CLIP-ReIdent: Technical Report

Konrad Habel, Fabian Deuser, Norbert Oswald
University of the Bundeswehr Munich
Institute for Distributed Intelligent Systems (VIS)
{konrad.habel,fabian.deuser,norbert.oswald}@unibw.de

September 20, 2022

Contents

1 Introduction

This technical report describes only changes we made for Phase-II of the MMSports 2022 Player Re-Identification challenge. We already published a paper "CLIP-ReIdent: Contrastive Training for Player Re-Identification" [1] for Phase-I (Paper submission) of that challenge and refer to that paper for the used architecture and our approach.

2 Implementation Details

We do not use the provided challenge toolkit, instead we use our own implementation available in the following GitHub repository: `https://github.com/KonradHabel/clip_reid`

Steps needed for training of the model:

1. get data: `download_data.py`
2. pre-process data: `preprocess_data.py`
3. training: `train.py`
4. evaluation: `evaluate.py`
5. generate submission.csv: `predict.py`

2.1 Download and Dataset preparation

The script `download_data.py` can be used to download and unzip the data from the given challenge toolkit repository on GitHub. This script also creates the folder structure used from all other scripts.

Because our code does not use the given challenge toolkit the script `preprocess_data.py` must be executed once to create the Pandas DataFrames for the dataloader. In this script it is also possible to specify the number of cross-validation splits besides the official test split.

Both scripts need to be executed only once to get the data, create the folder structure and get everything ready for training.

2.2 Training and Evaluation

For training the script `train.py` is used. The smaller ViT-B/16 model can be trained with mixed precision on GPUs with only 12GB VRAM when using a batch size of 16. As already mentioned in the paper, the effective batch size is doubled to 32 because we train on pairs of query and gallery images. We used for the training of the ViT-L/14 model V100 GPUs with 32GB VRAM. But it is also possible to use GPUs with lower VRAM like a single RTX 3090 or 2x12GB GPUs using DataPrallel.

Additionally for the CLIP ViT models we implemented the possibility to use "Gradient Checkpointing". With gradient checkpointing it is possible to train even on a single GPU with low VRAM the ViT-L/14 model with a batch size of 16 only with the disadvantage that the training takes longer.

We also implemented "Gradient Accumulation" but when using the InfoNCE loss [6] the batch size has an significant impact when calculating the loss, because all other samples within the batch act as negative samples for a given sample. Gradient accumulation should only be used if no multi GPU setup is available or even gradient checkpointing is not enough to save GPU memory.

We use the implementation of OpenCLIP [2]. With this implementations all pre-trained checkpoint of OpenCLIP but also the checkpoints of OpenAI [4] can be used. We use only pre-trained models of OpenAI, because when fine-tuning on the dataset, these checkpoints always leads to better results than there counterpart of OpenCLIP trained on Laion-400m [5].

To evaluate a trained model on the given test split or own cross-validation splits we provide the script `evaluate.py`.

2.3 Final Predictions

For calculation of the distance matrix and generation of the submission files, the script `predict.py` is provided. It is possible to set more than one checkpoint in the config. In this case a `submission.csv` file is created in all corresponding checkpoint folders and also for the ensemble of this checkpoints.

3 Training and Results

3.1 Training Parameters

The model architecture stays the same as for Phase-I (Paper Submission) but our code base changed and we also changed the learning rate schedule. The bigger ViT-L/14 reaches the best performance on the test split after 4-5 epochs. We use AdamW [3] as optimiser with a polynomial learning rate schedule with warm-up, reaching a maximum learning rate (LR) of 4×10^{-5} after one epoch and a minimum learning rate of 1×10^{-5} at the end of the training. We only train for 4 epochs with label smoothing of 0.1.

Table 1: Hyperparameters used for training.

Hyperparameter	Value
Batch size	16
Training epochs	4
LR warm-up epochs	1
Max. LR	4×10^{-5}
End LR	1×10^{-5}
Label smoothing	0.1
Weight decay	0.01
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	10^{-8}
Seed	1

3.2 Results

We provide for both phases of the competition our used checkpoints for the final results.

3.2.1 Phase-I (Paper Submission):

The models of Phase-I were trained on an early code base with slightly other hyperparameters described in the paper. These models are trained on the complete training data (train+test) with the results shown in Table 2. The best checkpoint of Phase-I was also submitted for Phase-II.

Table 2: Results on the Challenge-Set (Paper submission).

Model	mAP [%]	rank-1 [%]	rank-5 [%]
ViT-B/16	97.56	98.72	99.36
ViT-L/14	**98.44**	99.15	**99.79**

3.2.2 Phase-II (Prize Submission):

For Phase-II we only tested if an ensemble of two models leads to better results. We use for this ensemble one checkpoint trained on only the training data using the test data for evaluation and a second checkpoint trained on all data (train+test). This ensemble is trained with the final code base.

Table 3: Results on the Challenge-Set (Prize submission).

Model	mAP [%]	rank-1 [%]	rank-5 [%]
2xViT-L/14	98.18	**99.36**	99.57

4 Discussion

For the final discussion we want to address two topics. First we discuss if it is possible to get even better results or are the results on the dataset saturated. Second we want to discuss the reproducibility and the influence of different seeds for the random number generators when using our approach.

4.1 Saturation of Results

Reaching a mAP score of over 98% and a rank-1 score of over 99% is an indication, that the results are quite saturated on the given dataset. If we have a closer look at what our model predicts for out of fold samples, we can hardly find any images that are assigned to the wrong player. Even if there is more than one player in a gallery image, the model does not get confused if the player's query image is appropriate.

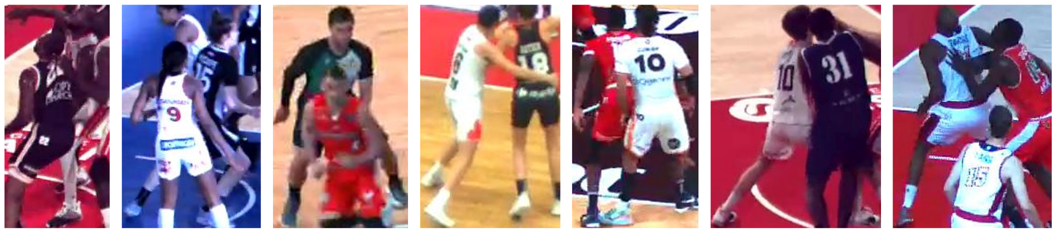

Figure 1: Examples for gallery images with more than one player.

But the model struggles, if the query image is bad or unclear towards which player we are looking for. Here are the most extreme examples of the challenge data with three query images where our model most likely will not find all corresponding gallery images.

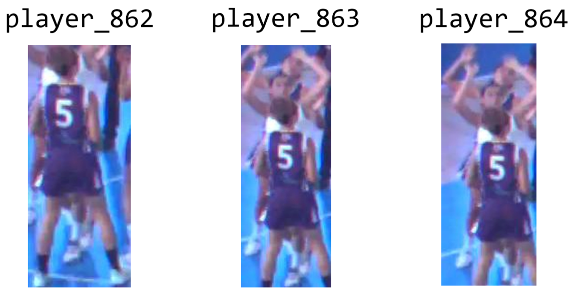

Figure 2: Unclear query images with more than one player.

Player ID 862 is an easy example, here it is clear that we are looking for the player in the foreground with number 5. But Player ID 863 and 864 are most likely the toughest samples. In this case, only a closer look at which player the bounding box belongs to would be helpful. Most likely, player 864 is the player in the background, as this bounding box is minimally shifted upwards. But for player 864 we do not know much about how she looks. We know she is the teammate of player 862 with the same jersey colour and has white shoes. Because the face are only some blurred pixels, the model will probably not find the correct gallery images for this player. Even for human annotators, this would be a complicated task.

Our ensemble produces a better rank-1 score, were as the single model of the first phase used for the paper leads to the best mAP score on the challenge set. But nearly all attempts for Phase-II to further improve the scores failed. More augmentation, a better sampling strategy also taking into account if the players within a batch comes from the same game or other loss functions does not lead to better results. It will be interesting to see how our approach performs on a more challenging not saturated dataset in the future.

4.2 Reproducibility

We provide the checkpoints of Phase-I and Phase-II in the GitHub repository. The results of Phase-I (Paper submission) were achieved with an early and slightly different version of our code base. For our final code base we fixed all random number generators to have reproducible results for a given seed. Only minimal differences can occur if using multi GPU training instead of single GPU training. This difference might come from the gradient reduction of all device gradients after the backward pass, when using DataPrallel of PyTorch for multi GPU training. The checkpoints of the ensemble were trained with our final code base using two GPUs and DataPrallel. Therefore, to reproduce the exact same results, training with two GPUs is required. We provide the config used for training in the checkpoint folders.

The initialisation of the random number generators for Python, Numpy and PyTorch has a significant effect on the results, despite we start from an already pre-trained network and do not not random initialise any additional parameters of the network when using the CLIP models. Different seeds and code changes can lead to deviations of up to 0.3% in mAP and rank-1 score for the results on the challenge set and even higher variations on the smaller test set.

As already described, when using the Info-NCE loss, all samples of the other players in the batch act as negative samples for an instance of a distinct player. That's one reason why the sampling order of different players per batch has a major impact on the final results. Additionally we random sample for a given query image a gallery image of the same player to create a pair of query and gallery image as instance for this player within in the batch. Both sampling relies on a random number selection, with only one constraint, that we have only one instance of the same player within the same batch. The fact, that we fine-tune a Vision-Transformer with a relative small subset of data, the used approach with the InfoNCE loss as training objective, random horizontal flip as augmentation and the custom sampling strategy leads to slightly different results when using different seeds.

Acknowledgement

The authors gratefully acknowledge the computing time granted by the Institute for Distributed Intelligent Systems and provided on the GPU cluster Monacum One at the University of the Bundeswehr Munich.

References

[1] K. Habel, F. Deuser, and N. Oswald. Clip-reident: Contrastive training for player re-identification. In *Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports (MMSports'22), October 10, 2022, Lisboa, Portugal*, 2022.

[2] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt. Openclip, July 2021.

[3] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[5] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[6] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.