Implementation of Different Rotation Based Methods for Unsupervised Domain Adaptation

1st Zahra Karimi
Politecnico di Torino
Turin, Italy
s302612@studenti.polito.it

2nd Mohammad Asgari
Politecnico di Torino
Turin, Italy
s289859@studenti.polito.it

3rd Gioele Giachino
Politecnico di Torino
Turin, Italy
s295380@studenti.polito.it

At this link you can find all the code produced for this project.

***Abstract*—Deep neural networks usually have a good ability to perform common tasks in the field of image processing if the required data is available. However, suitable and sufficient labeled data for their training may not always be available. For this purpose, the use of unsupervised DA methods can make it possible to use supervised labeled datasets as sources to predict unlabeled target datasets. The issue that has attracted our attention in this research is that usually the conventional methods in DA are not designed for problems with RGB images. In this project, we try to reduce the synthetic-to-real domain shift by exploiting the inter modal relation between the RGB and Depth image. Hence, our goal is to train CNN networks in order to recognize images, in addition to achieving a good pretext model in order to perform DA properly.**

I. Introduction

In the branch of unsupervised learning methods, self-supervised ones represent some kind of "middle solution" between the two sides of supervised/unsupervised approaches. The capability to learn features from both large scale image and video sources, mixed with the mitigation of costs and failures due to human error, constitute the main advantages in using this kind of algorithms. The practical applications of these self-supervised methods cover a plenty of real cases, such as image captioning, semantic segmentation, but also object recognition.

The process of learning a discriminative classifier or other predictors when there is a shift between the distributions of the training set and the test set is referred to as "domain adaptation" (DA).

RGB-D data, if we want to make a comparison with "barely" RGB ones, contains additional information(thanks to the presence of depth), such as the object shape. An important issue we have to cope with, when facing real-world environments, is the fact that objects are extremely susceptible to noises, so when developing our algorithms, we must implement data augmentation techniques in order to improve the robustness of our training set.

We can increase their performance by gathering more extensive datasets, gaining a deeper understanding of more robust models, and implementing more effective strategies to avoid issues such as the typical overfitting.

In addition to visual information, modern cameras can give us with more comprehensive information from a variety of angles regarding an object. The wide complexity of the object identification challenge, on the other hand, means that this problem cannot be defined even by a dataset as huge; hence, our model should also include a great deal of previous knowledge in order to compensate for all of the data that we do not hold.

II. Related Work

A. Unsupervised Domain Adaptation

In the context of shallow learning, a number of strategies for domain adaptation have been proposed.

The ability to learn a mapping between domains in situations in which the target domain data are either completely unlabeled (also known as unsupervised domain annotation) or have a small number of samples that have been labeled is one of the appealing aspects of the domain adaptation approaches (semi-supervised DA).

Although the proposed method is capable of being generalized to the semi-supervised case in a fairly straightforward manner, we will concentrate on the more difficult unsupervised case in the following sections [1], [2].

Numerous strategies aim to bound the target error by the sum of the source error and a notion of distance between the source and the target distributions in order to tackle the challenging task of DA. For the purpose of DA, several different notions of distance have been proposed (Ben-David et al., 2006, 2010; Mansour et al., 2009a,b; Germain et al., 2013) [3]–[5].

We should also take a look at a couple of examples that are related to this one. RGB data is converted to depth data by taking it to be the source and target domains, respectively, in Spinello and Arras [6], and Hoffman et al [7]. For istance, Li et al. [8] consider a situation in which the source dataset consists

of RGB-D photos but the target dataset only comprises RGB pictures.

B. Self-Supervised Visual Tasks

The development of self-learning mechanisms with unstructured data that can expand the research and development of low-cost generic AI systems stands as one of the top priorities of AI researchers. To address this issue, researchers are developing self-supervised learning algorithms capable of detecting small data variations.

The unsupervised problem is converted into a supervised problem by automatically generated labels. To utilize the vast amount of unlabeled data, it is necessary to establish the correct learning objectives in order to receive guidance from the data itself. The objective of the self-supervised learning method is to distinguish between hidden and unhidden portions of the input [9].

Self-supervised learning is a suitable method for regression and classification tasks, whereas unsupervised learning is useful for clustering and dimensionality reduction [10], [11].

III. Datasets

In this section, we present two different datasets used to evaluate our method on object categorization. We use the popular RGB-D Object Dataset (ROD) for the real data and the synthetic counterpart collected,named instead synROD.

1) ROD: This dataset has become the main reference dataset for RGB-D object recognition in the robotics community. It contains 41,877 RGB-D images of 300 objects commonly found in house and office environments grouped in 51 categories. Each object is recorded on a turn-table with the RGB-D camera placed at approximately one meter distance at 30◦ , 45◦ and 60◦ angle above the horizon.

2) SynROD: This dataset is collected by using a synthetic dataset, created using the combination of two different protocols. Firstly we collect the 3D object models from web resources and secondly for rendering 2.5D scenes we use a ray-tracing engine in Blender, in order to simulate photorealistic lighting. The final result of the selection stage is a set of 303 textured 3D models from the 51 object categories of ROD, for an average of about 6 models per category.

In our experiments, we evaluate RGB-D DA methods by considering synROD as the synthetic source dataset and ROD as the real target dataset.

IV. METHOD

A. Preparing Datasets

To make the two datasets comparable, we randomly select and extract approximately 40,000 objects crops from synROD to match the dimensions of ROD. In our experiments, we evaluate RGB-D DA methods by considering synROD as the synthetic source dataset and ROD as the real target dataset [14]. Their huge volumes does not give us the possibility to recall all the data available on the RAM space at the same time, and for this reason, it is necessary to split data in equal batches.

The Dataloader tool in PyTorch library gives us the chance to preparing data in consecutive batches, where only one batch is transferred to the RAM space at any time. During the preparation stage, in addition to the construction of orderly batches, some primary operations must also be done on the image. For instance, considering that the initial layer of the network used in this project corresponds to the input layer of the ResNet-18 network, all our selected images should be resized to the dimensions of the ResNet-18 network, i.e. 256*256.

With the goal of minimize the data dispersion of image inputs, it is imperative to normalize them. In general, the mean and standard deviation of the ImageNet image data are used to normalize the images (mean= [0.485, 0.456, 0.406], std= [0.229, 0.224, 0.225]), whereas we have normalized the image with our manually calculated mean and standard deviation for both ROD and SynROD dataset (Table I).

TABLE I
NORMALIZING THE RGB AND DEPTH IMAGES BY MEAN AND STANDARD DEVIATION FOR BOTH ROD AND SYNROD DATASETS

	ROD	SynROD
RGB-mean	[0.5547, 0.5310, 0.5190]	[0.3107, 0.2884, 0.2595]
RGB-std	[0.2122, 0.2184, 0.2499]	[0.2352, 0.2268, 0.2336]
Depth-mean	[0.7426, 0.2956, 0.4785]	[0.4422, 0.4014, 0.6852]
Depth-std	[0.2012, 0.2030, 0.2916]	[0.1382, 0.2507, 0.3628]

In order to be sure that each RGB and Depth images are correct and corresponds to each other, when working with the datasets, we plot and check a few images from one of the available batches on both ROD and SynROD dataset (Figure 1 and 2).

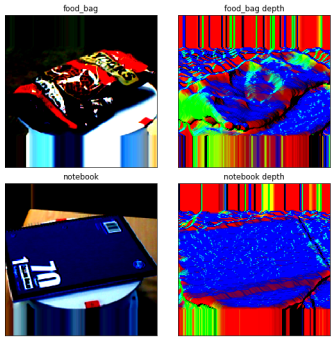

Fig. 1. : Test sample of RGB (left column) and RBG-D (right column) images on ROD dataset

B. Overview

1) Main Task: With object recognition we refer to the operation of identification of each and every noteworthy object in an image or a video; specifically, in our case we deal with image sources. This kind of procedure is first of all characterized by a learning part, where handling large scale

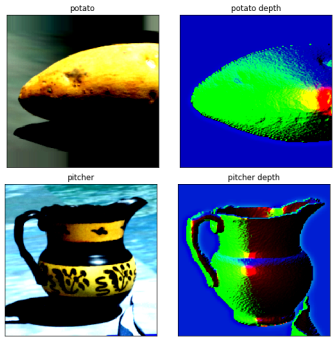

Fig. 2. Test sample of RGB (left column) and RBG-D (right column) images on SynROD dataset

datasets our AI system improves his capability to correctly recognize the different classes of objects.

2) Pretext Task: The pretext task consists of predicting the relative rotation between a pair of RGB and Depth images that have been independently rotated. Since the ground truth for this simple pretext task can be generated automatically from the data, we can train the network to predict the relative rotation using both source and target data in a self-supervised fashion [14].

Assuming that our data do not have labels, by using this method we can teach the network structure to find a relationship between RGB and corresponding Depth images that are independently rotated. To do that, our model should be able to recognize almost all the components of the image well. Our network learns to correctly distinguish image components and eventually predicts the rotation angle.

Every RBG and depth images can be rotated independently and their rotation is limited to 0,90,180 and 270 degrees; we want to measure relative rotation between those rotated images. We must consider a positive direction in the rotation, that is, we must measure how much rotation our depth image has had in relation to the RGB image in a state with a positive rotation. Done that, if the result is a negative number, we will sum to its value 360 degrees in order to have normalization and consistency among all data, for example, when it reaches -90 degrees, it changes to 270 degrees (360+t). In figure 4, we can see a test example consisting of rotated images, which shows the high level of accuracy guaranteed by this work.

In Figure 4, there is a test example of the production of a dataset consisting of rotated images, which shows that this work has been done with complete accuracy.

3) Network architecture : In figure 3, we can see a representation of the Convolutional Neural Network (CNN) built for our method. The network structure is composed of three main parts. In the first part, we make us of two CNN networks, which help us in obtaining image features. These networks are selected basing on the pre-trained model of ResNet-18 applied to the ImageNet dataset. In general, the first layers of neural networks are characterized by a good ability to retrieve general information related to the image after the learning process.

Applying the pre-trained ResNet model with the aim of extracting image features permits us to save processing time and consequently reach the result faster. For this purpose, the two ending layers of ResNet, which are global average pooling and soft max (which has learned to distinguish the images in the ImageNet groups) are removed from the network and we restrict our usage to the other layers. Both the RGB image and its corresponding Depth image are each one fed separately by a ResNet network with independent weights and then we concatenate their output results around the axis corresponding to the images channel, sending next them to the following step of the network.

The main head part is responsible for recognizing the group of each image, relying on the features obtained in the first step. For this purpose, by flattening the information obtained in the CNN structure, we transfer the data to the fully connected layer and finally reach a layer with SoftMax activation function and the number of neurons equal to the classes in the input images. In fact, in this layer, we determine the probability that the corresponding image belongs to each one of the different groups.

In summary:

- For what it regards the feature extractor, we make use of a two-stream CNN, generating RGB-D features via a late fusion approach. In a nutshell, the two identical CNNs process one the RGB and one the depth RGB-D image, and after that their outputs are concatenated in order to compose final features. Moving on, we observe the two heads, the main(M) and the pretext(P), respectively.
- Starting from the main head, this network works which a C-way classification problem, with C equals to the number of object classes which we want to predict. This problem is constructed as [gap, fc(1000), fc(C)] ,where gap indicates a global average pooling operation, fc(n) shows a fully connected layer with n neurons. For what concerns activation functions, fc(1000) are characterized by ReLU, whereas fc(C) use softmax.
- Ending with the pretext head, this one solve itself a 4-way classification problem, about the prediction of the rotation between the RGB and the depth RGB-D image, built as it follows:[conv(1 x 1, 100), conv(3 x 3, 100), fc(100), fc(4)]. All convolutional and fully connected layers use batch normalization and ReLU activation function, except for fc(4) that uses softMax activation function.

C. Implementation details

1) Loss Function: This function helps our network to check how well our estimate of the output of the network corresponds to reality. Each of the neurons in the output layer specifies a specific class in our dataset. When an image passes through the network, in the final layer, the number will be obtained as many classes as we have. Each number specifies the probability that the object belongs to the corresponding class. We also have the correct data class label as a tensor. The value of this tensor is equal to one in the index that specifies

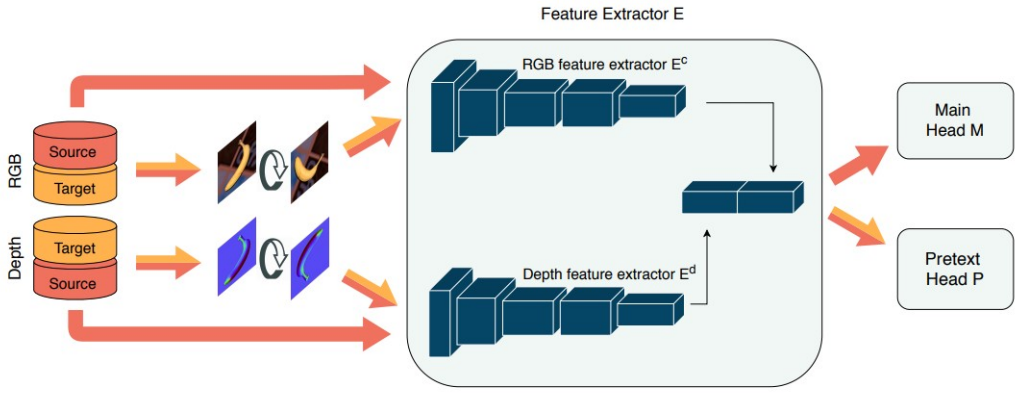

Fig. 3. Here we can observe the general structure of our method for RGB-D DA. Firstly, the blue squares represent the CNN, composed by a two-stream feature extractor E and by two network heads, the main and the pretext, respectively M and P. The first is trained using object recognition with the labeled source data (see the red arrow), while the second exploits both source and target samples. Finally, both RGB and Depth images are independently rotate before being transferred to the network

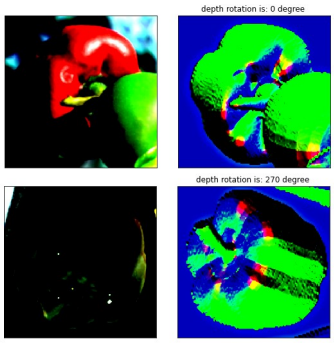

Fig. 4. Test sample of SynROD dataset: it shows the relative rotation between the RGB and the Depth image after they have been independently rotated; the RGB image is rotated by 0 ∘ compared to the depth image (top row), whereas the RGB image is rotated by 270 ∘ compared to the depth image (bottom row)

its correct class and is equal to zero in other indices. Having the output of the network and the actual class of the image, we should somehow be able to get a correct result of the network's performance. Here we face a classification problem in several categories, and one of the most used loss functions in these problems is cross-entropy loss, which measures the actual label of the image and the percentage of the belonging an image to other labels using the following formula (1,2).

$$l_m = -\frac{1}{N_s}\sum_{i=1}^{N_s} y_i^s . \log(\hat{y_i^s}), \tag{1}$$

$$l_p = -\frac{1}{\tilde{N_s}}\sum_{i=1}^{\tilde{N_s}} z_i^s . \log(\hat{z_i^s}) - \frac{1}{\tilde{N_t}}\sum_{i=1}^{\tilde{N_s}} z_j^t . \log(\hat{z_j^t}), \tag{2}$$

In fact, the cross-entropy loss function has been used three times. Once to measure the classification of images in the source dataset, and once to measure the performance of the network in estimating the rotation angle of the source dataset and finally measuring the rotation of the target dataset.

How loss function of these two methods (Lm and Lp) can affect the changes of our network weights is one of the important hyperparameters of our problem. In this implementation, similar to [14], this number is considered equal to one, and the effect of errors on the weights of our network will be the same.

2) Optimization: In this part, Stochastic Gradient Descent(SGD) method is used to optimize network weights. This method has two hyperparameters of learning rate and momentum, whose values are considered equal to 0.0003 and 0.9, respectively. In addition to changing the values of the hyperparameters of this method (SGD), for the purpose of comparison, we may be able to use other optimization methods, especially ADAM, which have a good ability to converge equations at a higher speed.

3) Prevent overfitting: One of the common problems that neural networks may face is the problem of overfitting on the test data set. This means that the network on which the train operation has been performed memorizes the input data and based on the memory, it can perform very well on the train data set. This means that the network on which the train operation has been performed memorizes the input data and based on the memory, it can perform very well on the train data set. However, since the network has not obtained the correct detection capability based on image features, the results obtained on the train dataset cannot be generalized, and as a result, the network will perform poorly on the test dataset. To solve this problem, various solutions can be used, including the use of drop out and weight decay. Also, to prevent overfitting, weight decay equal to 0.05 has been used. This weight actually applies a penalty on the weights of the network based on the following formula(3).

$$Lnew(w) = Loriginal(w) + \lambda w' w, \tag{3}$$

In the main implementation, by using dropout weight of 0.5, in the layers related to both pretext and main head tasks, half of the neurons of both networks are deactivated in each step of the simulation, which is a relatively large number.

V. EXPERIMENTS

In this section, we conduct experiments to validate the proposed DA method for both object recognition and predicting image rotation. More precisely, section V-A describes the Source-Only implementation, section V-B presents the result of using rotation classification as a pretext task and section V-C show the result of using rotation regression as a pretext task.

A. Source-Only implementation

In the first step of performing simulations, we perform the Source-Only implementation. When applying the Source-Only method without using DA, we merely train the network basing on the source dataset (SynROD) and we finally evaluate its performance using the target dataset (ROD).

The accuracy obtained by this method determines how much it is possible to correctly predict the class of each image without seeing the images of the target dataset. In this section, the source-only method is carried out in the form of e2e, which means that by entering train dataset images into the network and calculating the loss function, all network weights are simultaneously reviewed and changed by means of the optimization algorithm.

The obtained results indicate that, without using any method for DA, an accuracy of about 40 per cent can be reached in the target dataset. This number can provide us with a suitable basis for evaluating the performance of the network in the use of DA methods[TABLE II].

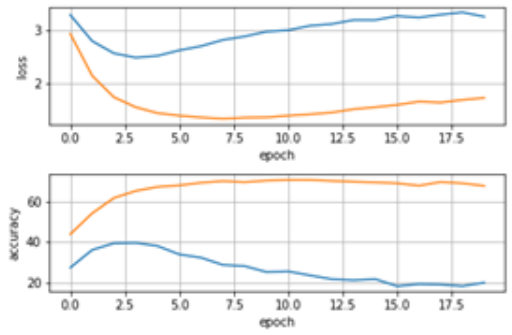

Fig. 5. Accuracy and average loss versus epoch for source only implementation with similar parameters as the article. Blue represents source test(SynROD), whereas orange indicates target test(ROD).

In this simulation, all network weights have been trained as e2e at the same time. As it is clear in Fig. 5, with the beginning of the learning process, the value of the loss average function in both data sets source test and target test shows a decreasing trend and this trend continues approximately until the third epoch. In the following, the algorithm gradually overfits on source train and as a result, its performance on both used test datasets gradually deteriorates (In this simulation, dropout has been done only in the structure of main head and pretext, we may be able to prevent the overfit of data on source train by using dropout in the structure of ResNet networks).

B. Using Rotation Classification as Pretext Task

By comparing the results obtained from the implementation of the DA method introduced in the main article [14], it is clear that our algorithm has been able to effectively improve its performance on the target dataset compared to the implementation in the source only method, and this means proving the effectiveness of relative rotation is a pretext.

In the second implementation, weight decay is reduced compared to the first simulation, and weight ratio of the loss cost function is doubled. (According to the trend of changes in the simulation chart, the simulation has stopped at a lower number of epochs than the first situation.) Paying attention to the results chart shows that the changes made have led to a decrease in the accuracy of the model by approximately 3.5 per cent.

C. Using Rotation Regression as Pretext Task

In this part, instead of using the classification problem as a pretext task, we are going to try it by means of a regression example. Basically, by having RGB and depth RGB-D images, each of these has been rotated between 0 and 360 degrees; then, we predict their relative rotation angle as a continuous number between 0 and 360 degrees. The structure of our pretext task network at this stage is considered as [conv (1 × 1,100), conv (3 × 3, 100), fc (360),fc(2),fc(1)]. In the second-to-last layer, we consider a linear layer with two output neurons. We want each one of these two neurons able to determine one of the two positions x and y in the Cartesian coordinate system on the plane. Moving on, by adding a layer characterized by atan2 activation function, having the position of the point on the plane, we want to determine its output as a continuous angle between zero and 2pi. (The important point is that in Pytorch, the input and output angles of trigonometric functions are defined in radians.)

Another key aspect in the definition of pretext task is to provide a suitable function to calculate the loss. The main problem in using the MSE loss function on the predicted angles and the actual angle is that some angles such as 1 and 360 degrees, while being very close to each other, produce a significant error. In order to provide a proper Loss function, we used the method presented by Hara et al. [13]. According to it, we consider each rotation angle as a point in the coordinate system as

$$v = (cos(\theta), sin(\theta)) \tag{4}$$

In this way, the angles from 0 to 360 degrees actually corresponds to the points in the diameter of a circle with a unit radius. Now we can define the loss function *Lp* as (5) + (6), basing on the distance between the points as follows(where p stands for *pred*):

TABLE II
EXPERIMENTAL RESULTS OF THE SOURCE-ONLY METHOD

Batch-size	Dropout	optimization	Optimization parameters	Loss function	W_Decay	Accuracy Source	Accuracy Target
64	0.5	SGD	Learning_rate = 0.0003, momentum = 0.9	Cross entropy	0.05	70	39.68
64	0.5	SGD	Learning_rate = 0.0001, momentum = 0.9	Cross entropy	0.05	69.36	30.17
64	0.2	SGD	Learning_rate = 0.0003, momentum = 0.9	Cross entropy	0.05	69	33

$$\frac{1}{\tilde{N}_s}\sum_{i=1}^{\tilde{N}_s}(sin_s\theta - sin_p\theta)^2 + \frac{1}{\tilde{N}_s}\sum_{i=1}^{\tilde{N}_s}(cos_s\theta - cos_p\theta)^2 \quad (5)$$

$$\frac{1}{\tilde{N}_t}\sum_{i=1}^{\tilde{N}_t}(sin_t\theta - sin_p\theta)^2 + \frac{1}{\tilde{N}_t}\sum_{i=1}^{\tilde{N}_t}(cos_t\theta - cos_p\theta)^2 \quad (6)$$

Similar to the pretext task based on classification, the calculated loss values for the transformed source and target datasets are this time computed with a new formula and added to the loss function associated with the main task with a specific weighting factor.

In Figures 8 and 9, the results of the simulation of calculating the rotation angle are drawn continuously, and weights of the loss function has different values. By comparing these results with the ones of source only method, we realize that the DA task defined in this implementation has not been able to generalize the results from one domain to another for our algorithm, because the accuracy of the target set has deteriorated with respect to the source only method.

By examining the amount of loss corresponding to the pretext task, we realize that it has been significantly reduced in the simulation process, which means that the task network has been able to learn the defined pretext to some extent. However, in practice, these results did not have a positive effect on the performance of our main task, which can indicate a problem in the way this task is defined.

By comparing both sets of simulated parameters, we can see that with the increase of the weight(lambda p) of the loss corresponding to the pretext task, the performance of our network has decreased in terms of accuracy, which also confirms the existence of a possible problem in the implementation of the loss function related to the pretext.

VI. CONCLUSION

In this project we try to solve the problem of DA for RGB-D data, using two benchmark datasets, ROD and SynROD. In order to achieve our goal, we train a network on the self-supervised task of predicting relative rotation between RGB and depth images, alongside the main image classification task. This task has been implemented by two different methods. Once we try to solve it as a classification problem with limited number of possible rotation between RGB and Depth pictures. As a second approach pretext task is considered in the ways of a continuous 360 degrees relative rotation(regression). The result of these methods have been evaluated based on mentioned datasets. As results show, using relative rotation as a classification for pretext task could enhance the model performance on target. However, by using continuous relative rotation we could not achieve better performance in our implementation.

REFERENCES

[1] Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:1180-1189, 2015.

[2] Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wort- man. A theory of learning from different domains. JMLR, 79, 2010.

[3] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jen- nifer Wortman Vaughan. A theory of learning from different domains. Machine Learning, 79(1-2):151–175, 2010.

[4]].Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In COLT, 2009a.

[5]].Pascal Germain, Amaury Habrard, Francois Laviolette, and Emilie Morvant. A PAC- Bayesian approach for domain adaptation with specialization to linear classifiers. In ICML, pages 738–746, 2013.

[6] L. Spinello and K. O. Arras, "Leveraging rgb-d data: Adaptive fusion and domain adaptation for object detection," in ICRA. IEEE, 2012, pp. 4469–4474.

[7] J.Hoffman,S.Gupta,J.Leong,S.Guadarrama,andT.Darrell,"Cross- modal adaptation for rgb-d detection," in ICRA. IEEE, 2016, pp. 5032–5039.

[8] X. Li, M. Fang, J.-J. Zhang, and J. Wu, "Domain adaptation from rgb-d to rgb images," Signal Processing, vol. 131, pp. 27–35, 2017.

[9] W. Jing and Z. Kuangen, "Unsupervised domain adaptation learning algorithm for rgb-d staircase recognition," arXiv preprint arXiv:1903.01212, 2019.

[10] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stephane Deny Self-Supervised Learning via Redundancy Reduction, Proceedings of the 38th International Conference on Machine Learning, PMLR 139:12310-12320, 2021.

[11] J. Xu, L. Xiao, and A. M. Lpez, "Self-supervised domain adaptation for computer vision tasks," IEEE Access, vol. 7, pp. 156 694–156 706, 2019.

[12] J. Xu, L. Xiao, and A. M. Lpez, "Self-supervised domain adaptation for computer vision tasks," IEEE Access, vol. 7, pp. 156 694–156 706, 2019.

[13] K. Hara, R. Vemulapalli and R. Chellappa, "Designing Deep Convolutional Neural Networks for Continuous Object Orientation Estimation", Center for Automation Research, UMIACS, University of Maryland

[14] M.R. Loghmani, L. Robbiano, M. Planamente, K. Park, B. Caputo and M. Vincze, "Unsupervised Domain Adaptation through Inter-modal Rotation for RGB-D Object Recognition"

TABLE III
EXPERIMENTAL RESULTS OF THE CLASSIFICATION ROTATION IMPLEMENTATION

Batch-size	Dropout	Optimization	Optimization parameters	Loss function	W_Decay	W_Ratio	Accuracy Source	Accuracy Target
64	0.5	SGD	L_r = 0.0003, momentum = 0.9	Cross entropy	0.05	0.1	66.63	57.05

TABLE IV
EXPERIMENTAL RESULTS OF THE CLASSIFICATION ROTATION IMPLEMENTATION WITH DIFFERENT PARAMETERS

Batch-size	Dropout	Optimization	Optimization parameters	Loss function	W_Decay	W_Ratio	Accuracy Source	Accuracy Target
64	0.5	SGD	L_r = 0.0003, momentum = 0.9	Cross entropy	0.04	2	64.49	53.5

TABLE V
EXPERIMENTAL RESULTS OF THE CONTINUOUS ROTATION IMPLEMENTATION

Batch-size	Dropout	Optimization	Optimization parameters	Loss function	W_Decay	W_Ratio	Accuracy Source	Accuracy Target
64	0.5	SGD	L_r = 0.0003, momentum = 0.9	Cross entropy	0.05	0.6	63.34	37.5

TABLE VI
EXPERIMENTAL RESULTS OF THE CONTINUOUS ROTATION SECOND IMPLEMENTATION

Batch-size	Dropout	Optimization	Optimization parameters	Loss function	W_Decay	W_Ratio	Accuracy Source	Accuracy Target
64	0.5	SGD	L_r = 0.0003, momentum = 0.9	Cross entropy	0.05	1	61.44	33.7

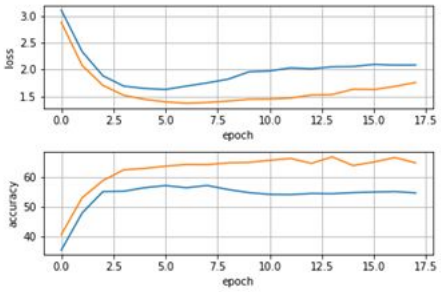

Fig. 6. Accuracy and average loss versus epoch for implementing rotation as classification for pretext. Blue represent source test(synROD) and orange is target test(ROD).

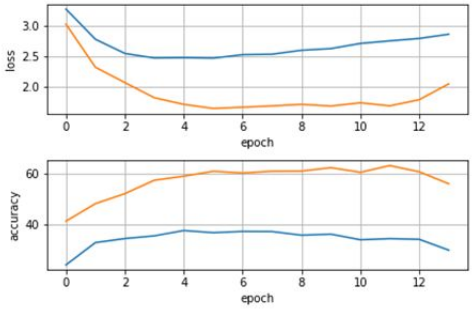

Fig. 8. Accuracy and average loss versus epoch for continuous rotation implementation. Blue represent source test (synROD) and orange is target test (ROD).

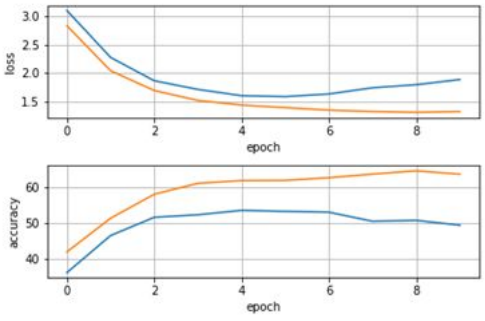

Fig. 7. Accuracy and average loss versus epoch for implementing rotation as classification for pretext with different parameters. Blue represent source test(synROD) and orange is target test(ROD).

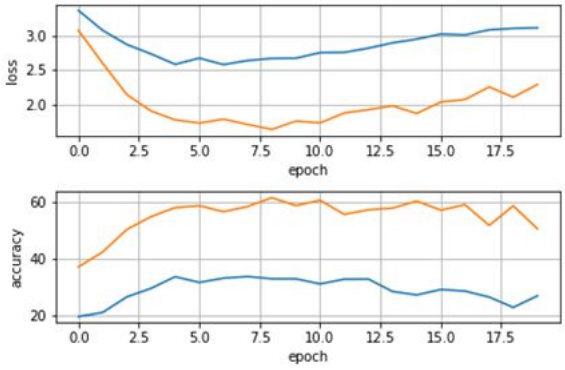

Fig. 9. Accuracy and average loss versus epoch for continuous rotation second implementation. Blue represent source test (synROD) and orange is target test(ROD).