TransMatcher: Deep Image Matching Through Transformers for Generalizable Person Re-identification

EECS 6322 (WI 2023) Reproducibility Challenge

Mithila Sivakumar
York University
msivakum@yorku.ca

Kelechi Otu
York University
kelechi@yorku.ca

Abstract

In this project, we present a reproduction study of the paper ***"TransMatcher: Deep Image Matching Through Transformers for Generalizable Person Re-identification"*** [9]. We implemented the TransMatcher model and reproduced the scenario where the model is trained on Market-1501 [13] dataset and tested on CUHK03-np dataset [7]. We evaluated the model's performance using the mean average precision (mAP), and Rank-1 metric and compared our results to those reported in the original paper. Our rank-1 and mAP are 16.5% and 17% respectively, which is 6% and 4% less than what is reported in the paper and hence, we were not able to successfully reproduce exact results as in the paper. All code and related materials are present here https://github.com/Mithila-Sivakumar/EECS-6322-WI-2023-Project

1 Introduction

In the field of computer vision, Transformers [12] have been gaining significant attention, particularly for feature representation learning such as image classification. However, the generalizability of them is unknown. So, in this paper, the authors investigate the possibility of applying Transformers for the problem of image matching and metric learning, given a pair of images.

First, the authors design two naïve solutions using Vision Transformer (ViT) [5] and vanilla Transformer and found that these do not improve the performance compared to S.O.T.A. (MGN, MuDeep, CBN, QAConv [8], QAConv-GS [10]). Hence, they propose a simplified decoder, which drops the full attention implementation with SoftMax weighting, keeps only the query-key similarity computation. In addition, they apply Global Max Pooling (GMP) inspired from QAConv-GS and 2 multilayer perceptron (MLP) heads to decode the matching result.

TransMatcher was built upon QAConv-GS, which has ResNet-50 [6] as their backbone network with three instance normalization (IN) layers further appended as in IBN-Net-b. The backbone network is pre-trained on the ImageNet dataset with states of the BN layers fixed. The authors show that this proposed technique achieves state of the art performance on datasets, with up to 6% and 7% percent performance gains in Rank-1 and mAP, respectively.

2 Reproduced Implementations and Experiments

For our project, we have chosen to recreate the main contribution of the paper, which is the custom Decoder implementation of TransMatcher. We proposed to reproduce two experiment scenarios from the paper. The first is, we train our TransMatcher model on the Market-1501 dataset and evaluate on the test set of the CUH0K3-np dataset. Second, we train our TransMatcher on the subset of the RandPerson [2] dataset and evaluate on the test sets of the CUHK03-np and Market-1501 datasets. Our

model's performance is evaluated with Rank-1 and mAP and is compared to the performance of state of the art (S.O.T.A) models in the same scenarios.

In the original TransMatcher, the authors have connected the encoder and decoder before stacking, in contrast to vanilla transformer where they connect after stacking. We have implemented our TransMatcher decoder on top of the official PyTorch project of QAConv-GS [4] just as the authors did with the original. We have used Resnet50-ibn-b [11] as the backbone of our network. We loaded the resnet50 model from its official github repository [1]. This backbone network is pre-trained on ImageNet with states of Batch Normalization layers being fixed. We have used layer3 feature map and also appended a 3x3 neck convolution layer to produce the final feature map as explained in the paper. The input image is resized to 384 × 128. The batch size is set to 64, with K=4 for the GS sampler. The network is trained with the SGD optimizer, with a learning rate of 0.0005 for the backbone network, and 0.005 for newly added layers. They are decayed by 0.1 after 10 epochs, and 15 epochs are trained in total. All these settings are exactly the same as expressed in the paper.

In our implementation of this model, we first pass the query and gallery images through the backbone network and the 3x3 neck convolution layer to get the final feature encodings. For the encoder, we have used the TransformerEncoderlayer class [3] from the official PyTorch implementation. In our case, there are 2 encoders and 3 decoders. where the input to the first decoder is from the 3x3 neck and input to the subsequent decoder will be from the corresponding encoder.

The decoder which we implemented has the following steps as shown in figure 1. The query and gallery encodings are passed through the fully connected layer 1 (FC1) to get the transformed features. Next, a dot-product is performed between the transformed features (which we have taken from QAConv-GS as explained in the paper). In the paper, they define a term ***"Prior Score Embedding"*** as similar to learnable FC weights of size (hw, hw), where h is height and w is width, so we have initiated the same using normal distribution between hw and hw and added it to nn.Parameter. This prior score embedding is then passed through a sigmoid, and the result is element-wise multiplied with the output of the dot-product in the previous step.

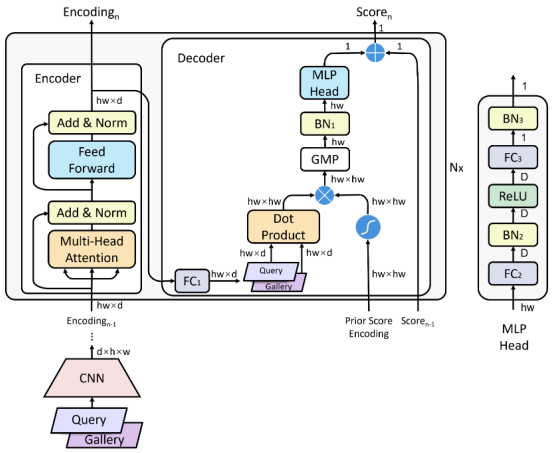

Figure 1: The structure of proposed TransMatcher for image matching from [9]

Train : Market, Test : CUHK03			
Method	**Venue**	**Rank-1**	**maP**
MGN	MM'18	8.5	7.4
MuDeep	PAMI'20	10.3	9.1
QAConv	ECCV'20	9.9	8.6
QAConv-GS	arXiv'21	16.4	15.7
TransMatcher	Ours	16.5	17

Table 1: Table showing comparison of Rank-1 and mAP (%) of our TransMatcher model with S.O.T.A

After that, a GMP layer is applied along the last dimension of hw elements. The authors have taken this from the QAConv-GS, so we have done the same. Finally, the output of GMP is passed through a Batch normalization layer (BN1) and then to two MLP heads, MLP head 1 (FC3, BN3) and MLP head 2 (FC2, BN2, ReLU). For the first decoder layer, the score is just the output of that decoder, else, the score is added with the prior score of the previous decoder. We re-used the dataloaders and parts of training and testing loop from QAConv-GS,

as the original TransMatcher also used them. For the loss function, the authors have used the pairwise matching loss based on binary cross entropy loss from QAConv-GS and we have reused the same.

We trained our model on the market-1501 dataset which contains 32,668 images of 1,501 identities captured from six cameras, with 12,936 images for training and 19,732 images for testing. We were able to successfully train the model on this dataset for 15 epochs. We tested our model on the CUHK03-np dataset which includes 1,360 persons and 13,164 images, with 767 and 700 subjects used for training and testing. The paper used only the detected subset and hence we tested using the same. As seen in Table 1, for this particular test scenario, our model, when tested on the CUHK03-np dataset, yielded ***16.5% Rank-1 and 17% mAP***.

We tried to train the model with RandPerson dataset. It is a synthetic dataset with 8000 persons and 1,801,816 images. The subset with 132,145 images is used for training. While we attempted to train, we faced a few issues which is discussed in the next section.

3 Discussion

We were not successful in reproducing the results of the experiments we wanted from the paper. For the experiment where we were supposed to train on the RandPerson dataset, we were not able to even get a result. Due to the large size of the dataset, Google drive automatically timed out and threw an error when we tried to load data from the folder. As a result, our training loop could not access the folder. This was an error that we could not foresee when we wrote the proposal, and given that google colab was the only way we could train this model due to our laptops machine constraints, we were left with no choice but to scrap this experiment.This left us with only one experiment that we could actually run.

As shown previously, when we trained our model on Market-1501 dataset and evaluate on CUHK03-np dataset we got results of ***16.5% Rank-1 and 17% mAP***. Although this is slightly better than the S.O.T.A, we could not achieve the whooping 5% and 6% increase in Rank-1 (22%) and mAP (21%) respectively, as they claimed in their paper. We are going to be charitable and assume the authors did not inflate their results, the following are reasons why we think our results differs soo much from theirs.

QAConv-GS has the same authors as this paper so when they built their TransMatcher model on top of the QAConv-GS framework they knew exactly what to do. This was very different for us. We found QAConv-GS code hard to understand with no proper comments or instructions. it took us days to get a working understanding of the framework and even then we found it extremely tough to build our model on top of it and to connect to it. Because of this difficulty, we believe that coding errors that we are not even aware of may have been introduced into our implementation.

There are also things that just weren't explained in the paper, for example, the prior score embedding weights initialization method was never mentioned. So we assumed normal distribution since it is usually the default. These weights are in every decoder layer and can greatly affect the outputs.

After we got our results we looked at their code to find reasons for the discrepancies. We saw that they use nestrov momentum for SGD but this was also never mentioned anywhere in the paper. For all these reasons, we think we could not exactly reproduce their result.

4 Contribution

A successful project is not only the result of the individual efforts of team members but also the combined contributions. In this project, both of us played a vital role in achieving the project goals and objectives. While we implemented the TransMatcher, Encoder and the backbone network together, Mithila, trained the model on Market-1501 dataset, while Kelechi tested on CUHK03-np detected dataset. The paperwork was split half-half.

5 Conclusion

In this reproduction study, we have implemented the TransMatcher model. While we were able to train the model on Market-1501 dataset, our test results on the CUHK03-np, however, were not the same as reported in the paper. Our rank-1 and mAP were 16.5% and 17% (Table 1) respectively, which is around 6% and 4% less than what is reported in the paper. With this study, we conclude that 1) It is not always possible to recreate/reproduce the model or network in deep learning space just by reading the published work. Some important details are missed in the paper and sometimes the writing is very vague. 2) It is hard to understand any existing projects used in the model and to create our implementation based on the flow of the existing project is complicated. 3) Some of the datasets they used might not be available now, or it may be too large making it unusable for training. In summary, while were able to create and train the TransMatcher model, we were not able to successfully reproduce the exact results.

References

[1] Enhancing learning and generalization capacities via ibn-net. https://github.com/XingangPan/IBN-Net.

[2] Randperson dataset. https://github.com/VideoObjectSearch/RandPerson.

[3] Torch.nn.modules.transformer. https://pytorch.org/docs/stable/_modules/torch/nn/modules/transformer.html#TransformerEncoderLayer.

[4] Transmatcher official github repository. https://https://github.com/ShengcaiLiao/TransMatcher.

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

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

[7] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 152–159, 2014.

[8] Shengcai Liao and Ling Shao. Interpretable and generalizable person re-identification with query-adaptive convolution and temporal lifting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 456–474. Springer, 2020.

[9] Shengcai Liao and Ling Shao. Transmatcher: Deep image matching through transformers for generalizable person re-identification. *Advances in Neural Information Processing Systems*, 34:1992–2003, 2021.

[10] Shengcai Liao and Ling Shao. Graph sampling based deep metric learning for generalizable person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7359–7368, 2022.

[11] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 464–479, 2018.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N

Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[13] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.