Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2021 | LIU-IDA/STAT-A--21/017--SE

Person Re-Identification in the wild

– Evaluation and application for soccer games using Deep Learning

Vasileios Karapoulios

Supervisor : Amirhossein Ahmadian
Examiner : Johan Alenlöv

External supervisor : Ludwig Jacobsson

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`.

© Vasileios Karapoulios

Abstract

Person Re-Identification (ReID) is the process of associating images of the same person taken from different angles, cameras and at different times. The task is very challenging as a slight change in the appearance of the person can cause troubles in identifying them. In this thesis, the Re-Identification task is applied in the context of soccer games. In soccer games, the players of the same team wear the same outfit and colors, thus the task of Re-Identification is very hard. To address this problem, a state-of-the-art deep neural network based model named AlignedReID and a variation of it called Vanilla model are explored and compared to a baseline approach based on Euclidean distance in the image space. The AlignedReID model uses two feature extractor branches, one global and one local feature extractor. The Vanilla approach is a variation of the AlignedReID which uses only the global feature extractor branch of the AlignedReID. They are trained using two different loss functions, the Batch Hard and its soft-margin variation. The triplet loss is used, where for each loss calculation a triplet of images is used, an anchor, a positive pair (coming from the same person) and a negative pair.

By comparing the metrics used for their evaluation, that is rank-1, rank-5, mean Average Precision (mAP) and the Area Under Curve (AUC), and by statistically comparing their mAPs which is assumed to be the most important metric, the AlignedReID model using the Batch Hard loss function outperforms the rest of the models with a mAP of 81% and rank-1 & rank-5 above 98%. Also, a qualitative evaluation of the best model is presented using Grad-CAM, in order to figure how the model decides which images are similar by investigating in which parts of the images it focuses on to produce their embedding representations. It is observed that the model focuses on some discriminative features, such as face, legs and hands other than clothing color and outfit. The empirical results suggest that the AlignedReid is usable in real world applications, however further research to get a better understanding of the generalization to different cameras, leagues and other factors that may affect appearance would be interesting.

Acknowledgments

I would like to thank Amirhossein Ahmadian, who was my supervisor at Linköping University for his help and his valuable feedback throughout my thesis.

I would also like to thank my extrenal supervisor at Signality, Ludwig Jackobsson, for his support through this period. I feel glad for trusting me in this challenging task and all the guidance I received to accomplish it.

Finally, I am thankful to my parents for their full support in order to pursue this master's programme and their continuous encouragement to fulfill my goals.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Nowadays there is a massive increase in practices and technologies that perform very well in person identification. However, in many real-world applications, cameras are not able to capture the person very clearly continuously. Thus, lately there is a huge progress in the field of Person Re-Identification (ReID) using deep neural networks. Re-identification is the process of associating images of the same person taken from different angles, cameras and at different times. Person re-identification is implicitly related to tracking which is a very challenging task. Tracking is used to differentiate the persons in a scene and identify their movements. Originally, the most common application is for surveillance purposes.

For this purpose, models are trained on this task of re-identification. However, there are some challenges that rise. First and foremost, the resolution of the images/videos is low in most cases which makes it harder for the model to detect features. Moreover, in many cases the person might have different poses, might be captured by different viewpoints or even some parts of the body might be occluded. These make the task of the identification even tougher. Finally, there are usually multiple cameras used, thus any change in the outfit in between the captures of the cameras, for example putting on a jacket or changing clothes, can confuse the identification and as a result the same person is identified in the different scenes as two different persons.

Furthermore, the recent years re-identification started being applied in sports in order to keep track of the player's movement during the game. By tracking each player, there are a lot of information that can be extracted for each individual and this information can later be used for different purposes. For instance, in soccer there can be information stored regarding the positioning of a player, his performance, his strengths and limits. Having this information, coaches can give performance feedback to the players so that they can improve or take advantage of their movement in the field to adjust the team's tactics and take decisions. Also, this gives the opportunity to teams to hire new players based on their previous stats which provide an objective rating regarding performance.

In addition, it is a very handy tool for media and betting. In the case that this is able to be applied live during a game, it can offer an engagement to the fans and a better experience to the viewers. For example, a viewer could keep track of the stats of their favorite player each second of the game, how much distance they covered so far, how fast they run and

their movement in the field. Also, this information could compose a wide betting part, where people can bet for instance which player will run faster.

The aim of this thesis is to identify players in a soccer game. By identification, only relative identification is needed, which means a separation of each player on the field from each other. The absolute identification (classification) is done already, and it is achieved by detecting the jersey number of each player. The reason why the relative identification is useful, is because the jersey number is most of the time not visible by the camera.

We would like to associate all the images of the same person together and assign them to the same track and as soon as one of the images is identified by the model used for recognition, we will know that all the images that are associated with that player (exist in the same track) belong to the same person. This could also work the other way around, in the scenario that all players are identified in the beginning, right before the match starts, and as a consequence we do not require the model that recognizes the players anymore because all the consequent detections will be associated and identified immediately.

A challenge here is that the players of the same team wear the same outfit and colors. Thus, the model needs to learn some more discriminative features of their body, other than simply detecting the outfit or the appearance.

1.2 Problem Foundation

The task of associating images of the same person with other appearances, more specifically, is related to retrieving all (or top N) images from the collection that belong to the same person. To begin with, we have a dataset D that consists of tuples in form of (Img, (Player_ID, Team_name+Game_ID)) pairs, where Img is the image of the player in form of a crop containing the person, Player_ID is the ID of the player and Team_name+Game_ID is the team plus the match that the player is playing in (where match is a unique ID). We will train the model using these information and the image is going to be used to produce the embedding and the (Player_ID, Team_name+Game_ID) pairs are going to be used for the loss function.

The output O of the model is a vector which serves as an embedding representation of the images. Eventually we expect the model to learn a new space where the embedding representations of the images have a meaning, that is the embedding representations of similar images will be close in that space and the dissimilar images will be further. We will call the distances between the embeddings of similar images «positives» and the distances between the embeddings of dissimilar images «negatives», referring to positive and negative pair matches. During testing, we would like to provide an image Img' as an input to the model and predict the embedding representation O of it, using our model. We will do it for all our test images Img'. We will use one embedding as a query Q and we will retrieve the top-50 most similar embeddings using the Euclidean distance. Then, we will create a ranked list of the similar images retrieved to the query Q. The top image on the list will be the most similar one and ideally we expect the positives to be on top of the list.

In reality, this testing phase will be performed in a single game. The reason is that we would like to separate the players during a game, thus we only need to separate the images of the players that participate in the same game. For this study however, we have more than one games to test on, so we will use them one by one in order to evaluate the performance of the model on different games. We will use 40 images per player for each team. This means, that we will evaluate the performance using (40 images per player) * (11 players per team) * (2 teams per game) = 880 images at a time.

Furthermore, we will investigate the interpretability of the model by plotting the activations A of the last convolutional layer of the 5th Block over the original image as a heatmap. Through this, we will try to interpret how the model produces the embeddings by observing which parts of the image the model identifies as important and actually differentiate each player.

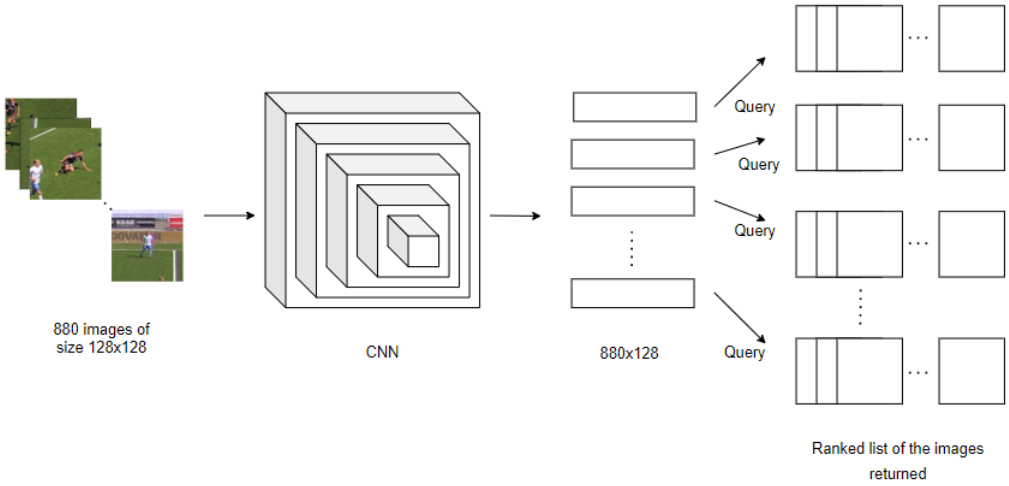


Figure 1.1: The above figure represents the testing phase graphically. This procedure is repeated for each game in the test data. The input is a batch of size 880, which contains 40 images for each of the 11 players of the two teams participating in the game.

1.3 Related Work

In general, state of the art approaches propose models, the so called embedding networks, based on convolutional neural networks. They are proven to be able to learn efficiently features in an image and in the end they contain a layer that serves as an embedding. And here is the importance of the embeddings. That is, to accomplish this relative identification in our case, we will use those embeddings learned to compare the images of the players. This way, we will try to learn efficiently the embedding representations of each player-image in order to be able to find similar images using the distance in this embedding space. All the proposed methods below, differ in the part before the embedding layer (the last layer), which means they propose different type of architectures before the layer that corresponds to the embedding.

One approach as described in [21] [16] is to use the so-called pose-guided methods. These methods refer to training a neural network not only on extracting features of the image but also on different poses and viewpoints of the persons and thus being able to recognize them based on the way they stand since «a person's pose and orientation can greatly affect the visual appearance in the image» [16]. The training on different poses is done in multiple ways. For example, in [21] it is done by training the network to detect different parts of the body, also referred as key-points, so that the key-points extracted are robust to different poses and viewpoints. The network is trained to detect the head, the upper-body and the lower-body and then combine the features extracted from the image as a total with the features extracted by the detected parts to form the embedding.

In contrast to this, in [16] they propose a different kind of pose information. That is a different set of key-points, which indicates if the person's orientation is in front, back or side. The model this way learns to extract features based on the viewpoint of the person.

Another approach as described in [7] [11] is the mask-guided methods. These methods consist of applying the semantic segmentation technique to the image in order to remove the background noise from the image and extract a segment with the person's body only. Intuitively, there is a binary mask applied to the image in order to remove the background. Because of the fact that other methods, like the pose-guided ones, rely on detecting the parts placing a bounding box around them, the parts will contain not only the human body but also some background. Or they may even be inaccurate. And this can make the task very hard, especially if there is another person's body captured within that box. Thus, these methods

try to remove as many pixels do not belong to the person as possible and let the model this way learn more efficiently the features.

As proposed in [7], the model is trained on the segmentation task in order to extract features from the segment of the whole body and local features from the segmentation of the human body parts. The advantage of this is that the parts that are extracted via segmentation do not contain background noise and thus make it easier for the model to extract local features in each body part. Another similar approach is the one proposed in [11], where in this case the difference is that the network is trained to segment the whole body of a person and the features learned are related to the whole body of the person.

Stripe-based methods are similar to the pose-guided methods described above and they are described in [22] [19]. The main difference is that in these methods the feature extraction from the parts of the body is performed by horizontally splitting the image into parts, whereas in the pose-guided methods the model is actually trained to detect those parts. And by splitting the image horizontally into parts, the model tries to learn some more discriminative features, as in pose-guided methods, on different poses and viewpoints.

In [22] the model proposed consists of two parts, one for extracting features from the whole image and one extracting features from the parts that the image was split in. In contrast, [19] propose a model that aims to recognize the person by just using the features extracted by the local parts, where those parts are produced again by splitting the image horizontally.

GAN-based (Generative Adversarial Network) methods are extensively explained in [26] [12]. The idea of these methods is to take advantage of GANs in the very first step to improve the quality of the image, change the person's appearance using style transfer or even change the person to a desirable pose. That way, some common challenges can be overcome, such as the feature extraction of the persons will be "invariant" to colors and patterns in clothing or the model will not have to worry about poses and viewpoints. Also, that way the training dataset is enriched with more images.

In [26] the model proposed is trained to generate new images using style transfer, which works as a data augmentation technique. The model thus learns to extract features that do not have to do with clothing colors. On the other hand, [12] propose a model that is trained to generate images on a pose and viewpoint of interest. For example, for an input image it generates an image that contains the same person from the front side. Thus, the model afterwards, does not have to consider poses and viewpoints.

Last but not least, there are the global feature-based methods [5] [18], which basically rely on the feature extraction of the person's body using the whole image. It is the least complex method of all mentioned so far, however it does not lack a lot in performance. In [5] the model proposed is simply a convolutional network to extract the features from the image, while in [18] the approach incorporates an eigenlayer, in which the aim is the inputs – which are the features extracted from the image – to be orthogonal. The challenge that the approach tries to overcome is the difficulty of an unbalanced dataset, that is the embedding representations of images-persons with similar features will be highly correlated. For instance, if we have a lot of images that are similar, such as persons that wear red or pink clothes will cause the weights to be correlated. This way, the embedding representations of the images will be less correlated.

All of the above proposals theoretically perform very well in the ReID task. However, in our case there are some limitations that are impossible to overcome. Most of the approaches require some extra labeling in the data that we do not have. For example, pose-guided methods propose models that are trained on detecting the pose/viewpoint of the person in the image. This is achieved by adding a detection task, which needs target values such as categories of the viewpoints (front,back or side) or target values related to classification of parts plus the coordinates of each part. Also, the mask-guided methods require masks in order to train the model for the segmentation task.

We also decided to use data augmentation during training and avoid training another network (GAN) for this purpose. Thus, the ones that we can actually take advantage of are the stripe-based and the global feature-based methods. We decided to use both of them so that we perform a comparison to figure which one performs better. Regarding stripe-based methods, we decided to use the approach proposed in [22], AlignedReID, as it is a very interesting approach using the alignment between the images of the local parts. For the second model, the model proposed in [18] is suggested for the case that there is an imbalanced dataset which is not the case. We could use an approach like [5], but instead we concluded using a modification of the AlignedReID in order to adapt it to the global feature-based methods and make it essentially similar to [5]. This way, we will have strong grounds for the comparison between the two models and the two methods in general.

1.4 Objective

As stated above we will evaluate two of the current state of the art approaches and the baseline model. The baseline simply uses the Euclidean distance in the image space. We will train a proposed Vanilla model, which will be a variation of the AlignedReID, by removing a branch and adapting it more to a different group of methods (global feature-based), as well as the AlignedReID model [22] with a minor modification. This way, we will have reasonable grounds to do some statistical model comparison between the models and conclude which one performs better in this specific task. Moreover, we will use two loss functions for the deep learning models, the Batch Hard Loss and its soft-margin variation as proposed in [5]. We will try to figure which of the two is more efficient in learning the embedding representations.

We will try to perform some innovative evaluation between the AlignedReID model based on [22], the Vanilla model and the baseline. It is very interesting to evaluate how these approaches perform in the available data we have, as the deep learning methods mentioned above have been evaluated so far in some open-source datasets, which are often well-structured by their nature (e.g. Market 1501 [25], MARS [24], DukeMTMC [14] and relatively easier to achieve high performance. In our dataset, the persons we want to separate look alike, in terms of outfit and colors.

Afterwards, there will be a qualitative evaluation of the best model's performance by applying the Grad-CAM [1] tool. We believe that it is of importance to also focus on the interpretability of the model to assess its quality, as it is not explicitly done by other people for the specific model architectures we will implement. Mainly the reason could be that Grad-CAM is conveniently applied to models that contain a classification layer, because the most common way of applying this tool involves using the target class. The Grad-CAM tool is related to visualizing the activations of the last convolutional layer, producing a heatmap over the original image showing in which parts of the image the model focuses to produce the embedding. Last, we will visualize the embeddings in the new space learned to identify the distances between similar and dissimilar images.

We will prove which of the models is better in this task by statistical comparisons. For instance, we will do some hypothesis testing on the distribution of positive/negative embeddings. As stated in Section 1.2 positives are the distances of the pairs of images that belong to the same person and negatives are the distances of the pairs of the images that belong to different persons. After having formed the embedding space, we will be able to retrieve the groups of the positives and the negatives. These two groups form two distributions where the value in the x axis represents the distance from a given embedding. The hypothesis will refer to the fact that the mean of the distribution of the embeddings of images that belong to the same person (positives) should be lower than the mean of the distribution of the embeddings of the images that belong to different persons (negatives). Also, those two distributions ideally should not overlap. Finally, we will inspect the form of the embeddings learned in

this new space and we will argue which model is able to separate better the embeddings of similar images based on the clusters they form.

The goal of the thesis as stated previously is to associate images of the same person based on previous appearances. That is, given an image we would like to retrieve the top-50 images of the same person. For this purpose, we will use the distance to compare the embeddings and retrieve a list ordered from the most similar (lowest distance) to the least similar (highest distance). We will use several metrics to evaluate the retrieved images, such as AUC (Area Under The Curve), rank-1, rank-5 and mAP (mean Average Precision).

In summary, this thesis addresses the following points:

1. Train and evaluation of the Baseline, Vanilla and AlignedReID models for person re-identification in soccer games using the Batch Hard loss function and its soft-margin variation for the last 2 models. The losses will use a triplet of images each time, during the learning process.

2. Qualitative evaluation using Grad-CAM to produce heatmaps over the images in order to interpret where the model focuses on the images to produce the embedding representations. The aim is the model to learn some more discriminative features other than the color of the outfit or the jersey number.

3. Statistical comparison between the models and statistical evaluation of the results of the best model. The comparison will be performed with a hypothesis test using the resulting evaluation metrics produced by the models to investigate if there is a significant difference between the performances. The statistical evaluation refers to performing a hypothesis test between the distributions of the distances of the images that belong to the same persons and the distances of the images that belong to different persons. This way it will be confirmed if the model learned how to separate similar/dissimilar persons by having the distances of dissimilar images significantly greater than the distances of similar images.

2 Data

2.1 Data Sources

The data that will be used are provided by Signality and they were extracted from football matches of the Swedish Football League "Allsvenskan". There are approximately 60 games used to extract the data, consisting of matches between 16 teams. Also, there are 11 players per team and approximately 1500 images (crops) per player per game. Moreover, there is a dictionary corresponding to each image, containing information like Game ID, Player ID, Jersey Number, Team Name etc.

2.2 Raw Data

More specifically, there is a deep learning model that is trained by Signality on detecting the players during a soccer game and it places a bounding box around the detected person. Afterwards, the content of this bounding box plus an extra padding that is added around that box are returned as an image and stored along with a dictionary containing the necessary information. That information can be seen in detail in the next table 2.1.

Game ID	Team name	Team	Jn	Height	Width	Padding
f539feeb	Malmö FF	Home team	18	117	163	40
c3937f609	Örebro SK	Away team	21	79	81	48
e32abdd1b	Kalmar FF	Away team	5	68	132	52
2dc5f8123	IFK Norrköping FK	Home team	1	74	130	50

Table 2.1: This is an example of the corresponding information in the dictionary that belong to the image crops. Jn denotes the jersy number. The numbers and IDs are random, as they do not really provide any specific information here.

In Figure 2.1, we can observe how an image in the dataset looks like. The information like height, width and padding are related to the extra padding that is added by the pretrained detector model to the detected bounding box, before the image is stored. This information is visualized in Figure 2.2.

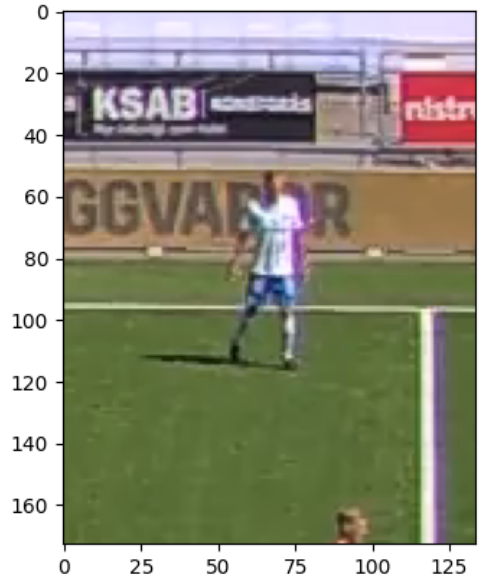


Figure 2.1: This is an example of an image crop in the dataset. This is the image that the model used for detection produces, including padding.

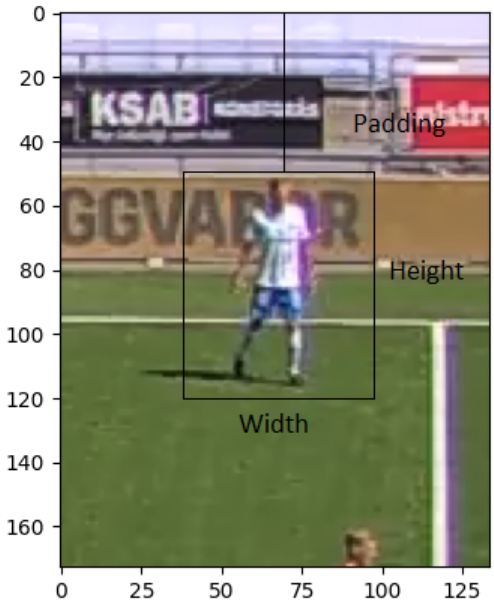


Figure 2.2: This is an intuitive example of the height, width and padding on the detected player. We will eventually keep the part of the image inside the box.

2.3 Noisy Data

As mentioned before, the data is produced by another model that is used to detect the players. Even tough there is human intervention in the annotation, there are still images that are wrongly registered to some players. We assume though that it does not affect the training, since the number of the wrongly annotated images is not significant and the images are sampled randomly per batch. However, it may affect slightly the evaluation of the performance, in terms of producing deceptive numbers for the metrics that will be used. There have been observed cases where the model returns correctly similar images but some of them are wrongly annotated and thus are considered wrong for the evaluation. Also, the opposite is possible, which means that the model returns wrong images as similar but due to annotation they are considered correct in the evaluation. Nevertheless, no such cases have been observed.

2.4 Data Preprocessing

To use the data introduced above, we need to do some preprocessing. First, we would like to remove the padding that is added to the detection. That way the crops we will have will contain mostly the body of the player and not much of the background. Additionally, we standardize first the image in the interval [0,1] and then we normalize the image using mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] per channel because we would like each feature to have similar range so that the gradients are more stable. Finally, since all images vary in height and width, we need them to be of equal sizes, so that they can be provided to the network as inputs. Thus, we resize every image to 128x128. We would like to avoid using bigger images, because that would result in adding more parameters to the network. In general, the images have an average size of approximately 140x190.

Regarding the formulation of the dataset, we read all the json files serially and we create a dictionary. This dictionary is composed of keys, that consist of a string denoting "TeamGameID", and values that are typically another dictionary (nested) that contains jersey numbers as keys and the index of the image as values. It is necessary to use the GameID information, since there can be cases when players of the same team wear different outfits in a different match, or some specific players differ in appearance, for instance they have different hairstyle or different accessories. For this reason, it would be confusing to provide the network with the information that a player that satisfies the above examples in different games has the same appearance. We instead want the network to learn different embedding representations for practically the same player but with different appearance.

Furthermore, we apply some augmentations in order to improve the performance of the model. These augmentations consist of bounding box augmentations, cutouts, rotations and flips. Each of the augmentations are applied with a probability of 0.5 each time. A bounding box augmentation refers to manipulating the bounding box slightly using translation. Translation is defined as moving the image randomly in either X or Y axis. Cutouts refer to randomly generated areas in the image that are replaced with noise. This way we want to make the model more flexible to recognizing persons when parts of the image are occluded. Last but not least, we also use rotations and flips of the image to introduce some diversity and variance to the data, aiming for a better generalization by the model.

The images in Figure 2.3 show the results of some random augmentations of the data. In all images, the bounding box augmentation is applied, but the effects are not always clearly visible since the offset is generated randomly and it is usually small. For example, we can clearly observe the difference in the bounding box augmentation in the two top images. The top left image includes cutout augmentation, the top right includes a rotation, the bottom left cutout and rotation and the bottom right is flipped.

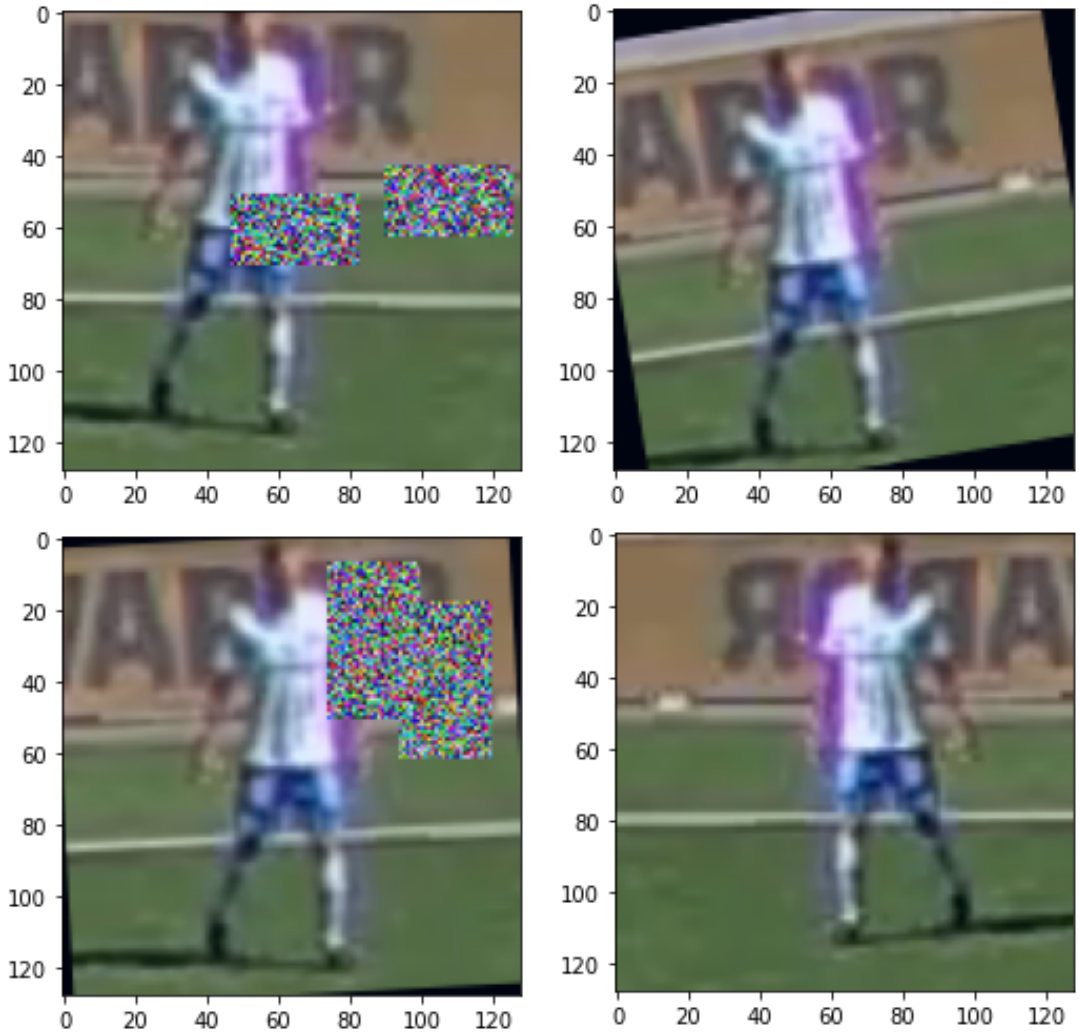


Figure 2.3: Demonstration of different augmentations on the same image. For the sake of plotting, the normalization is removed.

Finally, we split the dataset into training, validation and test set. We will use 10% of the games as test data, 20% as validation data and 70% as training data. That means, including the possible augmentations, we have approximately 5 million images for the training. Validation and test data will contain some of the teams and all their players that the model has been trained on, but in different (unseen) games. It is also possible that some of the teams have the «away» outfit in those data, while trained on the «home» outfit and reverse. The validation data will be used during the training to keep track of the model's performance in generalizing and save the best model based on the distance between the distribution modes, and we will evaluate the model on the test data.

3 Theory

3.1 Neural Networks

A neural network is a mathematical model that is able to recognize relationships in data and learn how to predict based on the knowledge acquired through a process called "training". It consists of neurons, weights that correspond to the connections between the neurons and biases connected to each neuron. The structure of the neural network can be separated to layers, which are groups of neurons that are not connected directly. The layers can be distinguished between input layers, hidden layers and output layers. A simple neural network is shown in 3.1.

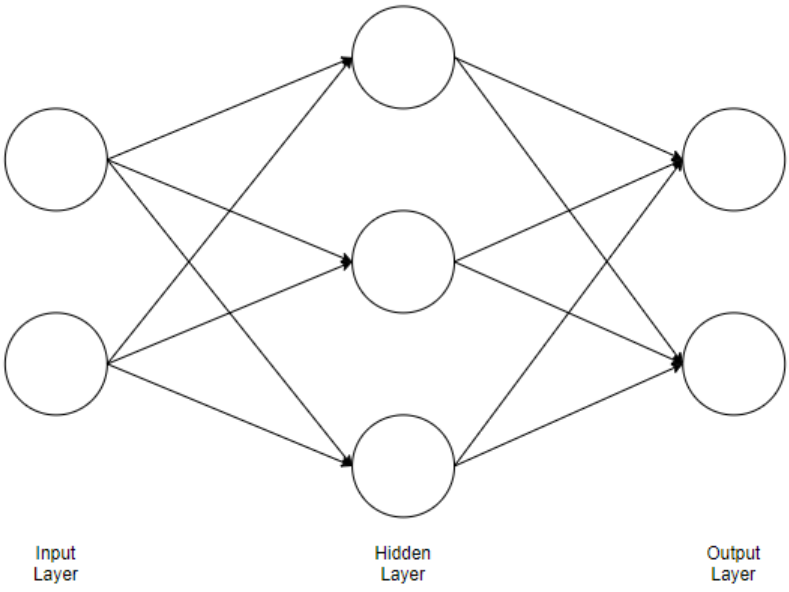
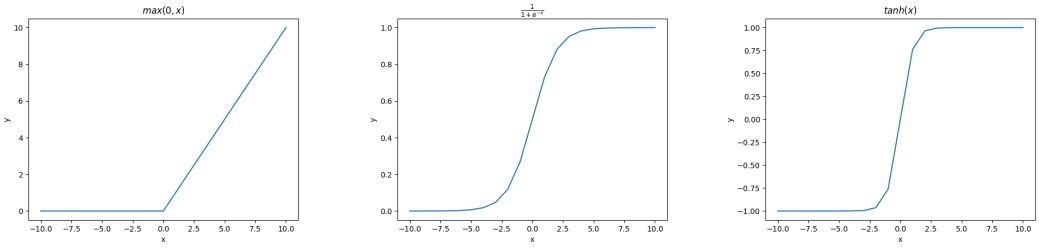


Figure 3.1: This is a simple neural network structure. It consists of an input layer of two inputs, a hidden layer composed of three neurons and an output layer composed of two outputs.

The neural networks also contain activation functions, which are non-linear transformations of the input of each neuron. The most commonly used are ReLU, Sigmoid and TanH [3]. They are important because they introduce some non-linearity to the model in order to learn complex patterns in the data. In Figure 3.2, we can observe the behavior of these activation functions.



ReLU Sigmoid TanH

Figure 3.2: ReLU, Sigmoid and TanH activation functions graphically.

The information flow from the input layer to the output layer and the network produces perdictions. This is called a forward pass and it consists of forwarding the inputs of the previous layer through some transformations to the next layer, until the outputs. The transformations consist of passing the activations α through the activation function h of the neurons of the following layer. The output of the neurons h of the following layer is then fed to the following layer the same way and so on.

In the formulas below a full forward pass is presented. Equation (3.1) shows the calculation of the activations j of the hidden layer. Essentially, the activation is calculated by summing the product of the weights times the outputs of the previous layer that are connected to the specific node plus a bias connected to this node. Equation (3.2) shows the activation function and Equation (3.3) shows the activations of the output layer. Finally, formula 3.4 shows the prediction, which is the output of the activation function of the output layer.

Let's assume that we have the neural network architecture in Figure 3.1. We refer to the input layer as i, the hidden layer as j and the output layer as k. The activations of the hidden layer j as calculated by

$$a_j = \sum_i w_{ji}^{(1)} x_i + b_j^{(1)}, \tag{3.1}$$

where a_j corresponds to the activations of the hidden layer, w_{ji} corresponds to the weights of the input layer, x_i corresponds to the inputs and b_j is the bias of the hidden layer. The subscript ji refers to the connection from layer i to layer j.

The activations then, pass through an activation function (non-linearity). This function can be ReLU, Sigmoid, TanH or any other non-linear function that is used. We call the output of this function z and it is calculated using

$$z_j = h(a_j), \tag{3.2}$$

where j corresponds to the hidden layer and h denotes the non-linear function.

Afterwards, the activations of the output layer a_k are calculated using

$$a_k = \sum_j w_{kj}^{(2)} z_j + b_k^{(2)}, \tag{3.3}$$

where a_k corresponds to the activations of the output layer, w_{kj} corresponds to the weights of the hidden layer, z_j corresponds to the outputs of the hidden layer (which are the non-linear transformations) and b_j is the bias of the hidden layer. The subscript kj refers to the connection from layer j to layer k.

Finally, depending on the nature of the problem, regression or classification, further activation function can be applied to a_k or not. And this output is the prediction of the network. Let's assume that we apply a function again to a_k. For instance

$$y_k(x) = h(a_k) \tag{3.4}$$

,where h denotes the non-linear function.

The forward pass can be expressed in a single line as

$$y_k(x) = h(\sum_j w_{kj}^{(2)} h(\sum_i w_{ji}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)}) \tag{3.5}$$

The neural networks learn to predict correctly through a process called training. Training involves calculating the best parameters (weights, biases) of the model and this is done by minimizing the error of the output. The error is measured by a metric called loss function and it intuitively shows how far the model is from the correct predictions. There are a lot of loss function that are used depending on each task. To minimize the loss, gradient descent is used.

During training, the networks performs forward and backward passes in order to fix the parameters. The forward pass is present earlier, while the backward pass involves taking the gradient of the loss function with respect to each of the parameters and updating them. This procedure is also known as backpropagation. In the following equations, E refers to the loss calculated using the forward pass. In Equations (3.6) and (3.7) the weights and biases updates are performed using a gradient decent step using

$$w^{t+1} = w^t - \eta_t \nabla E(w^t), \tag{3.6}$$

$$b^{t+1} = b^t - \eta_t \nabla E(b^t), \tag{3.7}$$

where, t denotes the time step, w denotes a weight term, b denotes a bias term, η denotes the learning rate which is a hyperparameter value chosen accordingly to each task and $\nabla E(x)$ is the gradient of the loss with respect to parameter x. [1]

Gradient descent can be applied in different ways, such as in batches, mini batches or single examples. Batch Gradient Descent refers to taking all the training data into account in order to take a step, while Mini Batch Gradient Descent involves splitting the training data into smaller batches and each step is taken for each batch. Finally, Stochastic Gradient Descent refers to taking only one example into account for each step [10] [3].

3.2 Convolutional Neural Networks

The main idea that is used in the approaches is the Convolutional Neural Network (CNN). The CNN is a type of neural network that is very powerful and preferred in tasks related to Computer Vision, such as Image and Video recognition. The advantage over simple feed-forward neural networks (described in Section 3.1) is that they are able to learn efficiently spatial features through the application of filters [17] [3]. They consist of convolutional layers which are used to learn features of the image, pooling layers that are used to introduce an invariance to small shifts and distortions of the images, by down sampling the input image, and fully connected layers.

[1] Of course all these computations in the forward and backward passes are performed in matrices, but this notation is used for better understanding.

Convolutional Layers

Convolutional layers perform an operation called «convolution» to an image input. Actually, the operation is cross-correlation, which refers to the process of sliding a filter over the image and computing the sum of products at each location. Convolution, is similar to cross-correlation with the difference that the filter is first flipped, but for simplicity the process in neural networks has been established as convolution. The filters are also called kernels, which vary in size and their parameters (filter values) are trainable.

More specifically, the convolution operation is the scalar product between the filter coefficients and image values in each neighbourhood. By neighbourhood we refer to the group of values that the filter is applied on. The filter slides over all pixels and each result is saved in the filter center. Another hyperparameter is the stride, which determines how the filter moves, meaning every how many pixels the filter is placed.

In Figure 3.3, there is an example of the convolution operation demonstrated. An image, is usually an RGB image, which means that it is separated by three color planes (Red, Green and Blue). Thus, the input technically is of size HxWxC, where H is the height of the image, W is the width and C is the number of channels (3 in the case of RGB). In this case, the filters are also in the shape of NxMxC and the operation is performed simultaneously in all C channels. However, for simplicity the process refers to an image of shape HxWx1.

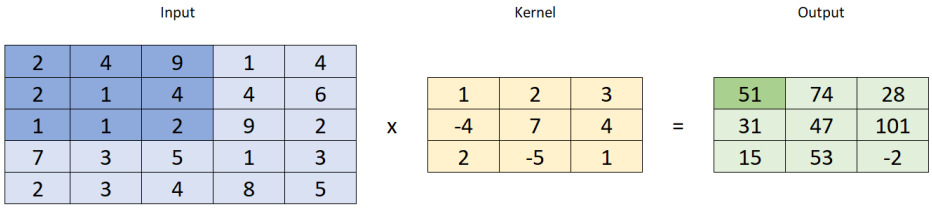


Figure 3.3: This is an example of the convolution operation. The filter is of size 3x3 and the output is the result of the convolution.

The outputs of the convolution correspond to the features extracted by the image. Despite the fact the convolutional layer is used to capture important features it is also used to reduce the spatial size of the images without losing critical information [15] [3]. Convolutional layers are usually also followed by a Batch Normalization layer and an activation function, usually ReLU.

Batch Normalization

Batch Normalization is a technique used to normalize the outputs of each convolutional layer for every mini batch. By normalization, it is meant to rescale the data to have mean of zero and standard deviation of one. To achieve that, the mean and the standard deviation per dimension is calculated. This technique improves the convergence time and the stability during the training. It typically makes the distribution of inputs to layers more stable and reduces the internal covariate shift [6]. Normalizing the inputs makes the optimization easier, since the loss function behaves better.

Pooling Layer

The pooling layers are similarly used to reduce the spatial size of the convolved features [15] [3]. It also introduced invariance to small shifts and distortions of the images, by down sampling the input image. There are different types of pooling layers, such as Max Pooling, Average Pooling, Global Max Pooling and Global Average Pooling [20]. The operation is similar to the convolution, with the difference that the kernel now slides on the image and

keeps the maximum or average value depending on the type of the pooling that is applied, instead of applying a filter. The global poolings refer to using a kernel equal to the size of the input image.

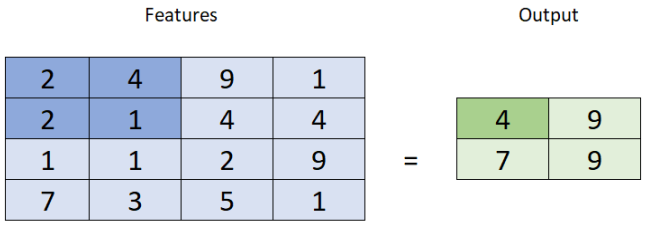


Figure 3.4: Max pooling operation.

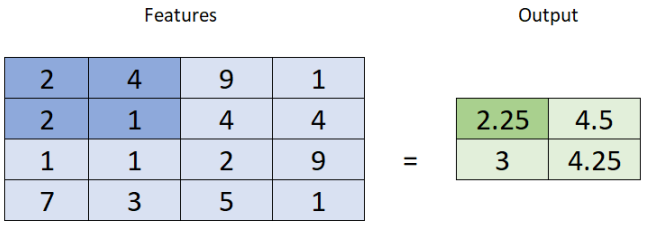


Figure 3.5: Average pooling operation.

3.3 Grad-CAM

Grad-CAM (Gradient-weighted Class activation Mapping) is a tool that uses the gradient information in the last convolutional layer of the model to provide information regarding where the model focuses in the image to take its decisions. It is a powerful tool used for the interpretability of the models and it is often applied in classification tasks.

To implement Grad-CAM, we essentially need to run a forward and a backward pass to obtain the gradients with respect to the feature maps. Let's assume that we have a classification task and we do the forward pass and we get the prediction y^c class score. Class score y^c is the probability that the outcome belongs to class c. We just need the score of the predicted class to run the backward pass, before softmax. Softmax is a function used to normalize the output of a network to a probability distribution over predicted output classes. Technically, it converts a number of values into values that sum to 1. This is calculated using

$$\sigma(z_i) = \frac{e^{(z_i)}}{e^{\sum_{j=1} z_j}} \tag{3.8}$$

Going backwards, we obtain the gradients of the target class y^c with respect to the feature maps A^k of the last convolutional layer [1].

The following formula shows how gradients are obtained via backpropagation.

$$\frac{\partial y^c}{\partial A^k} \tag{3.9}$$

Then, the weights w_k^c are calculated by applying global average pooling on the gradients.

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{3.10}$$

To obtain the Grad-CAM, the weights w_k^c are multiplied with each activation map A^k, which intuitively shows how important each channel is for the predicted class, and they are summed.

$$\sum_k w_k^c A^k \tag{3.11}$$

Finally, the summation is followed by a ReLU in order to keep the features that have positive influence on the heatmap.

$$H^c = ReLU(\sum_k w_k^c A^k) \tag{3.12}$$

,where H^c is the heatmap for the predicted class c.

After having obtained the heatmap, we can plot it over the original image and observe in which parts of it the model focused to predict class c.

3.4 Transfer Learning

We will use transfer learning to implement the models. Transfer learning is a technique where a model that is trained beforehand on a task with possibly different data, is used as a starting point to be retrained and fine-tuned on a different task, using significantly less data. The pre-trained model that we will use is ResNet50 trained on ImageNet, because it is a very efficient model for extracting features in images, as proven in [4]. ImageNet is a large dataset of approximately 14 million images and 21000 classes [2]. The main advantage of using transfer learning in our model is that it would take a lot of time to train a network like this from scratch and since these networks are trained in huge datasets, probably we would not achieve as high performance as the pre-trained one.

3.5 Triplets

Triplet loss is a function used to learn efficiently the distances between an anchor point, a positive input and a negative input. The aim is to minimize the distance between and the anchor and the positive input and maximize the distance between the anchor and the negative input. Choosing the right triplets to calculate the loss has a great impact in the learning process of the model. If we just provide the network with all positive (belonging to the same person) and all negative (belonging to a different person) pairs, eventually when all the positive points have been seen by the network all of the positive pairs will be close and form a cluster.

The drawback of this approach is that since we have a huge dataset, it would require a very long training process. Also, providing continuously easy comparison it would force the network to learn how to map correctly the most trivial cases [5], for example people that wear different outfits or different colors, and lack in separating more similar appearances. Thus, training over hard triplets becomes crucial.

We aim providing the network with hard positives, which refer to pictures of the same person that appears the most different among the other pictures of the same person, usually due to pose or viewpoint variation and hard negatives which refer to the most similarly looking different person. Of course, these pairs are chosen within the same team and game for each anchor since our goal is to differentiate players within a team and between the teams. Anchor is considered the image used each time as a target to finding the similar and dissimilar images. The second task will be automatically handled if the network is able to separate players of the same team.

In Figure 3.6 the Triplet concept is presented. The aim of the figure is to provide an intuition on how the triplet loss works. The model tries to pull closer similar images and push away the dissimilar ones by fixing the distances between them as shown above.

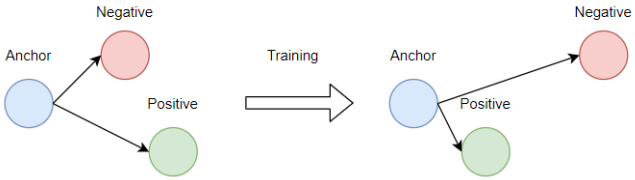


Figure 3.6: On the left a triplet is presented before the training. On the right we observe how the distances change after the model learns to set low distances on similar images and high distances to dissimilar ones.

4 Method

As mentioned in Chapter 1 we will evaluate the performance of the Baseline, Vanilla and AlignedReID models. In this Chapter, we will go through the implementations of the Vanilla and the AlignedReID model. We will provide details regarding their architectures and some important parts, for instance the learning rate that is used, the optimizer and the loss functions.

The aim of the thesis is to identify the players, in terms of separating them, in a soccer game. We would like to associate all the images of the same person and assign all of them to tracks, one for each person. Then, by identifying one image in each track, we will know the identity of all the detected players since all of the images are assigned to the tracks and they are associated with one of the images where the person that it depicts is recognized.

Eventually, we will have a model that will take as input images of a game and it will predict the embedding representations of them. Then, it will set each one of them as Query and return the top-50 most similar images from the same game, which means the images given as input, and it will produce a ranking list for every Query.

The AlignedReID model is proposed in [22] and we will slightly modify it for our purposes. The AlignedReID model contains two outputs, one for global feature extraction and one for local feature extraction. They are both used to calculate the loss during training, but the actual embedding output is contained in the global feature extraction. Thus, we modify this branch to output a vector of 128 length. The Vanilla model, is also proposed which is based on AlignedReID and it can be considered more simple. It only contains the global feature extraction part and this is also the only output. The vector output is also of length 128. We assume that embeddings of size 128 are enough to map our data in this task.

Regarding the Baseline model, we simply use the Euclidean distances in the images space to find the most similar images. It is equivalent to flattening the image arrays and calculating the Euclidean distances, producing a ranked list for each Query.

4.1 Vanilla Model

Regarding the approach we will start by implementing the vanilla model. For this implementation, we will follow a similar idea as proposed in the global feature-based methods. That is, there will be a convolutional neural network in order to extract the features of the image and eventually it will lead to a layer that will serve as an embedding layer. For the convolutional neural network (CNN) part we will use the well-known residual network (ResNet50), as stated before, consisting of 50 layers. The architecture can be seen in Figure 4.1 and the image is taken from [4].

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.1: ResNet50's architecture. In the beginning there is a convolutional layer followed by a max pooling layer. The 50-layer ResNet then contains convolutional layers as shown in the respective column, for example a block of 3 convolutional layers followed by a block of 4 convolutional layers etc. The notation (n x n, m) refers to m number of filters of size n x n.

As shown in the architecture (we are interested in column "50-layer"), the model consists of convolution layers followed by an average pooling layer plus a fully connected layer in the end. It also includes skip-connections (also called residual connections), which are used to prevent exploding or vanishing gradients. They are just shortcuts, that skip the next 2 layers each time and perform identity mapping which is added to the output [4]. We need only the feature extractor part of ResNet50 for our implementation, so we remove the 2 last layers. Instead, we will add a global pooling in the end, same way as it is proposed in [22] for the global feature part, followed by a fully connected layer consisting of 128 neurons. That way, we will have strong grounds to compare this model with the next one that will be implemented, and it will contain in addition another branch for the local feature extractor. The length of the embedding layer is decided to be 128, because we assume that this length works well for our task, thus it is not needed to add more neurons and more parameters in the model.

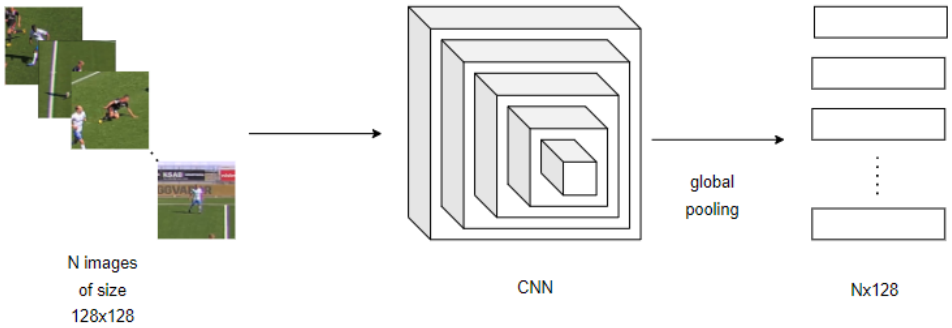


Figure 4.2: Vanilla model's architecture

4.2 AlignedReID Model

For the AlignedReID model implementation, we decided to use a similar architecture as proposed in [22]. It belongs to the stripe-based methods and it aims to enable the model to learn more discriminative features through the local feature extractor branch. Once again, for the convolutional part we will use the same pre-trained network (ResNet50), both because it is a strong pre-trained network for feature extraction and because we used it in the vanilla model, thus the comparison between the models will be more valid. In Figure 4.3 we can observe the architecture proposed in [22] and the image is taken from [22].

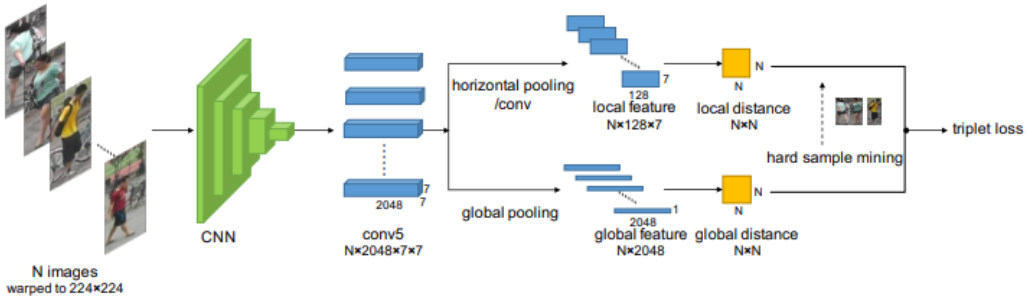


Figure 4.3: AlignedReID's model architecture. As we observe, there is a CNN in the beginning, which is the ResNet50, to extract the features and two branches one for global features and one for local features. The embedding layer is found exactly after the global pooling layer.

Regarding the global feature branch, we use the same exact branch as in the vanilla model. The difference in the AlignedReID model is that we also use another branch for local feature extraction. It consists of horizontal pooling, which equals to taking the mean of the features horizontally, and a 1x1 convolution in order to lower the parameters of the model. This way, as we can see in the image, we aim to have learned some more discriminative features by splitting the image into several horizontal parts. We will use both global and local branch outcomes to compute the loss during training by adding their global and local distances, but we will only use the global branch to calculate the loss for the validation and test set. It is shown in [22] that using the local branch in addition to the global branch does not affect much the evaluation part.

Local Parts Alignment

For the local features an alignment distance is used. When separating two images horizontally in parts, we would like to find the distance of the body parts that match to each other. For this purpose, an algorithm is used as introduced in [22] which tries to match the local parts from top to bottom between two images. Eventually, the distance between those local parts will be considered as the sum of the distances of the aligned parts. In Figure 4.4, the algorithm for the alignment and the distance calculation is introduced graphically. The image is taken from [22].

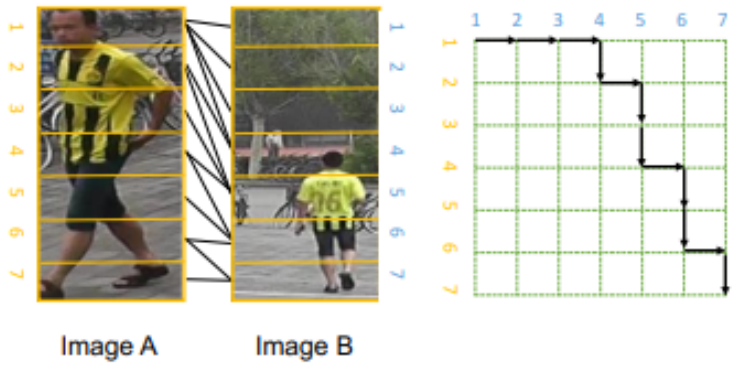


Figure 4.4: AlignedReID's algorithm for alignment and distance calculation. We can see the alignment between the parts that Images A and B are split. Also, the matrix corresponds to the distance between a part i of Image A and one part j of Image B. The arrows in the matrix, denote the total distance between the two images which is the shortest path.

In order to calculate the distance between the two images based on the local parts, we first create a matrix i x j as shown in Figure 4.4, where i is the number of parts in Image A and j is the number of parts in Image B. Each element of the matrix, corresponds to the distance of the between i-th part of Image A and j-th part of Image B. Those distances are also normalized by an element-wise transformation, as proposed in [22]. Again, Euclidean distance is used. The Equation used is

$$d_{i,j} = \frac{e^{||f_i - g_j||_2} - 1}{e^{||f_i - g_j||_2} + 1}, \tag{4.1}$$

where f_i is the local feature i of image A and g_j is the local feature j of image B.

After the distance matrix is created, we can find the distance by finding the shortest path in the matrix starting from (1,1) up to the bottom right corner of the matrix. Of course, there will be alignments between non-corresponding parts included in the shortest path but as mentioned in [22] they are needed in order to maintain the order of the vertical alignment of the parts.

4.3 Triplet Loss

Our loss will be the one mentioned in [5] and it is called Triplet Loss and we will use the Batch Hard version (TriHard). Also, we will use both variations mentioned in the paper, the default one that adds a margin in the formula and the soft-margin. We chose the margin to be 0.3 as in [5]. Triplet loss sets each embedding in the batch as an anchor and uses the distance between the anchor and an embedding that belongs to the same person, called positive embedding, as well the distance between the anchor and an embedding that belongs to a different person, called negative embedding. And the resulting loss value occurs by summing all the individual triplet losses. To measure the distance between the embeddings, Euclidean distance is used. This way it tries make sure that the distance between the anchor and the positive embedding is smaller than the distance between the anchor and the negative embedding.

Intuitively, this way, it tries to pull together embeddings that belong to the same person and push away embeddings that belong to different persons. The Batch Hard version refers to the way the positive and negative embeddings are picked in each set. Briefly, in each batch the hardest positives and the hardest negatives are picked for each anchor. The way the triplets are picked and their importance is further described in Section 4.5. We present below

the loss functions that will be used in the Vanilla model. The Equation of the default Batch Hard loss is

$$\sum_{i=1}^{P}\sum_{a=1}^{K}[m + \max_{p=1...K} D(f_\theta(x_a^i), f_\theta(x_p^i)) - \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D(f_\theta(x_a^i), f_\theta(x_n^i))], \tag{4.2}$$

where P represents the persons in a batch, K the number of the images of each person, m is a constant margin set arbitrarily to 0.3 in this case, D represents the Euclidean distance, f_θ the embedding of the image x and x_j^i corresponds to the j-th image of the i-th person in the batch. Maximum and minimum distances refer to the hardest positive and hardest negative images respectively [5].

In the next Equation (4.3), the soft-margin variation is presented. The idea of this variation is to make the model avoid the already correct triplets and this is done by replacing the formula above with a smooth approximation by using the softplus function as described in [5]. It is calculated using

$$\sum_{i=1}^{P}\sum_{a=1}^{K}[ln(1 + exp(\max_{p=1...K} D(f_\theta(x_a^i), f_\theta(x_p^i)) - \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D(f_\theta(x_a^i), f_\theta(x_n^i))))], \tag{4.3}$$

where again P represents the persons in a batch, K the number of the images of each person, D represents the Euclidean distance, f_θ the embedding of the image x and x_j^i corresponds to the j-th image of the i-th person in the batch. Maximum and minimum distances refer to the hardest positive and hardest negative images respectively.

The losses we will use in the AlignedReID will be similar to the ones used in the Vanilla model. We will use both version of the Batch Hard Triplet Loss. The difference now, is that during the training we will also add the distances between the local features in the Equation. We will add the distance of the local features between the anchor and the positive to the distance of their global features and we will subtract the distance of the local features between the anchor and the negative from the distance of their global features. The Equations of the Batch Hard loss and the soft-margin variation are presented in (4.4) and (4.5) respectively.

$$\begin{aligned}\sum_{i=1}^{P}\sum_{a=1}^{K}[m + \max_{p=1...K} D(f_\theta(x_a^i), f_\theta(x_p^i)) + \max_{p=1...K} D'(f_\theta(x_a^i), f_\theta(x_p^i)) - \\ \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D(f_\theta(x_a^i), f_\theta(x_n^i)) - \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D'(f_\theta(x_a^i), f_\theta(x_n^i))],\end{aligned} \tag{4.4}$$

where P represents the persons in a batch, K the number of the images of each person, m is a constant margin set arbitrarily to 0.3 in this case, D represents the Euclidean distance of the global features, D' represents the distance of the local features, f_θ the embedding of the image x and x_j^i corresponds to the j-th image of the i-th person in the batch. Maximum and minimum distances refer to the hardest positive and hardest negative images respectively.

$$\begin{aligned}\sum_{i=1}^{P}\sum_{a=1}^{K}[ln(1 + exp(\max_{p=1...K} D(f_\theta(x_a^i), f_\theta(x_p^i)) + \max_{p=1...K} D'(f_\theta(x_a^i), f_\theta(x_p^i)) - \\ \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D(f_\theta(x_a^i), f_\theta(x_n^i)) - \min_{\substack{j=1...K \\ n=1..K \\ j\neq i}} D'(f_\theta(x_a^i), f_\theta(x_n^i))))],\end{aligned} \tag{4.5}$$

where once again P represents the persons in a batch, K the number of the images of each person, D represents the Euclidean distance, D' represents the distance of the local features, f_θ the embedding of the image x and x_j^i corresponds to the j-th image of the i-th person in the batch. Maximum and minimum distances refer to the hardest positive and hardest negative images respectively.

4.4 Ranking

To retrieve the ranked list of the similar images, we will use Euclidean distance in the embedding space. First of all, during testing we will predict the embeddings of the images for the test data. As mentioned earlier, we will evaluate the performance at one game at a time. Afterwards, we will set each embedding-image as a "query" and we will retrieve the top 50 images that are most similar to the query image. To achieve that, we calculate a distance matrix between all the embeddings and we sort each row of the matrix with ascending order. Each row corresponds to the distances of the embedding i, where i is the row, from every embedding j, where j is the column of the distance matrix. Of course, there will be zeros in positions (i,j) where i=j and when the rows are sorted all zeros will be in the beginning of each row respectively.

We also need to keep track of the images that these embeddings correspond, so that we can plot them. For this purpose, when storing the embeddings predicted we also store the path to the image that each embedding corresponds. Moreover, when a row is sorted we keep track of the indices that are allocated so that we match the embeddings with their respective images. Finally, when plotting the images we place a green or red border around the image if the retrieved image belongs to the same person or not respectively.

4.5 Implementation Details

In this section we will describe the implementation details, which apply to both Vanilla and AlignedReID model.

Learning Rate

We will use the One Cycle Policy as described in [17] which is a way to boost the convergence of the training and avoid saddle points. It can also work as a regularization technique in some cases and prevent overfitting, as it helps escape steep areas of the loss. The idea of this policy is setting a lower bound and an upper bound for the learning rate and switching values between this interval in cycles, starting from the minimum value towards the maximum and reverse. We concluded using 0.000001 as a lower bound and 0.0003 as an upper bound, as we observed it offered better convergence after experimental runs.

Optimizer

We will use the Adam optimizer (Adaptive Moment estimation) for our training, which is a very popular optimization algorithm. It is an algorithm that requires only first-order gradients and it computes individual adaptive learning rates for different parameters using estimates of first and second moments of the gradients [8]. We will also use mini Batch Gradient Descent, meaning that each step of the gradient will be performed after each mini batch.

Batch Sampler

Regarding the batches' formulation, we will use 128 batches per epoch and batches of size 256. Each batch samples randomly 4 teams that will be used within it. When sampling teams, the GameID is considered, which means that the same team can be chosen more than once in the

same batch as long as the data (images) refer to a different game. For every team, 8 players as well as 8 crops per player are sampled at random and these 256 crops compose the batch.

An intuitive representation of an example batch is shown in the following Table 4.1.

Index in batch	Team + GameID	Player (jersey number)	Image Index
1	IFK Norrköping FK 2dc5f8123	1	1028
...	...	...	...
64	IFK Norrköping FK 2dc5f8123	26	50032
65	Kalmar FF e32abdd1b	5	23930
...	...	...	...
128	Kalmar FF e32abdd1b	13	69038
129	Örebro SK c3937f609	9	18930
...	...	...	...
192	Örebro SK c3937f609	21	34296
193	Malmö FF ff539feeb	4	512
...	...	...	...
256	Malmö FF ff539feeb	18	278

Table 4.1: Example of a batch. The numbers and IDs are random, as they do not really provide any specific information here.

The training data we have can be divided to more than 128 batches. That means that we do not go through the whole training set per epoch, mainly due to computational costs. However, since we randomly sample each team and the players per batch, and the size of the training data is huge and it includes augmentations, we assume that the same image is not picked repeatedly during training. It can be re-picked, but not regularly which makes it not harmful for the training. It is also discussed in [9] that training on a finite set, which means going through a fixed set per epoch, yields the same generalization as training on an infinite set like in our case, where we pick different data every epoch.

Triplets Extraction

Within each batch, we set every person as an anchor, serially one at a time, and assign to it a hard positive and a hard negative pair to calculate the loss. When searching for the hard positives we compare the distances of the images of the same player (consequently of the same team) with the anchor and pick the one that has the greatest distance. When searching for the hard negatives we compare the distances of the images that belong to only different players (within the same team) with the anchor and pick the one that has the less distance. To perform the distances calculations, we used the Euclidean distance between the embedding vectors.

Grad-CAM Implementation

Regarding Grad-CAM, we decided to adapt it more to a classification problem in order to extract the gradients. The idea is that we do not have a target label for each image in our current task, thus we cannot calculate the gradients for each image. One workaround would be to use weight transferring as proposed in [1], but this would need to retrain the network in order to save the weights. Due to the computational cost and the required time it is needed for this approach, we eventually decided to simply add a classification layer and retrain the network for only one team. The target classes are 11 and they are practically IDs for each player of the team. By retraining the network, we mean loading the weights we retrieved from our training of the embedding network and freezing all the layers except the new clas-

sification layer. The aim is just to lead the weights of the new layer to the correct direction so that we get the gradient we need to calculate the heatmaps.

5 Results

In this chapter, the results are presented, which refer both to the process of training and the evaluation. Also the metrics that will be used to evaluate the performance of each model are described in detail.

We will compare the results of three models, the Baseline, the Vanilla and the AlignedReID. For the deep learning models, there will be two loss functions used per model. Thus, in total there will be a comparison between five models, the Baseline, the Vanilla model with the Batch Hard loss function and its soft-margin variation and the AlignedReID with the same loss functions. The loss functions were described extensively in 4.3.

Regarding the evaluation part, Rank-1, Rank-5, mAP(mean Average Precision) and AUC (Area Under Curve) metrics will be used. The metrics will be explained first, before presenting the results, so that there is a clear understanding of what the results show. Also, we will perform hypothesis testing and PCA (Principal component analysis) to further evaluate the performance, as well as Grad-CAM.

5.1 Evaluation Metrics

Rank-1, Rank-5, mAP are some common metrics that are used for evaluating the performance of models that perform this task.

- Rank-1: It is the percentage of the queries that have a correct retrieval on the top of the list, thus it takes into account only the top retrieved image.
- Rank-5: It is the percentage of the queries that have a correct retrieval on the top 5 images of the list, thus it takes into account only the first 5 retrieved images.
- mAP: Mean Average Precision is commonly used to measure the overall performance. It refers to calculating the mean of the APs for all queries. The AP is equivalent to calculating the area under the precision-recall curve [23]. It intuitively shows how many of the correctly retrieved images are concentrated on the top of the list. The Equation of AP can be seen in Equation (5.1).

$$AP = \frac{1}{T} \sum_{k=1}^{K} Precision@k \times I(k) \tag{5.1}$$

In the above formula, T corresponds to the number of the correct images that can be retrieved for each query in total. Precision@k refers to the ratio of the number of images that can be retrieved on top-k images (the number of the correct images that exist) divided by the total of number of images that are returned on top k. I(k) is an indicator function that is 1 if the image returned at position k is correct and 0 otherwise. Below, in Figure 5.1 we can see graphically the AP.

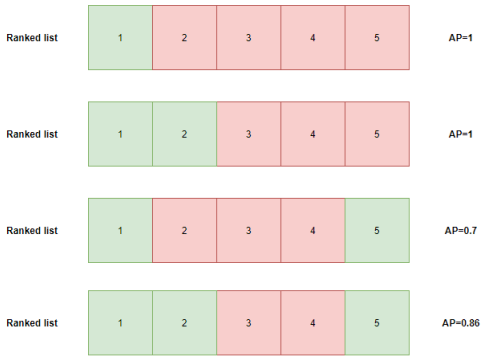


Figure 5.1: AP calculation graphically for each rank list. K is equal to 5 for every ranked list. The correctly retrieved images are indicated by green color and the wrongly retrieved images are indicated by red color.

As mentioned before, the evaluation is done per game. For each image query, there is a list of the retrieved images returned and on that list the evaluation is performed.

Moreover, the AUC will be used as a mertic. However, since we do not deal with a classification task, this metric will be calculated under the assumption that we have two classes: positive and negative distances. Thus, thresholds will be set repeatedly on the distances between pairs of images and we will try to classify the distances into positives and negatives. A curve is plotted by connecting the points using the True Positive rates (TPR) as y axis and the False Positive rates (FPR) as x axis for each different threshold. The model with the greatest Area Under Curve (AUC) is assumed to be the best classifier. By this, we will try to get an intuition of how the distributions of positives and negatives are separated.

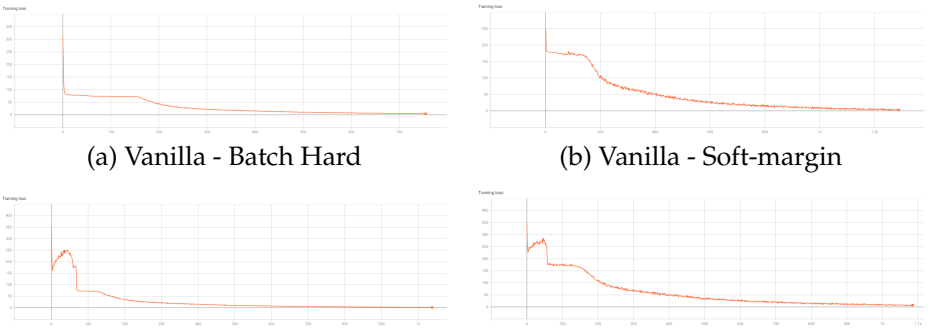
5.2 Training

The models are trained until they converge. However, the best model is saved based on the validation set using the distance between the modes of the distributions of positives and negatives. Positives refer to the distances between pairs of images belonging to the same person and negatives refer to the distances between images belonging to different persons.

During training the training loss - per epoch - is plotted, the distance between the distribution modes (of positives/negatives) on both training and validation set, the histograms of the distributions for both training and validation set and the AUC on the validation set. Also, per batch, the 2-norm of the embeddings is plotted, the distances of the embeddings, the losses and the non-zero losses, all on the training set.

In the images in Figure 5.2 we can observe the training losses of the 4 models, the Vanilla model with Batch Hard and its soft-marging variation and the AlignedReID with the same losses. We can see through the behavior of the training losses how the loss functions work. Both of them in the beginning try to pull the embeddings towards 0, which explains why this plateu is formed in the first 150 epochs. In the case of the AlignedReID in images (c) and (d), we also observe a sudden increase before forming the plateu, which refers to the fact that it also uses the distances of the local features to calculate the loss function. Then, the models start actually learning and eventually converge. This can also be understood by Figures 5.3

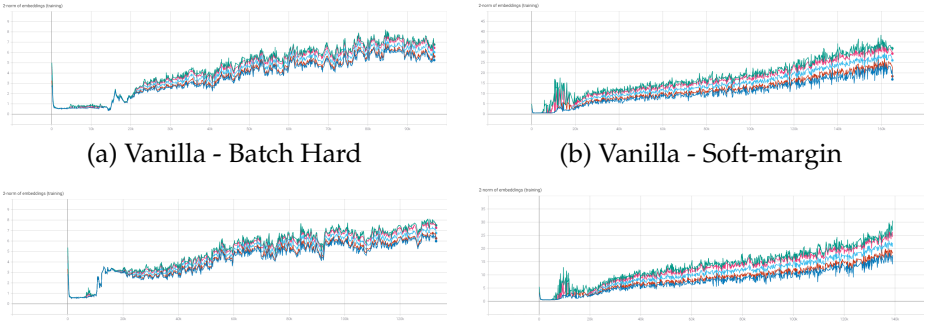
and 5.4. In the beginning the 2nd norm of the embeddings and their distances (pairwise) drop towards 0 and then (after 150 epochs) they start to expand and having different distances between them.



(a) Vanilla - Batch Hard (b) Vanilla - Soft-margin

(c) AlignedReID - Batch Hard (d) AlignedReID - Soft-margin

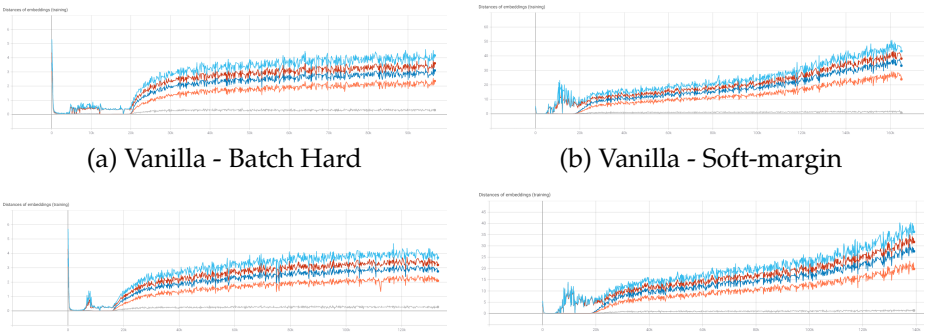
Figure 5.2: In (a) and (b) we observe the training loss for the Vanilla model using the Batch Hard loss function and its soft-margin variation respectively. In (c) and (d) we have the same demonstration for the AlignedReID.



(a) Vanilla - Batch Hard (b) Vanilla - Soft-margin

(c) AlignedReID - Batch Hard (d) AlignedReID - Soft-margin

Figure 5.3: Each of the curves represents a quantile of the distribution of the 2-norm of the embeddings used in each batch. Those quantiles are the 0-th, 5-th, 50-th, 95-th, 100-th bottom to top.

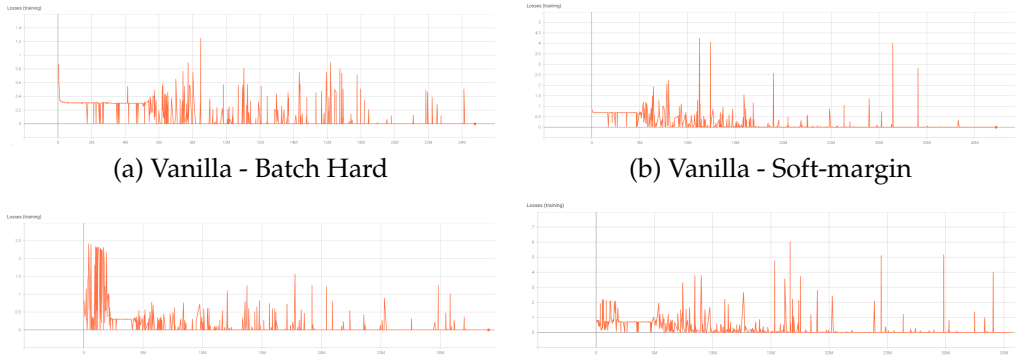


(a) Vanilla - Batch Hard (b) Vanilla - Soft-margin

(c) AlignedReID - Batch Hard (d) AlignedReID - Soft-margin

Figure 5.4: Each of the curves represents a quantile of the distribution of the pairwise distances of the embeddings used in each batch. Those quantiles are the 0-th, 5-th, 50-th, 95-th, 100-th bottom to top.

In Figure 5.5 the losses for each triplet used within each batch are shown and also in Figure 5.6 the percentage of the non-zero losses throughout the training (per batch). More precisely, non-zero losses are the losses that are greater than 0.001, since it's very difficult to have losses equal to 0. These plots provide information regarding the convergence of the model's training.



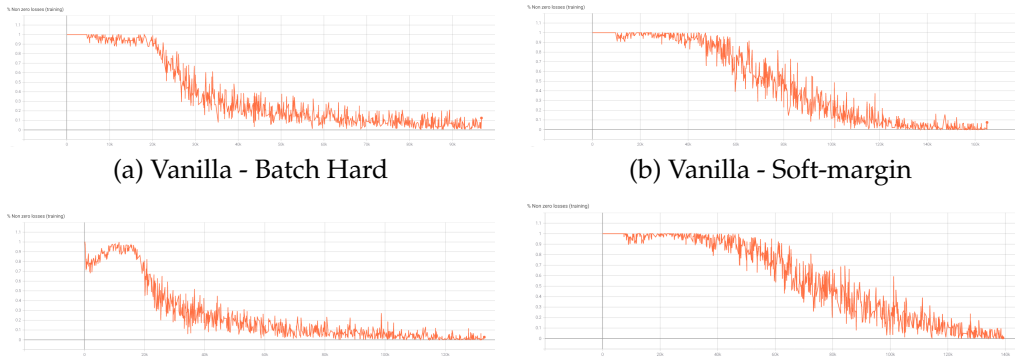
(a) Vanilla - Batch Hard

(b) Vanilla - Soft-margin

(c) AlignedReID - Batch Hard

(d) AlignedReID - Soft-margin

Figure 5.5: Losses (per triplet).



(a) Vanilla - Batch Hard

(b) Vanilla - Soft-margin

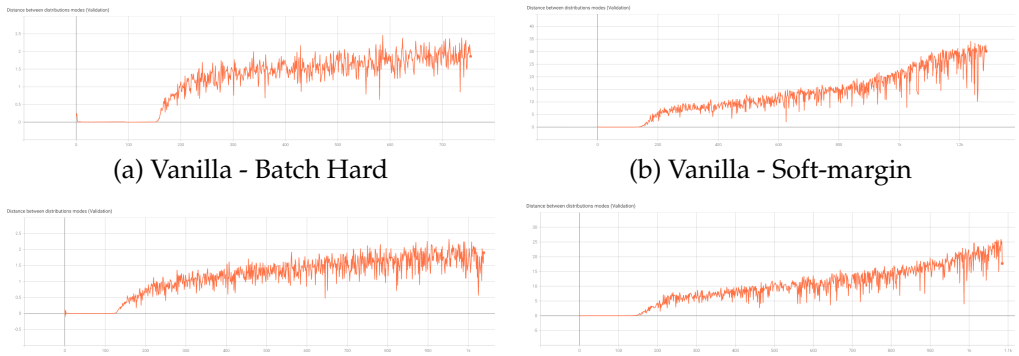
(c) AlignedReID - Batch Hard

(d) AlignedReID - Soft-margin

Figure 5.6: Percentage of non-zero losses - greater than 0.001 (per batch).

We mentioned earlier that we use the distance between the modes of the distributions of positives and negatives in the validation set to save the best model. We can see in Figure 5.7 how this distance increases graphically over epochs. It is desired for the distance between the modes to increase, as that means that the distributions of the distances between the positive and negative distances get more and more separated. It has no signs of overfitting, as the distance between the modes on the validation set is expected to decrease dramatically in case of overfitting. Moreover in Figures 5.8, 5.9, 5.10 and 5.11 we can observe the histograms of the distributions of positives and negatives for the training and the validation set, for all 4 models. Observing the distributions, we can again confirm that in the beginning of the training for approximately 150 epochs, the model pulls all the embeddings towards 0, thus their distances are very low (close to 0). And then the negatives start to overcome the positives and they are pushed away, while the positives are pulled together close to 0.

We can notice a difference between the two losses. The Batch Hard loss forms embeddings that have smaller distances than the embeddings formed using the soft-margin variation. However, the overlaps between the two distributions do not differ significantly.



(a) Vanilla - Batch Hard

(b) Vanilla - Soft-margin

(c) AlignedReID - Batch Hard

(d) AlignedReID - Soft-margin

Figure 5.7: Distance between distribution modes of positive and negative distances for the validation set.

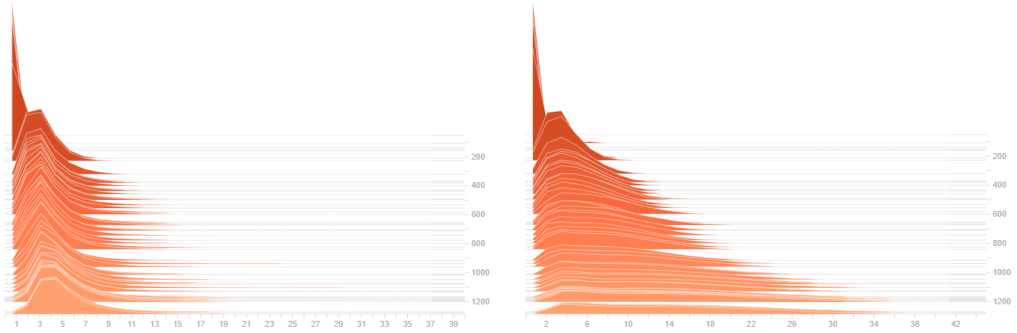
(a) Distribution of positives - Training

(b) Distribution of positives - Validation

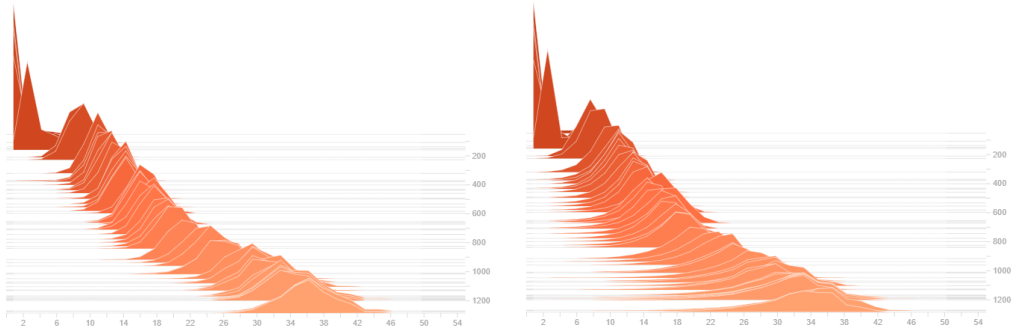
(c) Distribution of negatives - Training

(d) Distribution of negatives - Validation

Figure 5.8: Distributions of positive and negative distances for the Vanilla model using Batch Hard loss (per epoch top to bottom).

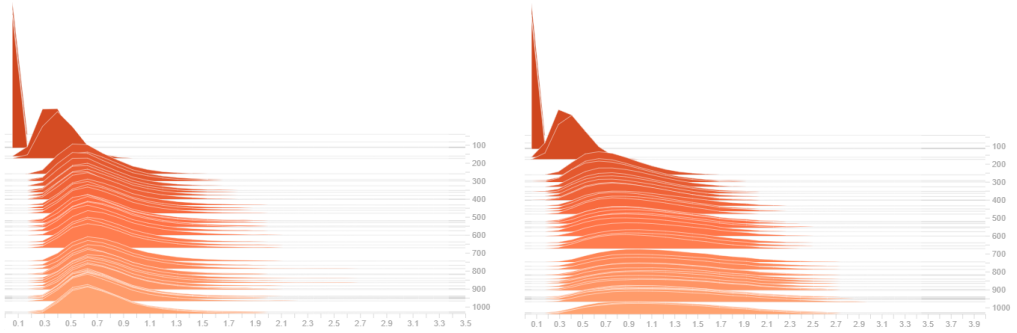


(a) Distribution of positives - Training (b) Distribution of positives - Validation

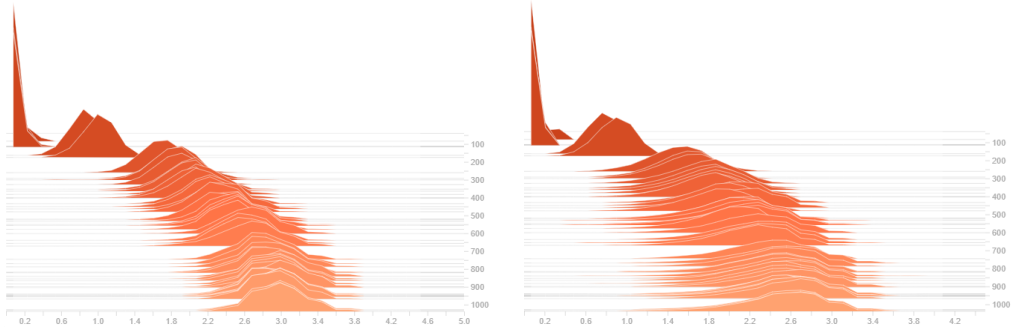


(c) Distribution of negatives - Training (d) Distribution of negatives - Validation

Figure 5.9: Distributions of positive and negative distances for the Vanilla model using the soft-margin variation of Batch Hard loss (per epoch top to bottom).

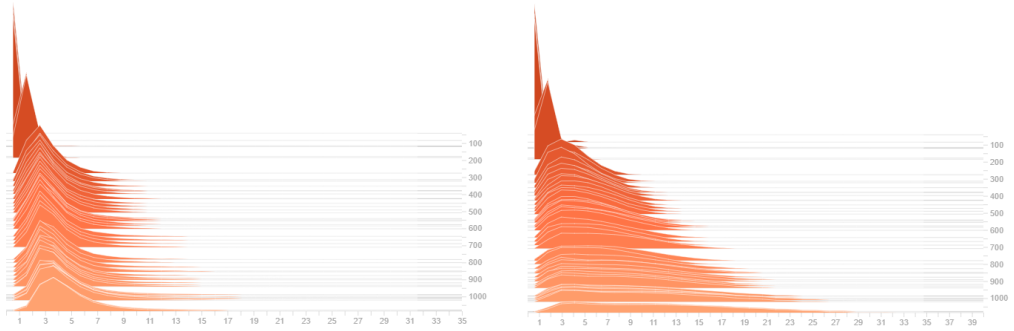


(a) Distribution of positives - Training (b) Distribution of positives - Validation



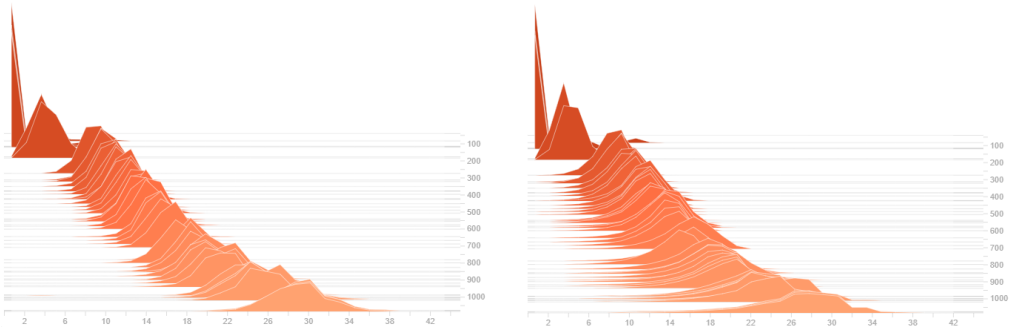
(c) Distribution of negatives - Training (d) Distribution of negatives - Validation

Figure 5.10: Distributions of positive and negative distances for the AlignedReID model using Batch Hard loss (per epoch top to bottom).



(a) Distribution of positives - Training

(b) Distribution of positives - Validation



(c) Distribution of negatives - Training

(d) Distribution of negatives - Validation

Figure 5.11: Distributions of positive and negative distances for the AlignedReID model using the soft-margin variation of Batch Hard loss (per epoch top to bottom).

5.3 Evaluation

For the evaluation of the performance of the models the test set is used which consists of 17 games. The splitting on the dataset was described in Chapter 2. The same exactly images are used for all the evaluations and we will test the 4 models trained as described in Section 5.2 plus the Baseline approach, which refers to comparing the Euclidean distance in the image space. The AUC metric will not be used for the evaluation of the Baseline approach, because in this case it is not informative at all. The result of the evaluation can be seen in Table 5.1. The average of the metrics for all models is presented in Table 5.2.

We observe that clearly the Baseline model underperforms compared to the rest of the models. Vanilla and AlignedReID seem to perform quite well, especially the ones using the Batch Hard loss. We notice that the AlignedReID model using the Batch Hard loss has the best performance of all 4, which is indicated also in the table by green color.

In Figure 5.12 the evaluation more in detail on some of the games used in Table 5.1 is observed. We choose to visualize Game 3, Game 5, Game 14 and Game 15. We can see how the top-10 images of the ranked lists look like in different games. The images presented in the lists are the original ones, which means that no preprocessing is applied on them. The query images are on top and are placed in a blue border and below each query there are the first 10 images retrieved. The correctly retrieved images are placed in green border and the wrongly retrieved ones in red border.

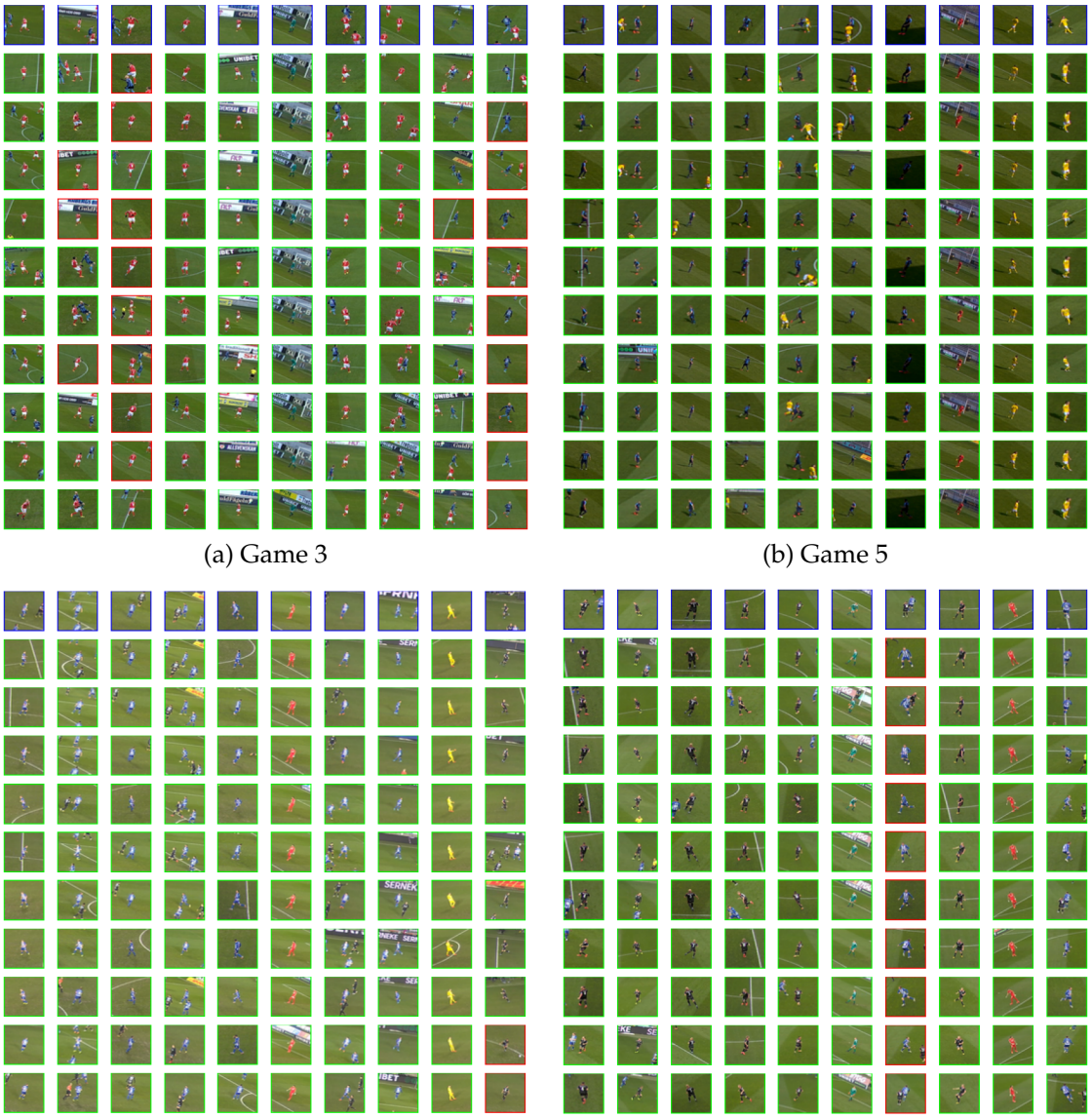
	Baseline			Vanilla								AlignedReid							
				Batch Hard				Soft-margin				Batch Hard				Soft-margin			
	rank 1	rank 5	mAP	rank 1	rank 5	mAP	AUC	rank 1	rank 5	mAP	AUC	rank 1	rank 5	mAP	AUC	rank 1	rank 5	mAP	AUC
Game 1	48.2812%	72.0312%	11.9703%	92.03125%	94.0625%	63.6443%	0.8589	89.375%	95.3125%	55.9941%	0.8782	99.2187%	99.8437%	96.8462%	0.9524	89.0625%	97.1875%	57.8917%	0.8895
Game 2	35.2272%	63.9772%	9.5258%	97.8409%	98.9772%	81.4713%	0.9038	97.6136%	99.2045%	80.4851%	0.9412	97.8409%	99.2045%	82.0034%	0.859	96.7045%	99.4318%	78.1772%	0.9195
Game 3	37.9545%	62.7272%	9.4547%	91.9318%	97.7272%	67.2067%	0.8353	87.5%	94.8863%	58.319%	0.8333	89.659%	96.25%	62.8708%	0.8829	87.95%	95.56%	58.2%	0.8392
Game 4	52.3863%	74.7727%	15.7778%	99.4318%	99.7727%	89.9888%	0.9712	98.75%	99.2045%	83.5889%	0.7732	100.0%	100.0%	97.4387%	0.9926	98.0681%	98.5227%	82.9215%	0.7449
Game 5	48.75%	71.0227%	13.4965%	95.5681%	97.9545%	67.7949%	0.8798	94.0909%	96.9318%	66.2973%	0.941	100.0%	100.0%	97.4924%	0.9954	94.0909%	97.0454%	63.0192%	0.9377
Game 6	46.1363%	74.4318%	13.6192%	89.5454%	97.5%	57.4561%	0.8778	87.6136%	95.0%	54.0701%	0.8887	99.7727%	99.7727%	96.8443%	0.9307	88.18%	96.13%	53.77%	0.8805
Game 7	33.75%	67.9687%	7.0989%	95.4687%	98.4375%	79.9374%	0.9405	95.9375%	97.8125%	78.2063%	0.8732	99.375%	99.375%	96.3636%	0.9957	95.1562%	97.5%	77.4485%	0.9495
Game 8	46.9318%	72.6136%	15.948%	97.9545%	98.8636%	89.3956%	0.9795	97.159%	98.6363%	85.7062%	0.8777	98.0681%	98.5227%	90.4017%	0.8774	97.3863%	98.5227%	83.8973%	0.7511
Game 9	39.2045%	63.75%	10.2964%	94.4318%	98.8636%	78.501%	0.9539	94.5454%	97.9545%	75.2336%	0.9407	92.9545%	98.1818%	73.4085%	0.8303	95.5681%	98.75%	74.5101%	0.8725
Game10	31.9318%	52.6136%	6.8933%	94.5454%	97.7272%	71.7132%	0.84	93.75%	97.3863%	70.6616%	0.8792	94.7727%	97.8409%	70.2678%	0.9042	94.3181%	96.3636%	69.8261%	0.9158
Game11	47.0454%	73.409%	12.2068%	97.6136%	99.2045%	80.1893%	0.9084	96.4772%	98.2954%	77.7404%	0.8671	95.909%	95.909%	74.1325%	0.9272	96.5909%	98.8636%	78.2983%	0.8782
Game12	45.909%	66.1363%	12.3075%	96.4772%	99.2045%	82.2586%	0.8185	96.4772%	98.409%	76.4256%	0.8299	97.5%	98.9772%	84.5554%	0.8879	95.5681%	98.8636%	75.4015%	0.8257
Game13	43.6363%	68.1818%	9.8561%	98.2954%	98.6363%	85.6790%	0.8379	97.6136%	98.8636%	82.6419%	0.9468	97.7272%	98.9772%	80.8714%	0.9508	97.6136%	98.75%	82.4492%	0.8127
Game14	55.909%	70.0%	14.3627%	93.9772%	98.409%	67.8858%	0.8979	92.3863%	98.0681%	59.9047%	0.8156	97.159%	98.9772%	74.7433%	0.9388	91.0227%	97.0454%	60.1252%	0.8322
Game15	43.1818%	67.5%	11.3913%	98.6363%	99.4318%	85.3892%	0.8634	97.8409%	98.8636%	77.5382%	0.7238	98.1818%	99.2045%	87.9707%	0.8993	97.5%	98.6363%	78.8037%	0.7508
Game16	40.0%	61.9318%	11.2193%	96.1363%	98.75%	82.8247%	0.8177	95.4545%	97.7272%	82.4573%	0.9125	96.8181%	98.0681%	83.1164%	0.966	95.909%	98.5227%	82.4937%	0.9092
Game17	62.159%	80.3409%	17.9815%	99.659%	99.659%	86.0395%	0.8865	97.7272%	98.2954%	82.2870%	0.7232	99.3181%	99.5454%	90.8235%	0.9286	98.0681%	98.75%	82.1353%	0.7496

Table 5.1: Evaluation of the approaches on 17 games.

In Table 5.2, the average of all the metrics is computed in order to compare the models. In general the Vanilla and the AlignedReID models are close in performance, with the second one outperforming more using the Batch Hard loss function.

	Baseline	Vanilla		AlignedReid	
		Batch Hard	Soft-margin	Batch Hard	Soft-margin
rank 1	44.4015%	95.659%	94.463%	98.6495%	94.3495%
rank 5	68.6047%	98.3254%	97.5787%	98.9726%	97.8232%
mAP	11.9048%	76.7438%	72.5985%	81.0291%	72.0704%
AUC	-	0.8879	0.8636	0.9276	0.855

Table 5.2: Average of the metrics for each model. The best performance on each metric is denoted with green color and the worst with red color.

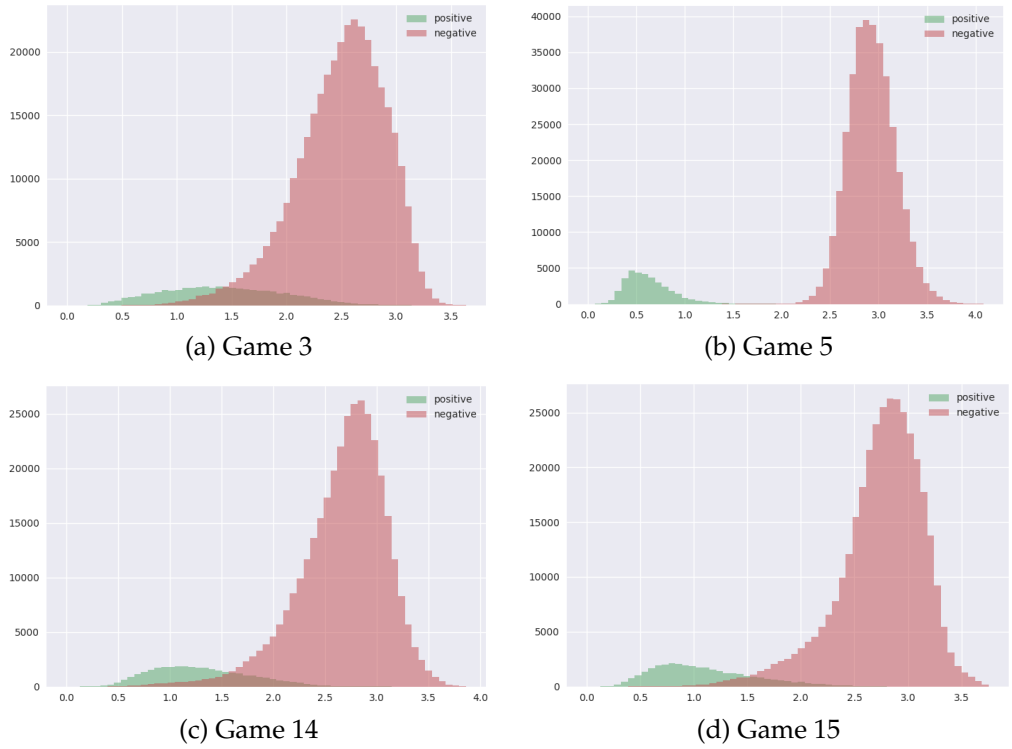


(a) Game 3 (b) Game 5

(c) Game 14 (d) Game 15

Figure 5.12: Random 10 ranking lists per game. Query image in blue border, correctly retrieved images in green border and wrongly retrieved in red border. The model used for this evaluation is AlignedReid trained using Batch Hard loss.

In Figure 5.13 the distributions of their positive/negative distances respectively for each game are shown. We mention again that, positives are considered the distances of the embedding representation of the images that belong to the same person and negatives the distances of the embeddings of images that belong to different persons.



(a) Game 3

(b) Game 5

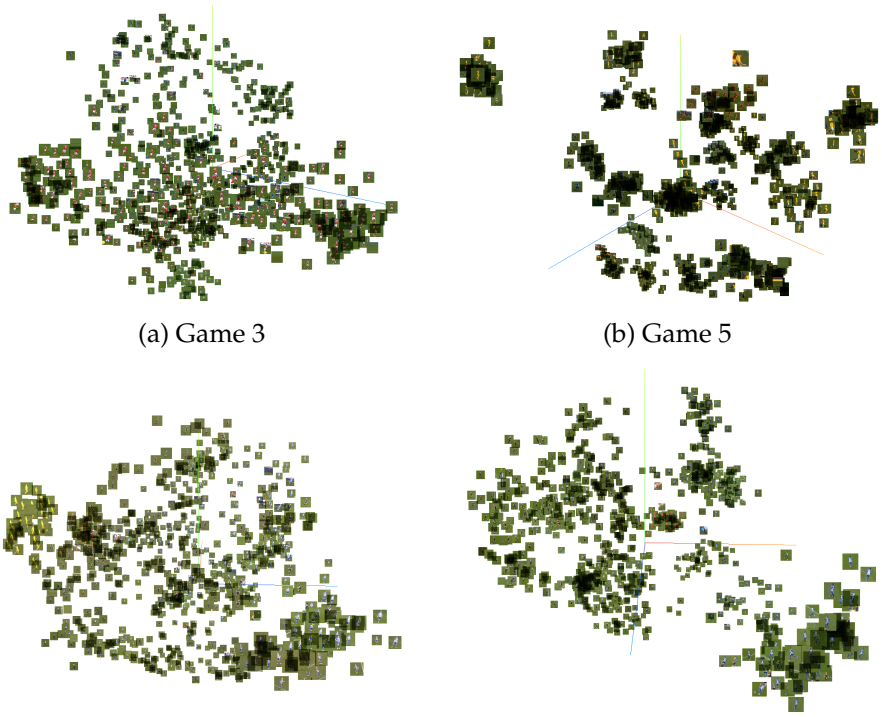
(c) Game 14

(d) Game 15

Figure 5.13: Distributions of positive and negative distances per game. Positive distances are the distances of images of same persons and negative distances are the distances of images of different persons.

Finally, we present an intuition of their distances in the space by performing a Principal Component Analysis (PCA). The embedding representations of the images used in the evaluation of each game are visualized, using the 3 most significant components. In Figure 5.14 we can see this visualization, respective for each game used in Figures 5.12 and 5.13.

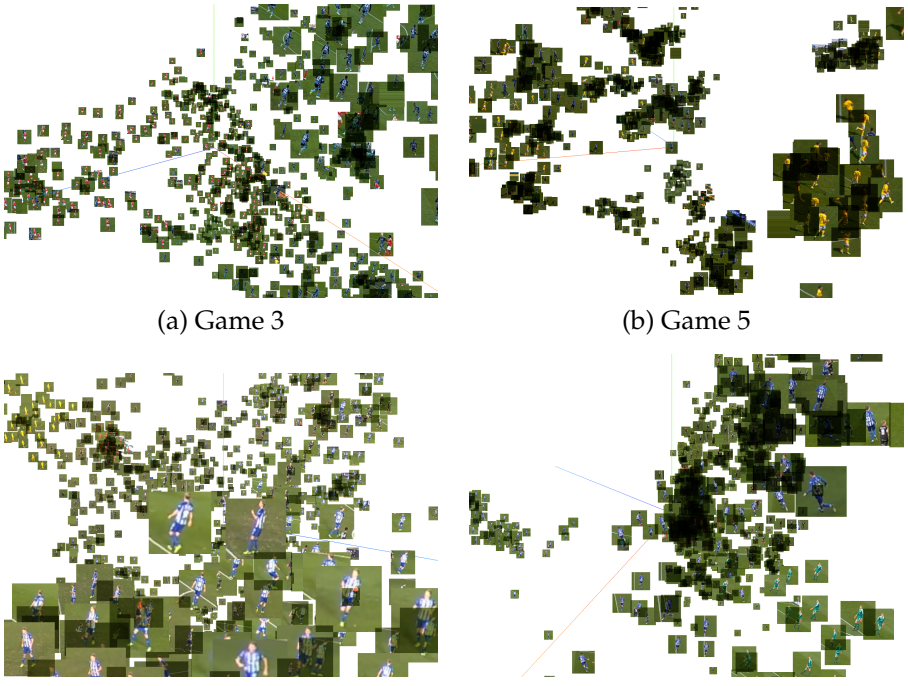
Moreover, since the images in Figure 5.14 are captured to show how the embeddings are scattered around the space and they are not very clear to observe the images of the players, in Figure 5.15 there is a closer inspection of some random parts, respective to each game again.



(a) Game 3 (b) Game 5

(c) Game 14 (d) Game 15

Figure 5.14: Principal Component Analysis per game.



(a) Game 3 (b) Game 5

(c) Game 14 (d) Game 15

Figure 5.15: Principal Component Analysis per game (closer capture).

Another interesting part is the ranking lists of the Baseline model. In Figure 5.16, the list of Game 1 produced by the Baseline model is presented. Despite the fact that it clearly underperforms compared to the other models, it manages to retrieve correctly some images for each query. This is reasonable since the images are captured every 5 seconds, thus there are at least 1 or 2 images per query that are very similar. Since the distances of the pixels of the images are compared, the images that are almost the same are returned correctly as similar, but the rest of the list is almost random and is based on the colors and the shapes that are included in the images. Also, it is noticeable that the goalkeepers are easy to separate since they wear different colors than the rest of the team.

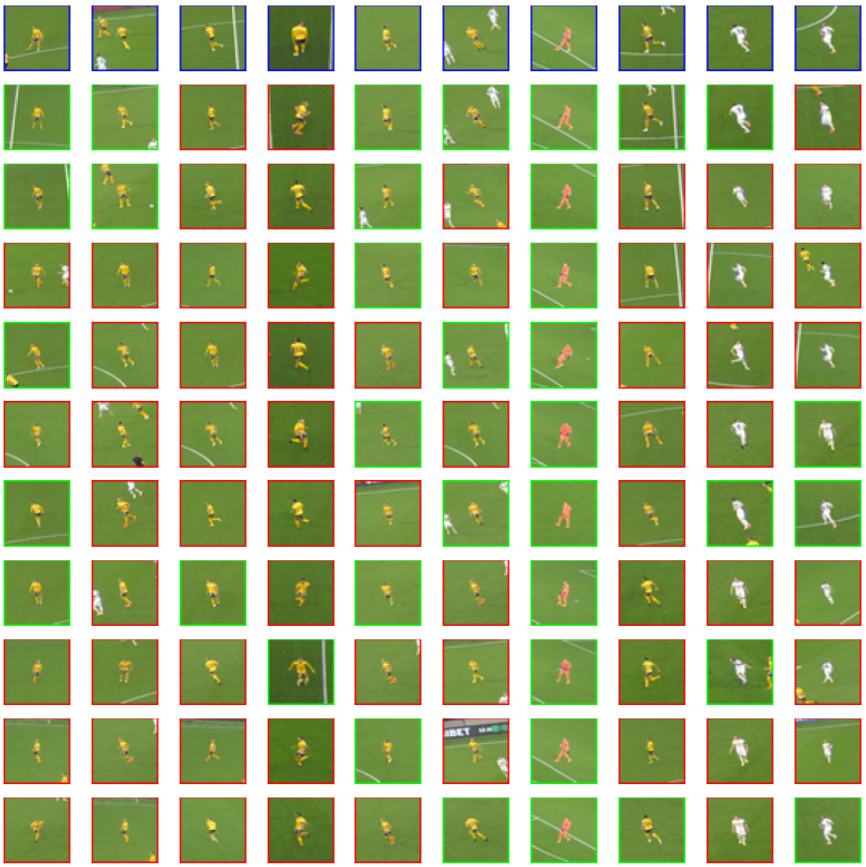


Figure 5.16: Ranking list of Baseline model for Game 1.

An interesting experiment is also to test if the AlignedReID model is able to retrieve correct images using image queries of players that wear either the home or away kit against images of the same players wearing the opposite kit. More specifically, images of players of Malmö FF are used, using the home kit as query, and retrieving images of the same players wearing the away kit.

We observed that the model was able to retrieve correctly images of the same players wearing clothes of different color. In Figure 5.17, there are a few lists presented using the AlignedReID model with Batch Hard. It seems that the model does not separate the players based on the color of their clothing.

Figure 5.17: Home kit players as queries - Away kit as retrieved images. Lists produced using AlignedReID trained with Batch Hard loss.

5.4 Grad-CAM Evaluation

Grad-CAM is also going to be applied on the images used during the evaluation. We will interpret how the model decides to produce the embedding representation of the images in order to place similar images close in space and dissimilar images further away. In Figure 5.18 we present some random images of players along with the heatmaps. The heatmaps are produced by the last convolutional layer of the 5th block of ResNet50. This shallower layer is chosen, as it seems that the feature map maintains a connection to the image space and it is more interpretable.

Shallower layers are also important as they visualize how the model concluded to produce the embedding. Regarding deeper layers, the relation of the feature map to the input image space seems to be less prominent.

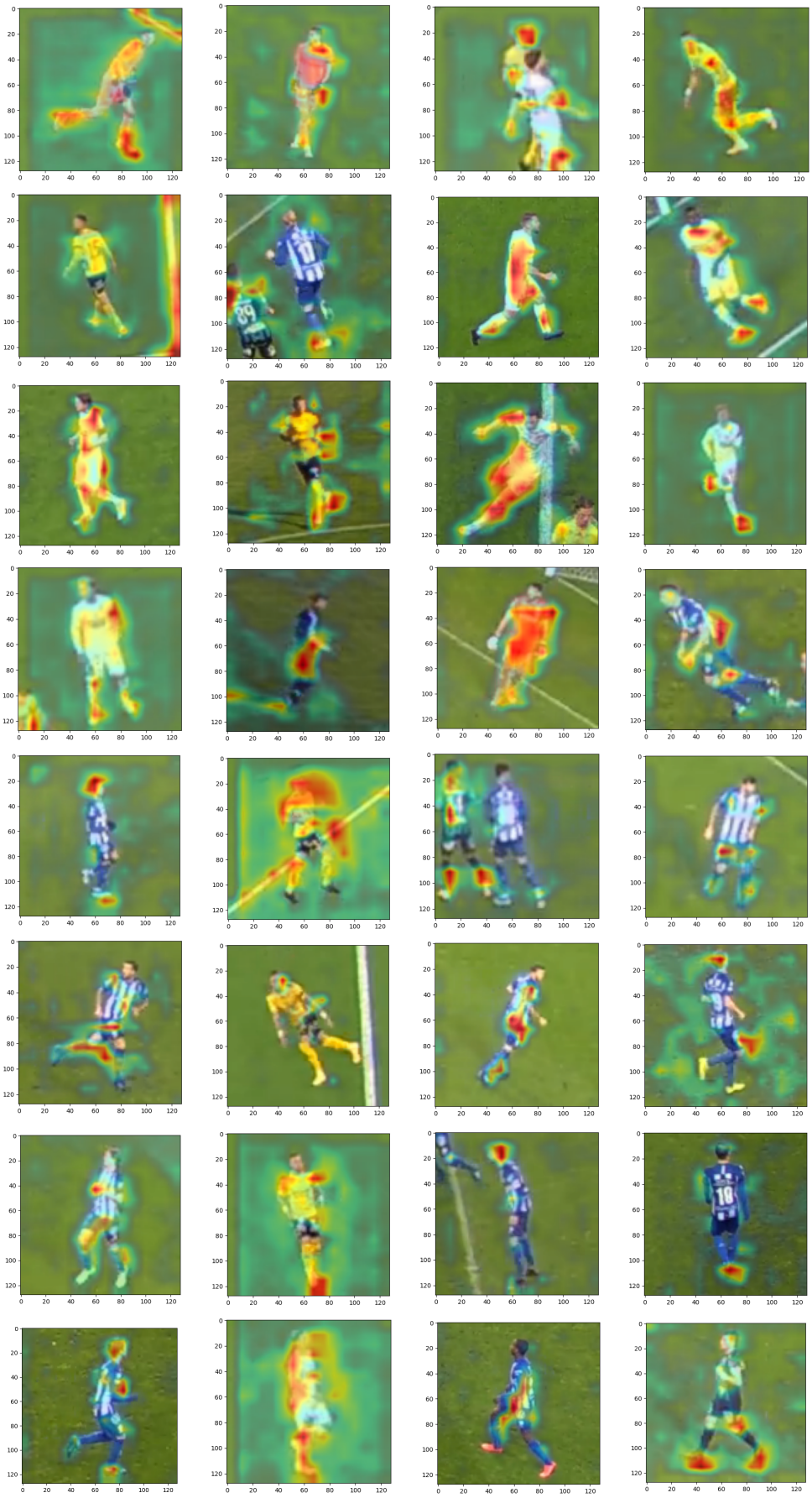


Figure 5.18: Heatmaps produced by the last convolutional layer of the 5th Block of ResNet50.

5.5 Hypothesis Testing

In order to argue which of the models performs better, we will also use a hypothesis testing on the mean Average Precisions (mAPs) of the two models we consider stronger. Those models are the Vanilla and the AlignedReID, using the Batch Hard loss function. We will try to figure if there is a significant difference in their performance, assuming that the most important metric used in the evaluation is the mAP and it is strong enough to decide which of the models performs better. The rest of the metrics are still important, but we assume that a model with significantly higher mAP is more preferable.

The Wilcoxon test will be performed, which is a non-parametric paired test. It is used to compare the medians of two distributions by using the ranks of the distances of the paired observations to calculate the statistic & p-value [13]. The test is conducted using the *wilcoxon()* function from the *scipy.stats* library.

Our data for the test are paired, since there are two mAP values for each of the models, AlignedReID and Vanilla using the Batch Hard loss. Our paired differences do not exactly form a symmetric distribution, but it is also expected as the number of the samples we have is very low, only 17. So, we assume that the differences of the mAPs come from a symmetric distribution and thus the constraint is satisfied.

The null hypothesis that the two distributions have the same median is tested, against the alternative that the mAPs of the AlignedReID have a median greater than the Vanilla model. The median of the mAPs of the AlignedReID is denoted by m_a and the median of the mAPs of the Vanilla model is denoted by m_v.

H_0: $m_v = m_a$
H_1: $m_v < m_a$

An one-tailed test is performed with $\alpha = 0.05$. The resulting p-value of the test is 0.0492. Since p-value < α, we reject the null hypothesis H_0 at 95% confidence, thus the median of the mAPs of the AlignedReID m_a is significantly greater than the median of the mAPs of the Vanilla m_v, thus the AlignedReID model is more preferable.

The same hypothesis is tested using a regular sign test as well, because it does not contain the strict assumption of the symmetry of the distribution of the differences. The sign test intuitively, counts the number of positive and negative differences and calculates the p-value using the binomial distribution table. In this case, n=17 (differences-number of trials), successes are equal to the smaller of either the positive or negative distances (successes=5 negative distances) and the probability used is 0.5. The resulting p-value is 0.0472 and since p-value < α, we reject again the null hypothesis H_0 at 95% confidence.

Furthermore, the evaluation results are investigated further, testing the hypothesis that the mean of distribution of the negatives should be significantly higher than the mean of the distribution of the positives. The distances of positives/negatives are going to be used, merged from all 17 games used in Table 5.1 for the AlignedReID model using the Batch Hard loss function. By this, we attempt to assure that the model is capable of distinguishing the images of the players efficiently and the representations of the players are not mostly close in distance. Thus, the model is able to produce different representations for players that look completely different (distant embeddings) and for players that actually look similar (closer embeddings).

In Figure , the distributions of the positive and negative distances are presented, out of the 17 games used in the evaluation. A t-test is performed to test the hypothesis that the two distributions have equal mean value. We assume that the distances are independent, with similar variance and we also assume that the samples do not violate the normality assumption due to the central limit theorem. That is when the sample size is too large, the means of the distributions are normally distributed, regardless the distribution of the population.

The null hypothesis that the two distributions have equal mean values is tested, against the alternative that the mean of the negative distances is greater than the mean of the positive distances. The mean of the distribution of positive distances is denoted by μ_p and the distribution of negative distances by μ_n.

An one-tailed test is performed again, with $\alpha = 0.05$. The resulting p-value of the test is almost equal to 0. Since p-value < α, we reject the null hypothesis H_0 at 95% confidence, thus the mean of the positive distances μ_p is significantly less than the mean of the negative distances μ_n.

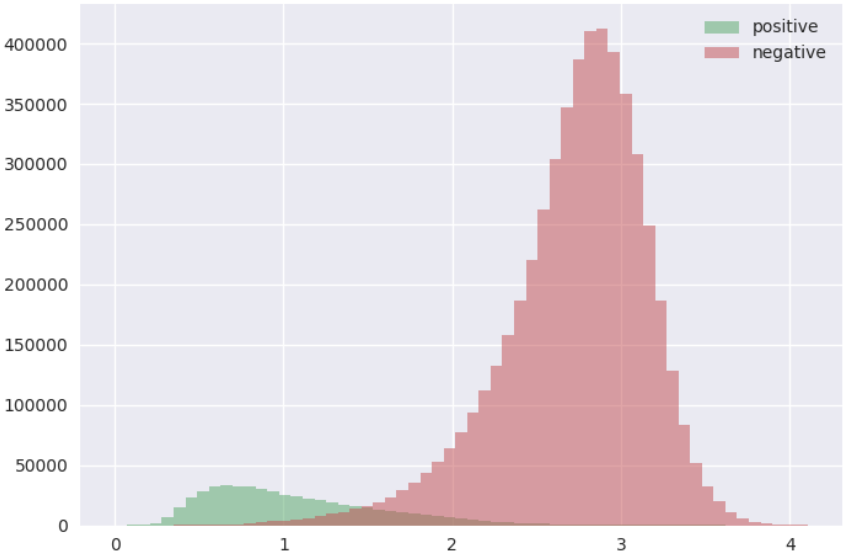


Figure 5.19: Distributions of positive/negative distances. Positive distances are the distances of images of same persons and negative distances are the distances of images of different persons.

H_0: $\mu_p = \mu_n$
H_1: $\mu_p < \mu_n$

We are also interested to investigating if each game individually has the same behavior, since our evaluation is done on each game separately. Thus, one-tailed t-tests are performed with $\alpha = 0.05$ for the distributions of positive and negative distances, similarly as above, but for each game. The distributions per game are similar to the ones presented in Figure 5.13. The resulting p-values for all 17 games as expected are almost equal to 0 in all cases. Thus we reject the null hypothesis H_0 at 95% confidence since p-value < α. Thus, the mean of the positive distances μ_p is significantly less than the mean of the negative distances μ_n for all 17 games.

6 Discussion

In this chapter we are going to discuss the results presented previously in Chapter 5.

6.1 Results

First and foremost, we observe from the evaluation on 17 test games (Table 5.1) that the AlignedReID model using the Batch Hard loss has the best performance regarding the metrics used. The Baseline model as expected performs the worst compared to the other 4 models. As mentioned before, it uses the Euclidean distance of the pixels of the images to retrieve the most similar ones. It is still able to retrieve occasionally some correct matches in the first 5 images, as observed in Figure 5.16, but this is due to the fact that the images used in the evaluation are captured every 5 seconds and thus there are at least 1 or 2 images that are very similar to the query each time.

Also, we observe that the AlignedReID and the Vanilla models using the soft-margin variation perform quite similar, with the AlignedReID being slightly better. On the other hand, using the Batch Hard loss function, the AlignedReID performs clearly better than the Vanilla, especially in terms of mAP. We expected the AlignedReID to perform better than the Vanilla, as it utilizes the alignment of the local features in the images, whereas we did not expect the Batch Hard loss to perform better than its soft-margin variation, based on the results of [5]. This outcome comes in contrast with the evaluation in [5], although it is completely possible as in [5], it is used on a different problem format.

However, AlignedReID is not extremely better than Vanilla as expected. This could be because the AlignedReID is utilizing the alignment of the local features, but the persons in all of the crops are almost perfectly centered. Thus, probably it is quite easy for the Vanilla model as well to separate the players. Maybe if the persons were not centered in the images, the alignment of the local parts could prove even more useful and outperform significantly the Vanilla model.

In the next two paragraphs we will present some comments regarding the training process of the models and how the loss function behave. We observe in Figures 5.3 and 5.4 that Batch Hard and soft-margin behave quite differently. Both of them in the beginning, pull all the embeddings close and towards 0, as it can be confirmed by their shrinkage in the first 150 epochs in Figure 5.3 and their distances are close to 0 in the same epochs in Figure 5.4. But then, after the network starts realizing that it has to push away the embeddings that belong

to dissimilar images, the embeddings start to expand as we can see in Figure 5.3 and there are various distances between the embeddings, shown in Figure 5.4.

Moreover, a difference between the two loss functions is that the soft-margin tends to form embeddings that are much more scattered in the space, which means that they have greater distances compared to the Batch Hard loss. For example, the positive distances using the Batch Hard loss are between 0 and 2, while using the soft-margin variation they are between 0 and 15. Specifically, in Figure 5.4 it is observed that the distances that belong to the 0-th quantile of the distribution of the distances between all the embedding used in each batch is quite different from the rest of the quantiles. This is explained by the fact that each batch is dominated by negative distances, as there are less positives for each image than negatives, thus most of the distances will be negative and they correspond mainly to the rest of the quantiles (5-th, 50-th, 95-th, 100-th). However, the positive distances are stable close to 0 as desired and they do not increase. This is visible both in Figure 5.4 and Figures 5.8,5.9,5.10,5.11 where we see the actual distributions of the distances per epoch. By observing the distributions carefully, we can confirm the fact that in the beginning all embeddings are pulled close, and then the distribution of the negative distances start overcome the distribution of positive distances and then the model start actually to learn.

By looking at the evaluation part, we can notice that the re-identification task is a quite tough task, especially in this context where many of the persons that need to be re-identified wear the same clothes and colors. In the results, we presented only the results of the best model, in terms of the evaluation metrics, so we will comment mainly on that. We can see that the model learned very efficiently how to separate the players. An interesting part is that the model seems to be invariant to colors, as in Figure 5.17 we observe that there are queries of players using the home kit clothes, but the retrieved images also contain the same person wearing the away kit clothing.

We noticed also, that the goalkeepers are an easy task for all 5 models, potentially because they wear gloves and also their clothing is different than the rest of the players. In Figures 5.13 and 5.14 we observe how the distances of the images look like. Firstly, in Figure 5.13 we see that often there is an overlap between the distributions of the positive and negative distances, but in general the result is quite satisfying. Moreover, in Figures 5.14 and 5.15 the distances of the embeddings in the new space is presented using the 3 most significant components of the Principal Components Analysis (PCA). We can notice that there are often clusters of the similar embeddings formed, especially in Game 5, which is almost ideal as the evaluation in this game is extremely good. Of course, this is just a 3D visualization to get an intuition of how they look like, and the distances and the way the embeddings are scattered are an approximation and they differ depending on the angle the image is captured.

As for the qualitative evaluation of the performance of the best model, which turned out to be the AlignedReID using the Batch Hard loss function, we presented the results in Figure 5.18. We observe that the model, mainly focuses on the face, the legs-shoes and the hands of the players in order to produce the embedding representations of the images. By looking at the heatmaps, we understand that the model detects the players by looking at some discriminative features, for instance the face or the body pose and it also focuses on specific parts of the body, such as legs or hands. In many cases, it detects the whole body of the person too. Observing the heatmaps, we cannot really tell if the color of the outfit or the skin tone affect the result. However, the majority of the images shows that the model prefers to focus more on the lower part of the body. There are also cases where the lines of the pitch are detected as well. It could be the case that the position of the person in the field also affects the embedding representation. Last but not least, it is shown in some of the heatmaps that other persons included in the image do also affect the outcome inevitably.

Finally, the results of the hypothesis tests showed that the AlignedReID performs significantly better than the Vanilla using the Batch Hard loss function. Based on the first hypothesis test that showed that the median of the mAPs of the AlignedReID model is significantly greater than the median of the mAPs of the Vanilla model and by comparing the average

values of the metrics that were produced during the evaluation, we have strong grounds to conclude that the best model for this task is the AlignedReID model using the Batch Hard loss. Moreover, observing the results of the rest of the hypothesis tests, that examined if the mean of the distributions of positive and negative distances differ, we conclude that the mean of the negative distances is significantly greater than the mean of the positive distances in all cases. This result assures that the embeddings the model creates and consequently the distances between them are not close at all, thus our results in the evaluation part, such as the ranking lists and the metrics calculated, are not based on "random" decisions of the model.

6.2 Method

In general, the results of the AlignedReID model using the Batch Hard loss function are satisfying and very promising. We observed in the evaluation part, that the model is able to retrieve almost always at least 1 correct image in the first 5 results, and in general the retrieved images are mostly correct, as the mAP is approximately 81%. Such a model could be very useful in real world applications.

A very good advantage of the model is that it is color invariant as well. That is, it does not decide which images to retrieve based on the colors or the outfit, but it focuses on some more discriminative features most of the time. This can be seen in Figure 5.17, as the queries are players wearing the home kit outfit, but the retrieved images belong to the same players wearing the away kit clothing. This is also confirmed by the heatmaps produced by Grad-CAM, as they do not seem to ever focus on the outfit of the player. In total it is a desired result, as we want to avoid using the color of the outfit to separate the players because this feature can potentially make the model prone to easy mistakes. However, it would be handy to incorporate this feature in a way that the model does never retrieve images of players of the opposite team, while it does not contribute too much to finding similar persons.

However there are also some limitations and disadvantages in the approach. First, the training of such models is usually very expensive and time consuming. And since the number of the data needed for this task cannot be reduced a lot, this issue is not resolved easily. Also, the training as mentioned before is done in this case on images of persons that play for teams in the Swedish league "Allsvenskan". That means, that the model is trained on a specific type of camera that is used in all stadiums. If we were to apply it to different type of cameras around the world, there is a chance that the results would not be that satisfying. It is usual in tasks like this, that the type, the quality and the resolution of the cameras affect significantly how the model perceives the images. This may be caused by the fact that the model learns some biases from the domain of the used dataset, for example the grass color.

6.3 The work in a wider context

Regarding the scope of this thesis, there is no noticeable violation of any ethical or societal aspects. The topic is soccer, where players expose themselves publicly through television and this is not different from what this thesis does. By observing the results, there is no obvious gender or race bias which is usually the case in such applications. However, this approach opens the way for more applications, not only in sports. In this case, such an application could potentially become unethical if not used appropriately, for instance if used for surveillance purposes.

In contrast, there is a positive outcome through this task. That is, players that are not quite famous currently have the chance to stand out through this application and gain some fame. This is achieved through the objective metrics that are provided, for instance the average speed of the player, the total distance covered etc.

7 Conclusion

To conclude, in this thesis 5 different model approaches were compared on the task of person re-identification. The specific use case evaluated in this thesis refers to associating images of the same players in a football game. A Baseline approach was established by simply using the the Euclidean distance in the image space, which is the distance between the image pixels. Additionaly, the Vanilla and the AlignedReID were developed, for which two different losses were used. The Vanilla model is a subset of the AlignedReID which contains only the global feature branch, whereas the AlignedReID contains also a local feature branch. Regarding the loss functions, the Batch Hard and its soft-margin variation were used for both approaches.

Our findings show that the AlignedReID using the Batch Hard loss function performs better than any other model. This is concluded by comparing the metrics used for the evaluation of the models, both by comparing the averages and by statistical comparison. In general, all of the models seem to perform decently. The ones using the soft-margin variation are almost equivalent, while the Vanilla model using Batch Hard also performs very well. All of the deep learning models performed much better than the Baseline model. As a result, stripe-based methods and global feature-based methods are appropriate for this task of re-identification. Especially, the AlignedReID model using Batch Hard seems to be the most appropriate and able to be applied in real world tasks efficiently.

Moreover, there was a qualitative evaluation performed using the best model. Specifically, Grad-CAM was used to inspect where the model focuses in the images to produce the embeddings of the images. Our results shows that it seems to ignore features like jersey numbers and clothing color. Instead, it focuses on faces, legs and hands while most of the time it prefers to focus on the lower part of the person's body.

Finally, by performing Principal Component Analysis (PCA) and by statistically comparing the distances of images of similar and different persons, it is concluded that the model actually learned a way to produce embedding representation that are close in space for images of the same person and further away for dissimilar images. Thus, the retrieval of similar images has a ground and is not based on random mapping of images.

As for further research, it is undeniable that the type of the camera and the quality/resolution of the images potentially affects the result in this kind of tasks. Thus, a further improvement would be to train the model using images from different type of cameras and investigate if the model is able to generalize well.

A different idea is to try an approach that incorporates some kind of pose estimation. Of course, an approach like this demands extra available data, but it seems that potentially it could perform very well on the Re-ID task as it could take advantage of the differences of the persons' poses and discover new ways to separate the players.

Finally, as we observed, the spatial information in deeper layers are less prominent, that's why Grad-CAM was applied in a shallower layer. Thus, a different approach in AlignedReID would be to apply the local parts alignment in either one of the shallower layers or additively in many layers. Potentially, it could provide better separation of the images by aligning the local parts in layers that the spatial information is more distinguishable.

Bibliography

[1] Lei Chen, Jianhui Chen, Hossein Hajimirsadeghi, and Greg Mori. *Adapting Grad-CAM for Embedding Networks*. 2020. arXiv: 2001.06538 [cs.CV].

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[5] Alexander Hermans, Lucas Beyer, and Bastian Leibe. *In Defense of the Triplet Loss for Person Re-Identification*. 2017. arXiv: 1703.07737 [cs.CV].

[6] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[7] Mahdi M. Kalayeh, Emrah Basaran, Muhittin Gokmen, Mustafa E. Kamasak, and Mubarak Shah. *Human Semantic Parsing for Person Re-identification*. 2018. arXiv: 1804.00216 [cs.CV].

[8] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[9] Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. *The Deep Bootstrap Framework: Good Online Learners are Good Offline Generalizers*. 2021. arXiv: 2010.08127 [cs.LG].

[10] Sushant Patrikar. *Batch, Mini Batch Stochastic Gradient Descent*. 2019. URL: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a.

[11] Lei Qi, Jing Huo, Lei Wang, Yinghuan Shi, and Yang Gao. *MaskReID: A Mask Based Deep Ranking Neural Network for Person Re-identification*. 2019. arXiv: 1804.03864 [cs.CV].

[12] Xuelin Qian, Yanwei Fu, Tao Xiang, Wenxuan Wang, Jie Qiu, Yang Wu, Yu-Gang Jiang, and Xiangyang Xue. *Pose-Normalized Image Generation for Person Re-identification*. 2018. arXiv: 1712.02225 [cs.CV].

[13] Denise Rey and Markus Neuhäuser. "Wilcoxon-Signed-Rank Test". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1658–1659. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_616. URL: https://doi.org/10.1007/978-3-642-04898-2_616.

[14] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. *Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking*. 2016. arXiv: 1609.01775 [cs.CV].

[15] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[16] M. Saquib Sarfraz, Arne Schumann, Andreas Eberle, and Rainer Stiefelhagen. *A Pose-Sensitive Embedding for Person Re-Identification with Expanded Cross Neighborhood Re-Ranking*. 2018. arXiv: 1711.10378 [cs.CV].

[17] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[18] Yifan Sun, Liang Zheng, Weijian Deng, and Shengjin Wang. *SVDNet for Pedestrian Retrieval*. 2017. arXiv: 1703.05693 [cs.CV].

[19] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. *Beyond Part Models: Person Retrieval with Refined Part Pooling (and a Strong Convolutional Baseline)*. 2018. arXiv: 1711.09349 [cs.CV].

[20] Christian Versloot. *What are Max Pooling, Average Pooling, Global Max Pooling and Global Average Pooling?* 2020. URL: https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/.

[21] Longhui Wei, Shiliang Zhang, Hantao Yao, Wen Gao, and Qi Tian. "GLAD". In: *Proceedings of the 25th ACM international conference on Multimedia* (Oct. 2017). DOI: 10.1145/3123266.3123279. URL: http://dx.doi.org/10.1145/3123266.3123279.

[22] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. *AlignedReID: Surpassing Human-Level Performance in Person Re-Identification*. 2018. arXiv: 1711.08184 [cs.CV].

[23] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. "Scalable Person Re-identification: A Benchmark". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1116–1124. DOI: 10.1109/ICCV.2015.133.

[24] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. "MARS: A Video Benchmark for Large-Scale Person Re-Identification". In: vol. 9910. Oct. 2016, pp. 868–884. ISBN: 978-3-319-46465-7. DOI: 10.1007/978-3-319-46466-4_52.

[25] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. "Scalable Person Re-identification: A Benchmark". In: Dec. 2015, pp. 1116–1124. DOI: 10.1109/ICCV.2015.133.

[26] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, and Yi Yang. *Camera Style Adaptation for Person Re-identification*. 2018. arXiv: 1711.10295 [cs.CV].

A Appendix

A.1 Plots

In this section there are included some of the plots presented in the Results chapter in full width. The purpose is to provide a more detailed observation of the graphs.

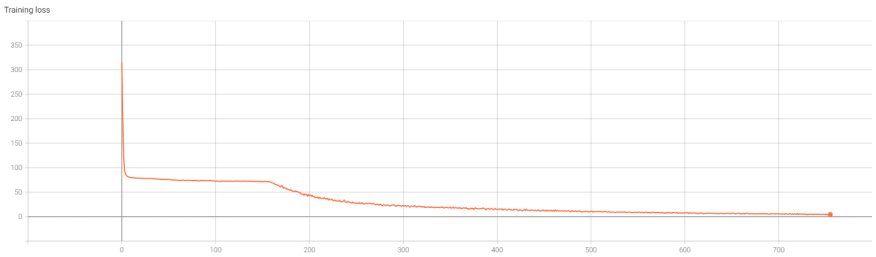


Figure A.1: Training loss of Vanilla model using Batch Hard loss.

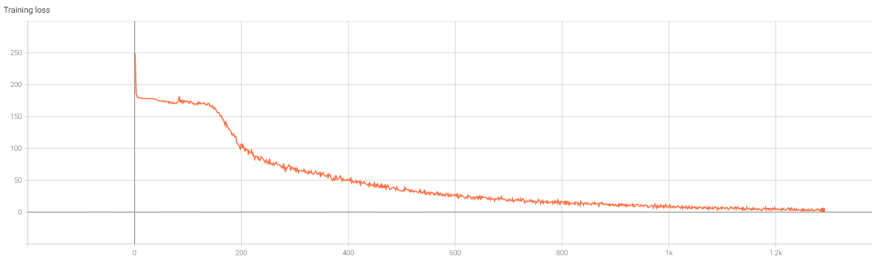


Figure A.2: Training loss of Vanilla model using soft-margin variation loss.

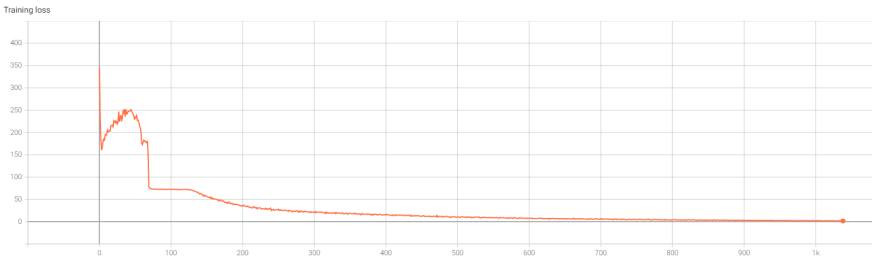


Figure A.3: Training loss of AlignedReID model using Batch Hard loss.

Figure A.4: Training loss of AlignedReID model using soft-margin variation loss.

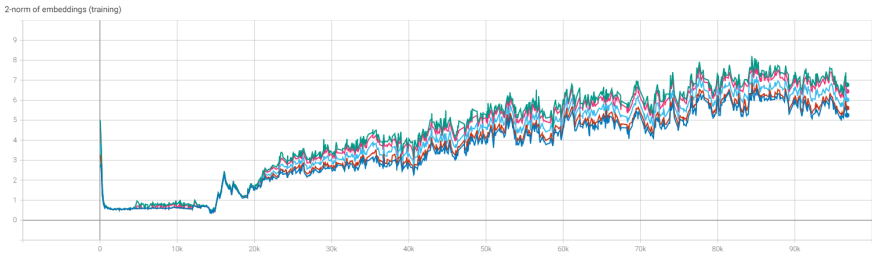


Figure A.5: 2-norm of embeddings of Vanilla model using Batch Hard loss.

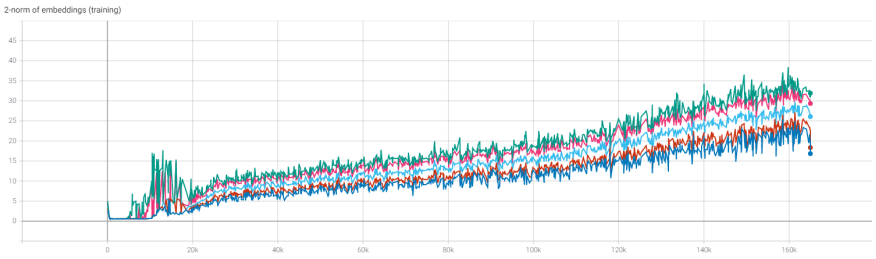


Figure A.6: 2-norm of embeddings of Vanilla model using soft-margin variation loss.

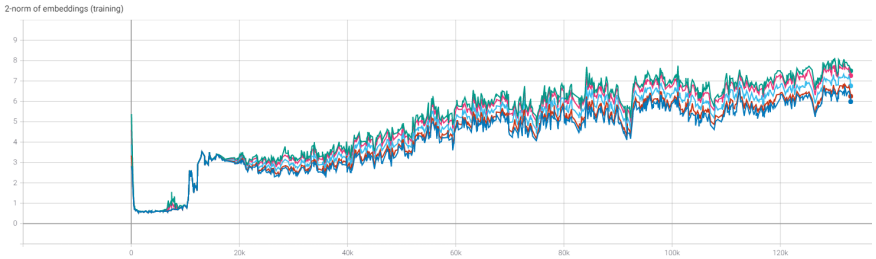


Figure A.7: 2-norm of embeddings of AlignedReID model using Batch Hard loss.

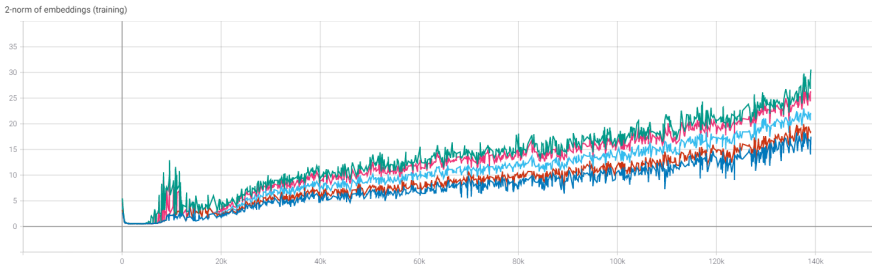


Figure A.8: 2-norm of embeddings of AlignedReID model using soft-margin variation loss.

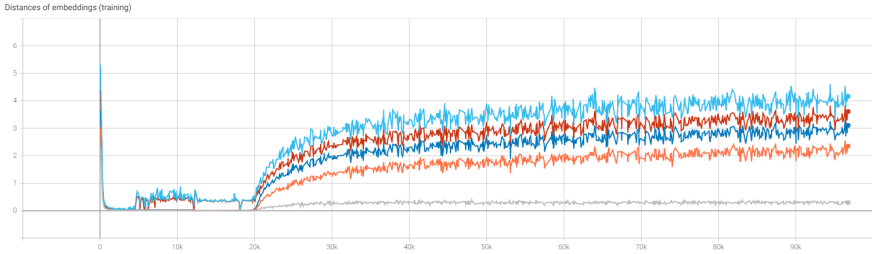


Figure A.9: Distance of embeddings of Vanilla model using Batch Hard loss.

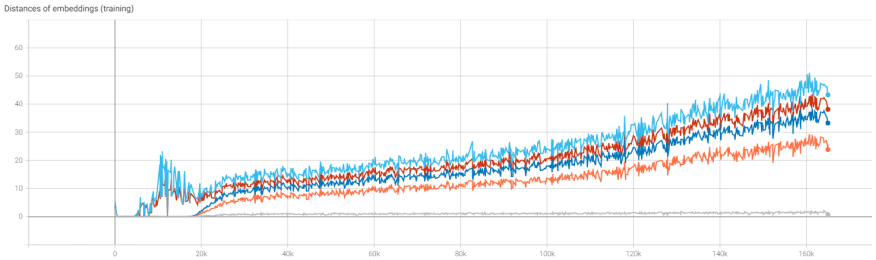


Figure A.10: Distance of embeddings of Vanilla model using soft-margin variation loss.

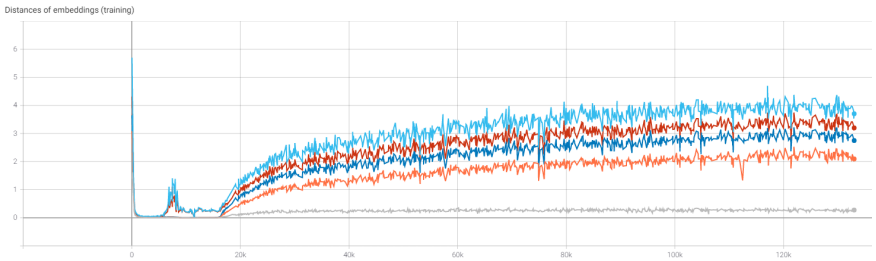


Figure A.11: Distance of embeddings of AlignedReID model using Batch Hard loss.

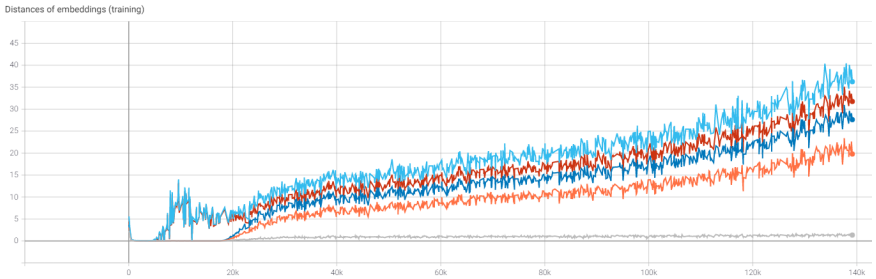


Figure A.12: Distance of embeddings of AlignedReID model using soft-margin variation loss.

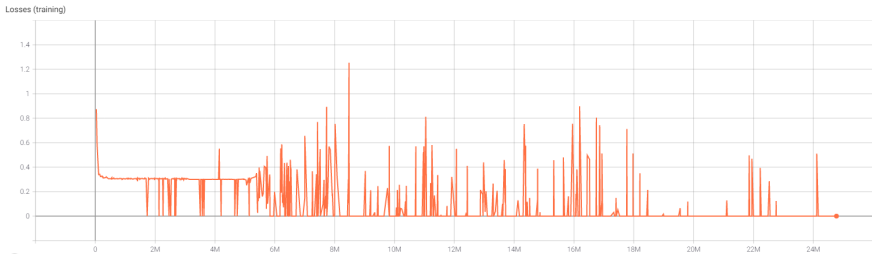


Figure A.13: Losses of Vanilla model using Batch Hard loss.

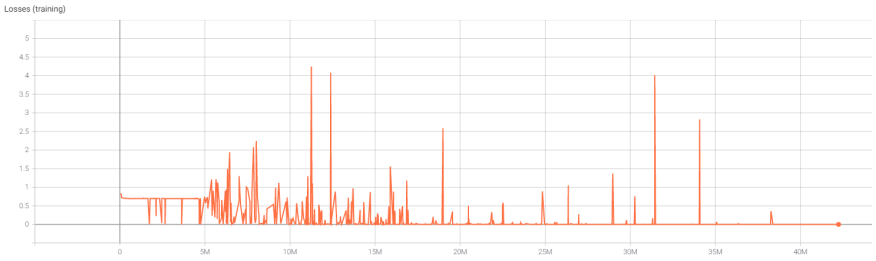


Figure A.14: Losses of Vanilla model using soft-margin variation loss.

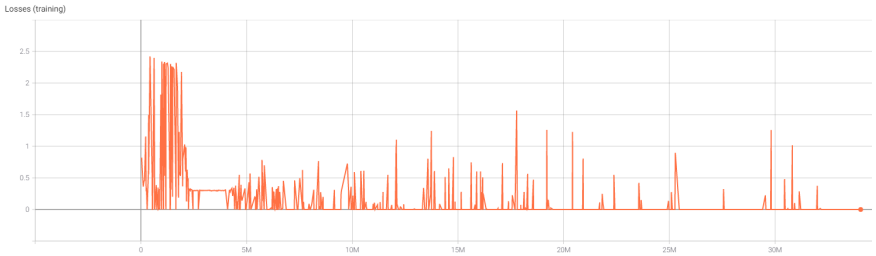


Figure A.15: Losses of AlignedReID model using Batch Hard loss.

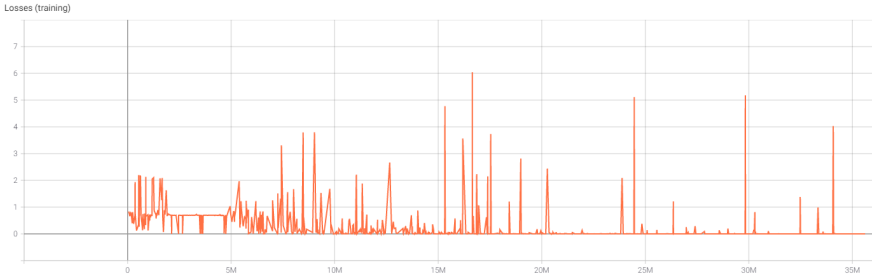


Figure A.16: Losses of AlignedReID model using soft-margin variation loss.

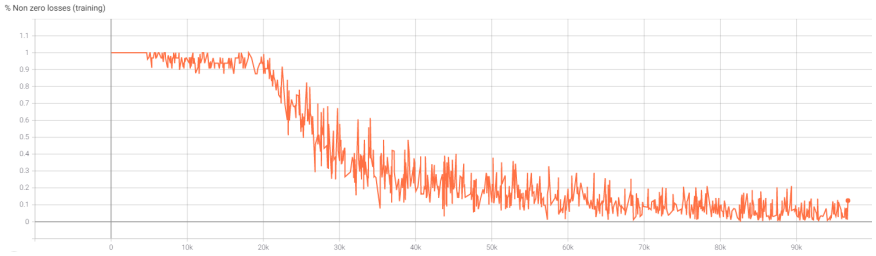


Figure A.17: Non-zero losses of Vanilla model using Batch Hard loss.

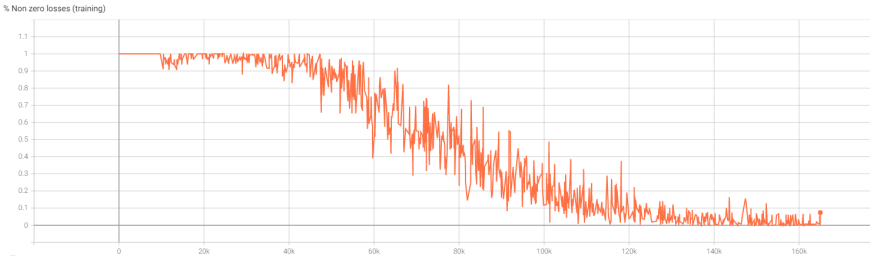


Figure A.18: Non-zero losses of Vanilla model using soft-margin variation loss.

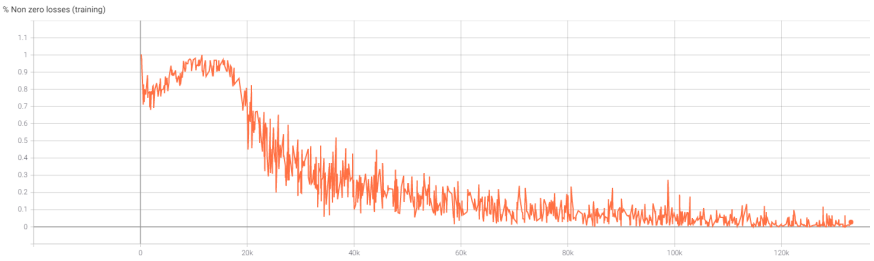


Figure A.19: Non-zero losses of AlignedReID model using Batch Hard loss.

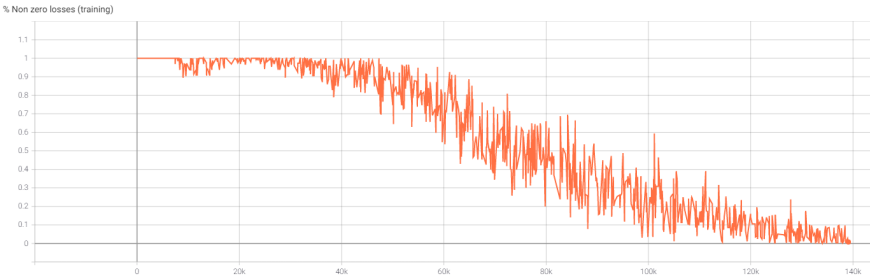


Figure A.20: Non-zero losses of AlignedReID model using soft-margin variation loss.

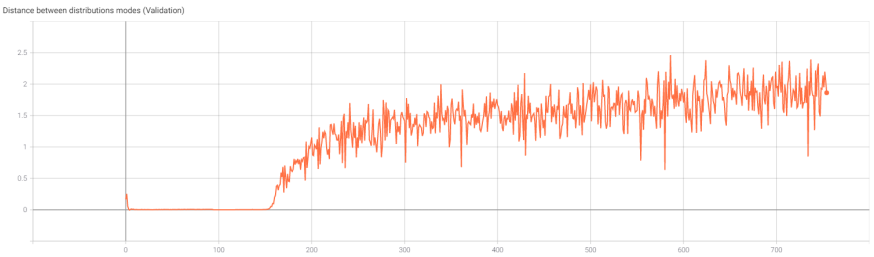


Figure A.21: Distance modes in validation set of Vanilla model using Batch Hard loss.

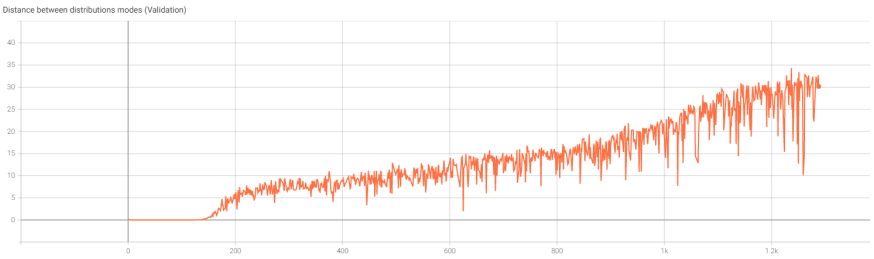


Figure A.22: Distance modes in validation set of Vanilla model using soft-margin variation loss.

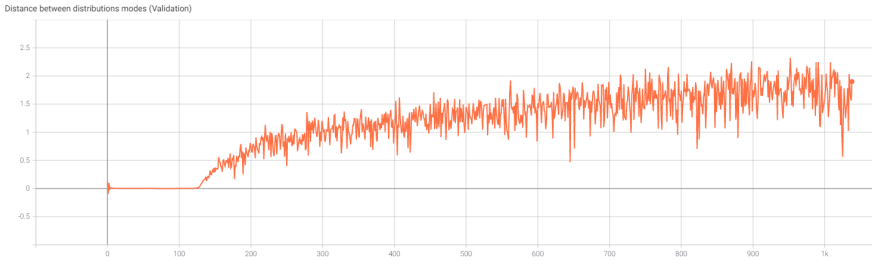


Figure A.23: Distance modes in validation set of AlignedReID model using Batch Hard loss.

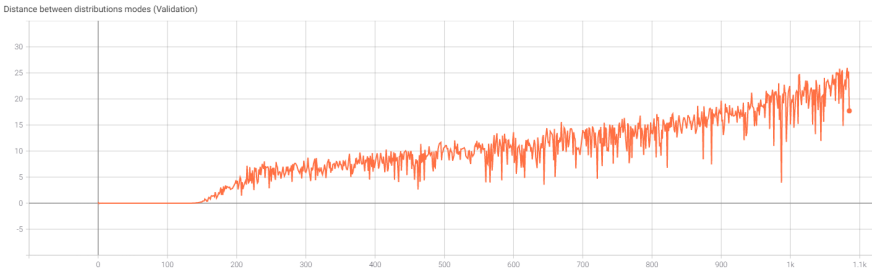


Figure A.24: Distance modes in validation set of AlignedReID model using soft-margin variation loss.