Unsupervised Domain Adaptation through Inter-modal Rotation and Jigsaw puzzle assembly for RGB-D Object Recognition
Machine Learning and Deep Learning project report

Hadi Nejabat
Politecnico di Torino
hadi.nejabat@studenti.polito.it

Abstract

Domain adaptation (DA) is a sub-discipline of machine learning which deals with scenarios in which a model trained on a source distribution is used in the context of a different target distribution. In general, domain adaptation uses labeled data in one or more source domains to solve new tasks in a target domain existing DA methods are not designed to cope with the multi-modal nature of RGB-D data, which are widely used in robotic vision. We propose a novel RGB-D DA method that reduces the synthetic-to-real domain shift by exploiting the inter-modal relation between the RGB and depth image. Our method consists of training a convolutional neural network to solve, in addition to the main recognition task, the pretext task of predicting the relative rotation between the RGB and depth image. Also proposing a variation to the pretext task by changing the self-supervision method from rotation to Jigsaw puzzle and comparing their results; Which resulted in 2% improvement in accuracy scores

1. Introduction

In the field of object recognition it is important for the system to be able to identify and understand the object and its surroundings. Yet, the loss of depth information caused by projecting the 3-dimensional world into a 2-dimensional image plane can negatively affect the recognition performance. This is where the Depth-Images comes to aid; By re-introduce the information about the camera scene distance as a depth image. While the RGB image contains texture and appearance information, the depth image contains additional geometric information and is more robust to lighting and color variations. However, this brings its own drawback, where the large amount of annotated data required to train CNNs can be very hard and costly to collect. This leads us to generate a large synthetic training set by rendering 3D object models with computer graphics software like *Blender*. However, the difference between the synthetic (source) training data and the real (target) test data severely undermines the recognition performance of the network. we can tackle this by Unsupervised Domain Adaptation (DA) which accounts for the difference between source and target data by considering them as drawn from two different distributions and derives the network to learn the object's shapes regardless of their backgrounds. Nevertheless, available DA methods implicitly assume that the data come from a single modality. We believe that this as-

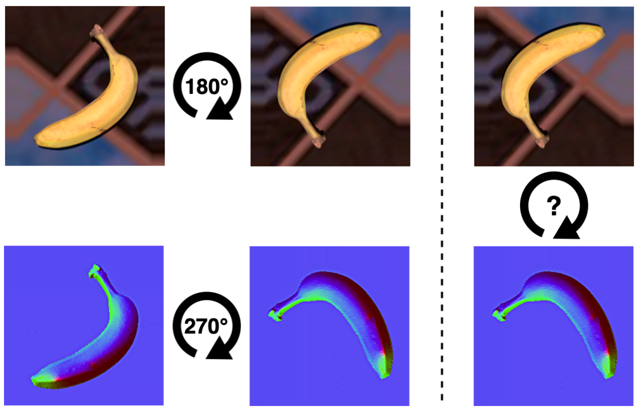

Figure 1. The self-supervised pretext task of rotation and prediction of relative rotation after both RGB and depth rotated independently

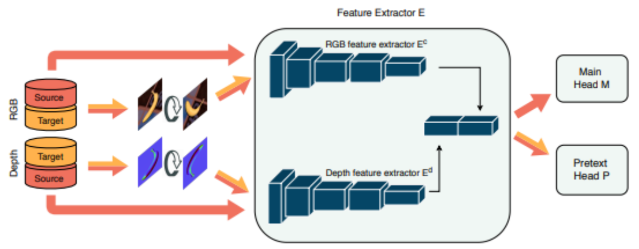

Figure 2. The original model to be re-implemented.

sumption leads to sub-optimal results when dealing with multi-modal data since the natural inter-modal relations of the data are ignored. In this project we are going to re-implement the DA method proposed by the given paper [5] consisting of training a CNN to solve a supervised main task and a self-supervised pretext (or auxiliary) task from pairs of RGB and depth images. And apply our variation to the pretext task of self-supervision and compare results. The pretext task is a self-supervised nature job which consists of rotating the RGB and depth image of a batch and ask the network to predict the relative rotation that re-aligns them (Figure 1). Consequently, both source and target data can be used to train the model on the pretext task, while the supervision of the source data is used to train the model on the main task (Figure 2). As the reference paper [5] proposes, To evaluate our method on object categorization no existing dataset presents both synthetic and real data. We use the popular RGB-D Object Dataset (ROD) [4] for the real data and collect the synthetic counterpart ourselves. Therefore, we propose synROD: a dataset generated by collecting and rendering 3D object models from the same categories as ROD using publicly available Web resources. Extensive experiments on these datasets show that our newly defined pretext task effectively reduces the synthetic-to-real domain gap and outperforms existing DA approaches that do not leverage the inter-modal relations of RGB-D data. Our project demand was simplified to using only the Real part hence working with synROD to ROD.

2. Related Work

2.1. Unsupervised Domain Adaptation

The main strategy of DA methods is to reduce the domain shift. And there are different groups of methods such as discrepancy-based methods, adversarial learning methods and leverage self-supervised learning methods. The mentioned approaches are based on the assumption that the data come from a single modality. The reference paper provided empirical evidence that these solutions are sub-optimal and better results can be achieved by leveraging the relation between the RGB and depth modalities. and we also exploit the chance of improving the model by changing the self-supervision pretext task from rotation to Jigsaw puzzle.

2.2. Self-Supervised Tasks

Self-supervised learning is used to compensate for the lack of annotated data by training the network on a pretext task for which the supervision (or ground truth) can be defined from the data themselves. One of the most effective self-supervised tasks consists of rotating the input images by multiples of 90 and training the network to predict the rotation of each image [3]. This pretext task has been successfully used in a variety of applications such as network pre-training and domain adaptation. Inspired by this success, we revisit the rotation prediction task for RGB-D data and propose a task that consists of predicting the relative rotation between the color and depth image.

3. Dataset

One of the novel actions of the paper [5] is to reach a curtain dataset which can be used for future research in RGB-D area. We will give a brief description of the dataset and principles on which it was created. the main goal is to exploit the synthetic-to-real domain shift, same objects classes have been collected in different conditions containing both the shape and the texture of the objects (meaning depth and color respectively). knowing this prevents us from using available datasets such as ShapeNet where they do not have a corresponding real dataset that shares the same classes. To solve this problem, we make our own dataset by collecting a new synthetic dataset called synROD. the objects are carefully selected from the 51 categories of ROD dataset, most popular RGB-D object categorization repository [2]. ”Next challenge is to create the 2.5D scenes to account for 3D nature of objects. This is done by using a ray-tracing engine in *Blender* to simulate photo-realistic lighting” [5]. multiple techniques were performed to obtain usual and realistic object posture.

3.1. Dataset import and preprocessing

Concerning the datasets, they are already available in two zip files that are stored in a Google Drive repository and were downloaded. ROD is used as unlabelled *target dataset* and contains RGB-D real images from 51 different classes all represented in both modalities (RGB and Depth). synROD is the synthetic counterpart, with the same number of classes, and its images are used as labelled *source dataset*.
to be able to access data, we must map the correct path to each image. this is done through reading and parsing the provided split file for synROD and a written module to generate the same file for ROD dataset sub-folders. One important action is to remove the four classes that are suggested to be discarded as they are underrepresented. Moreover, a typo in the ”pepper” dataset folders must be corrected to correctly map this class. As a result, we can observe that some images which are not present in both modalities are discarded, since they are decoupled and cannot be used for multi-modal purposes.

3.2. Data Agumentation

to compensate for having limited data,we employ data augmentation techniques (only on the source) used to automatically generate marginally modified images from the original ones at each epoch, but keeping the same label in order to increase the variety of the training dataset and avoid over-fitting. In details, we perform two random transformations (random crop and random horizontal flip) paying attention to apply the same one to the RGB and Depth images. This is done through a re-definition of the used transformations in order to decide whether to apply the same transformation or not on both the elements of the RGB-D pairs. The dataset objects defined in this way are ready to be fed to the Pytorch's dataloaders.

4. Method

In this section we are going to talk about the approach used to both re-implement the original model and build our variation for the multi-modal self-supervised DA. More specifically, *section 4.1* presents the Network architecture, while *section 4.2* shows the details of the implemented self-supervised tasks and *section 4.3* describes the loss functions and optimization techniques.

4.1. Network Architecture

Our goal is to train a neural network to predict the object class of the target data, using only labeled source data and unlabelled target data. As it can be seen from figure 2, the model is made of three principal sections which are the *Feature Extractor (E)*, the *Main Task (M)* and the *Pretext Task (P)*.

Feature Extractor (E): defined to extract the features from RGB and Depth images through two identical branches E^c and E^d; those features are then concatenated along the channel dimension and used as input to the pretext task P and further to main task M. as suggested we will implement our whole project with a pre-trained modified version the *ResNet18* architecture[**?**], without the final fully connected and global average pooling layers. The pre-training of network was done on *ImageNet* dataset[1], therefore the model has prior knowledge of features in use that will eliminate the problem of starting from scratch.

Main head (M): defined to performs the supervised recognition and classification task, it only uses the source dataset as labelled data during training and performs class prediction on the unlabelled dataset at test time. Its structure is as follow: first we have the average pooling layer with kernel size *(1,1)*, then a fully connected layer *fc(1000)* which uses batch normalization and the ReLU as activation function. We finally use the dropout layer with probability 0.5, followed by a *fc(47)* layer as classifier.

Pretext Task (P): defined to solve the 4-way classification problem of predicting the rotation between the RGB and depth image by performs a self-supervised task. It is composed of a convolutional layer with 100 *1x1* filters, followed by another convolutional with 100 *3x3* filters, a *fc(100)*, and the final *fc(n)*, where n is the number of output classes and depends on the chosen task for any given experiment. A value *n=4* is used for relative rotation. All the convolutional and fully connected layers use batch normalization and ReLU as activation function except for the last one, which uses softmax and is subject to dropout.

4.2. Self-supervised tasks

The self-supervised task is applied to reduce the domain shift between the two data modalities. we define a sample of the source images (x_s^c, x_s^d), where x_s^c is the RGB image and x_s^d is the Depth image.

1.Original(Rotation): Performing a *relative rotation* between the two modalities. We apply a clockwise rotation of a multiple of 90° on each of the single images: ($\hat{x} = rot90(90 \cdot k, x)$ where k in {0, 1, 2, 3}, x is the original image and $\hat{x}$ is the rotated version. Therefore, by knowing the independent values of k for the two modalities, namely k^c and k^d, we can construct the label for the pretext task with a simple computation: $y_s^{rot} = (k^c - k^d) mod(4)$, where *mod* is the integer modulo operator.

2.New variation(Jigsaw Puzzle): The second self-supervised task is the *Jigsaw Puzzle*. The idea came from the *Nowroozi et al.*[6] work that used Jigsaw Puzzle for Unsupervised Learning. We will use a similar approach in self-supervised Pretext task for DA. The main scheme is to Partition an image into multiple tiles and then shuffle these tiles on a 2x2 grid. The model is then tasked with un-shuffling the tiles back to the original configuration and predict which one of all possible permutations was applied to the input. This is done by creating batches of tiles such that each tile of an image is evaluated independently. The convolution output are then concatenated and then the permutation id which is known as class label in *SSClassifier* of our network is predicted. In order to investigate and better explore this solution we have adjusted the Data-Generator module's variables to provide the opportunity to apply a second configuration of our variation for better comparison. In this case the same procedure is applied but by creating a 3x3 puzzle grid and selection an acceptable smaller subset of permutations. Yet, due to extensive training time we weren't able to add it to our experiments. More details on this task and the algorithm used to create and shuffle the tiles will be explained in section 5.3.

4.3. Loss function and Optimizer

Theoretical concept analysis: Since our model consists of two heads working in parallel together, The loss functions used for our experiments is calculated by first defining separate objective functions with *Cross Entropy* for where using labels for training is allowed, such as training the main task with source data, or training the pretext task with transformed data of domains. Moreover, since target data contains important information for the main task but cannot be used as labeled for this part, *Entropy* has been used as a loss component to measure how clearly the classifier can make a prediction on it. This is the section where we will derive the system to remove the domain shift between our modalities.

Code implementation:
denoting($S = \{((x_i^{sc}, x_i^{sd}), y_i^s)\}_{i=1}^{Ns}$), as The labeled source and ($T = \{((x_i^{tc}, x_i^{td}), y_i^t)\}_{i=1}^{Nt}$) as the unlabeled target data.from S and T we can generate the transformed set of source and target data that are used in to define the self-supervision task. Following this we will train our CNN to minimize the objective function

$$Loss = L_m + \lambda_p * (\hat{L}_{ps} + \hat{L}_{pt}) \tag{1}$$

where L_m and L_p are the cross-entropy loss of the main and pretext task, respectively. and λ_p is a weight to regulate the contribution of the corresponding pretext loss term. more precisely:

$\mathrm{L}_m = -1/N_s \sum_{i=1}^{Ns} y_i^s . \log y_i^s$

$\mathrm{L}_p = -1/\tilde{N}_s \sum_{i=1}^{\tilde{N}_s} z_i^s . \log \tilde{z}_i^s$- $1/_t \sum_{i=1}^{\tilde{N}_t} z_i^t . \log \tilde{z}_i^t$

where $\tilde{y}^s = M(E(x^{sc}, x^{sd}))$ and $\tilde{z}^* = P(E(\tilde{x}^{*c}, \tilde{x}^{*d}))$. At test time, the pretext head P is discarded and the predictions of the target data are computed as $\tilde{y}^t = M(E(x^{tc}, x^{td}))$. We also include entropy-minimization with weight 0.1 as a DA-specific regularization. The pseudo-code is presented in Algorithm 1.

Algorithm 1 RGB-D Domain Adaptation

```
Input:
    Labeled source data set $S = \{((x_i^{sc}, x_i^{sd}), y_i^s)\}_{i=1}^{Ns}$
    Unlabeled target data set $T = \{((x_i^{tc}, x_i^{td})\}_{i=1}^{Nt}$
Output:
    Object class prediction for the target data $\{\tilde{y}_i^t\}_{i=1}^{Nt}$
1:  procedure TRAINING(S,T)
2:      Get transformed set $\tilde{S} = \{((\tilde{x}_i^{sc}, \tilde{x}_i^{sd}), z_i^s)\}_{i=1}^{\tilde{N}s}$
3:      Get transformed set $\tilde{T} = \{((\tilde{x}_i^{tc}, \tilde{x}_i^{td}), z_i^s)\}_{i=1}^{\tilde{N}t}$
4:      for each epoch do
5:          Load a batch from S
6:          Compute loss $L_m$
7:          Load a batch from $\tilde{S}$ and $\tilde{T}$
8:          Compute loss $L_p$
9:          Update loss $L_p = *0.1 \leftarrow$ Regularization DA
10:         Update M weights with $\nabla L_m$
11:         Update P weights with $\nabla L_p$
12:         Update E weights with $\nabla L_m$ and $\nabla \hat{L}_t$
13: procedure TEST(T)
14:     for each $x_i^{tc}, x_i^{td}$ in T do
15:         Compute $\tilde{y}_i^t = M(E(x_i^{tc}, x_i^{td}))$
```

M = Main task, P = Pretext task, E = Feature Extractor

All the hyper-parameters shown in *table 1* and The expressions of the losses are described in detail in section 5.

Each presented model is trained using a *SGD optimizer*, by considering a L2 regularization of 0.05 which is passed as weight decay=0.05 and also added a dropout=0.5 to ensure minimization of chance of overfitting. Moreover, The training process may be forced to an early stopping, in cases when the main loss on target does not show improvements after it reaches the patience parameter equal to several epochs. This has the added benefits of reducing execution time and the chance of overfitting on target data.

5. Experiments

we are going to present the performed experiments and their results. all numerical details of our hyper-parameters that used to carry the experiments are shown in *table 1*; and description of each of our experiments are as follows.

5.1. Source only baseline

This experiment is useful to establish a baseline score that allows to understand if the domain adaptation (later applied) will improve the model by reducing the domain shift. For this experiment the auxiliary task is therefore not included and our model will consist of the *Feature Extractor (E)* and the *Main Task (M)*. The source data ($S = \{((x_i^{sc}, x_i^{sd}), y_i^s)\}_{i=1}^{Ns}$), limited to the training split, are forwarded through the model ($\tilde{y}_i^s = M(E(x_i^{sc}, x_i^{sd}))$) in order to train the network. Since it is suggested that we assume that validation set is the same as test set we can go ahead at the end of each epoch and perform the validation on the entire target set and analyze the validation losses to check the need of early stopping. In order to understand how the model behaves during the training, a plot describing trends of both training loss and the validation accuracy is shown in figure 3. As it can be observed there is a rapid decrease of the training loss while the validation accuracy increases and reaches the peak (47.8%) at epoch 4. Going on with the training, the training loss is still decreasing but the validation accuracy starts decreasing too (the model starts to overfit) and the training is stopped 10 epochs later. while this might be an acceptable starting point, we will perform a hyper-parameter tuning with experimental parameters shown in *table 1*. it is evident that the tuned hyper-parameters are quite standard and it is worth mentioning that since our hardware memory bottlenecks batch size,we were able to apply Batch size up 128. Yet, for further sections where the model also includes the pretext task, we are able to assign only Batch size=32. Moreover, by monitoring figure 4 we can notice a different behavior of the models with lower learning rate and weight decay where a drop shows model leading to an overfit with those hyper-parameter sets.

Experiment	Param. Set	Batch size	LR	Weight loss Pretext task	Weight loss Entropy	weight_decay	dropout	Max. Validation Accuracy (%)	Diagnostics
Source only Baseline	S1	64	0.01	N/A	N/A	N/A	N/A	**48.89 @Epoch 4**	Overfitting
	S4	64	0.001	N/A	N/A	N/A	N/A	46.77 @Epoch 4	Overfitting
	S2	32	0.0001	N/A	N/A	0.005	N/A	41.2 @Epoch 18	Distrubtion due to unrepresentative classes
	S5	64	0.00001	N/A	N/A	0.05	N/A	44.36 @Epoch 19	Reuqire smoother learning
	S3	32	0.0001	N/A	N/A	0.05	0.5	47.78 @Epoch 20	Acceptable for Baseline
DA - Rotation	S7	32	0.0001	1	0.1	0.05	0.5	**61.12 @Epoch 6**	Original papers inputs
	S10	32	0.00001	1	0.1	0.05	0.5	60.06 @Epoch 9	Lower LR
	S6	32	0.0001	0.8	0.1	0.05	0.5	59.61 @Epoch 7	Lower wieght to tune objective func.
	S9	32	0.0001	1	0.1	0	0.5	59.99 @Epoch 6	No reguraizer for main task
	S8	32	0.0001	1	0.2	0.05	0.5	58.63 @Epoch 5	Not acceptable generalization
DA - Jigsaw Puzzle	S11	32	0.0001	1	0.1	0.05	0.5	**62.92 @Epoch 3**	Inputs from previous best result
	S12	32	0.0001	1	0.1	0	0.5	60.52 @Epoch 4	No reguraizer for main task
	S13	32	0.0001	0.8	0.1	0.05	0.5	61.78 @Epoch 3	Lower wieght to tune objective func.
	S14	32	0.0001	0.9	0.2	0.05	0.5	62.8 @Epoch 3	Lower wieght to tune objective func.
	S15	32	0.0001	1.5	0.1	0.05	0.5	60.4 @Epoch 3	higher wieght to tune objective func.

Table 1. Representation of all input hyper-parameters for each experiment.

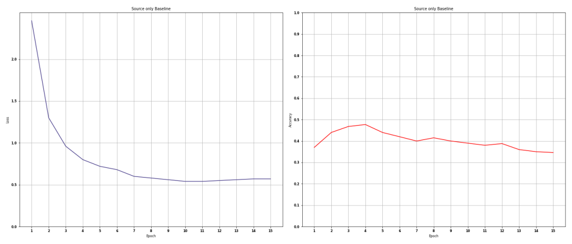

Figure 3. First achieved results of source only base model.

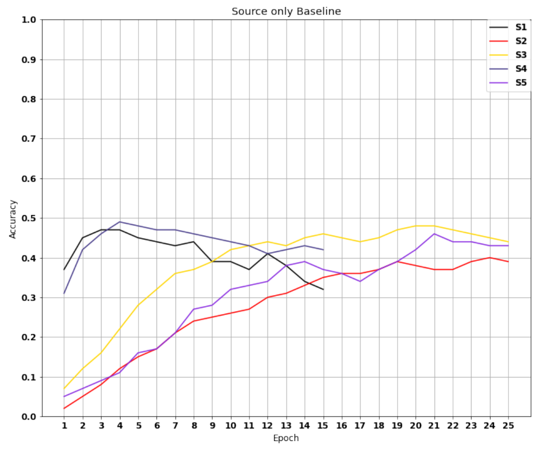

Figure 4. Source only hyper-parameters tuning scores: validation accuracy.

5.2. Domain adaptation - relative rotation

This section is the re-implementation of the original paper's [5] Experiments, so not only the *Feature Extractor (E)* and the *Main Task (M)* are assigned, but also the *Pretext Task (P)*. The self-supervised task used in this section is the relative rotation; where at each training epoch an iteration is performed on four data loaders in parallel: source, target, source transformed, target transformed. During the training step, the data are forwarded through the network according to the pseudo code described in algorithm 1, the single losses are computed. Then the gradient is computed and the weights, and all the network parameters, are updated during the back-propagation. Note that, during the training, a validation is made, similarly to the one in sub-section 5.1.As represented in *table 1* a hyper-parameter tuning phase is performed after each experiment with consideration to the behaviour of our model in previous experiment; In this part we have a wider range of parameters to tune considering the available weights for regularization of entropy loss and degree of influence of rotation loss. The main idea is to study the model's behaviour by assigning a different relevance to the auxiliary task; that is an open problem of this field. The results are shown in the table, where it is visible that the hyper-parameter set presented in the paper proved to be the most suitable. The plot also shows that models can oscillate after reaching convergence, and therefore this encourages the usage of early stopping techniques.

5.3. Domain adaptation - Jigsaw Puzzle

The model's structure is similar to the previous model in Relative Rotation presented in subsection 5.2, while its self-supervision task is changed to new variation of jigsaw puzzle. In both configuration of this variation, shuffling permutation is applied on both modalities.
In the first configuration of the variation, the pretext task is a 2x2 puzzle grid and in the second configuration, the pretext task is a 3x3 puzzle grid. Both follow the same procedure to generate the puzzle grids and then divide them to separated tiles and shuffle them with respect to an anchor point to the relative distance from the anchor tile be the measure of reconstruction.

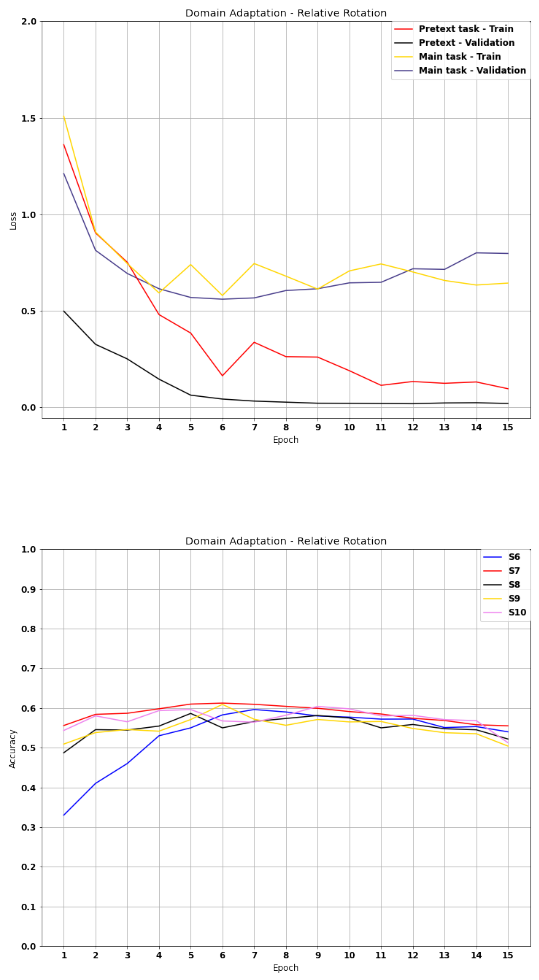

Figure 5. Scores of the DA:Relative rotation model. First graph's curves represent the trend of the single losses calculated in the Main and the Pretext tasks. Second graph shows hyper-parameters tuning scores: validation accuracy applied on same model.

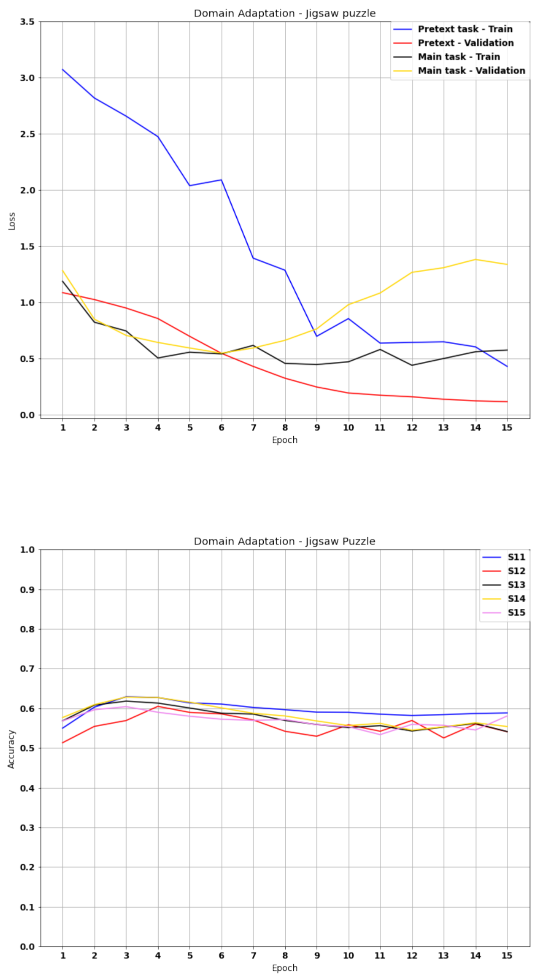

Figure 6. Scores of the DA:Jigsaw Puzzle model. First graph's curves represent the trend of the single losses calculated in the Main and the Pretext tasks. Second graph shows hyper-parameters tuning scores: validation accuracy applied on same model.

As for defining labels each grid's shuffle composition will be selected from a pool of 4!=24 available permutations in the 2x2 grid and from a smaller subset of 9! permutations in the 3x3 grid. Although this task will show much better behaviour if we take into account all the possible permutations but since training time will be considerably higher it is suggested to took a subset of 100 i.e. From 9!, use 100. following this,we pass the data for train we select a random composition which consists of 4 input patches according to permutation and those will define the distance the tile must sit in x and y direction. ask network to return the probability value of each index.

at last the maximum are selected and the combination of the indexes create the puzzle composition and correspond to the label defined from the start of grid creation. Next steps are identical to the relative rotation case. It is important to analyse another aspect of this experiment, what we decide to select as our class label for analyzing the pretext task. for this experiment we took the same approach of the original paper where they assigned 4 classed to multiples of 90; whilst here we label each assembly of puzzle permutations to 24 classes, while this lead us to better performance behaviour than relative rotation, it can be observed that our model still suffer from same overfitting in problem that we experienced in the rotation experiment and this can be solved by two separate solutions for each task. For relative rotation, we can replace Rotation as Regression where the objective function will be to minimize the exact number of rotations instead of classes. Also, same idea can be implemented in Jigsaw puzzle by replacing the total x and y distance each tile moved from original position as our measure and try to minimize that.

5.4. Results

we have reported our final achieved quantitative results of each section of the experiments in table 2 to have an overview and be able to make a comparison between our scores versus the ones obtained in the original reference paper.

The source only experiment has proven to work, having a similar accuracy for various hyper-parameter sets, which also includes the experiment in the reference paper.
The domain adaptation with Relative Rotation was able to achieve a comparable score with respect to the baseline paper. However, given the circumstances such as having available a subset of whole dataset which used in original paper and not having clear insight of the network's architecture used by *Loghmani et al*, reproducing the original model is near to impossible.
Moreover, the proposed method was able to promote the baseline by 14% in accuracy by integrating the pretext task which shows improvement in the reduction of domain gap by aligning the distributions of features between the two domains. With the purposed variation, domain adaptation with Jigsaw Puzzle, the accuracy is further improved by nearly 2% which is a considerable. Furthermore, it appears that with higher values of λ_p for the pretext task leads to lower accuracy even if the loss trend of the pretext task would suggest the opposite.

In order to further investigate, it can be seen that our variation was able to counter overfitting better than the rotation case but it still starts to overfit from epoch 6. Several techniques have been utilized to prevent overfitting in both cases which includes applying L2 regularization and data augmentation. However, we realized to improve this further, we need more fundamental approaches like deeper network (e.g Resnet50) or changing model's structure (e.g. VGG, DenseNets).

Experiment	Our results	Ref. results*
Source only Baseline	47.78%	47.33%
DA - Relative Rotation	61.12%	66.68%
DA - Jigsaw Puzzle	62.92%	N/A

Table 2. Results comparison.
(*) Results achieved in the reference paper[5]

This must be taken under consideration that due to extensive training times, and lack of access to high performance hardware it has not been possible to do repeated runs for various hyper-parameters in order to find the optimal set and providing confidence intervals.

6. Conclusions

This project consists of 3 core sections, proposing two approaches to tackle the problem of RGB-D DA. we first employ the new synthetic collected dataset from the original paper *synROD* to be explored with different DA techniques. Then we have shown experimentally that both our self-supervised tasks successfully reduce the domain shift and outperforms all considered baselines. Moreover, our purposed variation was able to improve the Relative Rotation by providing better validation accuracies, reaching the score of 63% in accuracy which is comparable to the obtained score for relative rotation in the reference paper. Although we were able to make improvements to the model with our variation and verify that relative tasks are powerful DA tools; Further research would be needed to unlock the full potential of self-supervision with jigsaw puzzle, by experimenting on a 3x3(or possibly 4x4) puzzle grid as smaller patches can extract more structural details from the image. Another possible improvement can be experimenting with deeper network (e.g Resnet50).

In conclusion, all the self-supervised tasks actually improved the model performances with respect to the source-only baseline, so it can be said that learning visual representations by performing self-supervised learning can be the key to many different Computer Vision problems.

References

[1] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database, 2009.

[2] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[3] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[4] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057. IEEE, 2014.

[5] 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, 2020.

[6] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.