COSE474-2024F: Final Project Report
Traffic Sign Image Classification

Matteo Felipe Merz

1. Introduction

Traffic sign recognition is a key topic in computer vision. On the road, traffic signs serve multiple important purposes such as regulating speed and warning drivers of potential hazards that lie ahead. Whether in ADAS (Advanced Driver Assistance Systems) or fully autonomous driving, accurate and reliable road sign recognition plays a vital role in ensuring driver safety. Especially warning signs, which inform the driver about potential dangers that lie ahead, are very important for ensuring the safety of the passengers and other road users.

The goal of this project is to achieve high accuracy image classification on a dataset of warning traffic signs. Warning traffic signs are visually highly similar, which complicates the classification process. To overcome this challenge and accomplish a high classification accuracy, a pre-trained model is fine-tuned on the dataset.

This project consists of two parts. Firstly, the preparation of the data for the fine-tuning process. This includes adding textual descriptions to the data points. Secondly, the model is fine-tuned on the dataset. This process includes careful tuning of hyper-parameters and utilizing contrastive learning to improve the model accuracy.

2. Methods

The identification of road signs poses multiple particular challenges. Signs are sometimes occluded by greenery or other drivers and their condition can vary due to their age or pollution. The classification should also work independent of the current weather or time of day. However, the highly similar nature of the signs poses the biggest challenge for the classification.

The model used for the image classification is the CLIP (Contrastive Language-Image Pre-Training) model developed by OpenAI. (Radford et al., 2021) CLIP is a powerful multi-modal model, that connects visual and textual data. It is trained on 400 million text, image pairs and performs especially well in a zero-shot manner. This means that CLIP can even classify images into classes, without being specifically trained on them. While this provides massive benefits for image classification tasks, there are some limitations to CLIP's capabilities. While CLIP can generalize well to common objects, it can struggle with understanding more abstract concepts and very fine-grained classifications. To apply the model in cases, where Zero-Shot classification does not work, the model needs to be fine-tuned to the specific dataset. (Goyal et al., 2022) Fine-tuning takes advantage of the already pre-trained model and adapts the model weights to better fit the classification task at hand. To achieve this, the model gets trained on the data from the dataset. Fine-tuning can lead to significantly improved performance on highly specialised tasks, while not being as computational intensive as training a model on the data from scratch.

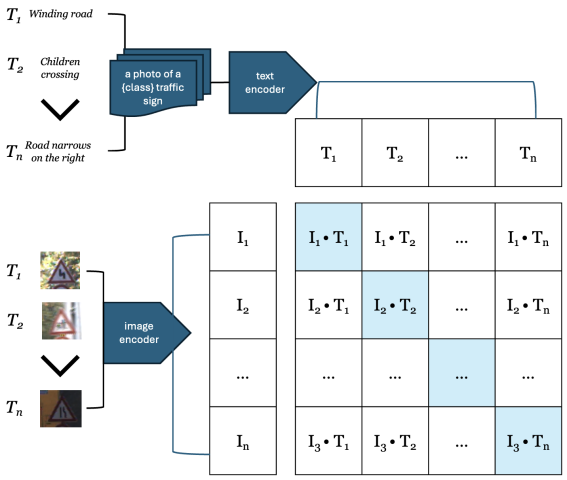

Figure 1. Fine-Tuning Pipeline. The diagonal values are maximized while the other values are minimized.

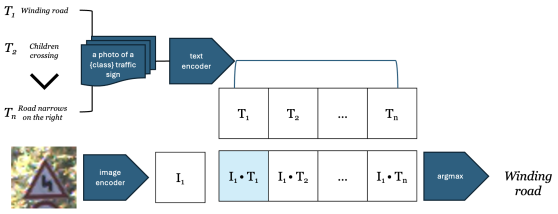

Figure 2. After fine-tuning, an image and all class labels are passed into the encoders. The highest value denotes the inferred class.

The process for fine-tuning the model is very similar to the initial pre-training of the model. (Goyal et al., 2022) In each iteration, image and text embeddings are computed for a batch of random pairs from the training data. Based on these embeddings, logits are calculated for both images and text, representing how similar each image is to each piece of text. The loss is calculated by averaging the cross-entropy loss of both directions. As in the pre-training, this encourages the model to maximize the similarity between matching text and image pairs. The resulting model is ready for Inference on the new dataset. The algorithm used for the pre-training of the CLIP model can be found in the Appendix 1.

Applying this kind of fine-tuning into the classification of street signs is a novelty, as many fine-tuning solutions only fine-tune CLIP's image encoder and ignore the text encoder entirely. Also using textual descriptions instead of labeled data in the classification of text signs is not very common.

3. Experiments

The GTSRB (German Traffic Sign Recognition Benchmark) dataset is used for training and testing the model. (Stallkamp et al., 2012) In total, the GTSRB contains more than 50,000 images and 40 classes of German traffic signs. It focuses on providing a lifelike database of traffic signs and is therefore well suited for this project. The images in the GTSRB are labeled with a number representing their specific class. For this project 13 classes were selected. They are all warning signs that are commonly used to warn about possible upcoming dangers, like poor road conditions or pedestrians interfering with the flow of traffic. An overview of the classes used in this project can be found in Figure 3. The design of these signs is very similar, with all of them being red triangles with graphics indicating the specific danger ahead. Additionally, the traffic signs convey abstract concepts and are probably not part of CLIP's pre-training. This makes accurate zero-shot classification unlikely and highlights the need for fine-tuning.

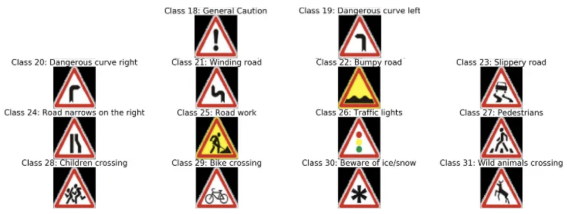

Figure 3. Classes that are used in the image classification.

The model training and image classification was performed in Google's web-based Jupyter notebook environment 'Google Colaboratory'. All the computations were performed on a Nvidia T4 GPU. For this project 23 compute units were used in total.

The first step to achieve the image classification is the preparation of the dataset. The GTSRB does not contain text descriptions as class labels and only has a number as the class id associated with each data entry. To leverage the full power of the text and image encoders of the CLIP model, the first step is to map a text description to each of the data points. The description consists of the prefix 'a photo of a ', a short description of the specific danger that the sign is warning about (e.g. 'Children crossing') and the suffix ' traffic sign'. Using natural language descriptions over simple labels, like the name of the class, can evidently lead to increases in model accuracy, as reported by Radford et al. (2021). The next step is the fine-tuning of the model using the technique described in the methods. The optimizer used for the fine-tuning is the AdamW optimizer, which is well suited for fine-tuning CLIP. (Loshchilov & Hutter, 2019) Through trial and error, a learning rate and a weight decay of 1e-5 were chosen to optimize the learning and reduce over-fitting. The model is only fine-tuned for one epoch as it already reaches a validation accuracy of over 99% after the first epoch and to limit the computational overhead caused by fine-tuning.

Before fine-tuning the model, the pre-trained CLIP model reaches an accuracy of around 19.16% on the test dataset. After fine-tuning for one epoch, the model reaches an accuracy of 94.98%. Fine-tuning for more epochs can improve this accuracy even further. The improvement highlights the effectiveness of the fine-tuning process. The current state of the art model for the GTSRB, a CNN with three spatial transformers, achieves an accuracy of 99.71% over all 40 classes of the dataset. (Arcos-García et al., 2018)

Four random data points were selected from the test dataset to perform a qualitative comparison between the two models. The pre-trained model inferred only one class correctly and chose all classes with a low confidence. This indicates that the model is most likely only guessing the correct class.

Figure 4. Qualitative results of zero-shot image classification.

After the fine-tuning is complete the model correctly infers the right class for every data point and does so with a confidence of over 99%.

Figure 5. Qualitative results of image classification using the fine-tuned model.

In both the comparison of quantitative and qualitative results, the impact of fine-tuning for even a single period on pre-trained models like CLIP can be clearly observed. While the performance of the model does not hold up to the state of the art solution, the achieved accuracy after only one epoch of pre-training is still impressive. Overall, fine-tuning the model achieves the desired high accuracy image classification of the warning traffic signs.

4. Future Direction

This project proves the validity and efficiency of the fine-tuning method for adapting a pre-trained model to a specific classification task. One future implication of the results of this study could be the development of an Object Detection model based on the fine-tuned CLIP model. This would enable the detection of multiple warning signs in one traffic scene, which is crucial in real world applications. In the context of this project this direction was also explored, but only with limited success.

Figure 6. Object Detection experiment result conducted with the fine-tuned CLIP model.

Another application of the high accuracy classification would be to combine the detected class label with other objects that are detected in the scene to give specific advice on how the driver should react to be prepared for possible dangerous situations. For example the detection of a person riding a bicycle in combination with a 'Bicycle crossing' sign, could alert the driver to slow down and pay close attention to this specific bike rider.

References

Arcos-García, Á., Alvarez-Garcia, J. A., and Soria-Morillo, L. M. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural Networks*, 99:158–165, 2018.

Google. URL https://colab.research.google.com/.

Goyal, S., Kumar, A., Garg, S., Kolter, Z., and Raghunathan, A. Finetune like you pretrain: Improved finetuning of zero-shot vision models, 2022. URL https://arxiv.org/abs/2212.00638.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

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

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2012.02.016. URL https://www.sciencedirect.com/science/article/pii/S0893608012000457. Selected Papers from IJCNN 2011.

A. Fine-tuning Algorithm

Algorithm 1 Fine-Tuning CLIP on a new dataset

Require: Pre-trained CLIP model, epochs E, dataset $\mathcal{D} = \{(I_k, T_k)\}_{k=1}^{N}$, learning rate η, temperature τ, weight decay λ

Ensure: Fine-tuned CLIP model

Initialize optimizer AdamW(θ, η, λ)

for epoch e in $1, \ldots, E$ **do**

  Shuffle dataset $\mathcal{D}$

  for each mini-batch $\mathcal{B} = \{(I_b, T_b)\}_{b=1}^{B}$ in $\mathcal{D}$ **do**

    Compute image embeddings

$$Z = \text{ImageEncoder}(I_b)$$

    Compute text embeddings

$$W = \text{TextEncoder}(T_b)$$

    Compute logits for image-to-text:

$$\text{logits}_{\text{img}} = \frac{ZW^\top}{\tau}$$

    Compute logits for text-to-image:

$$\text{logits}_{\text{txt}} = \frac{WZ^\top}{\tau}$$

    Set ground truth labels $= \{0, \ldots, B-1\}$

    Compute loss:

$$\mathcal{L}_{\text{img}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\text{logits}_{\text{img}}[i, \text{labels}[i]])}{\sum_{j=1}^{B} \exp(\text{logits}_{\text{img}}[i, j])}$$

$$\mathcal{L}_{\text{txt}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\text{logits}_{\text{txt}}[i, \text{labels}[i]])}{\sum_{j=1}^{B} \exp(\text{logits}_{\text{txt}}[i, j])}$$

$$\mathcal{L}_{\text{total}} = \frac{\mathcal{L}_{\text{img}} + \mathcal{L}_{\text{txt}}}{2}$$

    Backpropagate loss $\mathcal{L}_{\text{total}}$

    Update model parameters using AdamW

  end for

end for

Return fine-tuned CLIP model

B. Project Links

The project report source files and editing history can be accessed on Overleaf.

The project repository can be accessed on GitHub.