sample, and ranking texts based on a given image sample. For simplicity, the training procedure disregards the fact that multiple text fragments can refer to the same image. This means that one image will appear multiple times as distinct images in the training dataset if that same image appears in multiple social media posts with different text fragments. The model was saved at the end of each training to be able to use further for evaluation.

3.3.2 Multi-label classification

To address the second research question presented in Section 1.2, it was of interest to first explore how image-text representations from CLIP can be used for a classification task specific to the threat domain. It was also of interest to see how this performance was changed when a fine-tuned model as retrieved from Section 3.3.1.1 was used to generate these image-text representations. Two different ways of using the CLIP embeddings for classification were developed – one classification method that utilizes CLIP's innate zero-shot abilities, and one classical supervised classification method. We refer to these two as *Approach A* for the zero-shot CLIP classification, and *Approach B* for the supervised classification. In other words, these two approaches are different methods to address one identical classification problem. The datatset used for both approaches was the Google Vision dataset, see Section 3.1.2 and the labels used for classification were hence **Manmade Disaster, Protest** and **Military Event**.

The classification in Approach A is done by embedding a sample and all possible labels to find the label that lies closest in the feature space. The classification in approach B on the other hand, is carried out using a linear classification layer that has been trained to predict event labels using the CLIP embeddings as input. Both approaches take an image embedding and text embedding as input, after they have been merged into one feature vector. This merge has been done in two ways to compare how they affect the results. Following previous works [10][18], the first method was feature-wise **addition**. The second method was to take a feature-wise

average. Approach A and B will be described further in detail below.

3.3.2.1 Approach A: Similarity based classification with CLIP

Approach A addresses how image-text representations from CLIP perform in a classification task specific to the threat domain and how this performance was changed when a fine-tuned model as retrieved from Section 3.3.1.1 was used to generate these image-text representations.

The pipeline used to measure the performance of this zero shot event classification look as follows:

1. *Create embeddings for text-image pair.* Separate text-embeddings e_t and image-embeddings e_i for the text-image pair were created using CLIP's text and image encoder respectively.

2. *Feature-merging.* A feature-wise average or sum of the text- and image embeddings ($\hat{e}_{t+i} = e_a$) was calculated.

3. *Create embeddings for chosen event types.* Firstly, we decide on which event types to use in the event classification, e.g. protests, police violence, military activity. In our case, *manmade disaster, protest* and *military event* were chosen. Then, a text embedding e_{event} for each of the possible event type was created using CLIP's text encoder. Since this is a zero shot approach, in theory new labels could be defined each time.

4. *Calculate cosine-similarity.* The similarity, $s_{a \to event}$, between e_a and each of the event embeddings e_{event} was calculated.

5. *Find prediction.* The prediction was found by specifying a threshold for each class, as in a classical multi-label classification set-up.

The pipeline is depicted in Figure 3.3, where each step is marked with its corresponding number.

This pipeline that is explained above, and depicted in Figure 3.3, clarifies the way in which we evaluate the performance of CLIP as a zero shot event classifier. The pipeline was implemented using both B-CLIP and SM-CLIP, with the aim of later being compared. Before doing so, a sub-experiment was conducted of choosing the best calculation method for feature-merging, indicated as *step 2* in Figure 3.3.

As mentioned in related works, [10] and [18] both use a feature-wise sum when conducting feature-merging. This experiment explores the difference, if any, between using the feature-wise sum and feature-wise average to merge the text and image feature vectors. The reason for using the average in contrast to other studies, is that we believe that might be more beneficial when the text and image reflect the same or a smiliar concept, compared to in [10] when they combine the information about two different concepts by using the plus operator on the two feature vectors.

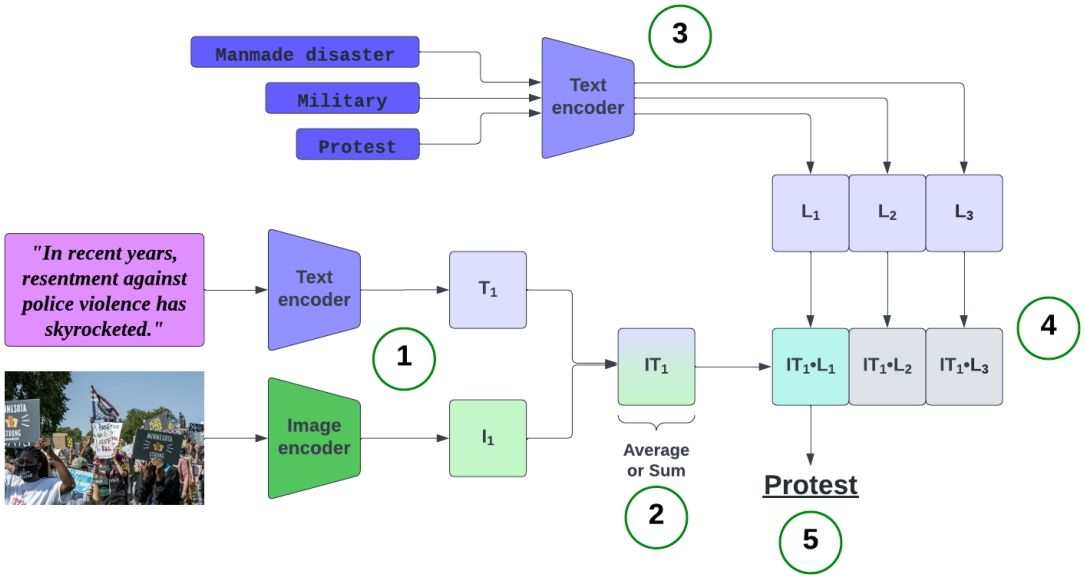


Figure 3.3: A descriptive image of the pipeline used in that what we refer to as *Approach A*.

As a final part to this experiment we also explored how the performance was affected by fine-tuning CLIP, through comparing when using original CLIP and the fine-tuned CLIP model.

3.3.2.2 Approach B: Feature based classification with an extended CLIP architecture

To give yet another dimension to the second research question presented in Section 1.2, that Approach A addresses, a parallel pipeline for event identification was developed, based on adding a linear classifier layer onto CLIP.

The input given to the linear classification layer is either the averaged or summed image and text embeddings, e_a, explained in step 2 of Approach A (see Figure 3.3). The output is a predicted probability for each event type. Since the linear layer needs to be trained, this approach carries one training step and one evaluation step implying that it is not a zero shot approach, but instead a more traditional supervised approach. The training step was conducted on one linear layer of 512 input neurons, using PyTorch lightning, on one RTX A5000 GPU. We used an Adam optimizer [25] with a flat learning rate of 10^{-3}, batch size of 32 and 67 epochs. The training objective was to minimize the Binary Cross Entropy loss after applying a sigmoid layer on the output. The steps taken in the evaluation pipeline are described below:

1. *Create embeddings on text-image pair.* Separate text-embeddings e_t and image-embeddings e_i for the text-image pair were created using CLIP's text and image encoder respectively.

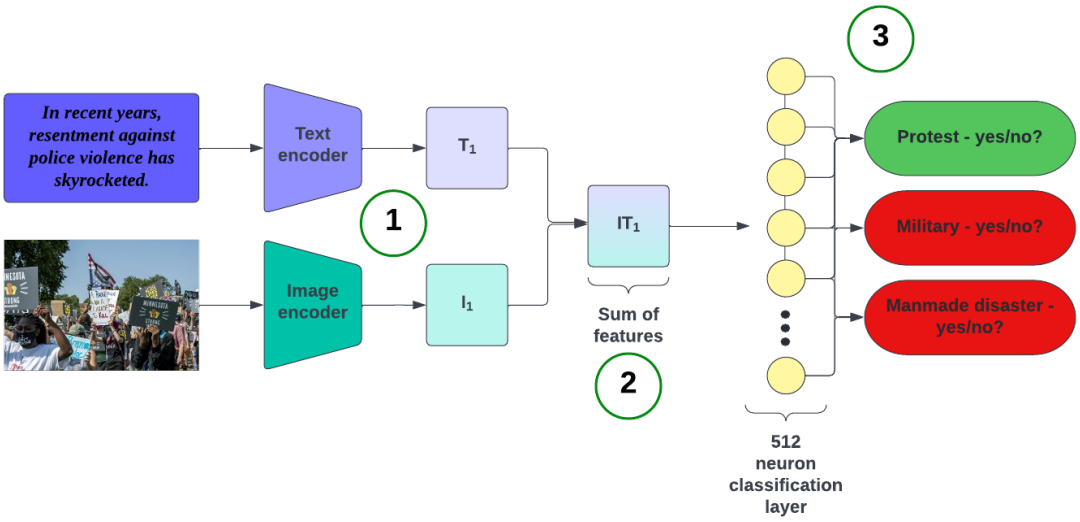


Figure 3.4: A descriptive image of the pipeline used in that what we refer to as *Approach B*.

2. *Calculate sum.* An feature wise sum of the text- and image embeddings ($\hat{e}_{t+i} = e_a$) was calculated.

3. *Find prediction.* With $\hat{e}_a$ as input, the linear layer outputs logits that are transformed to probability values for each of the possible event types p_{event}, using a softmax function. The prediction was then found by specifying a threshold for each class, as in a classical multi-label classification set-up.

Similarly to Approach A, this pipeline was evaluated with both the pre-trained CLIP as well as the fine-tuned CLIP. The pipeline is depicted in Figure 3.4, where each step is marked with its corresponding number.

4

Results

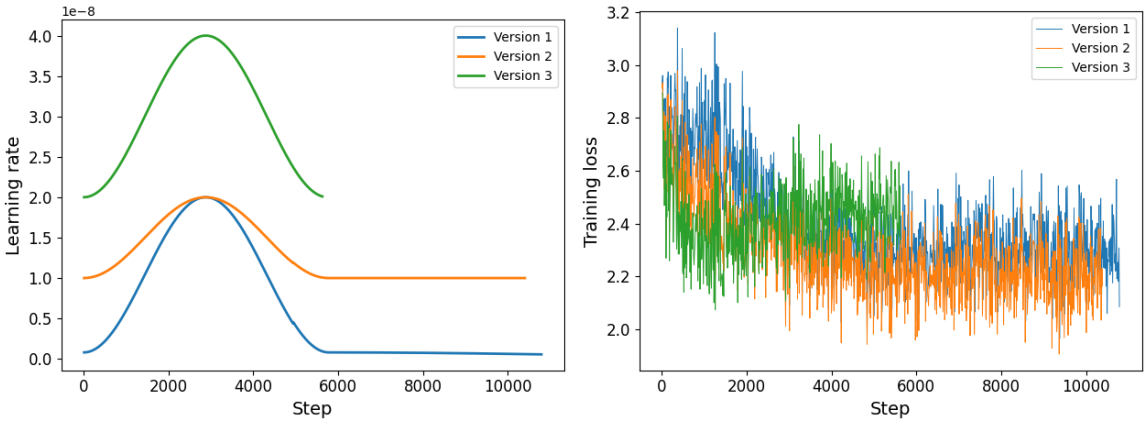
This chapter will present the results from the experiments described in Section 3.3.

4.1 Using CLIP for Zero-shot transfer

Achieving a successful fine-tuning of the pre-trained B-CLIP, required many trials with different set-ups. Three of these are shown in figure 4.1, where the learning rates, training loss, validation loss and mAP for the image-to-text pairing is presented for each training. The training was done on 18 151 890 samples collected in September and October. The validation was done on 148 666 samples collected in December, to minimize the risk for overlap in the training and validation data. Also the test data was taken from December.

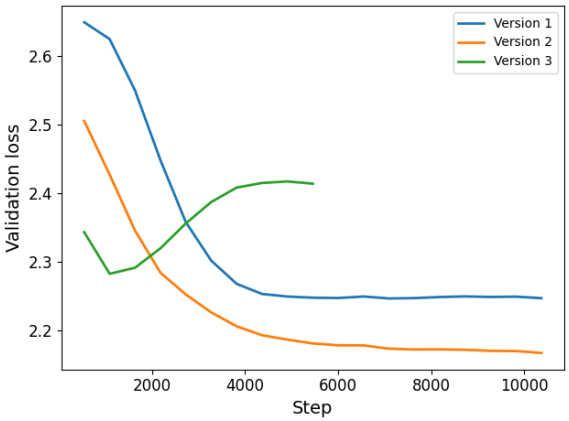
The only varying parameter in between the three trainings presented is the learning rate. All three use a "1cycle learning rate scheduler", as presented earlier, but use different initial, maximum and final values for the learning rate. The plots for training loss, validation loss and mAP show that there is a clear correlation between the performance of the training behaviour and the learning rate. The training using the highest initial, maximum and final learning rate (*Version 3*, colored in green) was interrupted prematurely, since it was obvious that the training had failed and wouldn't result in a competitive model when compared to the pre-trained model and the other two fine-tuned models.

Although the plots give a quite clear indication that model *Version 2* visualized in orange would perform the best, an evaluation was also carried out on the test set to get objective results, which is presented in Table 4.1. The evaluation on the test set followed the image-to-text pairing and text-to-image setup explained in Section 3.3.1. The same evaluation was also carried out using B-CLIP to use as a baseline for the performance. The evaluation of the best model, *Social Media-CLIP (SM-CLIP)*, was finally carried out on three differently sized test sets, to see how the size of the dataset affected the results.

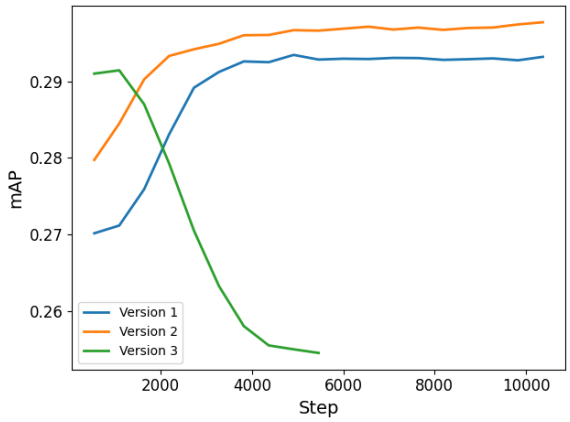


(a) Learning rate

(b) Training loss



(c) Validation loss



(d) mAP

Figure 4.1: Plots showing learning rate, training loss, validation loss and mAP for three different training sessions, which all three had different learning rate schedulers. From these plots, it's easy to see how the learning rate affects the behaviour during training. The green training was interrupted early due to that it showed unsuccessful training behaviour already after a few hundred training steps.

	Image 2 Text			**Text 2 Image**	
	mRR	mAP	top-1 accuracy	MRR	top-1 accuracy
Version 1	22.00	22.00	16.75	20.15	15.04
Version 2 (SM-CLIP)	**22.25**	**22.25**	**16.87**	**20.45**	**15.23**
Version 3	21.85	21.85	16.60	20.02	14.96

Table 4.1: Resulting metrics for the three models showed in figure 4.1. The table shows metrics both for the image to text pairing ("Image 2 Text") and for the text to image pairing ("Text 2 Image"). The best performing model, i.e. the orange one, is now named SM-CLIP and is further used in the remaining experiments.

	Image 2 Text			Text 2 Image	
	mRR	mAP	top-1 accuracy	MRR	top-1 accuracy
B-CLIP[1]	9.44	9.44	6.58	8.33	5.66
SM-CLIP[1]	**10.56**	**10.55**	**7.36**	**9.19**	**6.27**
B-CLIP[2]	11.47	11.45	8.16	10.25	7.12
SM-CLIP[2]	**12.80**	**12.79**	**9.12**	**11.30**	**7.88**
B-CLIP[3]	20.04	20.03	15.13	18.54	13.79
SM-CLIP[3]	**22.25**	**22.25**	**16.87**	**20.45**	**15.23**

Table 4.2: Resulting metrics for the B-CLIP and SM-CLIP on three differently sized datasets. The table shows metrics both for the image to text pairing ("Image 2 Text") and for the text to image pairing ("Text 2 Image"). 1: 1 822 967 samples, 2: 955 391 samples, 3: 110 055 samples.

4.2 Multi-label event classification

The second part of the results regard the usage of CLIP as a feature encoder for multi-label classification on threat related events, more specifically *Manmade disaster*, *Military* and *Protests*. This evaluation consists of a few separate experiments, explained in detail in Section 3.3.2. Firstly, the results for the experiments using approach A will be presented. The experiments on approach A include an evaluation of the best way to merge the text and image features as well as an evaluation of the effects of prompt engineering. Lastly, a comparison of using B-CLIP versus SM-CLIP will be presented. For approach B, results will be presented for the experiment on feature merging followed by a comparison of using B-CLIP versus SM-CLIP as the text and image encoders to create the input for the classifier.

4.2.1 Approach A

As explained in the methodology, Approach A (see Figure 3.3) clarifies the way in which we evaluate the performance of CLIP as a zero shot event classifier. In this section we present the results of the experiments of choosing the calculation method for feature-merging as well as the comparisons of the two CLIP models (B-CLIP vs. SM-CLIP) using the pipeline presented in Approach A.

To evaluate the calculation method for the feature-merge, we tested calculating using an average or sum. The features were created using B-CLIP or SM-CLIP. The results are compiled in Table 4.6. The Precision-Recall curves for the best calculation method, for both B-CLIP and SM-CLIP are presented in Figure 4.2.

4.2.2 Approach B

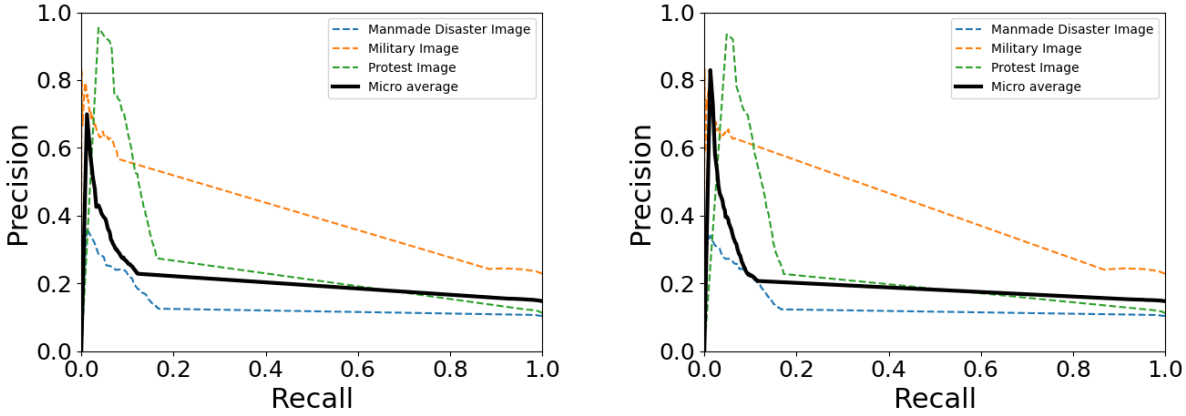
This section describes the results from the experiments for the supervised classifier, i.e. Approach B (see Figure 3.4).

	AUC-PR			
	B-CLIP		SM-CLIP	
	Sum	Average	Sum	Average
Manmade disaster	0.1226	0.1345	0.1243	0.1345
Military	0.4047	0.4064	0.4229	0.4240
Protest	0.2368	0.2617	0.2141	0.2371

Table 4.3: AUC-PR values for **Approach A**, for each of the models, along with the difference when using **average vs sum** as calculation method for feature-merging. The AUC-PR values are given per label.

	AUC-PR	
	B-CLIP	SM-CLIP
	Average	Average
$\overline{\text{Labels}}_{micro}$	0.2093	0.2004

Table 4.4: A balanced average of the AUC-PR values over all labels for **Approach A**. This is computed for each of the models when using **average** as calculation method for feature-merging.



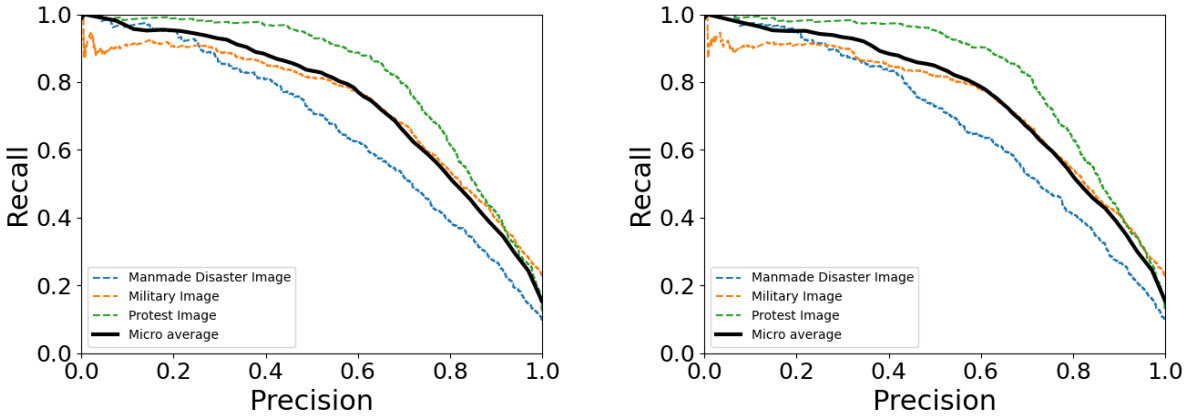
(a) B-CLIP **(b)** SM-CLIP

Figure 4.2: Precision-recall curves for **Approach A** when using **average** to calculate the feature-merge. The colored graphs show the performance with respect to each label, and the black graph shows the micro average of them.

Although we have assumed that the text and image in each sample-pair are related, as mentioned in Section 3.1.2, we cannot be certain that this is the case. Admitting this uncertainty, we have chosen to train the linear layer using *sum-calculation* for the feature-merging, since this would allow for contrasting concepts to be learnt more intuitively as described in Section 3.3.2.1. Furthermore, the technical difference between a summation calculation and average calculation is simply a constant factor of two, and knowing this, it is reasonable to assume that the network would learn

this and adjusting the weights accordingly. Moreover, the difference between sum and average calculations are marginal as reflected in Table 4.6.

The main part of this approach was, as mentioned in Section 3.3.2.2, to give another dimension to the research question addressed by Approach A (question 2 in Section 1.2). Hence, the main part of this experiment regards the comparison between B-CLIP and SM-CLIP. To compare, both a visual and numerical result is presented. Figure 4.3a and Figure 4.3b provide the Precision-Recall curves of the predictions when using 100 different thresholds between 0 and 1, with features being created from B-CLIP and SM-CLIP respectively. Table 4.5 show the numerical values representing the Area Under the Curve (AUC) for each of the plots.



(a) Precision-Recall curves with B-CLIP

(b) Precision-Recall curves with SM-CLIP

Figure 4.3: Precision-recall curves when evaluating using the orignal, or finetuned CLIP model.

	AUC-PR	
	B-CLIP	SM-CLIP
Madmade disaster	0.6667	**0.6793**
Military	0.7339	**0.7372**
Protest	0.8170	**0.8247**

Table 4.5: AUC-score for **Approach B**, for each of the models over each individual label. The best results are highlighted in the table.

	AUC-PR	
	B-CLIP	SM-CLIP
Average_{micro}	0.7412	0.7480

Table 4.6: A balanced average AUC-PR values for **Approach B**, for each of the models when using sum as calculation method for feature-merging.

5

Discussion

This chapter will discuss the results presented in Chapter 4, starting with a discussion of the obtained results of the fine-tuning followed by a discussion of the results from the event classification in approach A and B.

5.1 Using CLIP for Zero-shot transfer

The main aim for this project was to explore whether it is possible to successfully fine-tune CLIP within a more specific domain, as explained in Section 1.2. Hence, the core of this project consisted of fine-tuning CLIP on social-media, threat associated data. The results presented in Section 4.1, more specifically Table 4.1 and 4.2, show that a fine-tuning on threat associated data indeed improves the performance on zero shot image-to-text matching and text-to-image matching. However, the improvements are quite modest, of 1-2% depending on the size of the dataset used for evaluation. Table 4.2 shows, very intuitively, that the performance of the models increase when a smaller dataset is used for performance. This is intuitive, since the evaluation is based on a ranking principle that for each query has to rank all available documents in the entire test set, which naturally becomes more and more difficult when the size of the dataset increases.

When it comes to the comparison of the three different trained models, presented in Figure 2.4 and Table 4.1, it is quite surprising to see that version 3, i.e. the one that showed strange behaviour, performs almost equally well as the other two, by only 0.15 % lower performance than Version 1. This difference is so small that it is almost statistically insignificant. The similar performance of Version 1 and 2 is, on the other hand, not as surprising, since they show very similar behaviours during training. As previously explained, the only factor that differs for these three models is the learning rate. Version 3 had an initial and final learning rate of 2e-8 and a maximum learning rate of 4e-8, which is only a factor of 2 larger than the corresponding values for Version 2, which used 1e-8 as initial and final learning rate, respectively 2e-8 for maximum learning rate. The behaviour of Version 3 shares similarities with typical overfitting behaviour, when the model learns the training data too well and loses its ability to generalize on new data, so the performance on the validation data decreases. On the other hand, overfitting shouldn't occur so early in the training process. Instead, it could be an example of so called *catastrophic forgetting*, which

is when the model conceptually forgets its pre-trained knowledge and overwrites it with knowledge specific for the fine-tuning data. One explanation for this could be that the learning rate is too big, which allows the model to quickly travel from its original global minimum in the parameter space to a new local minimum which is better suited for the new training data.

Another point to notice is how the validation loss is very low already at the start of the continued training, which is the case for all three Versions. This might be an indication of CLIP already having been exposed to the type of data used in this project, since if it wasn't in alignment to the data used in the pre-training, the validation loss would have higher initial value. The very small improvements in performance after fine-tuning is yet another argument that the format or characteristics of the data used for fine-tuning is already somewhat familiar to the pre-trained CLIP. Based on these results, it seems reasonable to use data that is even more specialized in future attempts to fine-tune CLIP to hopefully achieve greater improvements.

Nevertheless, the three different versions show that the fine-tuning is extremely sensitive to the size of the learning rate. In general, very (!) small learning rates had to be used to successfully train the models, compared to baseline learning rates that normally are used, which usually lie between 1e-2 and 1e-5. This might although be quite reasonable – first of all since this case regards a fine-tuning of an already extensively trained model, and second of all since it is a huge model.

An aspect of the fine-tuning that would have been interesting to explore is whether it would be possible to use a multi-training-task instead of the single image-to-text matching used in this work. [28] uses a pre-trained CLIP model and presents a second learning objective when continuing the training. Their second learning objective is described as using two event graphs, one for text and one for image, and aligning those two during training. By doing so, they achieve better performance on zero-shot event extraction. Such an approach could possibly benefit this study as well.

5.2 Multi-label event classification

To address the second research question this project aimed to answer, **Approach A** was developed. The first part of the research question reads "*How can text and image representations from an ITA-model be used for multi-label classification ...*". This inspired experimenting with the calculations behind feature-merging. Seeing that current work such as [10] and [18] carry out an element-wise addition to merge the features, one could assume that this technique would yield a higher performance. However, as reflected in Table 4.6 the performance when considering the AUC values instead was higher in the case of average calculations.

In the case of additive calculations, one conceptually "adds" information from different modalities. One example of this is presented using ALIGN in [10], where they feed their model a multi-modal query of an image of a pair of grey shoes, along with

a text string "red". The image and the text string are then encoded using ALIGN's image and text encoders. The image-features would then reflect the concept of "shoes", and the text features would reflect the concept of the color "red". Then, the feature-merge using feature-wise addition would translate to the combination of "shoes" and "red", i.e. red shoes. And intuitively, the ALIGN model outputs a new image of a red pair of shoes. Technically, this means that the image-features that represent one specific place in the embedding space, added with the text-features that represent another place in the embedding space, would result in a feature vector that represents a new, third place in the embedding space reflecting the combined information of "red shoes".

In the case of when both the text and image contain the same concept, i.e. if the image shows a dog, and the text is a string "A dog", both the image-features and text-features would reflect the same concept, and both feature representations would probably lie very close in the embedding space. Using feature wise average to merge the features in this case, would result in a combined feature vector whose position in the embedding space is very close to the original locations of "dog" and the image of a dog. If the text and image however reflect non-similar concepts, the average calculation would map to a different place in the embedding space somewhere in between the two original places. Seeing that our data samples are text-image *pairs*, they ought to reflect the same concept. With this reasoning, it makes sense that the average calculations perform slightly higher.

Since the images and texts in the Google Vision dataset are similar in some cases, but more contrasting, i.e. different, in nature than first anticipated, one could argue that this is the reason to why the performance between average and additive feature-merging are as small as they are. This uncertainty opens the door to relevant future work. With the discussion carried out until now, one could assume that an additive calculation method of the feature-merge would be more appropriate in the case when the input-pairs are *contrasting* in nature. The reason being that the information of both modalities would be used to map to a new place in the embedding space. An average calculation method however, might be more appropriate in the case of the input-pairs being *similar* in nature, since they would always map to similar place in the embedding space.

The overall performance of using CLIP's innate zero-shot capabilities for multi-label event classification is however low. When investigating the Google Vision dataset used, it was visible that the annotations from both Google Vision and Recorded Future were partly unreliable. The reason we call the dataset unreliable is that for quite many samples neither the image-labels or text-labels seem to reflect the context appropriately. That is, they do not reflect the high level concepts such as events occurring, but instead low level concepts such as objects present. For the case of the image labels, they seem to be in fact more accurate in object detection, while the text labels reflect keywords rather than the contextual event. This inevitably affects the performance of the evaluation. Because of the unreliability in the dataset, it is difficult to properly conclude whether the approach itself is useful or not.

The second part of the research question addressed by Approach A reads "*... how is the performance affected by the previously mentioned, domain-specific, fine-tuning*". Judging by results presented in Table 4.6 and precision-recall curves presented in Figure 3.3, both B-CLIP's and SM-CLIP perform quite poorly as zero-shot multi-label classifiers. The ideal case for a Precision-Recall plot is to have a curve that follows the upper and right boundaries of the plot as closely as possible. The larger the area is under the curve, the better the performance. SM-CLIP shows a slight improvement for the Military events, while lowering its performance for Protest events. The marginal difference in performance might be explained by the fact that the improvement of SM-CLIP itself is marginal as discussed in Section 5.1, which can be seen in Table 4.2.

Another aspect worth exploring for approach A is the effect of prompt engineering. As the authors of CLIP used different prompt templates when evaluating the model as a zero-shot classifier, achieving positive results [1], the hypothesis is it would be similar for our case since the event labels used were short in nature. However, due to time constraints the aspect could not be explored further.

With the aim of putting the performance of Approach A in perspective, **Approach B** was developed. As can be seen in Table 4.5, the performance of multi-label event classification improves by almost 4 times when using Approach B compared to Approach A. Adding a classification layer and training it with a specific objective allows for better overall predictions. The network still seems to have difficulties in distinguishing between the labels in an ideal manner, seen in Figure 3.4. Keeping in mind the high quality performance of CLIP's zero-shot image-to-text matching abilities in the fine-tuning setup, it seems more reasonable that the results reflect the quality of the dataset. As previously discussed, there are discrepancies in the labeling of the dataset – partly because of flaws in the labelling, and partly because of that the image and text are unrelated in many cases, which naturally would confuse the model. The fact that the performance of approach B indeed is much better than Approach A is another argument that the labeling is poor, since Approach A is completely dependent on the actual semantic of the label, while approach B only looks for patterns in the data that are common for each label without actually paying attention to the meaning of the label. The differences in performance hence prove that there are patterns in the data, but that they are partly unrelated to the labels. Another aspect of the somewhat inadequate performance in both approaches, is the overlap in content between the event labels. The data was collected during a time when a large part of news articles and other posts were related to the invasion of Ukraine. Naturally, *protests*, *military activity* and *manmade disaster* occur frequently in the reports of the situation in Ukraine. As a consequence of this, it is natural to believe that there are a lot of similarities between samples in the different event categories, since many of them are related to the same subjects. In particular, it is natural that there is a large overlap between the *Military* event samples and the *Manmade disaster* samples, since many of the *Manmade disaster* samples regard explosions or fires, which themselves often are a consequence of bombs (in the context of Ukraine) which can be considered a *Military activity*. This ambiguity can be a factor of confusion of the model.

When it comes to comparing SM-CLIP and B-CLIP in approach B, we see that SM-CLIP marginally outperforms B-CLIP as reflected in Table 4.5 for all event types. This is an argument that the fine-tuning of CLIP indeed having a positive effect in the domain of threat intelligence, although a very small such, which follows the argumentation previously stated.

6

Conclusion

From the discussion carried out we can conclude that CLIP can be successfully fine-tuned on threat associated data to improve its zero-shot performance, which answers the first research question. However, the marginal improvements suggest that the data used for the fine-tuning has been seen by CLIP already in the pre-training, which leaves questions on the value of fine-tuning CLIP on social media data. The experiments conducted shows clearly that such a fine-tuning is highly affected by the learning rate, where a relatively low value should be chosen for catastrophic forgetting to not occur. Furthermore, fine-tuning CLIP gives slight improvements in using it as an image and text encoder for the a supervised classification task.

As for the investigation of how the image and text embeddings created by CLIP can be used for multi-label event classification, it is clear that Approach B outperforms Approach A. But as reflected in the discussion, there is an uncertainty in the dataset, which gives incentive to further explore this aspect. The question also adressed how fine-tuning would affect the performance of the classification. Due to the marginal differences seen from the fine-tuning, one can conclude that fine-tuning on social media data does not affect the performance when using CLIP's features in a classification set up.

Bibliography

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," 2017. [Online]. Available: https://arxiv.org/abs/1705.09406

[6] Y. Aytar, C. Vondrick, and A. Torralba, "See, hear, and read: Deep aligned representations," 2017. [Online]. Available: https://arxiv.org/abs/1706.00932

[7] H. Yu and J. M. Siskind, "Grounded language learning from video described with sentences," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 53–63. [Online]. Available: https://aclanthology.org/P13-1006

[8] Z. Wang, J. Yu, A. W. Yu, Z. Dai, Y. Tsvetkov, and Y. Cao, "Simvlm: Simple visual language model pretraining with weak supervision," 2021. [Online]. Available: https://arxiv.org/abs/2108.10904

[9] Microsoft, "Turing bletchley: A universal image language representa-

tion model by microsoft," https://www.microsoft.com/en-us/research/blog/turing-bletchley-a-universal-image-language-representation-model-by-microsoft/, 2021.

[10] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," 2021. [Online]. Available: https://arxiv.org/abs/2102.05918

[11] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, 2008, pp. 722–729.

[12] D. Won, Z. C. Steinert-Threlkeld, and J. Joo, "Protest activity detection and perceived violence estimation from social media images," 2017. [Online]. Available: https://arxiv.org/abs/1709.06204

[13] A. H. Hossny and L. Mitchell, "Event detection in twitter: A keyword volume approach," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018, pp. 1200–1208.

[14] N. Ding, Z. Li, Z. Liu, H. Zheng, and Z. Lin, "Event detection with trigger-aware lattice neural network," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 347–356. [Online]. Available: https://aclanthology.org/D19-1033

[15] S. Liu, Y. Li, F. Zhang, T. Yang, and X. Zhou, "Event detection without triggers," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 735–744. [Online]. Available: https://aclanthology.org/N19-1080

[16] Q. Lyu, H. Zhang, E. Sulem, and D. Roth, "Zero-shot event extraction via transfer learning: Challenges and insights," in *ACL/IJCNLP (2)*, 2021, pp. 322–332. [Online]. Available: https://doi.org/10.18653/v1/2021.acl-short.42

[17] Z. Wang, X. Wang, X. Han, Y. Lin, L. Hou, Z. Liu, P. Li, J. Li, and J. Zhou, "CLEVE: contrastive pre-training for event extraction," *CoRR*, vol. abs/2105.14485, 2021. [Online]. Available: https://arxiv.org/abs/2105.14485

[18] e. a. Tiberiu Sosea, "Using the image-text relationship to improve multimodal disaster tweet classification," *Proceedings of the ISCRAM Conference*, vol. 18, 05 2021.

[19] M. Tong, S. Wang, Y. Cao, B. Xu, J.-Z. Li, L. Hou, and T.-S. Chua, "Image

enhanced event detection in news articles," in *AAAI*, 2020.

[20] G. S. Cheema, S. Hakimov, E. Müller-Budack, and R. Ewerth, "A fair and comprehensive comparison of multimodal tweet sentiment analysis methods," 2021. [Online]. Available: https://arxiv.org/abs/2106.08829

[21] I. C. Education, "Natural Language Processing (NLP)," 2020. [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing

[22] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4291–4308, oct 2021. [Online]. Available: https://doi.org/10.1109%2Ftnnls.2020.3019893

[23] J. Wehrmann, C. Kolling, and R. Barros, "Adaptive cross-modal embeddings for image-text alignment," 11 2019.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[26] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2017. [Online]. Available: https://arxiv.org/abs/1708.07120

[27] K. Naoki, "Cosine annealing with warmup for pytorch," https://github.com/katsura-jp/pytorch-cosine-annealing-with-warmup, 2020.

[28] M. Li, R. Xu, S. Wang, L. Zhou, X. Lin, C. Zhu, M. Zeng, H. Ji, and S. Chang, "Clip-event: Connecting text and images with event structures," *CoRR*, vol. abs/2201.05078, 2022. [Online]. Available: https://arxiv.org/abs/2201.05078