Metric Learning for Person ReIdentification task

by

Abhishake Kumar Bojja

Contents

List of Figures

Metric Learning for Person ReIdentification task

1.1 Introduction

Machine Learning tasks like classification, clustering typically require a measure of the distance between the given data points. This metric is generally chosen in the literature based on the given task and the domain of the data. Some standard distance metrics include Euclidean, Cosine. The data distribution in many practical tasks changes from the training data distribution, and we don't want to harm the model's performance because of this change. So, a good distance metric is required to represent the input data and achieve excellent performance [3, 4]. Metric Learning solves this problem by learning the distance metric from the data distribution.

Metric Learning projects the input data into high dimensional latent space vectors and reduces the distance between the latent space vectors of similar objects and increases the distance for different objects.

In this report, we study Metric Learning for Person Re-Identification Task, and the corresponding code written in Tensorflow 1.15 is available on GitHub at `https://github.com/abojja9/SiamesePersonReID`.

1.2 Person Re-Identification Task

Person Re-Identification task aims to determine whether an image or a video sequence of a specific person exists in a given database and determine his identity. It has

various applications in security and surveillance activities. One solves this problem by calculating the distance between the query person image and the images in the database. It is a challenging problem due to the presence of images from different camera sensors, viewpoints, and illumination changes. Since we are looking to match similar images, we solve this task with Metric Learning.

1.3 Data

We use MARS (Motion Analysis and Re-identification Set) Dataset [2] to learn this task. The dataset contains 1261 person IDs and more than one million examples. The provided training set contains 500,000 samples, and the test set contains around 700,000 samples and is of resolution (256×128). But due to the constrained resources and limited time, we use 100,000 samples with 100 unique IDS for training the deep learning model. My training set contains 80,000 examples, and the validation set has 20,000 samples. We store the data in the form of tf_records, and the code to generate them is available in the file $data\_tfrecord.py$.

1.4 Network Architecture

We use the Siamese Network [1] to perform the Person ReIdentification task. A Siamese network consists of twin networks with shared parameters and is joined by a merged network with a distance metric function at the top. The network takes a pair of images and are passed to each of the twin networks to produce a high dimensional representation for the inputs. The metric function then computes the distance between these features. Our network architecture is shown in the Figure 1.1. It consists of a base network and a merge network. Our base network consists of four convolutional blocks, followed by a dense layer to represent the image in high dimensional space. Each convolutional block consists of a series of conv2D, maxpool2D, batch norm, and relu layers. The merge network consists of layers. The first layer calculates the absolute difference between the high dimensional features of both input images. The second layer is a dense layer that produces a single unit output passed to the Sigmoid layer to classify input images. The network architecture is implemented as a function with the name *network* in the file $train\_ops.py$.

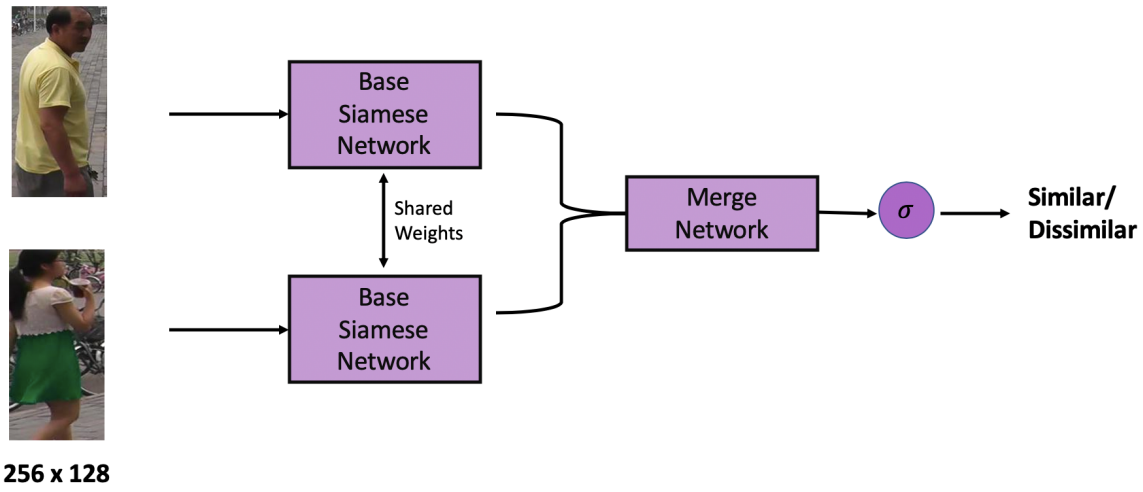

Figure 1.1: Siamese Network architecture.

1.5 Loss function

Loss functions play a key role in the success of any Machine Learning task. In our task, the objective of Metric Learning is achieved by the loss function. There are many loss functions available for the Metric Learning task in the literature. The most commonly used loss functions with Siamese networks for metric learning tasks are Identity loss and Contrastive loss.

1.5.1 Contrastive Loss

Contrastive loss is a distance-based loss function. For a given pair of input images, the contrastive loss is defined as

$$L_{contrstive} = Y_{\text{true}} * D^2 + (1 - Y_{\text{true}}) * (\max(\text{m} - D, 0))^2 , \tag{1.1}$$

where, D is the Euclidean distance between the Embedding features output from base Siamese network. Y_{true} is the ground truth label and is equal to 1 if both the input images are similar and 0 if dissimilar. In the above equation, the first term contributes to the similarity loss. If the input images are similar, the loss will be 0 only when the network produces the embeddings whose distance is 0; otherwise, loss will be

accumulated. The second term contributes to dissimilar loss and is 0 only when the gap between the embeddings representations is greater than the margin m. When the distance between the embedding vectors is less than m, then the dissimilarity loss is definite. Thus Contrastive loss tries to minimize the gap between embedding representations for similar images and enlarge the distance for different images.

1.5.2 Binary verification Loss

Binary verification Loss treats the Person Re-Identifaction task as a binary Image classification problem. The ground truth label y_i is assigned to 1 if the input images are similar and 0 if dissimilar. For an input image pair x_i with label y_i and predicted probabilities p_i the cross entropy loss (L_{ce}) is defined as below.

$$L_{ce}(y, p) = -\sum_{i} y_i \log(p_i) \ . \tag{1.2}$$

The Contrastive Loss and Binary verificarion loss are implemented as *contrastive_loss.py* and *identity_loss* repectively in the file *train_ops.py*.

1.6 Training Settings

The training code is implemented in the file *train_net.py*. The optimizer we used is Adam and is implemented under *_build_optim* function of *SiameseNet* class. We start with a lower learning rate and are set as 0.00005. From my limited experiments, the loss was getting higher with larger rates. The batch size is set to 32 and margin m is set as 1.0

1.6.1 Evaluation Metrics

We use accuracy as a measure to understand how good our model performs. The *accuracy* function is implemented in the file *train_ops.py*.

1.6.2 Tensorboard Logs

The training logs are written into the tensorboard to track the training progress, and the corresponding code is implemented in *_build_model* function of *SiameseNet* class of *train_ops.py*.

1.7 Experiments

With a limited amount of time and resources, I had done a few experiments to train the Siamese network model. My first experiments include training models with a contrastive loss function.

1.7.1 Contrastive Loss Experiment

For the Contrastive loss experiment, loss and accuracy plots are shown in figure 1.2. The training loss (red color in the bottom left) is much lesser than Validation loss (pink color in the bottom right), and Training accuracy reached above 90 (%) within the initial phase of training. This shows that that model is overfitted to the training data. This model cannot generalize well during the inference phase. The overfitting problem can be reduced by adding regularization to the model while training.

1.7.2 Contrastive Loss with Regularization techniques Experiment

To regularize the model, we used Dropout and Data augmentation techniques. We have added Dropout layers at the end of every Convolutional block and the Dense layer of the Base Siamese Network discussed in section 1.4. The dropout rate is set to 20 (%). For data augmentation, we randomly flip images horizontally.

By adding the regularization, the accuracies and losses in both the training and validation phases for the model are comparable as shown in the figure 1.2. The new model achieves the best validation accuracy (grey color in the figure) around 82 (%) within fewer iterations than the one without regularization (pink color). The model with regularization is my current best model.

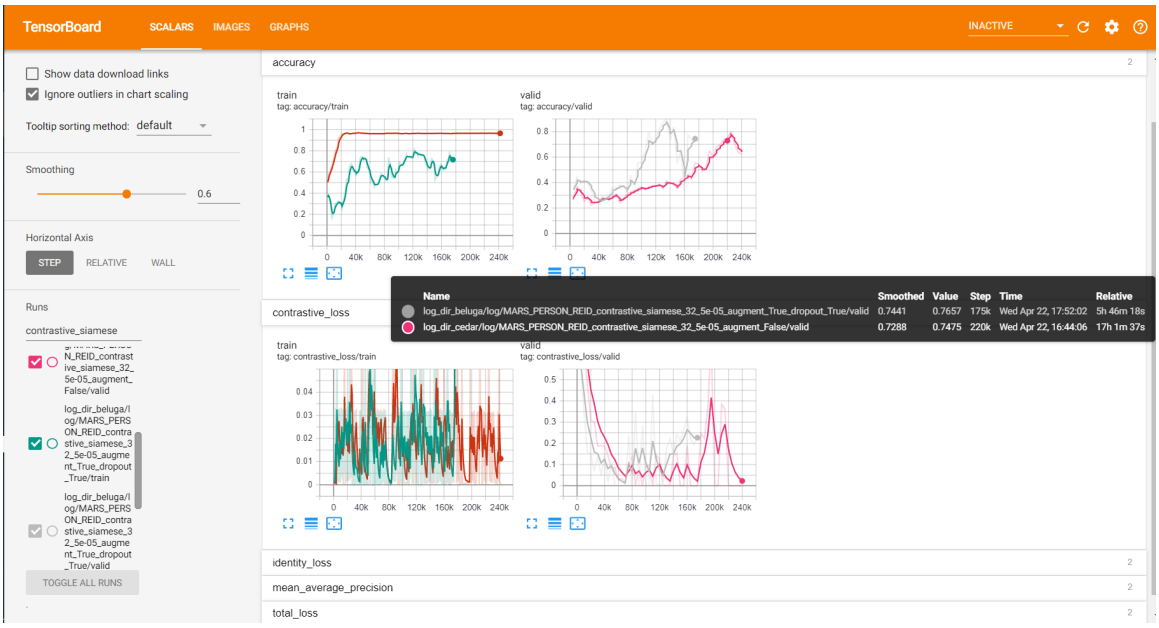

Figure 1.2: Experiments with Data Augmentation

1.8 Qualitative Results

The evaluation framework to generate distance metrics and qualitative results given a pair of images is implemented in the class *EvalSiameseNet* of *test_net.py*. The distance metrics are defined in the function *contrastive$_l$oss*. The Qualtitative results of the model are shown in the figures 1.3. The model does a decent job in identifying similar and dissimilar images. We can see that the model has learned distinguishing between both the similar and dissimilar images. On the top of each image, the distance metrics are dispalyed. .

1.9 Methods to imporove the results

We can explore the following methods to improve the performance of the model on Person ReIdentification task.

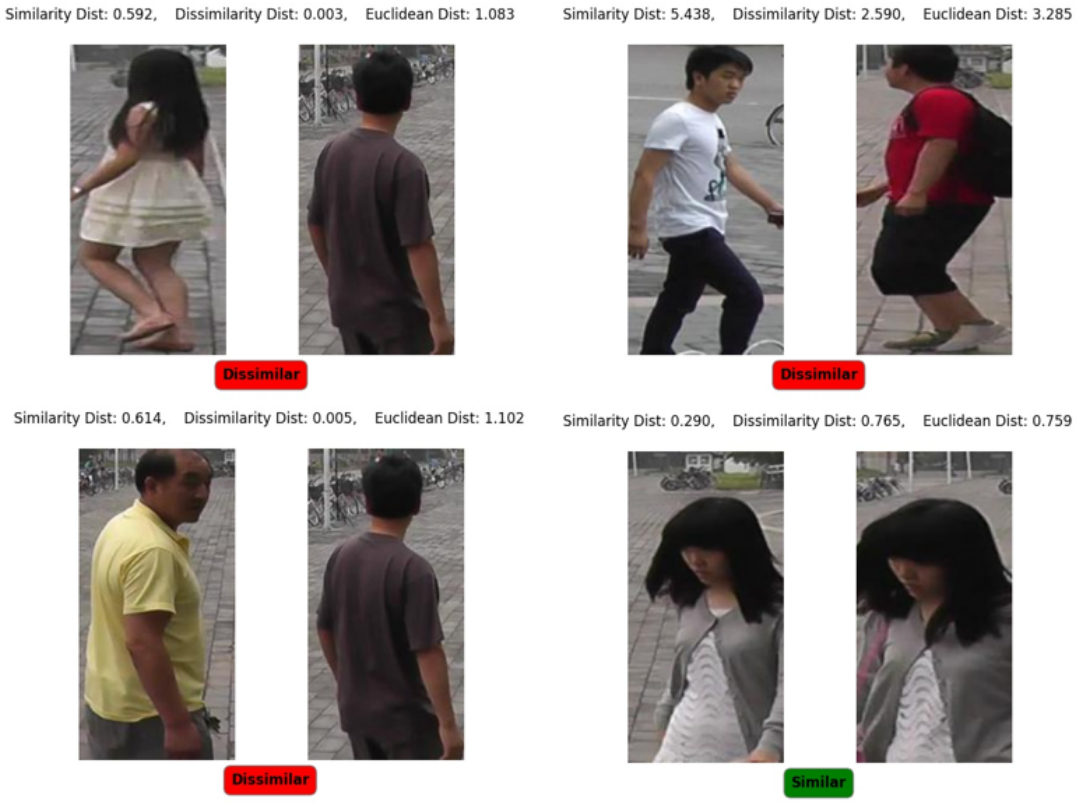

Figure 1.3: Qualitative Results

1.9.1 Transfer Learning

The success of the model in our task depends on the high dimensional embedding features generated from the base Siamese network. Using a pre-trained model can give a better representation of the images in high dimensional space and also help to train the model quickly. We can replace the base Siamese network with a model with pre-trained weights from Imagenet or any model trained for the Person ReIdentification task.

1.9.2 Increase the Training Data

Currently, I'm using only 20 % of the training data from the MARS dataset. The model can be trained with the full dataset to improve the performance of the model.

1.9.3 Explore Loss Functions

For my experimentation, I'm using Contrastive loss. More experiments can be conducted to train the model with only Identity loss and also the combination of both losses.

A Triplet Loss [5] uses triplets of images (original, positive pair image, negative pair image) in every iteration, and the loss tries to minimize the distance between the original and positive pair image and increase the gap between the original and negative pair image. Since the model sees similar and dissimilar images, the model can learn better representations.

1.9.4 Data Augmentation and other techniques

More data augmentation techniques can be added to improve the model's generalizing capability. Also, the current model has only a few layers. The number of layers can be increased, and hyperparameter tuning can be conducted to improve the model's performance.

1.9.5 Modifications to Merge Network

The merge network takes the absolute difference between embedding features of input images. The absolute difference can be replaced with other distance functions like cosine distance, which can provide more information on how close the embedding vectors are.

1.10 Conclusion

In this project, I have implemented the training and evaluation framework for the Person Re Identification task and trained the designed Siamesese network on MARS dataset. I have provided the initial results and suggest methods to improve the performance of the model.

Bibliography

[1] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[2] *MARS: A Video Benchmark for Large-Scale Person Re-identification*, 2016. Springer.

[3] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[4] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[5] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification: A survey and outlook. *arXiv preprint arXiv:2001.04193*, 2020.