UPPSALA UNIVERSITET

Serienummer

Examensarbete 30 hp

Månad och 2022

Unsupervised Image Classification Using Domain Adaptation

Joel Finn Bjervig

Våra program civil- och högskoleingenjörsprogram (Klicka och välj program)

UPPSALA
UNIVERSITET

Engelsk titel

Ange författarens förnamn och efternamn

Abstract

Klicka och ange text.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/Visby

Handledare: Förnamn Efternamn Ämnesgranskare: Förnamn Efternamn

Examinator: Förnamn Efternamn

Detta examensarbete är sekretessbelagt t o m Klicka och välj datum eller skriv i fältet. (Tas bort om ej sekretessbelagt)

Abstract

Framgången inom maskininlärning och djupinlärning beror till stor del på stora annoterade dataset, som MNIST, ImageNet, Caltech-256, och Cifar-10. Att skapa stora dataset genom att tilldela etiketter till data är väldigt resurskrävade och kan till viss del undvikas genom att utnyttja datans statistiska egenskaper. En domän består utav ett funktionsrum och en marginell fördelning utav datasettet. Genom att överföra information från en *källdomän* till en liknande *måldomän*, är det möjligt för en model, tränad i källdomänen att generalilsera sig över till måldomänen. Samlingsnamnet för sådana metoder heter domän anpassning. Detta examensarbete använder kovariansen som mått för att anpassa en bildklassificerings model tränad på ett kamera dataset, till att prestera på ett liknande dataset med bilder från LiDAR sensorer. Två domän anpassning metoder har jämförts med varandra, samt med en model tränad på källdatan (kameradatan) genom övervakad inlärning utan domän anpassning. Alla metoder opererar på något vis med ett djup faltningsnätverk (CNN) och behandlar ett binärt klassificeringsproblem: är det en fotgängare eller ett fordon på bilden? För binära klassificeringsmodeller, är slumpmässig klassificering 50% noggrannhet. Den första metoden är en så kallad ytlig metod, där själva anpassningsmetoden inte ingår inuti den djupa arkitekturen av modellen, men efteråt. Den andra metoden är inkoorporerad i den djupa arkutekturen, och den tredje sista modellen är endast själva faltningsnätverket utan en metod för domänanpassning. Utan en domän anpassningsmetod minskar klassificeringsnoggrannheten i måldomänen till 63.80% medan den ökar i källdomänen. Den ytliga metoden upprätthåller klassificerings noggrannheten i måldomänen till 74.67% och den djupa metoden presterar bäst med 80.73% noggrannhet vid slutet av träningen. Förvirringsmatrisen visar även på hur modellen utan domän anpassning klassificerar forgängare felaktigt som fordon mycket mer ofta än de andra modellerna.

Acknowledgements

First, I must mention my thesis partner Mikael Westlund at KTH, and his excellent collaborative spirit. We parted ways halfway through the project to focus on different methods but kept in touch. I would also like to thank my thesis supervisor Arash Owrang Ph.D., with whom I had many insightful discussions. I would like to thank the entire perception team of the autonomous driving department at Scania for supporting my master thesis project. I am also very grateful of my subject reader Dominik Baumann, Ph.D. from Uppsala University, who was always keen to discuss with me and answer my questions. And finally, I would like to thank my friend and colleague Nils Guamelius, who worked on another Master's thesis project in the same department at Scania. We spent a lot of time together, working on our separate theses, and supporting each other.

Contents

1 Introduction

Large datasets such as X, Y, and Z have been essential in the success of the machine learning field as a whole but has been very resource costly to produce

There are vast amounts of data being collected and stored every day, and for supervised learning tasks, annotated data is needed. Data annotation is often outsourced to companies that specialize in such tasks, and while it produces valuable datasets, it is resource costly and a relatively inefficient process. To circumvent the manual labor of annotation, one may adopt approaches that instead utilize the intrinsic structure of the data, and thus alleviate the dependency on labels. Such methods go under the category of unsupervised machine learning, which experiences only data features from which it learns structural properties [1, Chapter 5]. In recent years, the prospect of transfer learning has emerged, which seeks to leverage already labeled data or utilize what has been learned in one setting of distribution P_1, to improve the performance on another similar dataset of distribution P_2. Measuring the "similarity" of datasets is a central area of research as it tells whether there exist any discrepancies between the two datasets. For example, if two experiments are supposedly executed by the same protocol, but the collected data from each experiment show a significant discrepancy, it signifies a difference in measuring equipment, techniques, or the environment in which the experiment was performed. If this is the case, they should not be jointly analyzed in a study (before being transformed in any way such that they share the same distribution). When the source and target domains are represented by the same feature space, but not by the same marginal distribution, it is called homogeneous transfer learning. The difference in marginal distributions is then the cause for what is called a domain shift. Further, if the type of prediction or inference being made (i.e. the task) is the same, we enter the realm of domain adaptation. In this thesis, the second order statistics, or the covariance, is the central measure that the domain adaptation (DA) methods presented in this report use. Pretraining a convolutional neural network on the Cifar-10 dataset is used to give an improved weight initialization, on which the DA methods will be performed. Both a so-called shallow method and a deep method are compared to a supervised learning method and put against each other with respect to their classification accuracy in the target domain.

2 Problem Description

Consider the task of training a model to discriminate and classify images from a *target* dataset $\mathcal{D}_t$ that has no accompanying labels. Without labels, a supervised learning approach is not possible, so we seek to leverage the statistical properties of the dataset, called unsupervised learning. It is also possible to utilize other datasets that have labels, to train a model which will act as a foundation on which the unsupervised learning can take place. Supervised learning tasks a similar *source* dataset $\mathcal{D}_s$ can help unsupervised learning tasks in a variety of ways. Pretraining a neural network with a large dataset $\mathcal{D}$ can generalize the model and improve the initial guess for the optimizer to reach an optimal solution faster. Another way to help the learning of a classifier on $\mathcal{D}_\sqcup$ without a corresponding label set is to train a model on a source dataset $\mathcal{D}_s$ with labels, (and preferably with the same number of classes), and simultaneously incorporate statistical knowledge from $\mathcal{D}_t$, such that the model will adapt to this *target* dataset. The source data consists of annotated camera images, and the target data contains LiDAR intensity images with no labels. Despite the two domains portraying similar scenes (streets, highways, traffic events) and depicting a lot of the same objects (pedestrians, bicyclists, cars, heavy vehicles), there is still a non-trivial discrepancy, or *domain shift* between the two datasets. The task is to minimize this domain shift such that we can use the supervised learning task in the source domain, alongside statistical information from the target domain to yield a classifier that performs well on the LiDAR dataset. The problem lies in balancing the discriminative task from the supervised learning on the camera dataset, and the domain adaptation task from the unsupervised learning task on the target dataset.

3 Related Work

The field of domain adaptation (DA) has a diverse set of methods, both shallow and deep, with the aim to bridge the discrepancy between datasets. The most established DA methods are presented in [2] from 2017. There exists several benchmark datasets with can be combined to form multiple domain shift scenarios.

3.1 Shallow Domain Adaptation

CORAL and Subspace Alignment (SA) ([3]) are both feature space alignment methods, where CORAL aligns the feature spaces directly via the second order moment of the data, and SA aims to align domain-specific subspaces. More specifically the source data is projected onto the source subspace and the target data onto the target subspace. These subspaces are spanned by a set of d eigenvectors corresponding to the d largest eigenvalues obtained from Principal Component Analysis (PCA) of the source and target data, respectively. The linear transformation $\boldsymbol{A}$ aligns the basis vectors of the subspaces such that the Bregman divergence (see definition 4 in Section A) between the two subspaces is minimized.

3.1.1 Unsupervised Feature Transformation Methods

An alternative to feature alignment methods, which aims to transform the source domain to the target domain, are feature transformation methods. There a transformation is applied on both the source and target domain to minimize the discrepancy between the two distributions. When such a method does not use labels in learning the transformation, it is regarded as an unsupervised feature transformation method. Many methods rely on minimizing some type of divergence or distance measure between sample means and variances, but [4] compares the data directly in the reproducing kernel Hilbert space (RKHS). In the CORAL paper [5], object recognition accuracies for the twelve domain shifts in the Office-Caltech 10 dataset are presented. The top three methods were CORAL, TCA, and SA. CORAL had the top accuracy score in 6 out of 12 domain shift scenarios with an average accuracy score of 46.7%, while SA performed the best in two scenarios (average accuracy 45.9%), and TCA in four scenarios (average accuracy 45.7%). For domain-specific accuracy scores and test results on other datasets, see the CORAL paper [5].

3.2 Deep Domain Adaptation

Models with deep architectures in general perform very well in discriminating complex data, be it binary or multi-class. The domain adaptation survey [2] highlights the ability of some deep models, with no domain adaptation, to outperform some shallow domain adaptation methods on most benchmarks. The ability of deep models to learn robust general features is very valuable and proves the power of deep architectures. Deep domain adaptation seeks to further boost the generalizability in either a semi-supervised or unsupervised manner. Semi-supervised domain adaptation can, for example, mix the source data with a few labeled target instances. When the target data has no labels, its statistical properties are used to minimize the domain shift of the deep features in the deep model.

3.2.1 Discrepancy-Based Methods

Inspired by shallow feature transformations, discrepancy-based methods operate in a deep learning framework and use a statistical distance measure between corresponding activation layers of two Siamese architectures. The categorical loss can be combined with the distribution loss such that the neural network learns a nonlinear transformation that minimizes the statistical distance between the source and target domain, while also discriminating between the classes. Deep Domain Confusion (DDC), proposed in [6], uses the Krizhevsky architecture [7] and adds a lower-dimensional, "bottleneck" adaptation layer after the fully connected layer fc7. An maximum mean discrepancy(MMD) loss is applied to the adaption layer to regularize the representation to become invariant to the source and target data. DDC uses one layer for computing the loss, but it is possible to extend the practice to several layers. Extending the idea of MMD, [8] proposes a Deep Adaptation Network (DAN) which incorporates the sum of MMD measures of several fully connected layers. Further, the authors explore other kernels, such as polynomial and the radial basis function (RBF). The resulting method outperforms DDC in all domain shift scenarios presented in [8], presumably because of the use of multiple layers instead of one.

3.3 Adversarial Generative Models

Models under this category rely on a generative adversarial network (GAN) architecture [9]. The generator produces artificial data instances that the discriminator tries to distinguish from the training data. The discriminator tells the generator (via the generator loss) that the artificial data instances are not plausible enough, by which the model parameters of the generator are updated via backpropagation. The discriminator loss updates the model parameters of the discriminator such that it improves at separating generated data from real training data. In the DA field, GAN can be used to facilitate domain confusion. A coupled GAN (CoGAN) framework is proposed by [10]. Each GAN generates images in their respective domain, and with weight sharing constraints, the model learns a joint distribution in the two domains from images separately drawn from the marginal distributions of each individual domain. Unsupervised Pixel-level Domain Adaptation with Generative Adversarial Networks (PixelDA) [11] has a generator function that maps a source image and a noise vector, to an "adapted" image that looks as if it was drawn from the target domain. Along with the source labels, the adapted source images create an adapted dataset on which a classifier can be learned in a supervised manner. In contrast to discrepancy-based methods, which join the DA task and the discriminate task in one single network, this GAN framework decouples the domain adaptation task from the discriminative task, allowing users to focus on training a classifier after the adapted dataset has been generated.

4 Transfer Learning

The concept of transfer of learning is traced back to the paper *The relation of special training and general intelligence* [12], published by the educational psychologist Charles Hubbard Judd in 1908 in the journal *Educational review.* The paper presents an experiment conducted several years earlier in which fifth- and sixth-grade students were given the task of throwing darts at a target submerged in 12 inches of water. The treatment group was taught about the theory of light refraction that made the underwater dartboard appear skewed. There was no difference between the accuracy of the control and treatment groups when the water depth was 12 inches. However, when the water depth was changed to four inches, the treatment group trained in refraction theory performed better than the control group.

4.1 Transfer Learning & Domain Adaptation

Following the notation and definitions of [13] and [14], the definitions of a domain and a task should be presented before defining transfer learning.

Definition 1 (Domain) *A domain is comprised by two parts, a feature space* $\mathcal{X} \in \mathbb{R}^d$ *and a marginal distribution* $P(\boldsymbol{X})$*. The domain can be written as* $\mathcal{D} = \{\mathcal{X}, P(\boldsymbol{X})\}$*, where* $\boldsymbol{X}$ *is an instance set defined as* $\boldsymbol{X} = \{\boldsymbol{x} \mid \boldsymbol{x}_i \in \mathcal{X}, \quad i = 1, 2, \ldots n\}$.

Definition 2 (Task) *A task is comprised by a label space* $\mathcal{Y} \in \mathbb{N}$ *and a decision function* $f : \mathbb{R} \mapsto \mathbb{N}$*. The task can be written as* $\mathcal{T} = \{\mathcal{Y}, f\}$*. The decision function is implicit and learned from experiencing corresponding features.*

In some cases, the decision function predicts the conditional distribution itself, i.e., $f(\boldsymbol{X}) = P(\boldsymbol{Y}|\boldsymbol{X})$, and in a classification task, the function predicts a label $f(\boldsymbol{X}) = \boldsymbol{y}_{pred}$. The prediction can be a binary vector of dimension n, or a logit $\boldsymbol{y}_{pred} \in 1, 2, \ldots n$. Consider two domains, a *source* domain $\mathcal{D}_s = \{\mathcal{X}_s, P(\boldsymbol{X}_s)\}$ with $\mathcal{T}_s = \{\mathcal{Y}_s, P(\boldsymbol{Y}_s|\boldsymbol{X}_s)\}$ and a *target* domain $\mathcal{D}_t = \{\mathcal{X}_t, P(\boldsymbol{X}_t)\}$ with $\mathcal{T}_t = \{\mathcal{Y}_t, P(\boldsymbol{Y}_t|\boldsymbol{X}_t)\}$. Now imagine that the target domain has no label set. If the source and target domains are equal, $\mathcal{D}_s = \mathcal{D}_t$ and the tasks are equal, $\mathcal{T}_s = \mathcal{T}_t$, then a supervised machine learning model trained on the source domain would generalize to the target domain. However, if there exists a *domain shift*, i.e., the domains are dissimilar $\mathcal{D}_s \neq \mathcal{D}_t$ (either $\boldsymbol{X}_s \neq \boldsymbol{X}_t$ or $P(\boldsymbol{X}_s) \neq P(\boldsymbol{X}_t)$), the model would not generalize well to the target domain. Transfer learning (TL) aims to use learned feature representations from the source domain to improve task performance in the target domain. More specifically, TL is defined in [13] as

Definition 3 (Transfer Learning) *Given a source domain* $\mathcal{D}_s$ *and learning task* $\mathcal{T}_s$*, a target domain* $\mathcal{D}_t$ *and learning task* $\mathcal{T}_t$*, transfer learning aims to improve the learning of the target predictive function* $f_t : \mathbb{R} \mapsto \mathbb{N}$ *in* $\mathcal{D}_t$ *using the knowledge in* $\mathcal{D}_s$ *and* $\mathcal{T}_s$*, where* $\mathcal{D}_s \neq \mathcal{D}_t$ *, or* $\mathcal{T}_s \neq \mathcal{T}_t$.

Transductive transfer learning refers to the case where the source and target tasks are equal $\mathcal{T}_s = \mathcal{T}_t$, but there is a non-trivial domain shift. When the feature spaces are the same $\mathcal{X}_s = \mathcal{X}_t$ but the marginal distributions are different $P(\mathcal{X}_s) \neq P(\mathcal{X}_t)$, we enter the practice of domain adaptation (DA). When labels for the target domain $\{\mathcal{Y}_t, P(\boldsymbol{Y}_t|\boldsymbol{X}_t)\}$ are not available, we speak of unsupervised DA, and when there is a scarce set of labels available in the target domain, we speak of supervised DA [13]. A prerequisite for successful domain adaptation is that there must be some similarities between the two domains. For visual applications, this would be low-level features such as shapes and edges, while high-level features, such as image background, lighting, color, etc. are allowed to differ.

5 CORAL

CORAL is short for covariance alignment which minimizes the Frobenius distance between the covariance matrices $\boldsymbol{C}_s$ and $\boldsymbol{C}_t$ of the source and target data $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ by finding the optimal linear transformation $\boldsymbol{A}^*$ acting on the source covariance. The method is a *feature alignment* method, i.e. it transforms the source data to align with the target data with respect to some measure, in this case, the second-order statistic. The optimal linear transformation can be found by considering the optimization problem

$$\boldsymbol{A}^* = \underset{\boldsymbol{A}}{\arg\min} \|\hat{\boldsymbol{C}}_s - \boldsymbol{C}_t\|_F = \underset{\boldsymbol{A}}{\arg\min} \|\boldsymbol{A}^T \boldsymbol{C}_s \boldsymbol{A} - \boldsymbol{C}_t\|_F, \tag{1}$$

where $\hat{\boldsymbol{C}}_s$ is the covariance of the transformed source features $\boldsymbol{X}_s$ and $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^2}$ is the matrix Frobenius norm, see Definition 5 in Appendix A. The optimization problem (1) is derived in [5], and has the solution

$$\boldsymbol{A}^* = \left(\boldsymbol{U}_s \boldsymbol{\Sigma}_s^{\dagger\frac{1}{2}} \boldsymbol{U}_s^\top\right) \left(\boldsymbol{U}_{t[1:r]} \boldsymbol{\Sigma}_{t[1:r]}^{\frac{1}{2}} \boldsymbol{U}_{t[1:r]}^\top\right), \tag{2}$$

where $(\boldsymbol{U\Sigma U}^\top)$ is the singular value decomposition (SVD) of $\boldsymbol{C}$, the superscript $\dagger$ denotes the Moore-Penrose psuedo inverse [15], and r is the lowest rank of the source and target covariance matrices $r = min(r_{C_s}, r_{C_t})$. However, computing the SVD of the covariance matrices is relatively expensive, and could be unstable. Therefore, it is sufficient to use the simpler transformation

$$\boldsymbol{A}^* = \boldsymbol{C}_s^{\dagger\frac{1}{2}} \boldsymbol{C}_t^{\frac{1}{2}}. \tag{3}$$

The squareroot of the covariance is performed by a blocking schur algorithm [16]. CORAL is summarized in Algorithm 1.

Algorithm 1: CORAL for Unsupervised Domain Adaptation

1 **Input**: Source data $\boldsymbol{X}_s$ and target data $\boldsymbol{X}_t$
2 **Output**: Adjusted source data $\boldsymbol{X}_S^*$
3 $\boldsymbol{C}_s = cov(\boldsymbol{X}_s)$
4 $\boldsymbol{C}_t = cov(\boldsymbol{X}_t)$
5 $\boldsymbol{X}_s = \boldsymbol{X}_s \boldsymbol{C}_s^{-\frac{1}{2}}$ ▷ Whitening source
6 $\boldsymbol{X}_s^* = \boldsymbol{X}_s \boldsymbol{C}_t^{\frac{1}{2}}$ ▷ re-coloring with target covariance

6 Deep CORAL

Extending the CORAL method, which applies a linear transformation, [17] proposes a deep approach to learning features that generalize well in the target domain by constructing a differentiable joint loss function from the classification loss ℓ_{class} and the CORAL loss ℓ_{CORAL}

$$\ell = \ell_{class} + \lambda \ell_{CORAL}, \tag{4}$$

where λ weights the relative importance between the domain adaptation task and the discriminative task in the source domain. Minimizing the CORAL loss is analogous to minimizing the L2 distance between the second order statistics of the source and target data

$$\ell_{CORAL} = \frac{1}{4d^2} \|\boldsymbol{C}_s - \boldsymbol{C}_t\|_F^2, \tag{5}$$

where d is the dimension of the features and F denotes the Frobenius norm, see Definition 5 in Appendix A. The covariances are calculated as

$$\boldsymbol{C}_s = \frac{1}{n_s - 1}\left(\boldsymbol{D}_s^T \boldsymbol{D}_s - \frac{1}{n_s}(\mathbf{1}^T \boldsymbol{D}_s)^T (\mathbf{1}^T \boldsymbol{D}_s)\right) \tag{6}$$

$$\boldsymbol{C}_t = \frac{1}{n_t - 1}\left(\boldsymbol{D}_t^T \boldsymbol{D}_t - \frac{1}{n_t}(\mathbf{1}^T \boldsymbol{D}_t)^T (\mathbf{1}^T \boldsymbol{D}_t)\right), \tag{7}$$

where n_s (n_t) is the number of data points in the source (target) domain, $\boldsymbol{D}_s$ ($\boldsymbol{D}_t$) the source (target) data, and $\mathbf{1}$ is a column vector with all elements equal to one. While [17] implements the deep CORAL method on a single fully connected layer, the authors add that it should be straightforward to apply the loss to several other layers.

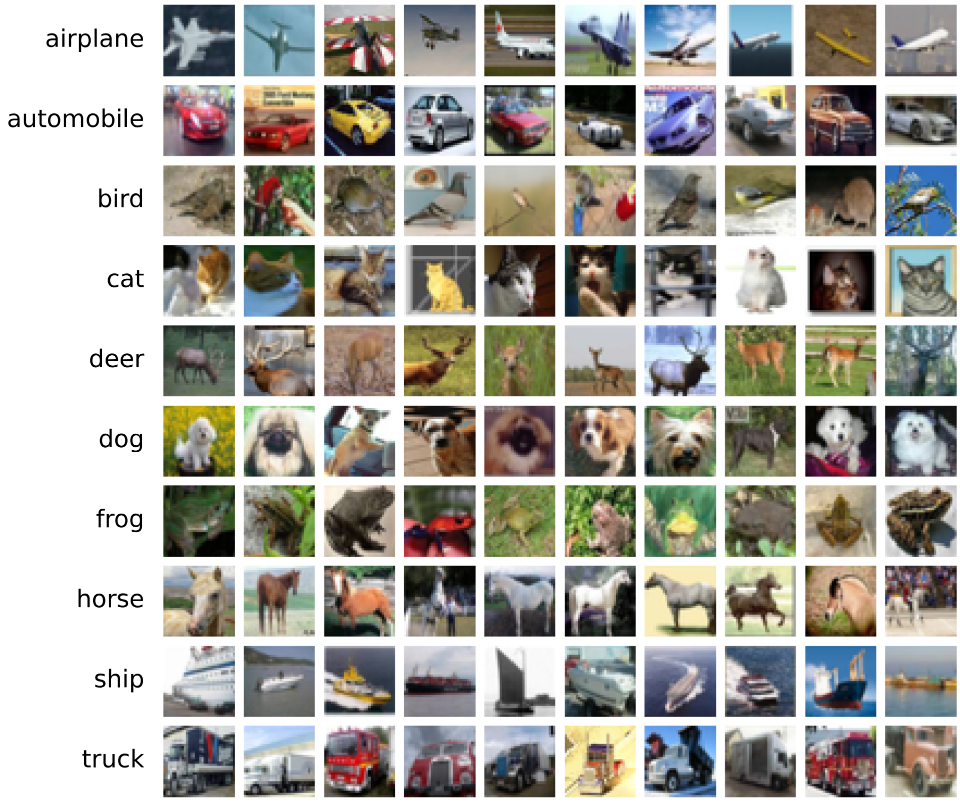


Figure 1: *Ten randomly picked images of each class in the Cifar-10 dataset.*

7 Data Acquisition and Preprocessing

The following section presents the acquisition or construction of the datasets which are used in this project. Before using the datasets, the image means are normalized. Since the covariance measure is central to the DA tasks, the covariances are not standardized.

7.1 Cifar-10

The Cifar-10 dataset [18] is a subset of the *80 million tiny images* dataset, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and contains 60 000 32×32 RGB images categorized into ten balanced (same number of instances in all classes) classes. There are 50 000 images for training, 5000 of them are validation instances, and 10 000 images for testing. Figure 1 shows the ten classes, with ten randomly selected images from each class. To match the dimensions of the LiDAR data, the Cifar-10 images are converted to one-channel grayscale images.

7.2 LiDAR Dataset

The balanced Lidar (light detection and ranging) dataset contains 4010 32×32 one channel images taken by the vehicles at the Scania autonomous driving department. Of these images, there are 3600 training instances, of which 10% are for validation 410 are test instances. The two classes (car and pedestrian) are shown in Figure 2, with ten random images from each class. The

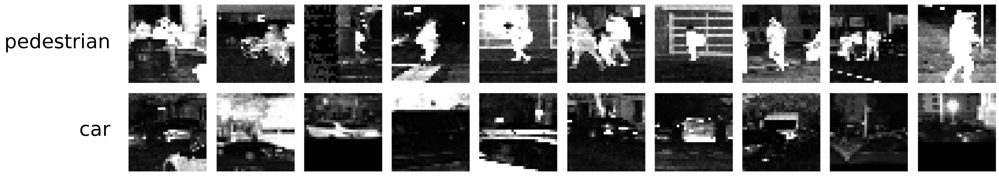


Figure 2: *Ten randomly picked LiDAR intensity images from each class in the LiDAR dataset.*

LiDAR intensity images come from LiDAR point clouds projected onto a 2D image, whose pixel values correspond to the intensity value for each point in the cloud. LiDAR works by emitting pulses of photons in a laser beam and measuring the time it takes for the photons to return to the transmitter. The light returns by reflecting off objects in the surrounding environment, and will sometimes reflect away from the transmitter, producing relatively sparse images, as shown in Figure 2. **CROPPING PROCEDURE??? THRESHOLD OF 10 POINTS IN IMAGE - SPARSITY**

7.3 Camera Dataset

The balanced camera dataset consists of 18 352 grayscale images with 32×32 pixels taken by the vehicles of Scania's autonomous driving department. The dataset is divided into 16 520 training instances, of which 10% are used for validation, and 1832 test instances. The two classes can be seen in Figure 7, with ten randomly selected images from each class. The cameras are located at all eight corners of the vehicles to provide a wide field of view of the traffic. A labeling company has created a set of corresponding Label files (a dictionary data type in a JSON file format). For one image there is a label file with xy coordinates for bounding boxes and labels for each corresponding object (among other properties of the objects in the scene). All relevant objects in the image are cropped to a square shape using the bounding boxes and resampled so that all cropped images are 32×32 pixels. The images are converted to grayscale so that they have the same shape as the LiDAR intensity images, and can be used by the same neural network architecture. The cropping process follows a set of rules, illustrated in Figure 3 and 5 alongside the resulting cropped images seen in Figure 4 and 6. The cropping rules are:
If the longest side length of the bounding box is

1. below the threshold of 10 pixels, the image is omitted from the dataset to uphold quality.
2. larger than the threshold but smaller than the desired crop size (32x32), then the boundaries for the object are expanded to the desired crop size.
3. expanded beyond the boundaries of the image, the expansion will stop at the image border and the expand in the other direction to create a square crop.
4. larger than the desired crop size, it is cropped square and resampled to the desired crop size using Lanczos interpolation [19].

The effect of each rule can be seen in effect for two traffic scenes, shown in Figure 3 and 5, with the resulting cropped images in Figure 4 and 6. Rule 1 which omits small objects from the image is shown in Figure 3 as red boxes which surrounding the most distant vehicles. Rule 2 is applied to the vehicles in images 1, 2, 3, 4, 8, 9, and 10 in Figure 4 because the crop bounds are not adjacent to the vehicles in the image, but are expanded to uphold resolution of 32x32 pixels. For the white vehicle in the lower left corner of Figure 3 (object number 6 in Figure 4),

Figure 3: *The orange rectangles illustrates the original bounding boxes from the label data, and the yellow squares illustrates the cropping method, except for resampling to desired resolution. Objects bounded by red boxes has not passed the size threshold and has therefore been omitted.*

rules 3 and 4 apply. Rule 3 places the square crop such that the vehicle in not centered, and is later downsampled. A relatively simple cropping procedure like this will also include some data instances that may degrade the quality of the dataset, and are therefore better discarded. It is inevitable that some objects will be obscured by something in the foreground, such as a traffic sign, a lamppost, or another vehicle, or pedestrian. Expanding a bounding box will also include other objects that are not relevant to the label of the cropped image. One could argue that the inclusion of images with obstructions improves the model because it accurately represents a real-world scenario.

8 Experimental Evaluation

The experiments test the shallow CORAL method, a supervised model on the source domain (camera dataset), and the deep CORAL method, involving the hyperparammeter tuning of the CORAL loss weight λ from equation (4). The models are evaluated and compared using the classification accuracy of the LiDAR test dataset, and the confusion matrix metric, that reveals classification biases in the models.

8.1 Pretraining the model

A convolutional neural network (CNN) is pretrained on the Cifar-10 dataset for $N_e = 15$ epochs. The architecture (seen in Figure 8) is inspired by the ”block” design proposed by the Visual

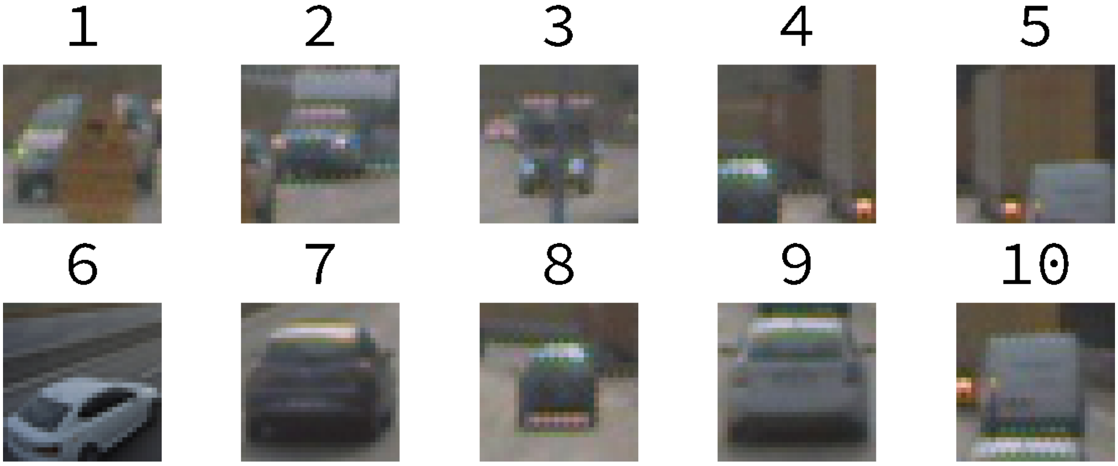


Figure 4: *Cropped images with equal size and shape.*

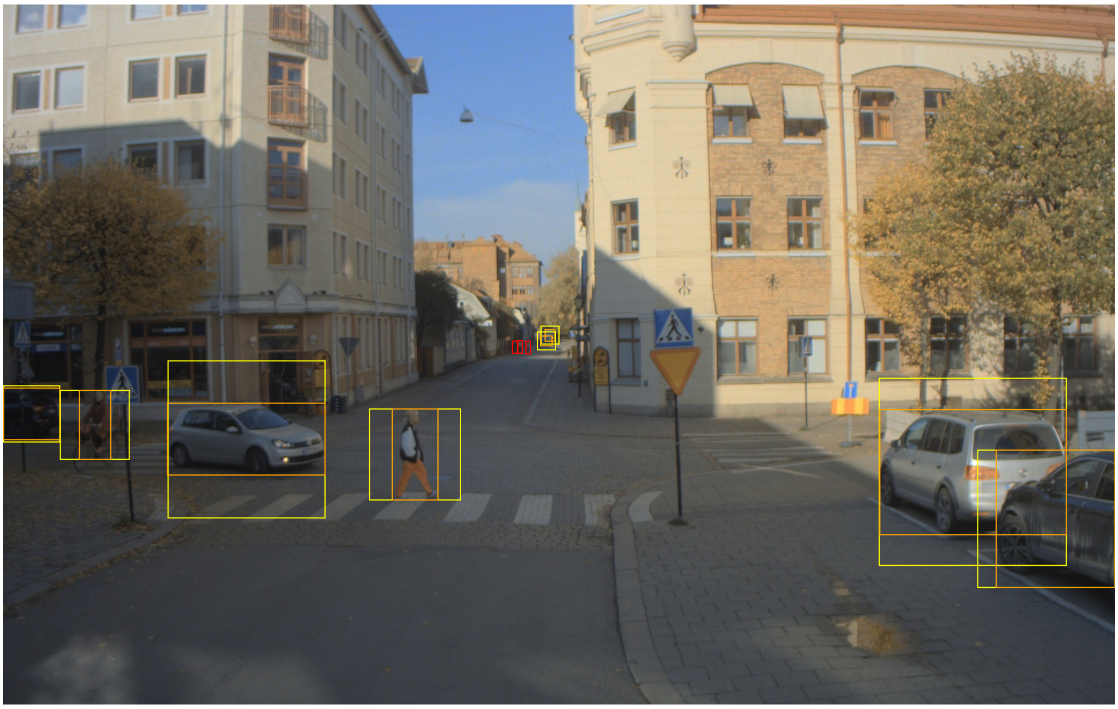


Figure 5: *Another example of a traffic scene with a total of five pedestrians present (cyclist included), three of which are omitted from the dataset because the corresponding bounding boxes are too small (colored in red).*

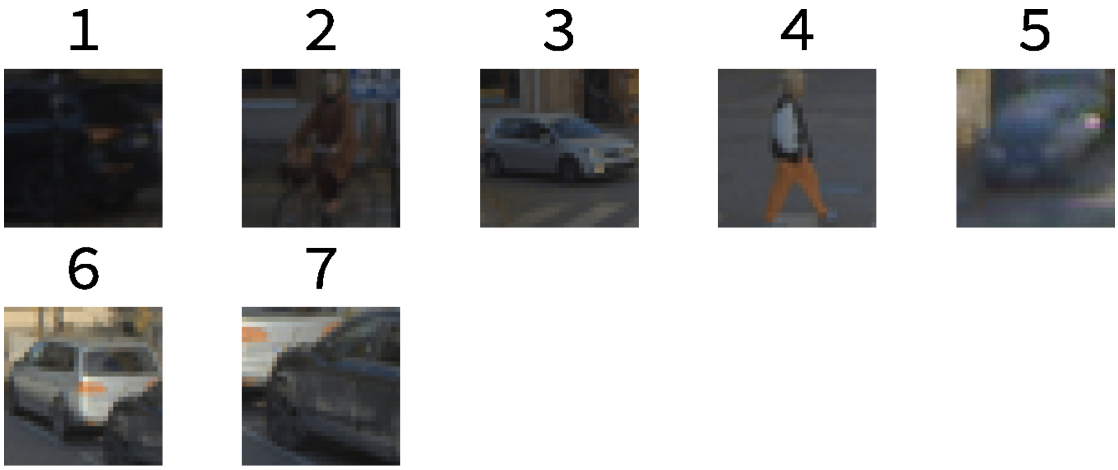


Figure 6: *Cropped images with equal size and shape.*

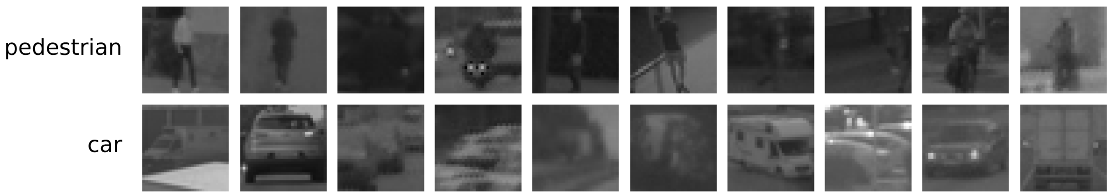


Figure 7: *Ten randomly picked grayscale images from each class in the Camera dataset.*

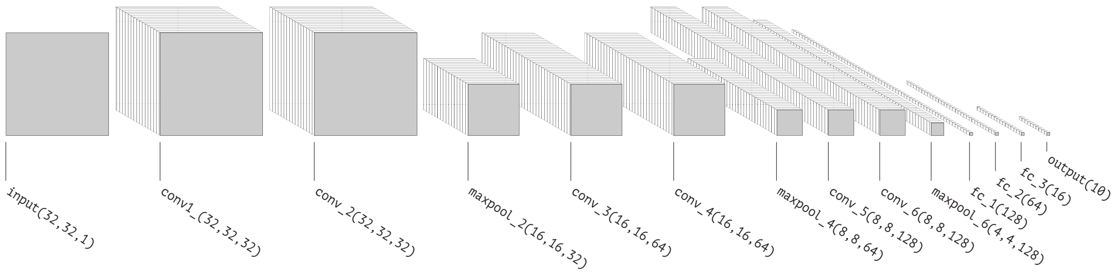


Figure 8: *Architecture of the convolutional neural network. Each layer and its output dimension is defined as* `layer_name(width,height,channels)`.

Geometry Group (VGG) at the University of Oxford in [20], known as VGG blocks. There are three VGG blocks which contains two convolutional layers followed by maxpooling, performed with a 2×2 window with stride 2, and a subsequent dropout layer with $p = 0.2$. All convolutional layers has filters with a receptive field of 3×3. The convolution stride is set to 1, and the spatial padding of the convolutions are set such that the resolution is preserved (`padding = 'same'`). After each convolutional layer there is also a batch normalization layer. The dropout layer parameter is set to a $p = 20\%$ chance of not updating the layer parameters. The dropout and batch normalization prevents, to a certain degree, the model from overfitting to the training data. Following the last maxpool layer `maxpool_6`, there are four fully connected layers, followed by batch normalization layers. All hidden layers has a rectified linear unit (ReLU) activation function. The input images has a batch size of 32, base learning rate is $\epsilon_0 = 0.01$ which decreases exponentially as $\epsilon_k = \epsilon_0 e^{-0.05k}$ for $k \in \{1, 2, \ldots, N_e\}$. The stochastic gradient descent (SGD) has a momentum of 0.9 and a weight decay of 10^{-5} (for more details on SGD, see Appendix Section B.1). The loss function is a categorical cross-entropy (CE) loss function (for more details on cross entropy, see Appendix Section B.2). The CNN architecture is not specifically tailored for optimal performance, but rather used as foundation on which the findings of DA methods can be evaluated. The accuracy on the Cifar-10 test dataset reaches 80.70%. Pretraining will give the optimizer a better start guess than normally distributed weights and biases, such that a minima is reached faster when finetuning the model on the source dataset. The pretrained model is used later to initialize the weights of other models with identical architectures, except for the output layer which is replaced to fit the number of classes needed for the new classification task. When the pretrained weights are used to initialize the weights of a model, they will be referred to as the Cifar-10 weights.

8.2 CORAL Model

The setup for the shallow CORAL model is illustrated in Figure 9. Red and blue arrows represent the camera and LiDAR data, respectively, and purple arrows after the CORAL method represents the aligned camera features. After the CNN has been pretrained, the camera and LiDAR images are passed through, where their ten dimensional deep features are extracted at the output layer. The extracted camera deep features are then aligned to the LiDAR deep features using the linear transformation according to Algorithm 1 on page 9. Thereafter, a linear classifier is trained on the aligned camera deep features and evaluated on the LiDAR deep features. Note that the CNN is not trained on the camera or LiDAR data after pretraining. The separate linear classifier learns to discriminate between the two classes (cars and pedestrians) using the camera deep features which are aligned to the LiDAR deep feautres w.r.t. their batch-wise covariances. Training

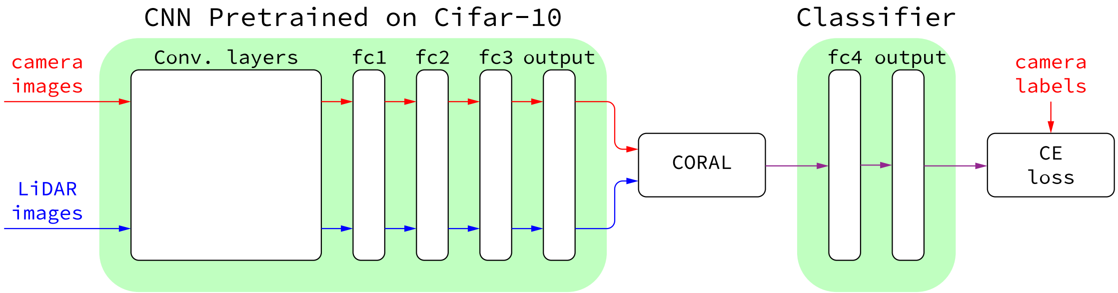


Figure 9: *The shallow CORAL model. Red arrows and text represent the camera data being, and blue represents the LiDAR data. When the camera features are aligned in the CORAL method, it is depicted by the purple arrows. The architecture of the pretrained CNN is seen in Figure 8*

on aligned camera deep features enables the linear classifier to adapt and increase classification accuracy in the target domain to 74.67% on the LiDAR test dataset.

8.3 Deep Model without CORAL Loss

The baseline model is a supervised CNN model with parameters initialized by the cifar-10 dataset. The model is further trained on the camera dataset as to solely rely on the similarity of source and target domain without any DA method. As the dataset has two classes, the output layer from the CNN of ten dimensions is replaced with a output layer with two dimensions. Due to the new output layer being randomly initialized, a higher initial learning rate is employed such that it will not take as long for the optimizer to reach an optimum. More specifically, the model has a SGD optimizer with initial learning rates $\epsilon_{0,fc4} = 0.01$ for the output parameters and $\epsilon_0 = 0.001$ for all other parameters. The momentum is set to 0.9 and the L2 weight decay coefficient to 10^{-5}. With each epoch, the learning rate decays exponentially as $\epsilon_i = \epsilon_0 e^{-0.05*i}$ for $\{i\}_{i=1}^{N_e}$, where $N_e = 30$ is the number of epochs performed. The setup for this baseline model is illustrated in Figure 10. First, the weights are initialized with the Cifar-10 weights and then the camera images are passed through the model in batches of 32 and evaluated in the CE loss function in a supervised manner. Training in the source domain while excluding the CORAL loss in the objective function increases the CORAL distance over the iterations, as seen in Figure 12. The CORAL distance increases in general during training, but jumps significantly, which might be mitigated by using a larger LiDAR batch size. The increased CORAL distance seems to correlate with a decreased validation accuracy in the target domain, seen in Figure 11. The training history shown in Figure 11 and 12 also suggests that one epoch of training generalizes the model relatively well to the target domain, before it becomes too specific for the source domain, which is also reflected by the shortest CORAL distance achieved is in beginning of training. As the CORAL distance and validation accuracy in the source domain increases, the validation accuracy in the target domain decreases to 67.05%, which further suggests that minimizing the CORAL distance would in opposition increase the validation accuracy for the LiDAR data. The 16 dimensional LiDAR deep features from the third fully connected layer (fc3) are visualized using T-distributed Stochastic Neighbourhood Embedding (TSNE) in Figure 19b. The embedded features are not well separated in the plot which is reflected in the relatively low validation accuracy in the target domain.

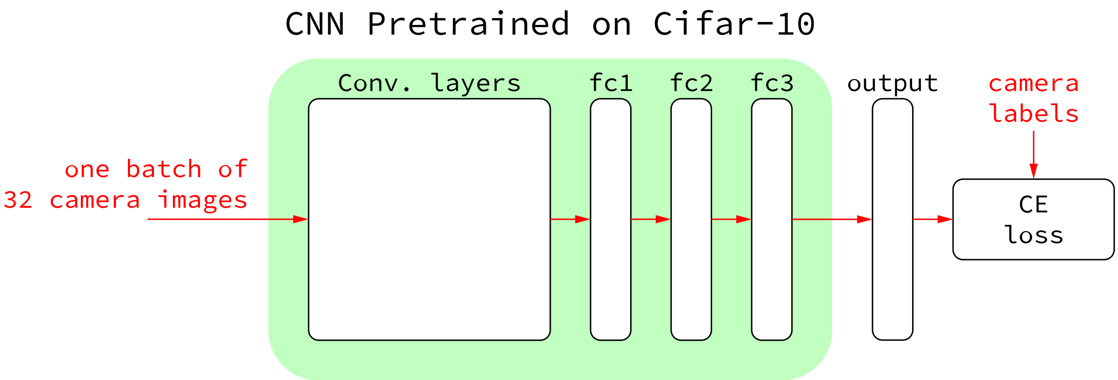


Figure 10: *The baseline model is transforming the camera data, shown with red arrows, in a supervised manner.*

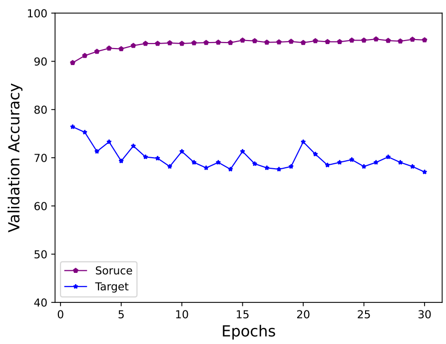


Figure 11: *With no means for domain adaptation, accuracy in the target domain steadily decreases as the model learns a classifier in the source domain.*

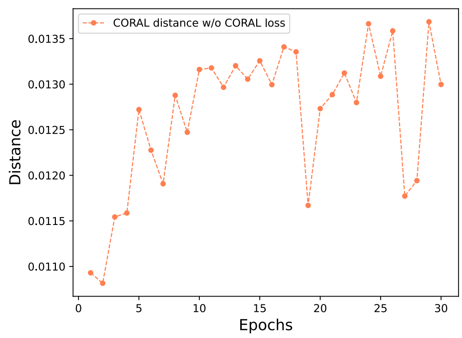


Figure 12: *As the CORAL distance (5) is not included in the loss function, it increases over the training iterations.*

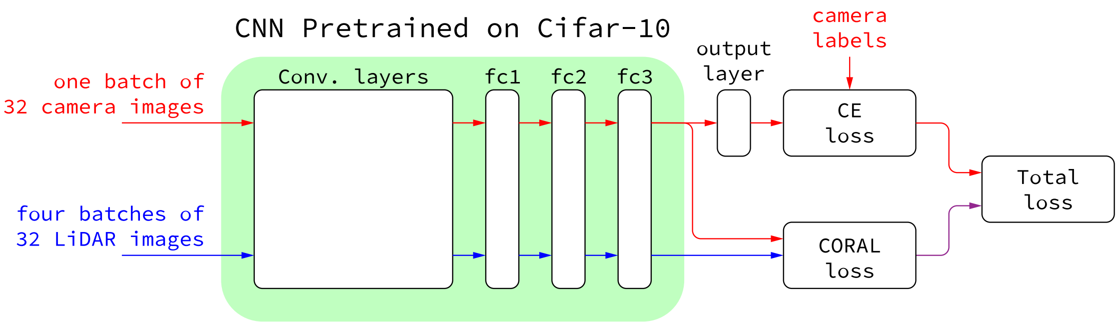


Figure 13: *The deep CORAL model. Red and blue arrows represent the camera and LiDAR data, respectively. The purple arrow represents the CORAL loss which is jointly evaluated with the CE loss in the total loss. The architecture of the pretrained CNN is seen in Figure 8*

8.4 Deep CORAL Model

The Deep CORAL model shares most of the hyperparameters with the model without a CORAL loss in Section 8.3, except for the number of epochs performed which is $N_e = 20$, and the additional CORAL weight λ from the joint loss function (4). The pipeline seen in Figure 13, begins similarly to the model without CORAL loss. The LiDAR and camera images are passed through the pretrained CNN, and extracted at fc3. Thereafter, the camera features are passed through the output layer, from which they are measured against the corresponding labels in the CE loss function. The batch of camera features from fc3 are also measured in the CORAL loss function four times against four different batches of LiDAR features, to get a better measurement for the domain adaptation task. The mean CORAL distance is then multiplied with a CORAL loss weight λ to control how much the optimizer should take the DA task into account when updating the weights. Lastly, the CE loss and CORAL loss are added together (total loss) from which the optimizer iterates towards an optimal set of weights and biases. A successful model should be able to discriminate between classes in the target domain by 1), the supervised learning task in the source domain and 2), the domain adaptation task, which adapts the classifier to perform well in the target domain.

8.4.1 Hyperparameter Tuning

The hyperparamter λ from the joint loss function in Equation (4) on page 19 is chosen such that a high validation accuracy on the target (LiDAR) data is achieved. Because it is unknown how the model behaves with respect to λ, 20 samples are drawn from the logarithmically uniform distribution

$$\lambda \sim f(x; a, b) = \frac{1}{x \log_e(\frac{b}{a})}, \tag{8}$$

with $a = 10^0$ and $b = 10^6$. The log-uniform distribution is an effective grid search because it spans several orders of magnitude using few samples, thus finding the best hyperparameter relatively quickly. For each sample of λ, the deep CORAL model is trained according to the specifications given in Section 6. The final validation accuracy on the source and target data, and the validation losses w.r.t. the hyperparameter can be seen in Figure 14 and 15. The goal is to yield the highest validation accuracy in the target domain, seen in blue in Figure 14, while maintaining a low total loss, seen in dark red in Figure 15. As the optimizer seeks to minimize the CE loss and the weighted CORAL loss jointly, the *weighted* CORAL loss is

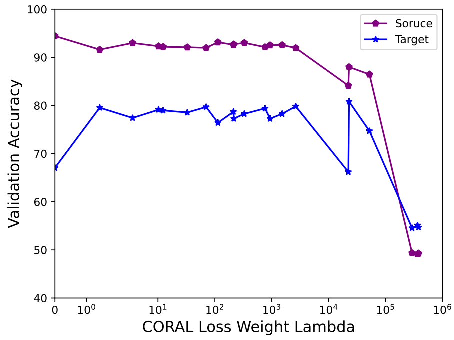


Figure 14: *Source and target validation accuracy at end of training with respect to the weight to the CORAL loss*

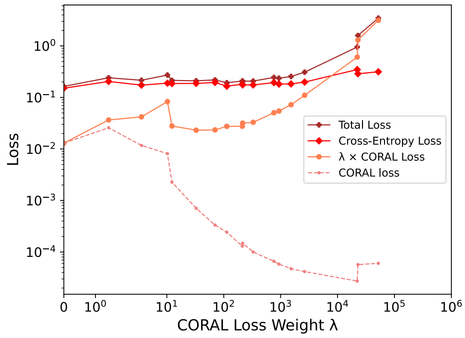


Figure 15: *Target test accuracy with respect to the weight for the CORAL loss function. The loss is undefined for the four last values of λ, and is therefore not included.*

also plotted (appropriately in the color *coral*) in Figure 15. For low values of λ, the CE loss is greater than the weighted CORAL loss. As λ increases, the weighted CORAL loss does too, as it is multiplied by λ. The minimal weighted CORAL loss produced by the optimizer is found at about $\lambda = 10^2$. Thereafter the weighted CORAL loss increases steadily, due to the CORAL loss (colored in *light coral* in Figure 15) not decreasing at the same rate as λ is increasing. Closest sampled value to $\lambda = 10^2$ is $\lambda = 70.229$, which is roughly where the minimal total loss is obtained as well. From excluding the CORAL distance from the objective function ($\lambda = 0$) to including the CORAL loss $\lambda \in (10^0, 10^3)$, there is a substantial increase in the validation accuracy on the target data, showm in Figure 14, where it remains stable at just below 80%. For $\lambda = 70.299$, the validation accuracy is 79.688%. If the hyperparameter is increased further, the task of minimizing the CORAL loss will dominate the objective function, compromising the discriminative task such that performance suffers to the point of random choice (accuracy near 50%) at about $\lambda \in (10^5, 10^6)$. The 16 dimensional LiDAR deep features from the third fully connected layer (fc3), with $\lambda = 70.299$, are visualized using TSNE in Figure 19c. Compared to the TSNE plot without CORAL loss in Figure 19b, the deep CORAL model has increased the interclass distance (distance *between* clusters of different classes, see Figure 18) and decreased the intraclass distance (distance among data instances *within* classes). The separation of the two classes is also reflected in the increased validation accuracy in the target domain. The reason why performance suffers at larger values of λ is due to the trivial solution solution of transforming all data instances, regardless of class, into almost the same value, making the covariance distance very small, and the accuracy near 50%. In other words, the interclass distance is minimized when the actual task is to maximize it.

9 Results

Initializing the model parameters by pretraining on the cifar-10 dataset provides a good foundation for all methods presented in the paper. This proves yet another aspect of transfer of learning, which is the generalizabillity from low level visual features learned by one domain (in which the the Cifar-10 data resides). The shallow CORAL model acts on the deep features produced by the the pretrained network and learns a linear classifier on the aligned source fea-

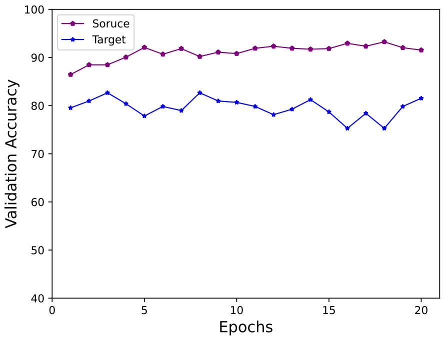


Figure 16: *Validation accuracy for source and target data with tuned hyperparameter* $\lambda = 70.299$.

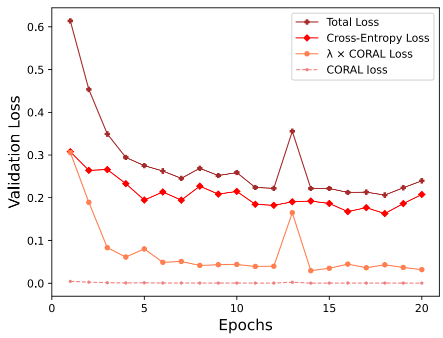


Figure 17: *Validation losses with tuned hyperparameter* $\lambda = 70.299$

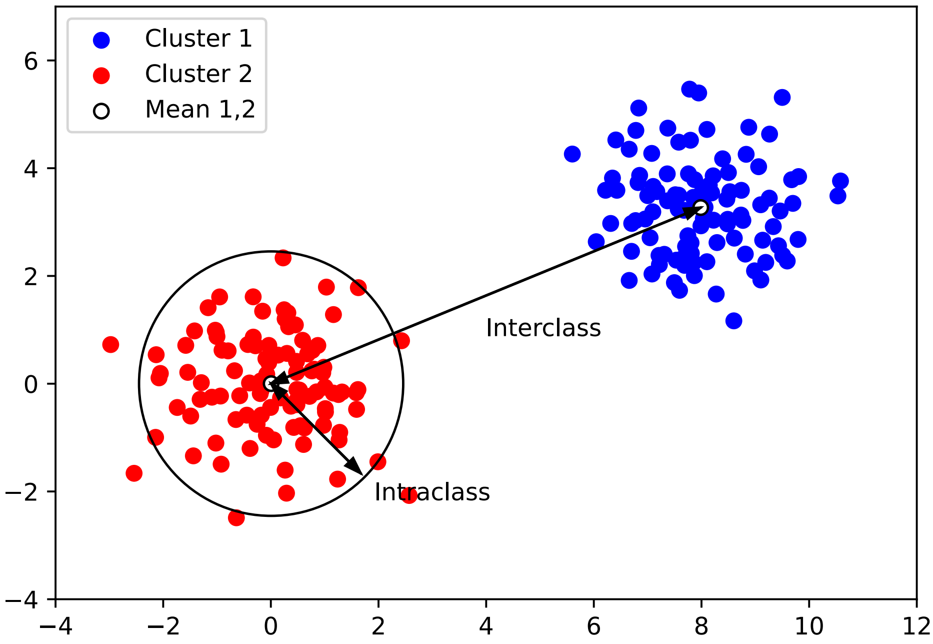
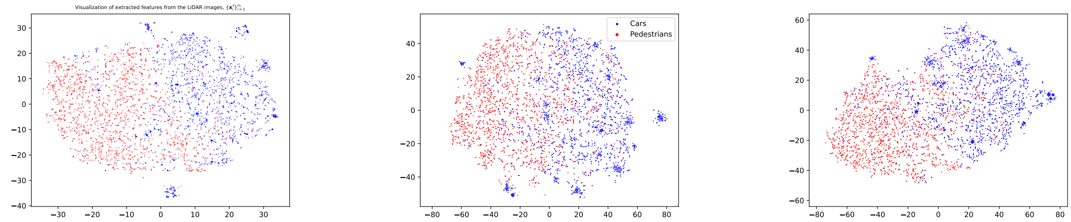


Figure 18: *Samples drawn from two different Gaussian distributions. Intraclass distance is the distance between the clusters (illustrated as the means of each cluster) and the Interclass distance is the distance between the points within a cluster. A successful classifier maximizes the interclass distance and minimizes the intraclass distance.*



(a) *Shallow CORAL model.* (b) *Deep model* (c) *Deep CORAL model.*

Figure 19: *TSNE of deep features from the third fully connected layer (fc3) of the deep model without a CORAL loss. The method embeds the deep features which has 16 dimensions to two dimensions, seen as points in the scatterplot.*

tures. The model yields a test accuracy of 74.67% on the LiDAR test dataset. The confusion matrix seen in Figure 20 gives insight into the class-wise predictions. horizontal axis is represents the predicted class, and the vertical axis represents the true class. Each quadrant displays all possible scenarios of predicting an image....the percentage of predictions of each class, both for correct and incorrect predictions. While the accuracy of the shallow CORAL model is relatively low, it has low bias towards any of the classes. The label "car" is predicted almost as many times as "pedestrian", regardless of it being a correct or false prediction. Pretraining is a very efficient way of initializing the model parameters, and provides a good starting guess for the SGD optimizer when the model is further trained for another, similar task. This is especially apparent in the training history of the finetuning model in Figure 11, where the LiDAR accuracy (in blue) is highest in the beginning of training, and deteriorates as the model is trained on the camera dataset, until the LiDAR accuracy reaches 67.06%. The lost generalizabillity is also reflected in the confusion matrix, seen in Figure 21, which shows a strong bias towards predicting the label "car". Specifically, it predicts car on 82.03% of the images, when 50% of the images contains cars. The deep CORAL model includes the CORAL distance in the loss function. With the tuned hyperparameter $\lambda = 70.299$, the Deep CORAL model increases the target accuracy to 79.688% and shifts the bias back to a more balanced confusion matrix, seen in Figure 22.

on the LiDAR test data of . for the deep models yields both for the relatively high classification accuracy on the target da Supervised learning on the camera dataset reaches a test accuracy on the LiDAR dataset of 67.06%, and wifinetuned model (supervised learning on the camera dataset) acting as a baseline Shallow CORAL reaches a test accuracy of percent, while the Deep CORAL model increases performance further, as the CORAL distance is included in the loss function such that the DA task is a non-linear transformation.

a baseline test, a non-CORAL test, and a CORAL test. The baseline serves to show the classification accuracy when no domain adaptation, nor pretrained network is used. The deep neural network is trained on the Bevda camera images and evaluated on the Lidar intensity images from Ouster. The non-CORAL setup has the nn pretrained on the Cifar-10 dataset, through which the camera and LiDAR images passed through and extracted at the Nth denselayer at which they are considered deep features. Thereafter a linear classifier (whichever) is trained on the camera deep features, and evaluated on the LiDAR deep features. Finally the CORAL test is setup similar to the non-CORAL test in that the nn is also pretrained on Cifar-10, and the camera and LiDAR deep features are obtained. At this stage however, the camera deep features are aligned to the target deep features and then used to train the linear classifier. Lastly the

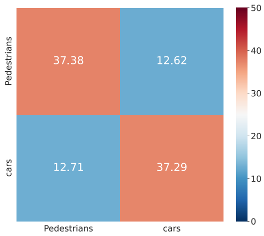


Figure 20: *Shallow CORAL method*

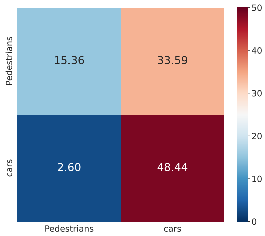


Figure 21: *of the deep model without a CORAL loss.*

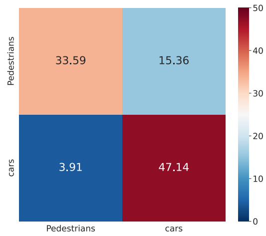


Figure 22: *of the deep model with* $\lambda = 70.299$.

Table 1: *Accuracy scores*

Shallow CORAL	**Finetuned**	**Deep CORAL**
74.67%	63.8%	80.73%

linear classifier is evaluated on the LiDAR deep features.

10 Future Work

multilayer deep CORAL semi-supervised learning

11 Conclusions

This thesis has studied how to minimize the discrepancy, or the *domain shift*, between two datasets from two different, but similar domains, via their second order statistic measure (the covariance). If one dataset does not have any labels (from the target domain), it cannot be used in a supervised learning model. But its statistical properties can be Incorporated in the model such that it adapts to the target domain, and thus increasing the classification accuracy.

The CORAL method (covariance alignment) finds the optimal linear transformation $\boldsymbol{A}$ which aligns the source data to the target data such that the l2-distance between the two covariances are minimized. Thereafter an the aligned source data is used in a supervised learning task to find a linear classifier, which would then have improved classification accuracy on the target data. Deep architectures has enjoyed great success in the field of machine learning, and also inspired the extension of the CORAL method to such a framework. The Deep CORAL incorporates a weighted CORAL loss with the cross entropy loss, such that the supervised learning task is accompanied with the domain adaptation task.

can be trained on a similar dataset which has labels (from the source domain), while also adapting to the can help the supervised model

The transfer of knowledge from one domain to another. The thesis has specifically

data aqusition

Appendix A Definitions

The following appendix is a collection of relevant definitions.

Definition 4 (Bregman Divergence [21, p. 16, section 3.5.5], [22]) *The Bregman divergence is*

$$d_\psi(P,Q) = \sum_{t\in\mathcal{X}} \Delta_\psi\{p(x), q(x)\},$$

where the operator $\Delta_\Psi(a,b)$ *is*

$$\Delta_\psi\{a,b\} \coloneqq \psi(a) - \psi(b) - \psi'(b)(a-b),$$

with the differentiable real-valued convex function $\psi(x)$ *of a nonnegative argument. The nonnegative function* $\Delta_\Psi(a,b)$ *measures how much the convex function* ψ *deviates at* a *from its tangent at* b*.*

Definition 5 (Frobenius Norm [23, p. 55]) *The Frobenius norm* $f : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$ *is defined as*

$$f(\boldsymbol{A}) = \|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2}.$$

Appendix B Algorithms and Functions

Here relevant algorithms and other functions are presented.

B.1 Stochastic Gradient Descent

The following algorithms and descriptions are taken from the Deep Learning book [1]. For very large datsets, regular gradient descent is very slow, as it computes every gradient in the loss function. Stochastic gradient descent (SGD) instead computes a gradient estimate by randomly picking a set of m examples from the training set, as shown in Algorithm 2. Although rapidly

---

Algorithm 2: Stochastic gradient descent (SGD). Update at training iteration k

---

Require: Learning rate ϵ_k
Require: Initial parameter $\boldsymbol{\theta}$
while *Stopping criteria not met* **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(i)}\}_{i=1}^{m}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\boldsymbol{y}^{(i)}\right)$.
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$.
end

---

increasing the efficiency of learning, SGD may still converge slowly if the loss function has an poorly conditioned hessian matrix. The momentum algorithm [24] shown in Algorithm 3 accelerates learning by introducing a velocity parameter $\boldsymbol{v}$ which accumulates an exponentially decaying moving average of previously calculated gradients. The term momentum originates from a physical analogy, where the gradient is a force moving a particle through the parameter space. Since momentum is velocity times mass, unit mass is assumed, and the momentum parameter α determines how fast the contribution of previous gradients will decay.

Algorithm 3: Stochastic gradient descent (SGD) with momentum, Update at training iteration k

Require: Learning rate ϵ_k, momentum parameter α
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity v
while *Stopping criteria not met* **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(i)}\}_{i=1}^{m}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L\left(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\boldsymbol{y}^{(i)}\right)$.
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\hat{\boldsymbol{g}}$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$.
end

B.2 Cross-Entropy Loss Function

The following definitions are from [25, p. 57]. The uncertainty of a random variable X can be measured by its entropy

$$\mathbb{H}(X) \triangleq \sum_{k=1}^{K} \log p(X=k) \log_2 p(X=k). \tag{9}$$

If the logarithmic base is 2, the unit for entropy is *bits* (or Shannons, named after the American mathematician and engineer Claude Elwood Shannon, who laid the theoretical foundations of information theory), and *nats* for the natural log. Minimizing the entropy is equal to minimizing the uncertainty of the random variable, meaning the outcome becomes more deterministic. To measure the similarity between two distributions p and q, the Kullback-Leiber (KL) divergence can be used:

$$\mathbb{KL}(p||q) \triangleq \sum_{k=1}^{K} p_k \log \frac{p_k}{q_k}, \tag{10}$$

where the summation is replaced by an integral for continuous probability density functions, which is known as the differential entropy. Expanding equation (10) yields

$$\mathbb{KL} = \sum_{k=1}^{K} p_k \left[\log p_k - \log q_k\right] = \sum_{k=1}^{K} p_k \log p_k - \sum_{k=1}^{K} p_k \log q_k = -\mathbb{H}(p) + \mathbb{H}(p,q), \tag{11}$$

where $\mathbb{H}(p,q)$ is the *cross-entropy*:

$$\mathbb{H}(p,q) \triangleq -\sum_{k=1}^{K} p_k \log q_k. \tag{12}$$

Minimizing The cross-entropy w.r.t. q is equal to minimizing the KL divergence.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[2] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[3] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[4] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[5] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[6] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[8] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 97–105, JMLR.org, 2015.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[11] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," 2017.

[12] C. H. Judd, "The relation of special training and general intelligence," *Educational Review*, vol. 36, pp. 28–42.

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[14] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[15] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 3, p. 406–413, 1955.

[16] E. Deadman, N. J. Higham, and R. Ralha, "Blocked schur algorithms for computing the matrix square root," in *Applied Parallel and Scientific Computing* (P. Manninen and P. Öster, eds.), (Berlin, Heidelberg), pp. 171–182, Springer Berlin Heidelberg, 2013.

[17] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 443–450, Springer International Publishing, 2016.

[18] "The cifar-10 dataset." https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2021-03-22.

[19] K. Turkowski, *Filters for Common Resampling Tasks*, p. 147–165. USA: Academic Press Professional, Inc., 1990.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[21] P. D. Grünwald and A. P. Dawid, "Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory," *The Annals of Statistics*, vol. 32, aug 2004.

[22] L. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967.

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, third ed., 1996.

[24] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[25] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, MIT Press, 2012.