Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2020

Classification of Clothing Attributes Across Domains

Denise Härnström

Master of Science Thesis in Electrical Engineering

Classification of Clothing Attributes Across Domains:

Denise Härnström

LiTH-ISY-EX--20/5276--SE

Supervisor: **Karl Holmquist**
ISY, Linköpings universitet
Johan Lind
NFC

Examiner: **Per-Erik Forssén**
ISY, Linköpings universitet

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Classifying clothing attributes in surveillance images can be useful in the forensic field, making it easier to, for example, find suspects based on eyewitness accounts. Deep Neural Networks are often used successfully in image classification, but require a large amount of annotated data. Since labeling data can be time consuming or difficult, and it is easier to get hold of labeled fashion images, this thesis investigates how the domain shift from a fashion domain to a surveillance domain, with little or no annotated data, affects a classifier.

In the experiments, two deep networks of different depth are used as a base and trained on only fashion images as well as both labeled and unlabeled surveillance images, with and without domain adaptation regularizers. The surveillance dataset is new and consists of images that were collected from different surveillance cameras and annotated during this thesis work.

The results show that there is a degradation in performance for a classifier trained on the fashion domain when tested on the surveillance domain, compared to when tested on the fashion domain. The results also show that if no labeled data in the surveillance domain is used for these experiments, it is more effective to use the deeper network and train it on only fashion data, rather than to use the more complicated unsupervised domain adaptation method.

Acknowledgments

I would like to thank everyone at NFC who helped me during this thesis work. I especially want to thank my supervisor Johan Lind, for interesting discussions and helpful suggestions on this work. I would also like to thank my examiner Per-Erik Forssén and supervisor Karl Holmquist at Linköping University, for their quick replies and feedback.

Linköping, February 2020
Denise Härnström

Contents

1 Introduction

Image classification is the task of assigning semantic class labels to images based on their visual content. Classifying clothes' attributes is important in several applications, for example shopping recommendations in the fashion industry and searching for suspects based on eyewitness accounts in the forensic field.

Deep convolutional neural networks (CNN) are often used to solve this task. This is a supervised machine learning method that requires a large amount of annotated data. Since labeling data can be time consuming or difficult, domain adaptation from a domain where labeled data is easy to get hold of, to a domain with little or no annotated data is of interest.

1.1 Background

The National Forensic Centre (NFC) is an organization within the Swedish police authority with a responsibility for forensics. This thesis is being conducted at NFC's section for information technology, specifically with the group that handles forensic image analysis.

NFC is working on a system that will be able to detect and classify clothing type and other clothing attributes automatically. This is of forensic interest for several reasons. Such a system will make it easier to collect statistics of clothing attributes at different times and places, in order to answer questions like how probable it is that a certain item of clothing is the same as one used in a crime. It can also be used to find suspects in surveillance video by filtering on semantic attributes or serve as features for tracking a person across different cameras.

This thesis aims to answer a few questions related to a sub-problem of automatically detecting and classifying clothes.

1.2 Problem Formulation

The aim of this thesis is to investigate classification of clothing patterns (e.g. stripes, dots) with the help of deep learning, and how the domain shift from a fashion domain to a surveillance domain affects a classifier.

Classification of clothing attributes in surveillance images can be used in a real world forensic application, but the problem with this domain is that it is difficult to obtain a sufficient amount of annotated data. Annotated fashion images are more accessible, but are different from surveillance images in that they are often of higher resolution and with cleaner background. In order to find out if it is worth spending resources on annotating a large amount of surveillance data, it is of interest to look at how this domain shift affects the classification results when there is no or very little annotated data in the surveillance domain.

Thus this thesis will investigate the following research questions:

- How is a model, trained to classify clothing patterns on fashion images, affected by the domain shift to a surveillance domain?
- Are there effective ways to adapt a model, trained to classify clothing patterns in fashion images, to the surveillance domain, when there is no, or very little annotated data in that domain?

1.3 Limitations

The possibilities of clothing attributes to classify are many and in order to limit the time having to annotate images, the clothing attributes have been limited to only a few pattern classes.

Because of time constraints and training data limitations, all models are initialized with pre-trained weights and then fine-tuned, instead of being trained from scratch.

There are many different domain adaptation methods, and trying them all out on this problem is not possible during this thesis work. The main focus is on testing a few methods that can easily be implemented in an already existing architecture, and trained the same as the existing architectures in order to compare the domain adaptation method to a baseline.

2

Theory and Related Work

This chapter aims to give some background theory and a short overview of work related to the method in chapter 3.

2.1 CNN Classifier

Convolutional Neural Networks (CNN) have been successfully used in image classification. The following section describes the main ideas related to the CNNs used in the experiments.

2.1.1 Loss Function

The most common loss function for a supervised classification problem is the negative log-likelihood, or the cross entropy between the training data and the model distribution [8].

Let X be the set of training data $X = \{\mathbf{x}_i\}_{i=1}^{n}$ and $Y = \{\mathbf{y}_i\}_{i=1}^{n}$ the corresponding set of labels, where $\mathbf{y} \in \{0, 1\}^K$. Then the cross entropy loss is

$$L_c(X, Y) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log f_k(\mathbf{x}_i) \tag{2.1}$$

where $f_k(\mathbf{x})$ is the probability of assigning sample $\mathbf{x}$ to label k, often the output from a softmax function.

2.1.2 Optimization

Stochastic Gradient Descent (SGD) is one of the most common methods for optimizing a CNN. In gradient descent the loss function is minimized, by taking small steps in the direction of the negative gradient. In SGD an unbiased estimate of the gradient is calculated over a minibatch of training samples. The step size is determined by the learning rate parameter. Momentum is often used to speed up SGD, by accumulating a moving average of past gradients. The momentum parameter decides how quickly the previous gradients decay [8].

Other optimization methods commonly used are RMSProp and Adam. Both use adaptive learning rates in order to improve optimization [8].

2.1.3 Dropout

Dropout is a method where units of the hidden layers of a network are randomly removed, by setting their activations to zero, to reduce the generalization error. It can be seen as an ensemble method, where the ensemble consists of all subnetworks that are formed by removing units of the base network [8].

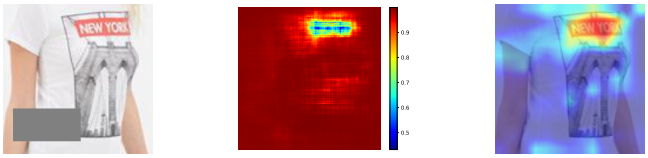
2.1.4 Batch Normalization

Batch normalization is used to improve training of very deep networks, by normalizing the network layers with mean and standard deviation calculated on a minibatch [8].

2.1.5 Visualization

To better understand CNNs, several methods to visualize the decisions they make exist. One approach is to visualize the regions of an image that are important when a CNN classifies it. Zeiler and Fergus [25] occlude part of an image, before using that as input to a CNN. Moving the occluder over the image and calculating the probability of the correct class for every position of the occluder, results in a heat map where the probability drops for the most important regions. An example of this can be seen in figures 2.1a and 2.1b.

A faster method to highlight important regions is Gradient-weighted Class Activation Mapping (Grad-CAM) [19]. With this method a similar heatmap, but with high values for important regions, is acquired with a single forward pass and a partial backward pass per image, see figure 2.1c.



(a) Occluded image *(b)* Probability of correct class *(c)* Grad-CAM

Figure 2.1: *Visualization of important class regions when classifying an image of class text with an AlexNet classifier, by occluding the image and calculating the probability of the correct class, (a) and (b), and by using Grad-CAM, (c).*

2.2 Evaluation Metrics

For a multiclass problem, the precision, recall and F1-score can be used as evaluation metrics, calculated per class. Precision is in this case the fraction of samples classified as the class by the model that are correct and recall is the fraction of actual samples from that class that were correctly classified [8]. The precision, P, recall, R, and F1-score, F, are then defined as:

$$P = \frac{tp}{tp + fp} \tag{2.2}$$

$$R = \frac{tp}{tp + fn} \tag{2.3}$$

$$F = \frac{2PR}{P + R} \tag{2.4}$$

where tp are true positives, fp false positives and fn false negatives.

2.3 Domain adaptation

Standard supervised learning methods assume that the training data and test data are drawn from the same distribution. If not, this distribution change or domain shift between training and test data might degrade performance. This might be because of differences in perspective, illumination and background [20].

This domain shift problem is addressed in domain adaptation. Domain adaptation methods are used when a task (such as classification) is to be performed in a target domain, where there is not sufficient labeled training data, but there is sufficient labeled data in one or more source domains. If the training data in target domain is unlabeled it is called unsupervised domain adaptation, while if the training data in target domain is labeled it is called supervised domain adaptation. Semi-supervised domain adaptation means that there is both labeled and unlabeled training data in the target domain [23].

Deep domain adaptation are deep learning architectures designed for domain adaptation. Wang and Dang [23] describes several different types of deep domain adaptation. Discrepancy-based methods, where the difference between target and source domains are made smaller by training a deep network on both source and target data based on some criterion, has been studied the most and is also the focus of this master thesis.

2.3.1 Domain Regularizers

While fine-tuning a deep network, the goal is in domain adaptation to learn domain invariant features. One way to do this is to insert a domain adaptation regularizer or loss to the total criterion to be optimized. For example, Correlation Alignment (CORAL) [21], minimizes the domain shift by aligning the second-order statistics of source and target distribution, while the Maximum Mean Discrepancy has the possibility to capture all the orders of statistic moments [23].

The following regularizers are used in the experiments described in chapter 3.

Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) [1] is a metric for comparing the distributions between two datasets by looking at the distance between the mean embeddings of the distributions. The MMD is defined by a feature map $\phi : X \mapsto \mathcal{H}$, where $\mathcal{H}$ is a reproducing Hilbert space, as

$$MMD = \|E_p(\phi(x)) - E_q(\phi(y))\|_{\mathcal{H}} \tag{2.5}$$

where x and y are random variables with distributions p and q, and $\|\cdot\|_{\mathcal{H}}$ is the norm induced by the inner product $< \cdot, \cdot >_{\mathcal{H}}$ on $\mathcal{H}$ as $\|x\|_{\mathcal{H}} = \sqrt{< x, x >_{\mathcal{H}}}$.

By using the kernel trick, with the kernel $k(x, x') =< \phi(x), \phi(x') >$, the squared MMD can be written as

$$MMD^2 = E_p(k(x, x')) + E_q(k(y, y')) - 2E_{p,q}(k(x, y)) \tag{2.6}$$

where x and x' are independent random variables with distribution p, and y and y' are independent random variables with distribution q.

The MMD is used by [6] as a domain shift regularizer, by calculating the metric between data from a source domain and a target domain. It is then used along with a standard classification loss calculated on the labeled source data, to train a shallow network. Similarly Deep Domain Confusion [22] does the same, but fine-tuning a deeper network initialized with pre-trained weights.

Gretton et al. [10] shows that an unbiased estimate of the squared MMD can be computed in linear time as

$$MMD_l^2 = \frac{2}{m} \sum_{i=1}^{m/2} h((x_{2i-1}, y_{2i-1}), (x_{2i}, y_{2i})) \tag{2.7}$$

with

$$h((x_i, y_i), (x_j, y_j)) = k(x_i, x_j) + k(y_i, y_j) - k(x_i, y_i) - k(x_j, y_i) \tag{2.8}$$

The choice of kernel is important, and [11] introduces a method for optimizing kernel choice, by defining the kernel k as a linear combination of a set of d base kernels $\{k_u\}_{u=1}^{d}$. That is

$$k = \sum_{u=i}^{d} \beta_u k_u \tag{2.9}$$

The optimal kernel parameter $\boldsymbol{\beta}$ for the MMD can then be learned by maximizing the test power, which is the same as minimizing the type-II error. The type-II error occurs when not rejecting the false null hypothesis that the two samples, x and y, are drawn from the same distribution, i.e. a false positive. Minimizing the type-II error leads to a quadratic optimization problem,

$$\min_{\mathbf{M}^T\boldsymbol{\beta}=1, \boldsymbol{\beta}\geq\mathbf{0}} \boldsymbol{\beta}^T(\mathbf{Q} + \epsilon\mathbf{I})\boldsymbol{\beta} \tag{2.10}$$

where $\mathbf{M} = [M_1, M_2, ..., M_d]^T$ is the squared MMD for each kernel, $\mathbf{Q}$ is the covariance matrix of $\mathbf{M}$ and ϵ is a small penalty to stabilize the problem.

The linear time estimate of the squared multi-kernel MMD with optimal $\boldsymbol{\beta}$, is used by [18] on the last layers of a CNN for domain adaptation.

Entropy

In semi-supervised learning, a decision boundary is learned from both labeled and unlabeled samples. By assuming that the points of each class form a cluster, the unlabeled samples can help find the boundary of each cluster by moving the decision boundary to a low density region - as to not cut a cluster (i.e. move through a high density region) [2].

Minimizing the conditional-entropy of the class distribution of unlabeled data is shown by [9] to be beneficial when the classes have little overlap in semi-supervised learning. The entropy can be seen as a measure of class overlap and by minimizing it the class conditional probabilities are encouraged to be either one or zero, thus pushing the decision boundary to a low density region.

Entropy minimization is later used by [18] in domain adaptation. The idea is that the classifier will generalize better to target data, if the decision boundary lies in low density regions of the target domain, and the entropy is therefore calculated on the unlabeled target training data.

Let $X = \{\mathbf{x}_i\}_{i=1}^{n}$ be a set of unlabeled training data, and $f_k(\mathbf{x})$ the probability of assigning sample $\mathbf{x}$ to label k, then the entropy loss is defined as

$$H(X) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} f_k(\mathbf{x}_i)\log f_k(\mathbf{x}_i) \tag{2.11}$$

2.4 Clothing Attributes

Classification of clothing attributes has previously been studied both in fashion applications, e.g. in shop retrieval or consumer-to-shop retrieval, and as a way to describe people for surveillance applications [3, 17].

In [17], the DeepFashion dataset and FashionNet are introduced. DeepFashion includes both online store images and consumer images and each image has a bounding box with one clothing type category and several clothing attributes (e.g. pattern, style) and clothing landmarks (e.g. left/right sleeve end). FashionNet is a deep model trained with DeepFashion data, in order to predict fashion landmarks and attributes simultaneously. DeepFashion2 [5] is similar to DeepFashion but also contains per-pixel masks and several bounding boxes per image.

Similar to this master thesis, [3] collects two datasets: fashion and surveillance, and uses a two stream network based on AlexNet along with a cost function to learn clothing attributes in both domains. The cost function is calculated between samples from the two domains and is based on a correlation, which in a supervised domain adaptation setting is the label similarity and for unsupervised domain adaptation is the visual similarity.

Other surveillance datasets are RAP [15, 16], which is labeled with attributes like color but not pattern, and PETA [4] which is also labeled with attributes, including two different patterns.

3

Method

In this chapter the methods used to answer the research questions specified in section 1.2 are presented.

3.1 System Overview

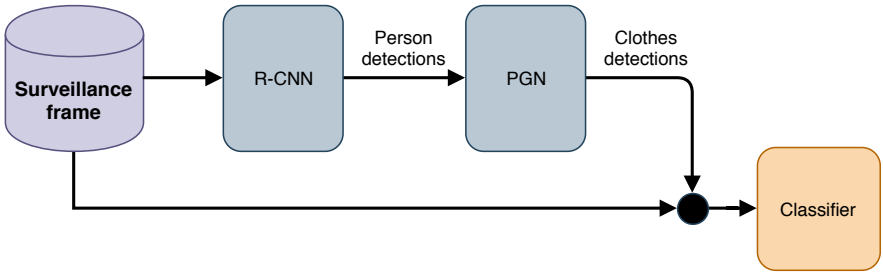


Figure 3.1: *Overview of system*

A graph of the system can be seen in figure 3.1. The task of the system is to classify clothing patterns in surveillance video. In order to do this, each surveillance frame is fed through a pre-trained Mask R-CNN [13] in order to detect persons. Each person detection is fed through a pre-trained Part Grouping Network [7], that segments out different parts of a person. A bounding box is created from each segment that belongs to a clothing type, and it is this bounding box cutout from the original surveillance frame that is classified. The training procedure of the classifier is explained in later sections.

3.2 Datasets

In order to investigate the domain shift, data from two different domains are used: the *fashion domain* and the *surveillance domain*. Each dataset is randomly divided into 70 % training samples, 15 % validation samples and 15 % testing samples per class. Each sample belongs to one of six pattern classes: checked, dotted, floral, solid, striped and text.

3.2.1 Fashion Domain

The fashion domain images are a subset of the DeepFashion dataset [17]. DeepFashion contains over 800,000 fashion images including both online store images and consumer images, labeled with 1,000 attributes. The attributes are annotated automatically with meta-data collected along with each image, and not annotated by humans. This means that the attributes are unbalanced and might contain errors. In order to deal with this, images with DeepFashion attributes matching any of the six clothing patterns classes described above, are sorted manually. In cases when there are many samples of a DeepFashion attribute, only a random subset of the samples is extracted.

Most clothing pattern classes consist of samples with one of several different DeepFashion attributes, as can be seen in table 3.1. The class solid consists of samples from DeepFashion that have not been labeled with any texture related attribute or any other attribute matching any of the clothing pattern classes.

This results in a total of 22,504 fashion images. The number of samples per class can be seen in figure 3.2 and some random examples of fashion images from each class can be seen in figure 3.4.

Table 3.1: *DeepFashion attributes to class mapping.*

Class	DeepFashion attribute(s)
checked	checked, checkered, grid, grid print, plaid
dotted	dot, dots, dotted, polka dot
floral	floral
solid	*no texture attributes*
striped	striped
text	graphic, logo

3.2.2 Surveillance Domain

The images in the surveillance domain are collected from 12 different surveillance cameras in public places. In order to get as much variation in the data as possible, the videos are captured at different angles, places; both indoors (metro stations) and outdoors (streets and under roof), and at different times during the year. The outdoor videos are both in sunny and cloudy weather.

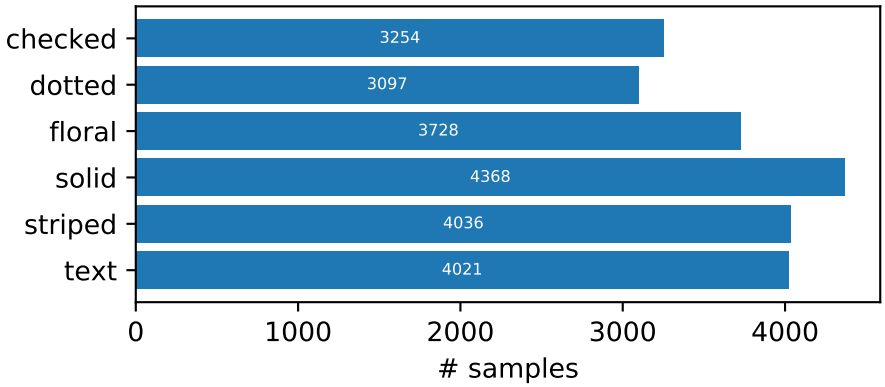


Figure 3.2: *Number of samples per class in the fashion domain*

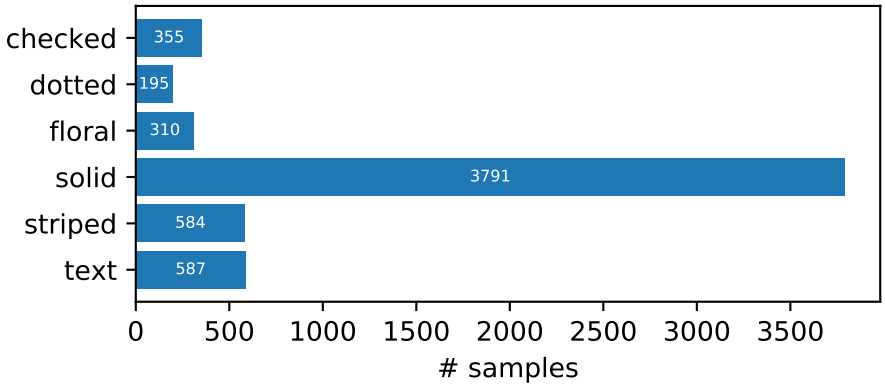


Figure 3.3: *Number of samples per class in the surveillance domain*

Each video is one hour long and is fed through a Mask R-CNN [13] at an interval of 10, 15 or 20 seconds in order to detect persons. All person detections are saved as cut out images of the detections, and the images that are deemed too small to be able to see clothing patterns are not used. The bounding boxes of clothes are automatically created by using a Part Grouping Network [7]. These bounding boxes are then manually labeled as one of the six clothing patterns, or not used if none of the classes apply.

This results in a total of 5,822 annotated images of clothes from surveillance videos, with a highly unbalanced number of samples per class, as can be seen in figure 3.3.

From the example images in figure 3.5 it is clear that the surveillance domain is different from the fashion domain because it for example has busier backgrounds, is captured at different angles and has more varying lighting conditions.

Figure 3.4: *Example images in the fashion domain*

Figure 3.5: *Example images in the surveillance domain*

3.3 Model Architecture

In the experiments, two different model architectures are used as a base. AlexNet with eight layers and the deeper ResNet50 with 50 layers. For the domain adaptation experiments they are used as a base for a domain adaptation network architecture.

3.3.1 AlexNet

AlexNet [14] won the ImageNet challenge in 2012. It contains eight layers: first five convolutional layers and then three fully connected layers, see figure 3.6.

It uses max pooling, ReLU activation and dropout regularization.

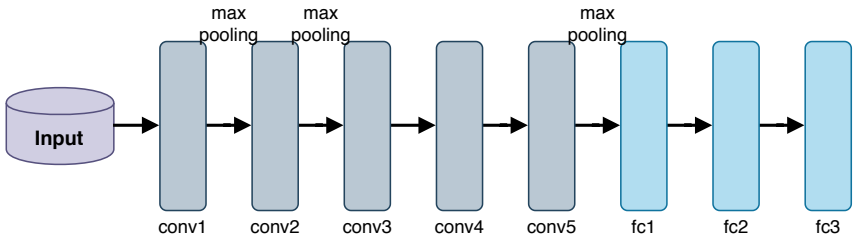


Figure 3.6: *AlexNet architecture*

3.3.2 ResNet50

Very deep convolutional networks have a problem with degrading accuracy, not caused by overfitting, but instead because they are more difficult to optimize. Deep Residual Networks (ResNet) was proposed to solve this degradation problem of very deep convolutional neural networks, by introducing residual mappings, or shortcut connections that can skip one or more layers [12].

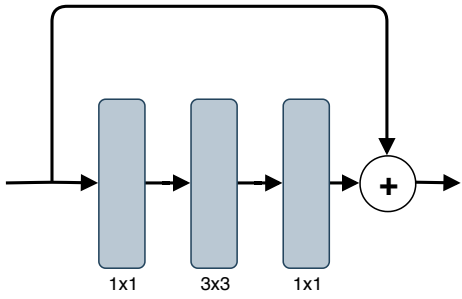


Figure 3.7: *ResNet50 building block*

ResNet50 has a total of 50 layers. It is made from a bottleneck building block, repeated differently for each main layer. Each bottleneck building block consists of 3 layers: 1x1, 3x3 and 1x1 convolutions, as can be seen in figure 3.7. This building block is then repeated to make up the model, see figure 3.8. The first main layer consists of a convolution layer and max pooling. The second main layer consists of 3 building blocks. The third main layer consists of 4 blocks. The fourth main layer consists of 6 blocks. And the fifth main layer consists of 3 blocks. The last layer is a fully connected layer with output equal to the number of classes.

After every convolution, batch normalization is used. ReLU is used as activation, and dropout is not used.

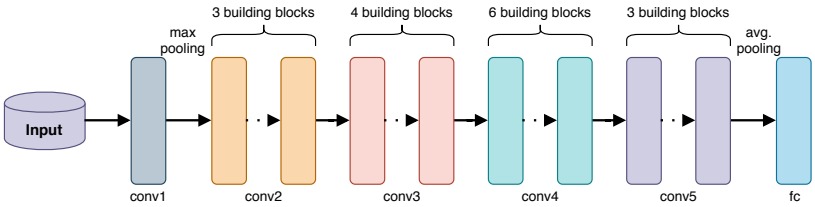


Figure 3.8: ResNet50 architecture

3.3.3 Domain Adaptation Network

The domain adaptation network is a CNN architecture that is used to learn domain invariant features, based on the architectures of Deep Domain Confusion [22] and Deep adaptation Networks [18]. As a base network AlexNet and ResNet50 described above are used.

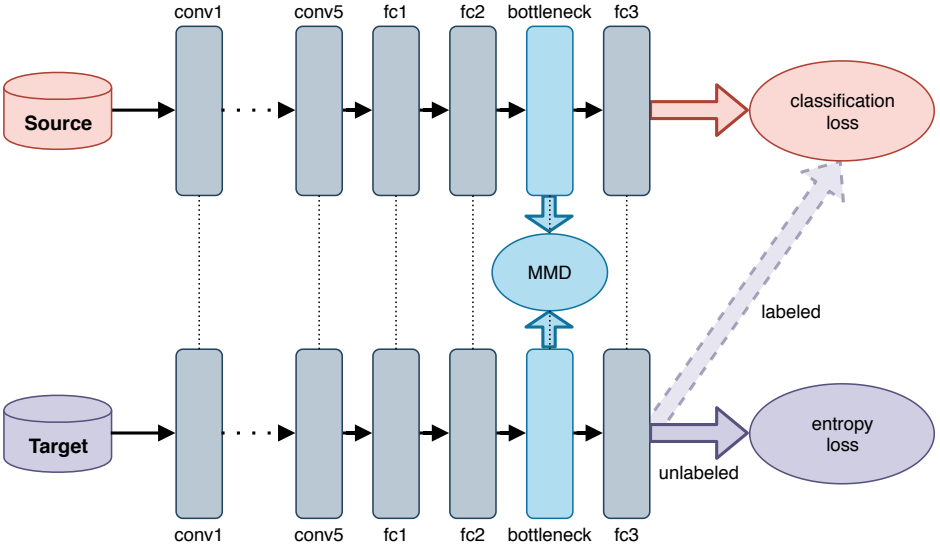


Figure 3.9: Domain adaptation network based on AlexNet

The architecture with AlexNet as a base can be seen in figure 3.9, but the domain adaptation networks are built on the same principle no matter what base network is used. It consists of two identical base networks, one for source and one for target data, with shared weights. Before the last fully connected layer is a fully connected bottleneck layer of size 256, as in [22], for which the output is used to calculate the MMD between the two domains. The classification loss is calculated for all labeled samples and as in [18] the entropy loss is calculated for all unlabeled samples.

This gives the total loss:

$$L = L_c(X_L, y) + \lambda MMD^2(X_S, X_T) + \gamma H(X_U) \tag{3.1}$$

where X_L is labeled samples, X_S is source or fashion samples, X_T is target or surveillance samples and X_U is unlabeled samples. For an unsupervised method, all labeled samples are source samples and all target samples are unlabeled. In a semi-supervised setting some of the target samples are labeled.

3.4 Training

The training procedure of the models are explained in this section.

3.4.1 Setup

Optimization is done with mini-batch gradient descent with a batch size of 18 and momentum 0.9. Because of the data imbalance, each batch is created by taking an equal number of samples per class, and during training the training data is undersampled: an epoch is done training when all the data from the smallest class has no more data. In each epoch the training samples are randomly shuffled, making it equally likely for each sample to be drawn.

For the domain adaptation, data is sampled from both the fashion and the surveillance domains. Using a batch size of n, n samples are sampled from the fashion data as described above and n samples are randomly sampled from the surveillance data. An epoch is done training when all the data from the smallest class in the fashion domain has been sampled. Because the surveillance dataset contains fewer samples, the surveillance data is oversampled and the model will use an equal amount of samples from both domains during training.

3.4.2 Preprocessing Data

Before a sample is fed to the network, it is resized to have a width and height of 224 pixels. For training, the data is augmented by randomly cropping the image and then doing resizing. Training data is also augmented by randomly doing a horizontal flip and randomly doing a perspective transform. The RGB values of the samples are normalized with mean and standard deviation of ImageNet.

When doing domain adaptation, the surveillance samples are also augmented by randomly rotating of a maximum of 15 degrees and randomly changing the brightness, contrast and saturation of the image. This is done because there are fewer surveillance samples than fashion samples.

All data augmentation is done with Torchvision standard transform functions [1], see the documentation for more information.

[1] https://pytorch.org/docs/stable/torchvision/transforms.html

3.4.3 Fine-tuning

Training a model from scratch is both time consuming and requires a large amount of annotated data, a fine-tuning method to train a model is therefore used.

The model is initialized with weights pre-trained on ImageNet and the last fully connected layer is changed to a new fully connected layer that is randomly initialized and has the same output size as number of classes. Since the first layers are more general and later layers more specific to the task [24] the idea is to transfer the more general features, by keeping those layers frozen and fine-tune the more specific features by continuing to train those layers.

Both of the models are fine-tuned in stages. First only the last layer is trained with a fixed learning rate for a number of epochs. Then the model at the epoch with the lowest validation loss has some more of the last layers trained with a lower learning rate. This is done several times. Each time taking the epoch with the lowest validation loss, unfreezing some more layers and continuing training with a lower learning rate than before.

The layers being trained, the number of epochs and the learning rate can be seen in table 3.2 for AlexNet and table 3.3 for ResNet50.

When the training is done, the model at the epoch with the lowest validation loss is chosen as the best model, and evaluated on previously unseen test data.

Table 3.2: *Fine-tuning approach for AlexNet*

Iteration	Learning rate	# Epochs	Layers trained
1	10^{-3}	5	FC3
2	10^{-4}	10	FC1-FC3
3	10^{-5}	20	Conv4-Conv5, FC1-FC3
4	10^{-6}	30	Conv2-Conv5, FC1-FC3

Table 3.3: *Fine-tuning approach for ResNet50*

Iteration	Learning rate	# Epochs	Layers trained
1	10^{-3}	5	FC
2	10^{-4}	10	FC, Layer 5
3	10^{-5}	20	FC, Layers 4-5
4	10^{-6}	30	FC, Layers 3-5

3.5 Experiments

In order to answer the questions in section 1.2 the following experiments are conducted.

3.5.1 Baseline: Fine-tuning in Fashion Domain

As a baseline, both AlexNet and ResNet50 are trained as described in section 3.4 with training and validation data from the fashion domain. To investigate how the shift from fashion to surveillance domain affects the classifiers, they are tested on data from the fashion and the surveillance domains separately.

3.5.2 Fine-tuning in Fashion and Surveillance Domains

As a simple way of improving the classifiers, the models are trained as described in section 3.4, with training and validation data from both the fashion and the surveillance domains. The new training and validation sets are created simply by merging the datasets from the two domains, and using the same data augmentation and sampling procedure as for the baseline. The models are then evaluated in each domain separately.

3.5.3 Domain Adaptation Method

In order to investigate if it is possible to improve classification in the surveillance domain by minimizing the domain discrepancy, the domain adaptation networks with AlexNet and ResNet50 as base are fine-tuned with the fashion data as source data and the surveillance data as target data, as described in section 3.4. For AlexNet the hyperparameters are tuned on labeled target validation data and can be seen in figure 3.4. The same parameters are then used for ResNet50.

Similar to [18], the MMD is a multi-kernel MMD with m Gaussian kernels with a varying bandwidth $[2^{-m/2}\sigma, 2^{m/2}\sigma]$, with a multiplicative step size of 2 and with σ as the median pairwise squared distances on the training data minibatch. Since calculating the optimal β for the MMD did not seem to improve the result on validation data, a fixed β is used to speed up calculations.

Table 3.4: *Parameters for domain adaptation method.*

Parameter	**Value**
λ	0.01
γ	0.01
Number of kernels	15
β	[1, 1, ..., 1]
Bottleneck layer width	256

To see the effects of the MMD and entropy losses, the domain adaptation networks are trained in three ways: with only the MMD loss, with only the entropy loss and with both losses. The models are then tested on previously unseen test data in the surveillance domain.

4

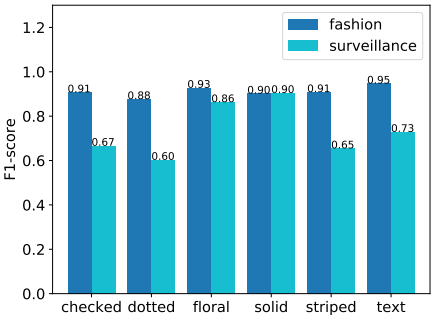
Results

4.1 Baseline: Fine-tuning in Fashion Domain

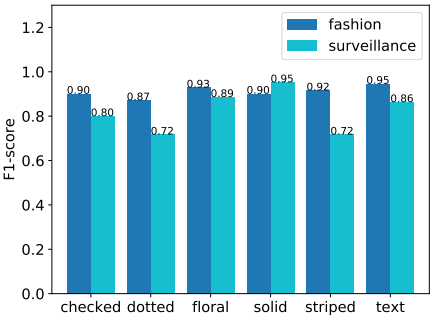
The F1-scores for each class on test data from the fashion and surveillance domains are plotted in figures 4.1a and 4.1c.

4.2 Fine-tuning in Fashion and Surveillance Domains

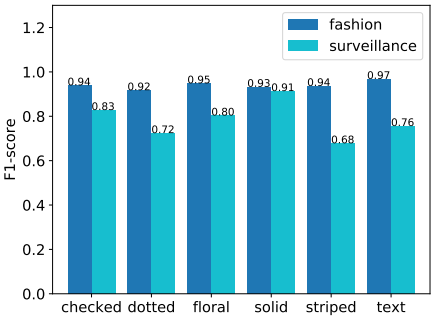
The F1-scores for each class on test data from the fashion and surveillance domains are plotted in figures 4.1b and 4.1d.



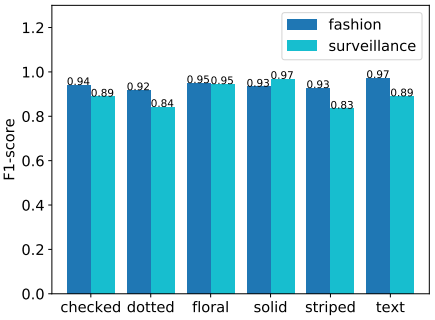
(a) AlexNet fine-tuned on fashion images.



(b) AlexNet fine-tuned on fashion and surveillance images.



(c) ResNet50 fine-tuned on fashion images.



(d) ResNet50 fine-tuned on fashion and surveillance images.

Figure 4.1: *F1-score per class for models trained and tested on images from different domains.*

4.3 Domain Adaptation Method

The average precision, recall and F1-score for unsupervised domain adaptation can be seen in table 4.1 with AlexNet as base network and in table 4.2 for ResNet50 as base network. The scores for semi and fully supervised domain adaptation with AlexNet as base network can be seen in tables 4.3 and 4.4. Baseline for unsupervised domain adaptation is the model fine-tuned on only fashion images and tested on surveillance images, that can also be seen in figures 4.1a and 4.1c. For fully supervised domain adaptation, baseline is the model fine-tuned on all fashion and surveillance images and tested on surveillance images, that can also be seen in figures 4.1b and 4.1d. For semi supervised domain adaptation, baseline is the model fine-tuned on all fashion images and 10 surveillance images.

The F1-score for each class, for models trained with different combinations of domain adaptation regularizers, on test data from the surveillance domain are plotted in figure 4.2.

Table 4.1: *Average metrics for unsupervised domain adaptation AlexNet*

Method	Precision	Recall	F1-score
Baseline	70.80	77.69	73.69
MMD	71.65	**79.05**	74.62
Entropy	**78.80**	73.12	75.34
MMD+Entropy	77.73	77.51	**77.27**

Table 4.2: *Average metrics for unsupervised domain adaptation ResNet50*

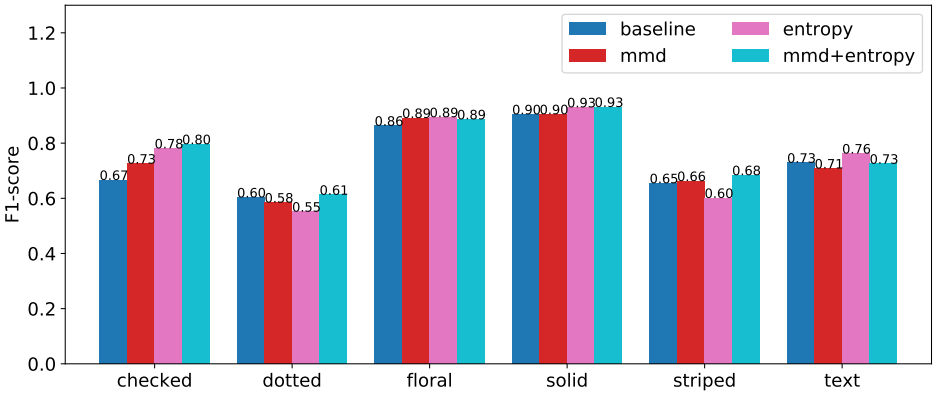
Method	Precision	Recall	F1-score
Baseline	**76.73**	**81.53**	**78.40**
MMD	69.35	81.12	72.90
Entropy	65.66	78.45	69.86
MMD+Entropy	67.40	81.06	71.75

Table 4.3: *Average metrics for semi-supervised domain adaptation AlexNet (10 samples per class)*

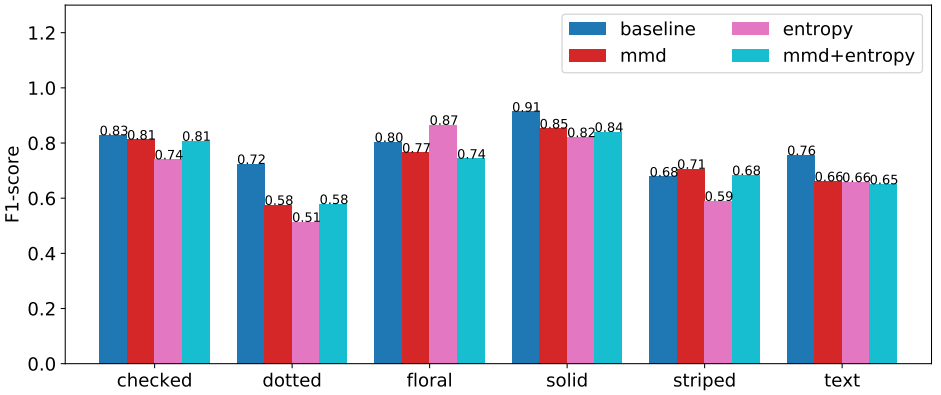
Method	Precision	Recall	F1-score
Baseline	74.75	76.81	75.61
MMD+Entropy	**79.87**	**77.94**	**78.70**

Table 4.4: *Average metrics for fully supervised domain adaptation AlexNet*

Method	Precision	Recall	F1-score
Baseline	85.48	**79.71**	82.27
MMD	**88.12**	79.15	**83.10**



(a) *AlexNet*



(b) *ResNet50*

Figure 4.2: *F1-score per class for different unsupervised domain adaptation methods, tested on surveillance images.*

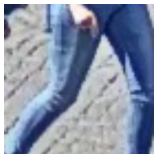
4.4 Classification Examples

In figure 4.3 the classification results on different surveillance test samples can be seen. AN is AlexNet fine-tuned on fashion samples, AN-D is AlexNet trained with the MMD and entropy regularizers with all surveillance samples unlabeled, AN-D-S is AlexNet trained with the MMD regularizer and with all surveillance samples labeled. RN is ResNet50 fine-tuned on fashion data, RN-D is ResNet50 trained with the MMD and entropy regularizers with all surveillance samples unlabeled and RN-S is ResNet50 fine-tuned on fashion and surveillance data. No examples for ResNet50 trained with the MMD and entropy regularizers with all surveillance samples labeled are shown, because ResNet50 did not improve with the regularizers.

Examples of surveillance images misclassified by baseline models, as well as the results after the background has been manually set to zero, can be seen in figure 4.4.

To see what AlexNet bases its decision on when wrong, Grad-CAM visualizations are shown for some mistakes in figure 4.5, where the regions of the images most important for classifications are highlighted. Grad-CAM visualizations for solid samples with the background set to zero can be seen in figure 4.6.

The classification examples are further discussed in chapter 5.



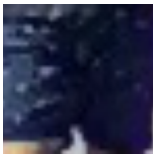
(a) True: solid
AN: checked
AN-D: ***solid***
AN-D-S: ***solid***
RN: text
RN-D: striped
RN-S: ***solid***

(b) True: solid
AN: checked
AN-D: ***solid***
AN-D-S: ***solid***
RN: ***solid***
RN-D: text
RN-S: ***solid***

(c) True: solid
AN: striped
AN-D: striped
AN-D-S: ***solid***
RN: striped
RN-D: text
RN-S: ***solid***

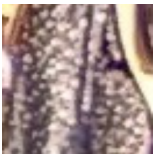
(d) True: solid
AN: striped
AN-D: ***solid***
AN-D-S: ***solid***
RN: striped
RN-D: text
RN-S: ***solid***

(e) True: solid
AN: floral
AN-D: ***solid***
AN-D-S: ***solid***
RN: floral
RN-D: text
RN-S: ***solid***



(f) True: dotted
AN: solid
AN-D: solid
AN-D-S: solid
RN: ***dotted***
RN-D: ***dotted***
RN-S: ***dotted***

(g) True: text
AN: dotted
AN-D: dotted
AN-D-S: ***text***
RN: ***text***
RN-D: ***text***
RN-S: ***text***



(h) True: floral
AN: dotted
AN-D: checked
AN-D-S: solid
RN: ***floral***
RN-D: ***floral***
RN-S: ***floral***

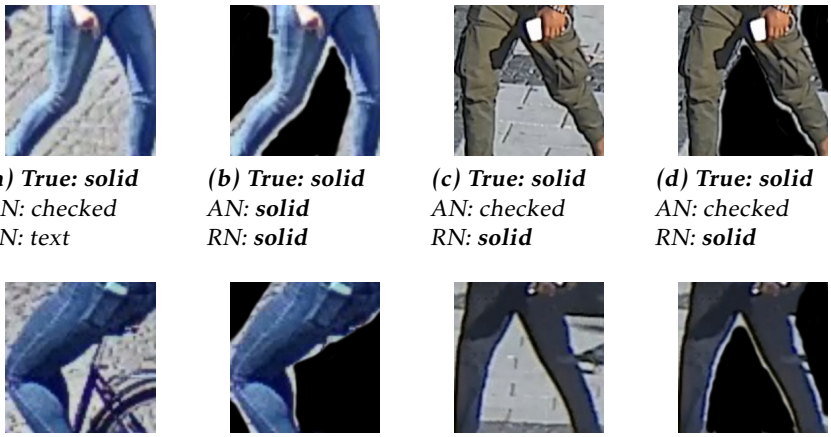
(i) True: checked
AN: striped
AN-D: ***checked***
AN-D-S: striped
RN: striped
RN-D: striped
RN-S: striped

(j) True: checked
AN: striped
AN-D: ***checked***
AN-D-S: ***checked***
RN: ***checked***
RN-D: ***checked***
RN-S: ***checked***

(k) True: striped
AN: checked
AN-D: checked
AN-D-S: checked
RN: checked
RN-D: ***striped***
RN-S: ***striped***

(l) True: striped
AN: checked
AN-D: ***striped***
AN-D-S: ***striped***
RN: checked
RN-D: checked
RN-S: ***striped***

Figure 4.3: *Example classification of different test samples done by AlexNet baseline (AN), AlexNet unsupervised domain adaptation (AN-D), AlexNet supervised domain adaptation (AN-D-S), ResNet50 baseline (RN) and ResNet50 finetuning with images from both domains (RN-S).*



(a) True: solid
AN: checked
RN: text

(b) True: solid
AN: ***solid***
RN: ***solid***

(c) True: solid
AN: checked
RN: ***solid***

(d) True: solid
AN: checked
RN: ***solid***

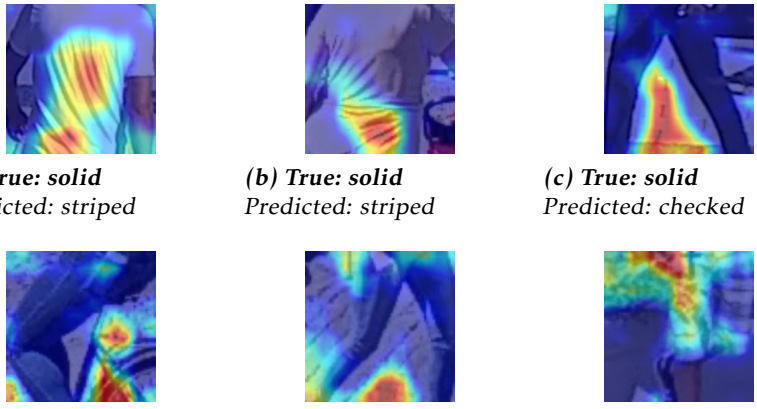
(e) True: solid
AN: checked
RN: checked

(f) True: solid
AN: ***solid***
RN: ***solid***

(g) True: solid
AN: checked
RN: text

(h) True: solid
AN: ***solid***
RN: ***solid***

Figure 4.4: *Classification results of original test samples and test samples with background manually set to 0.*



(a) True: solid
Predicted: striped

(b) True: solid
Predicted: striped

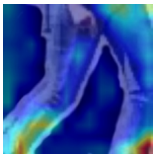
(c) True: solid
Predicted: checked

(d) True: solid
Predicted: checked

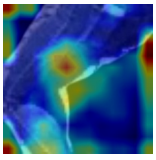
(e) True: solid
Predicted: checked

(f) True: solid
Predicted: floral

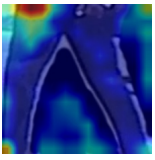
Figure 4.5: *Grad-CAM visualizations for AlexNet on incorrectly classified surveillance samples, highlighting the important regions for prediction.*



(a) ***True: solid***
Predicted: ***solid***



(b) ***True: solid***
Predicted: ***solid***



(c) ***True: solid***
Predicted: ***solid***

Figure 4.6: *Grad-CAM visualizations for AlexNet on solid samples with background set to 0, highlighting the important regions for prediction.*

5 Discussion

In this chapter the method, results and future work are discussed.

5.1 Method

In the following sections the datasets, model architecture and training are discussed.

5.1.1 Datasets

A problem with the datasets is that they are highly unbalanced: there are a different number of samples per class and also different number of samples per domain. The imbalance was for labeled samples (data from the fashion domain) handled during training by having each minibatch consist of an equal number of samples per class. This was not done in the surveillance domain during the unsupervised domain adaptation methods, simply because that would mean using the images' true labels during sampling, making the method not entirely unsupervised in the surveillance domain. The class imbalance was larger in the surveillance domain and this might have affected the results, by having the models learn the class with most samples (solid) better than the others. One way to deal with this could have been to assign the surveillance samples pseudo-labels and use these to do the equal sampling procedure between classes. This assignment can be done iteratively by using the pseudo-labels during the current epoch assigned by the model trained after the previous epoch, updating the pseudo-labels after every epoch. This was not tested because of time constraints.

There is a significantly higher number of samples in the fashion domain than the surveillance domain. When training the base networks on labeled data from

both domains, this was not accounted for and therefore more data from the fashion domain than the surveillance domain was used during training. Still, the results did improve, so this might not have had too much of an affect. Oversampling of the surveillance data, having the models see more of the surveillance data during training, might improve the results even more. Although it might also lead to overfitting on the surveillance training data since it has few samples for all classes but solid.

In the domain adaptation methods, this imbalance between the number of samples of the two domains was taken care of by oversampling in the surveillance domain, making the models see an equal amount of fashion and surveillance images during training. The surveillance images were also augmented more during training, to try and avoid overfitting to the surveillance training data.

While evaluating the models on the surveillance domain, the result might be misleading compared to the result on the fashion domain, simply because there are fewer testing samples for all classes except solid. The smallest class, dotted, has only 29 testing samples and therefore one sample has more impact on the total result for that class than for example solid, with 568 test samples, has on its class result. On the other hand, the solid samples have more impact on the total accuracy, and having a model be very good at classifying solid but bad at the other classes would still lead to high accuracy because of the imbalance. This is the reason why accuracy, recall and F1-score are presented in chapter 4 both per class and per class average, while total accuracy is not.

Another problem with the datasets is that there are some images that contain several different patterns, but only one pattern is labeled. This might both affect the models during training and affect the result during testing. Is it really an error when an image containing a striped shirt with flowers labeled as striped is classified as floral? This could be handled by having multiple labels, labeling it both as striped and floral. If there are two different garments with two different patterns in the image, it could be handled by training the models to recognize both clothing patterns and the type of garment (e.g. solid t-shirt and striped shirt).

Even though the datasets are annotated by a person, there are still mistakes in the labels. Some are just plain wrong (e.g. solid being labeled as checked), and then there are some examples where, for example, flowers in a dotted pattern has been labeled as 'dotted'. There are also samples that have the pattern in only a small part of the image that is sometimes cut out during training and/or testing. Which for example will make the model see a solid sample that is labeled as text. This is, of course, a problem during training, but does not have to be too much of a problem since the cutouts are random and the models should see the pattern more than not. It is a bigger problem if this happens while testing with only a few samples, which may make the results misleading.

5.1.2 Model Architecture

For the domain adaptation network Maximum Mean Discrepancy is used as a domain regularizer, calculated between two layers, unlike [18] where it is calcu-

lated for multiple layers. The reason for not doing the same in this thesis is partly because of speed, it takes longer to calculate it and sum it for several layers, and because the effect of the loss should transfer to earlier layers while updating the weights through back propagation. Perhaps using it on the last fully connected layer would make a difference in performance.

A potential problem with MMD is that from the theory it should be zero if the two distributions (source and target) are the same. But having each domain as the same distribution might not be a good enough approximation. In this case the source domain consists of fashion images both from online stores and consumer images. Perhaps these cannot be approximated as being drawn from the same distribution.

5.1.3 Training

While training the baseline, the training samples were augmented by doing random perspective transformations. This was done because it seemed reasonable that this could improve results on the surveillance data, because of the fact that one difference between the domains are the perspective. This might have closed the gap between the two domains, resulting in a much smaller difference between results of the baseline and the domain adaptation method, but also having the domain adaptation regularizer not work as expected. It would be interesting to see how not using this kind of data augmentation would affect both the baseline and the domain adaptation. Although it might just end up worse because of overfitting.

5.2 Results

Both AlexNet and ResNet50 has an average F1-score per class of over 90%. ResNet50 is slightly better than AlexNet, which is not very surprising since it has been shown to outperform AlexNet on ImageNet. Looking at figures 4.1a and 4.1c it is clear that both models trained on only fashion images has a decline in F1-score for almost all classes (solid being the only class with none or very little change in result) when testing on surveillance images. The decline is less for ResNet50, especially for the classes checked and dotted, which might mean that this model has managed to learn features that are more transferable across the two domains.

The domain shift has degraded the performance as expected. But what is it more specifically about the surveillance domain that makes the classifiers perform worse there? Figure 4.3 contains examples of typical classification mistakes. Just looking at these and similar examples, it seems that lighting conditions and background have a bigger impact on performance than the perspective. This might be because that the patterns do not change much with the perspective or perhaps because that the samples were augmented with random perspective transforms during training.

In figure 4.4 classification results of original test samples and test samples, with background manually set to zero, from the solid class can be seen. This was

done to figure out if the background affected the misclassifications. For all but one example, setting the background to zero resulted in the classifiers correctly classifying them as solid. This implies that it is indeed the background that is the reason for the solid examples being classified as checked or text. In figures 4.5c, 4.5d and 4.5e, the background is highlighted showing that it is this region that is important when AlexNet makes the decision to classify these solid examples as checked instead. In figure 4.6 Grad-CAM visualizations for the solid samples without background can be seen, and it is mostly the pants being highlighted, which then means that setting the background to zero made the decision be based on the garment and not the background. Except for in figure 4.6b, where some of the background is also highlighted.

The solid samples in figures 4.3c and 4.3d were by AlexNet predicted as striped. In figures 4.5a and 4.5b the regions that were important in this prediction can be seen. It is the draped parts of the garments that under these lighting conditions look like stripes that are highlighted. This seems reasonable, and indicates that the lighting conditions does play a vital role in the misclassifications.

Several of these mistakes are corrected with the domain adaptation methods, as can be seen in figure 4.3. What is perhaps most interesting is that there are several examples when unsupervised domain adaption with AlexNet as base network does a correct prediction. This means that at least for these examples, training with unlabeled surveillance samples did improve the results.

Looking at the results in figure 4.1 it is possible to see that the F1-score for surveillance images improved for all classes when training on both fashion and surveillance images for both models, while the F1-score for the fashion images did not change more than a little. This implies that the improvement in the surveillance domain is not because that the number of training samples increased, but because of the fact that the models saw samples from the same domain during training.

Unsupervised domain adaptation did not improve the results for ResNet50. This could be because the correct hyperparameters were not found (since they were based on AlexNet), or it could mean that the domain regularizer was not in between the layers that would be optimal. It could also mean that the features ResNet50 learnt were already more domain invariant than for AlexNet, getting less of a benefit from the domain adaptation regularizer. It might also be because of the fact that ResNet50 is a much deeper network, and thus needs more data in the surveillance domain. In the original article that used both MMD and entropy regularizers [18] the result improved for both AlexNet and ResNet50, although they did use different data so it is difficult to compare. More experiments have to be done in order to tell why the adaptation method did not work with ResNet50 as a base network.

It is interesting to look at how the different domain regularizers affected the result when using AlexNet as a base, as can be seen in table 4.1. Using only entropy minimization the precision improves, while the recall gets worse. While using only MMD both precision and recall improves slightly. While using both, the recall is almost the same while precision improves, giving the best F1-score out of all of the unsupervised methods. It is difficult to say why this is, but it

does seem like the two losses together balance the precision and recall in this experiment.

Comparing the unsupervised methods for both models it is still the ResNet50 baseline that has the best average F1-score. Fine-tuning a network without any domain regularizer is easier to implement and requires no surveillance data at all, so this seems like the best choice when moving forward. Although the best thing to do would be to also use annotated surveillance data for training. If looking at the F1-score ResNet50 is the best, but if speed is important it is worth noting that AlexNet is faster.

5.3 Future Work

Since it is difficult to tell why the domain adaptation did not work very well on this problem, it would be interesting to do the same experiments but on different clothing attributes, such as color or garment type. Perhaps other attributes change more between the fashion and the surveillance domains, leading to different results, or perhaps the method does not work well between the fashion and surveillance domains for any task.

In some cases the models wrongly classified a pattern by looking at the background, instead of the clothes in the image. This is something that could be improved by doing detection instead of classification or learning garment type along with pattern (e.i 'striped t-shirt', instead of only 'striped'). Perhaps then it would learn to ignore the background patterns and instead focus on the clothing pattern.

The class imbalance in the surveillance domain might have affected the results. It would therefore be interesting to see if the results improve by using the same amount of samples per class for every minibatch. It is also of interest to figure out why the domain adaptation regularizers made ResNet50 perform worse. Using more data in the surveillance domain should improve the performance if it is because of too little data. More effort should also be put into tuning the hyperparameters to make sure that they are optimal for ResNet50.

In a real world scenario, the unlabeled images in the target domain might not fit into any of the classes in the source domain. It would therefore also be of interest to investigate if it is possible to improve the results by using unlabeled images from the target domain, not corresponding to any class in the source domain.

It would also be interesting to look at how the results are affected by adding more data from the surveillance domain, by training on differently sized parts of the data and comparing the results. Does more data always make the classifier better, or is there a point when adding more data does not improve the results?

It would also be interesting to look at other types of domain adaptation than the one used in the experiments, with the fashion domain as source and the surveillance domain as target.

6

Conclusion

This thesis has investigated how the domain shift from fashion to surveillance images affects classification of clothing patterns. In order to do this a fashion dataset was assembled from the existing DeepFashion dataset and a new surveillance dataset was collected and manually annotated. The data was then used to train different deep networks on only fashion images and on both fashion and surveillance images, with and without using the labels in the surveillance domain. The following questions are answered:

- *How is a model, trained to classify clothing patterns on fashion images, affected by the domain shift to a surveillance domain?*

  Both AlexNet and ResNet50 perform worse on the surveillance images, with ResNet50 generalizing slightly better to the surveillance domain. It seems that the biggest problem in the surveillance domain is the background and the lighting conditions: the classifiers sometimes confuse background, shadows or highlights as a different pattern.

- *Are there effective ways to adapt a model, trained to classify clothing patterns in fashion images, to the surveillance domain, when there is no, or very little annotated data in that domain?*

  The simplest and most effective way, among the ones tested, is to train the model with annotated samples from both domains. If no labeled images in the surveillance domain are used, using both entropy and MMD as domain regularizers slightly improves the performance when using AlexNet as base, but for these experiments it was more effective to simply use the deeper ResNet50 and only train it on fashion images when looking at the average F1-score.

Appendix

A

All Results

A.1 AlexNet

The following section contains the results for the methods with AlexNet as a base network.

A.1.1 AlexNet Trained on Fashion Domain

Table A.1: *AlexNet trained on fashion domain and tested on fashion domain*

Class	Precision	Recall	F1-score
checked	93.29	88.32	90.74
dotted	89.46	85.99	87.69
floral	92.23	93.38	92.80
solid	86.11	94.66	90.18
striped	93.37	88.43	90.83
text	94.40	95.02	94.71
Average	91.48	90.97	91.16

Table A.2: AlexNet trained on fashion domain and tested on surveillance domain

Class	Precision	Recall	F1-score
checked	58.57	77.36	66.67
dotted	55.88	65.52	60.32
floral	83.67	89.13	86.32
solid	94.44	86.80	90.46
striped	67.90	63.22	65.48
text	64.35	84.09	72.91
Average	70.80	77.69	73.69

Table A.3: AlexNet trained on fashion domain and tested on surveillance domain, top-2 scores

Class	Precision	Recall	F1-score
checked	81.03	88.68	84.68
dotted	81.48	75.86	78.57
floral	88.00	95.65	91.67
solid	96.57	94.19	95.37
striped	88.89	82.76	85.71
text	80.20	92.05	85.71
Average	86.03	88.20	86.95

A.1.2 AlexNet Trained on Both Domains

Table A.4: AlexNet trained on both domains and tested on fashion domain

Class	Precision	Recall	F1-score
checked	91.54	88.73	90.11
dotted	88.81	85.56	87.16
floral	91.94	93.92	92.92
solid	87.83	92.52	90.11
striped	94.08	89.26	91.60
text	93.38	95.85	94.60
Average	91.26	90.97	91.08

Table A.5: *AlexNet trained on both domains and tested on surveillance domain*

Class	Precision	Recall	F1-score
checked	85.11	75.47	80.00
dotted	79.17	65.52	71.70
floral	92.86	84.78	88.64
solid	93.40	97.18	95.25
striped	79.17	65.52	71.70
text	83.16	89.77	86.34
Average	85.48	79.71	82.27

Table A.6: *AlexNet trained on both domains, with 10 target samples per class, tested on surveillance*

Class	Precision	Recall	F1-score
checked	79.17	71.70	75.25
dotted	57.58	65.52	61.29
floral	82.98	84.78	83.87
solid	94.20	91.55	92.86
striped	64.37	64.37	64.37
text	70.19	82.95	76.04
Average	74.75	76.81	75.61

A.1.3 AlexNet Domain Adaptation

Table A.7: *AlexNet unsupervised domain adaptation with MMD loss tested on surveillance*

Class	Precision	Recall	F1-score
checked	63.38	84.91	72.58
dotted	52.78	65.52	58.46
floral	89.13	89.13	89.13
solid	95.15	86.27	90.49
striped	68.29	64.37	66.27
text	61.16	84.09	70.81
Average	71.65	79.05	74.62

Table A.8: *AlexNet unsupervised domain adaptation with entropy loss tested on surveillance*

Class	Precision	Recall	F1-score
checked	78.85	77.36	78.10
dotted	55.17	55.17	55.17
floral	97.44	82.61	89.41
solid	90.37	95.77	92.99
striped	76.79	49.43	60.14
text	74.19	78.41	76.24
Average	78.80	73.12	75.34

Table A.9: *AlexNet unsupervised training with MMD and entropy loss tested on surveillance domain.*

Class	Precision	Recall	F1-score
checked	78.18	81.13	79.63
dotted	57.58	65.52	61.29
floral	92.86	84.78	88.64
solid	93.13	93.13	93.13
striped	77.94	60.92	68.39
text	66.67	79.55	72.54
Average	77.73	77.51	77.27

Table A.10: *AlexNet semi-supervised training with MMD and entropy loss (10 labeled target samples per class), tested on surveillance domain.*

Class	Precision	Recall	F1-score
checked	81.63	75.47	78.43
dotted	63.33	65.52	64.41
floral	92.50	80.43	86.05
solid	93.54	94.37	93.95
striped	72.73	64.37	68.29
text	75.49	87.50	81.05
Average	79.87	77.94	78.70

Table A.11: *AlexNet fully supervised domain adaptation with MMD (all labeled target samples), tested on surveillance domain.*

Class	Precision	Recall	F1-score
checked	90.48	71.70	80.00
dotted	82.61	65.52	73.08
floral	95.00	82.61	88.37
solid	92.87	98.59	95.64
striped	82.19	68.97	75.00
text	85.56	87.50	86.52
Average	88.12	79.15	83.10

A.2 ResNet50

The following section contains the results for the methods with ResNet50 as a base network.

A.2.1 ResNet50 Trained on Fashion Domain

Table A.12: *ResNet50 trained on fashion domain and tested on fashion domain*

Class	Precision	Recall	F1-score
checked	97.57	90.57	93.94
dotted	92.54	90.95	91.74
floral	94.02	95.71	94.86
solid	89.37	97.56	93.28
striped	95.69	91.74	93.67
text	97.00	96.68	96.84
Average	94.37	93.87	94.06

Table A.13: *ResNet50 trained on fashion domain and tested on surveillance domain*

Class	Precision	Recall	F1-score
checked	89.13	77.36	82.83
dotted	72.41	72.41	72.41
floral	68.18	97.83	80.36
solid	94.37	88.56	91.37
striped	69.05	66.67	67.84
text	67.26	86.36	75.62
Average	76.73	81.53	78.40

Table A.14: *ResNet50 trained on fashion domain and tested on surveillance domain, top-2 scores*

Class	Precision	Recall	F1-score
checked	94.00	88.68	91.26
dotted	93.10	93.10	93.10
floral	82.14	100.00	90.20
solid	98.36	95.07	96.69
striped	87.50	88.51	88.00
text	83.84	94.32	88.77
Average	89.82	93.28	91.34

A.2.2 ResNet50 Trained on Both Domains

Table A.15: *ResNet50 trained on both domains and tested on fashion domain*

Class	Precision	Recall	F1-score
checked	95.18	93.03	94.09
dotted	92.36	91.16	91.76
floral	94.04	95.89	94.95
solid	91.25	95.57	93.36
striped	94.82	90.74	92.74
text	97.19	97.35	97.27
Average	94.14	93.96	94.03

Table A.16: *ResNet50 trained on both domains and tested on surveillance domain*

Class	Precision	Recall	F1-score
checked	95.65	83.02	88.89
dotted	85.71	82.76	84.21
floral	95.56	93.48	94.51
solid	95.70	98.06	96.87
striped	89.47	78.16	83.44
text	86.17	92.05	89.01
Average	91.38	87.92	89.49

A.2.3 ResNet50 Unsupervised Domain Adaptation

Table A.17: *ResNet unsupervised training with MMD loss tested on surveillance*

Class	Precision	Recall	F1-score
checked	90.70	73.58	81.25
dotted	51.35	65.52	57.58
floral	63.77	95.65	76.52
solid	97.08	76.06	85.29
striped	62.28	81.61	70.65
text	50.92	94.32	66.14
Average	69.35	81.12	72.90

Table A.18: *ResNet unsupervised domain adaptation with entropy loss, tested on surveillance*

Class	Precision	Recall	F1-score
checked	72.73	75.47	74.07
dotted	43.90	62.07	51.43
floral	82.35	91.30	86.60
solid	96.22	71.65	82.14
striped	47.86	77.01	59.03
text	50.93	93.18	65.86
Average	65.66	78.45	69.86

Table A.19: *ResNet unsupervised training with MMD and entropy loss tested on surveillance*

Class	Precision	Recall	F1-score
checked	86.96	75.47	80.81
dotted	46.81	75.86	57.89
floral	62.69	91.30	74.34
solid	96.15	74.65	84.04
striped	61.47	77.01	68.37
text	50.31	92.05	65.06
Average	67.40	81.06	71.75

Bibliography

[1] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alexander J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. In *Proceedings 14th International Conference on Intelligent Systems for Molecular Biology 2006, Fortaleza, Brazil, August 6-10, 2006*, pages 49–57, 2006.

[2] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[3] Qiang Chen, Junshi Huang, Rogério Schmidt Feris, Lisa M. Brown, Jian Dong, and Shuicheng Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. In *CVPR*, pages 5315–5324. IEEE Computer Society, 2015.

[4] Yubin Deng, Ping Luo, Chen Change Loy, and Xiaoou Tang. Pedestrian attribute recognition at far distance. In *ACM Multimedia*, pages 789–792. ACM, 2014.

[5] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *CVPR*, 2019.

[6] Muhammad Ghifary, W. Bastiaan Kleijn, and Mengjie Zhang. Domain adaptive neural networks for object recognition. 2014.

[7] Ke Gong, Xiaodan Liang, Yicheng Li, Yimin Chen, Ming Yang, and Liang Lin. Instance-level human parsing via part grouping network. In *ECCV (4)*, volume 11208 of *Lecture Notes in Computer Science*, pages 805–822. Springer, 2018.

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

[9] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Neural Information Processing Systems (NIPS)*, pages 529–536, 2004.

[10] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13: 723–773, March 2012.

[11] Arthur Gretton, Bharath K. Sriperumbudur, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, and Kenji Fukumizu. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1214–1222, 2012.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[15] Dangwei Li, Zhang Zhang, Xiaotang Chen, Haibin Ling, and Kaiqi Huang. A richly annotated dataset for pedestrian attribute recognition. *CoRR*, abs/1603.07054, 2016.

[16] Dangwei Li, Zhang Zhang, Xiaotang Chen, and Kaiqi Huang. A richly annotated pedestrian dataset for person retrieval in real surveillance scenarios. *IEEE Transactions on Image Processing*, 28(4):1575–1590, 2019.

[17] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Micheal I. Jordan. Transferable representation learning with deep adaptation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12): 3071–3085, 2019.

[19] Ramprasaath Rs, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 10 2019.

[20] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, pages 213–226, 2010.

[21] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2058–2065, 2016.

[22] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *arXiv e-prints*, art. arXiv:1412.3474, 2014.

[23] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[24] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, pages 3320–3328, 2014.

[25] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.