ADVANCED METHODS OF ASYMMETRIC HETEROGENEOUS TRANSFER LEARNING

by

Magda Friedjungová

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Applied Mathematics

Prague, August 2020

Supervisor:
doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Department of Applied Mathematics
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Co-Supervisor:
Ing. Daniel Vašata, Ph.D.
Department of Applied Mathematics
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

This dissertation thesis deals with the application of asymmetric heterogeneous transfer learning in scenarios where transfer of data is crucial due to the preservation of the quality of the prediction model. We consider the existence of a source and target domain, where only the source domain is equipped with labels and is thus used to train the model to solve a supervised prediction task which is the same for both domains. In order to solve the same supervised task on an unlabeled target domain one wants to reuse the already trained model. A problem arises when the domains are different, i.e. they contain overlapping features with same joint distributions, overlapping features with different marginal distributions, or no overlapping features at all. This thesis considers two complex real-world tasks: missing features reconstruction and unsupervised domain adaptation.

First, we introduce the reader to the transfer learning field with comprehensive definitions of the basic concepts. We provide a survey of several methods used within the less known field of asymmetric heterogeneous transfer learning.

Next, we target the missing features reconstruction problem. We focus on a comprehensive survey of the missing data imputation domain and on the possibility of applying these methods to reconstruct entire missing features. We introduce our own imputation model called Wasserstein Generative Adversarial Imputation Network and provide its experimental comparison to state-of-the-art imputation methods. This model is able to solve the considered task based only on a general training strategy even in a scenario when we do not know which features are missing in advance.

Finally, we focus on unsupervised domain adaptation. Our aim was to solve the problem of mapping images from one domain to images of another domain without the need of paired or even labeled data. After a survey of the state-of-the-art methods, we propose a novel model called Latent Space Translation Network. This model greatly outperforms other state-of-the-art approaches.

This thesis therefore presents two novel methods, which tackle important real-world scenarios.

The main contributions of the dissertation thesis are as follows:

1. Comprehensive definitions of transfer learning concepts together with an overview of methods in the less known asymmetric heterogeneous field.
2. Novel imputation model, Wasserstein Generative Adversarial Imputation Network, to tackle the problem of missing features reconstruction. This model is able to work in a general set up when it is not known which features are missing in advance. In some cases the model gives better results compared to the state-of-the-art data imputation methods.
3. Linear and information imputability to order missing features. Thanks to this ordering, the sequential imputation methods are able to impute data more accurately.
4. Novel Latent Space Translation Network, which is able to solve an unsupervised domain adaptation task, where no paired or even labeled data are available. Proposed network greatly outperforms state-of-the-art approaches.

Keywords:
Heterogeneous Asymmetric Transfer Learning, Missing Features, Data Imputation, Non-Generative Models, Generative Models, Latent Space, Domain Adaptation.

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, assoc. prof. Marcel Jiřina. He has been a constant source of encouragement and professional advancements during my research.

Next, I am very grateful to my dissertation thesis co-supervisor, Dr. Daniel Vašata, who has been an amazing person and a brilliant researcher to work with. We spent a lot of time together researching, discussing and solving various difficulties. Also, he was always ready to overcome all my depressive moods and keep up the optimism in our research. I learned a lot from him and he gave me valuable feedback on all my steps throughout this study programme.

Special thanks go to the Department of Applied Mathematics, which maintained a very pleasant, flexible and inspiring environment for my research. The head, Dr. Karel Klouda, and other colleagues within this department provided me with moral support and boosted my mood whenever it was needed. These people not only worked with me, but we also spent a lot of our free time together. It may sound kitschy, but without exaggeration these years being part of such a team have allowed me to grow up both professionally and personally.

I would like to express special thanks to the department and faculty management providing most of the funding for my research. My research has also been partially supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS17/210/OHK3/3T/18, SGS20/213/OHK3/3T/18, and by the Czech Science Foundation as project No. GA18-18080S. Also, I would like to thank the faculty managment for the opportunity to attend several conferences in order to present my doctoral work. These visits provided lots of valuable experience.

Furthermore, I would like to express thanks to my closest friends, to name some of you, Kát'a, Bob, Adéla, Petra, Kuba, and Ivana, and all the others who did not doubt me and patiently endured all my lamentations.

Finally, my biggest thanks go to my family members for their infinite patience, support and care.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ACGAN	Auxiliary Classifier Generative Adversarial Network
ADDA	Adversarial Discriminative Domain Adaptation
ANOVA	Analysis of Variance
ARC-t	Asymmetric Regularized Cross-domain transformation
CoGAN	Coupled Generative Adversarial Network
CycleGAN	Cycle-Consistent Adversarial Network
DAE	Denoising Autoencoder
DAMA	Domain Adaptation with Manifold Alignment
DiscoGAN	Discovery Generative Adversarial Network
DTN	Domain Transfer Network
ECOC	Error Correcting Output Codes
ELFSR	Ensemble Learning via Feature-Space Remapping
FSR	Feature-Space Remapping
GAIN	Generative Adversarial Imputation Network
GAN	Generative Adversarial Network
HDP	Heterogeneous Defect Prediction
HeMap	Heterogeneous Spectral Mapping
HFA	Heterogeneous Feature Augmentation
HHTL	Hybrid Heterogeneous Transfer Learning
HI-VAE	Heterogeneous-Incomplete Variational Autoencoder
IFSR	Informed Feature-Space Remapping
JDOT	Joint Distribution Optimal Transport
***k*-NN**	*k*-Nearest Neighbors
LR	Logistic Regression
LSTNet	Latent Space Translation Network
MCAR	Missing Completely At Random
MHTL	Multi Home Transfer Learning
MICE	Multiple Imputation by Chained Equations
MIWAE	Missing Data Importance-Weighted Autoencoder
MLP	Multi-Layer Perceptron
MMDT	Max-Margin Domain Transforms
MOMAP	Multiple Outlook Mapping
MSE	Mean Squared Error
NB	Naive Bayes
NIR	Near Infrared
RF	Random Forest
RMSE	Root Mean Squared Error
SBADA-GAN	Symmetric Bi-directional ADAptive Generative Adversarial Network

SGVB	Stochastic Gradient Variational Bayes
SHDA-RF	Supervised Heterogeneous Domain Adaptation via Random Forest
SHFR	Sparse Heterogeneous Feature Representation
SSKMDA	Semi-Supervised Kernel Matching Domain Adaptation
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
UFSR	Uninformed Feature-Space Remapping
UNIT	Unsupervised Image-to-image Translation
VAE	Variational Autoencoder
VAEAC	Variational Autoencoder with Arbitrary Conditioning
WGAIN	Wasserstein Generative Adversarial Imputation Network
WGAN	Wasserstein Generative Adversarial Network
XGBT	Extreme Gradient Boosted Trees

Chapter 1

Introduction

In this chapter we introduce the reader to the transfer learning field together with the motivation for the proposed doctoral thesis, described in Section 1.1. In Section 1.2, the problem statement is provided. The main contributions of this thesis can be found in Section 1.3 and finally the thesis outline is presented in Section 1.4.

1.1 Transfer Learning

To solve machine learning tasks, we need a sufficient amount of data of high quality in order to construct a sufficient prediction model. One of the main prerequisites in most machine learning tasks is that all available data originates from the same domain. In practice, it is sometimes impossible to comply with this assumption. The distribution, the type, and even the dimensionality of features can change from one dataset to the next. In the big data era, we encounter a diversity of datasets from different sources in different domains. In many cases, we face a situation where we want to combine data from different domains represented by different feature spaces to solve machine learning tasks. This can be caused by e.g., lack of quality data or the impossibility to collect new data for the construction of a machine learning model, or perhaps we may want to use information from more than one dataset.

This thesis studies how to deal with the task when we assume two domains (as a start, but there may be more domains) that differ in feature representations, but the target supervised prediction task is the same. Specifically, we assume that both domains are rich with data, but only one is equipped with labels. One of the possible ways to predict labels on the unlabeled domain is to reuse the prediction model already learned in a labeled domain. To do so, the data representation of both domains has to be unified to a representation as used for the training of the model. It is due to the fact that models are not usually able to react to different data representations (e.g. missing features, different distributions). We found transfer learning helpful in this case.

Transfer learning is a machine learning method where a prediction model trained to solve some task is reused as the starting point for a model on a different task, or the solved tasks can be the same and the domains devoted for training the models can differ [89]. It means that transfer learning approaches enable us to modify the domain without labels in such a manner that the unlabeled domain takes advantage of an already trained prediction model on other domains. This thesis focuses on two tasks, in which we investigate their ability to be solved using transfer learning methods.

As the first task, we consider missing features reconstruction. This means that some features presented in the source domain are missing in the target domain. A typical example is the prediction of a classification model from sensor data, where a sensor breakdown leads to missing data in one or more features at the time of prediction. In such a case, it is crucial to somehow reconstruct the missing features, because the prediction model has already been learned and is not able to react to missing data. Furthermore, it is not possible to retrain this model due to its deployment in production.

The second task is focused on unsupervised domain adaptation where the features from the source domain are related to features from the target domain. As a typical example one can imagine photos taken by one camera and photos taken by a different camera. These photos can differ in resolution, chromaticity, sharpness etc. We want to perform image classification on these photos, i.e. assignment of the correct description of the scene shown in the photo, which can be a time and money consuming task due to its complexity and often requires the presence of a human. It may result in one or both photo domains being unlabeled and the goal is to learn unsupervised mapping between the domains. It can be used in a wide range of applications, such as image caption generation, collection style transfer, object transfiguration, season transfer, image colorization, object reconstruction, and photo enhancement.

This thesis is written with the assumption that the reader has a working knowledge of machine learning. For more information about machine learning, see e.g. [82].

1.1.1 Motivation

The success of machine learning algorithms depends on data representation as well as the quantity of available data for the training of a model. Both requirements are domain-dependent. However, transfer learning methods allow us to use data from different domains with different representations to solve the same machine learning task. Transfer learning saves time and human resources and provides the possibility to solve tasks without knowledge of the domain. Without transfer learning methods, the transfer of data is often solved manually. However, in some cases, this approach poses a combinatorial problem and almost always requires the presence of a domain expert. In the ideal case, it would be beneficial to find such a mapping, that would enable us to map the data between source and target feature spaces. The challenge is to overcome the differences between the domains so that a prediction model trained on one domain generalizes well to another domain [89]. Let us show some real scenarios where transfer learning methods are common.

Typical tasks for transfer learning within the computer vision field include **domain adaptation** and **image-to-image translation** [23]. In this case, images from various domains differ in external factors such as environments, lighting, background, sensor types, view angles, and post-processing. This causes a distribution shift or even different feature spaces such as mathematical sets (e.g. different image resolution) [141].

Another example is **cross-lingual text categorization, document classification** or **language processing** [143, 130]. For example, labeled English documents are widely available, but labeled Chinese documents are much harder to obtain. These documents, English and Chinese, do not share the same feature representation due to a different language. Transfer learning can use natural correspondences between feature spaces in order to create an automated classification model for Chinese documents.

One more example can be **sensors** used to localize and recognize human activities in two different apartments. The necessity to transfer data between different apartments is usually caused by a lack of data about one of the apartments [126, 125]. We can also face a cold-start problem (known from the domain of recommender systems [106]) in one of the apartments, which occurs when no data exist (due to new apartment, new users, etc.).

Recently, a very discussed domain is **clinical imaging**. Imagine a scenario when radiologists manually annotate tissues, abnormalities, and pathologies of patients. Biomedical engineers then use these annotations to train systems to perform automatic tissue segmentation or pathology detection in medical images. Now suppose a hospital installs a new MRI scanner. Unfortunately, due to the mechanical configuration, calibration, vendor, and acquisition protocol of the scanner, the images it produces will differ from images produced by other scanners. Consequently, systems trained on data from other scanners would fail to perform well on the new scanner [135]. Transfer learning methods may prevent this failure by adapting one domain to another.

Another example of differences between feature spaces are **image caption generators** from the field of computer vision. These generators generalize from the image domain to the text domain [54] and try to describe objects seen on the images.

Sometimes, we can face a **deployment shift**. The deployment shift problem happens when we transfer from synthetic data to live user data [56]. It happens when a model cannot be trained on the real usage data before deployment simply because it does not exist. A common practice is to generate a large quantity of synthetic training data that mimics the expected user behavior. Such synthetic data is crafted using domain-specific knowledge and can be time-consuming. It is also flawed in that it typically does not match the live user data generated by actual users. The real queries submitted to these systems are different from what the model designers expect to see. It means that there is a distribution mismatch between training and evaluation data, leading to a model that overfits the flawed training data and performs poorly on the test data. The transfer learning task which has to be tackled is adapting one data domain to the other.

Transfer learning methods can be used successfully in all these scenarios.

1.2 Problem Statement

In our research, we consider the following scenario: two data domains, source and target, and a prediction task, which is the same for both domains. We assume that only the source domain contains labels. We are able to learn the prediction model in this labeled domain. This prediction model is considered as a black-box throughout this thesis, i.e., cannot be changed in the sense of re-training. To predict labels on the unlabeled target domain, we can reuse the predictive model already learned in the labeled source domain. To do so, the data representation of the target domain has to be unified to the representation that was used for the learning of the model. This can be reached by mapping domains, one to the other. This specific area of transfer learning is called asymmetric heterogeneous transfer learning [27]. A complete explanation of the topic is given in Chapter 2.

We defined two tasks of asymmetric heterogeneous transfer learning to be solved in this thesis:

1. missing features reconstruction,
2. unsupervised domain adaptation.

In order to address these tasks, we set the following assumptions:

- both domains are rich on data,
- just one of the domains contains labels,
- the prediction task is classification,
- the already learned prediction model is considered a black-box, i.e., cannot be changed in the sense of re-training.

To evaluate the influence of the mapping on an already learned prediction model, we explore classification accuracy using the mapped dataset. The classification accuracy was used to evaluate proposed methods in both tasks, missing features reconstruction and unsupervised domain adaptation. The motivation behind this evaluation metric is that we consider an already learned and deployed model whose prediction accuracy must not drop significantly using the new unseen dataset.

1.2.1 Goals of the Dissertation Thesis

The main goal of this dissertation thesis is to improve the up-to-date knowledge in the asymmetric heterogeneous transfer learning field. The goals are summarized as follows:

- Precisely define heterogeneous transfer learning and related terms.
- Research and compare existing data imputation methods. Improve existing or introduce new algorithm.
- Improve existing or introduce new algorithm for the problem of unsupervised domain adaptation and compare to existing approaches.

1.3 Contributions of the Dissertation Thesis

In this section we briefly highlight the main contributions of the presented dissertation thesis as follows:

1. We provide comprehensive definitions of transfer learning concepts together with an overview of methods in the less known asymmetric heterogeneous field. Both can be found in Chapter 2.

2. We present our own imputation model, Wasserstein Generative Adversarial Imputation Network, to tackle the problem of missing features reconstruction. This model is able to work in a general set up where it is not known in advance which features are missing. In some cases the model gives better results compared to the state-of-the-art data imputation methods. For details see Chapter 3.

3. We propose linear and information imputability to order missing features, which are presented in Chapter 3 as well. Thanks to this ordering, the sequential imputation methods are able to impute data more accurately.

4. We present a novel Latent Space Translation Network, which is able to solve the unsupervised domain adaptation task, where no paired or even labeled data are available. The proposed network greatly outperforms state-of-the-art approaches, see Chapter 4.

All contributions of this thesis are properly published as can be seen in a list of reviewed publications of the author on page 85 and relevant publications are also mentioned at the beginning of the corresponding chapters.

1.4 Structure of the Dissertation Thesis

The thesis is organized into five chapters as follows:

Chapter 1 describes the motivation behind our efforts together with our goals. There is also a list of contributions of this dissertation thesis.

Chapter 2 introduces the reader to the necessary theoretical background of transfer learning, state-of-the-art methods, and solved scenarios.

Chapter 3 focuses on missing features reconstruction, including possible data imputation methods. We propose linear and information imputability to order missing features to be imputed. Also, we present our general imputation model called Wasserstein Generative Adversarial Imputation Network and compare it to state-of-the-art imputation methods on a feature reconstruction task.

Chapter 4 presents the domain adaptation task between labeled and unlabeled domains. To tackle this task we present our own Latent Space Translation Network and compare its results to state-of-the-art methods.

Chapter 5 summarizes the results of the presented research, suggests possible topics for further research and concludes the thesis.

Chapter 2

Asymmetric Heterogeneous Transfer Learning

This chapter is based on the following publications:

- Friedjungová, M.; Jiřina, M. An Overview of Transfer Learning Focused on Asymmetric Heterogeneous Approaches. In *Data Management Technologies and Applications.* Springer International Publishing, 2018.
- Friedjungová, M.; Jiřina, M. Asymmetric Heterogeneous Transfer Learning: A Survey. In *Proceedings of the 6th International Conference on Data Science, Technology and Applications.* SciTePress, 2017.
- Friedjungová, M. *Asymmetric Heterogeneous Transfer Learning.* Ph.D. Minimum Thesis, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic, 2018.

This chapter presents the necessary background of transfer learning and an overview of state-of-the-art methods. In particular, it is focused on asymmetric heterogeneous transfer learning, which is the primary topic of this thesis. The chapter is organized as follows. In Section 2.1, we introduce definitions and basic approaches necessary for this thesis. We also propose a division of different transfer learning settings based on our overview in this field. In Section 2.2, the state-of-the-art methods for asymmetric heterogeneous transfer learning are presented. In Section 2.3, we present the focus of this thesis on two tasks - missing features reconstruction and domain adaptation, on which we focus in more detail in separate chapters.

2.1 Preliminaries

Machine learning tasks have a common assumption that training and test data are drawn from the same feature space and the same distribution. Moreover, from a statistical point of view it is usually assumed that they are independent and identically distributed (i.i.d.) [26]. When this assumption does not hold, transfer learning comes into play. Let us start with definitions of basic terms where we follow [89].

Definition 2.1.1. A *domain* $\mathcal{D}$ is a pair $(\mathcal{X}, P(x))$, where $\mathcal{X}$ is the feature space of $\mathcal{D}$, and $P(x)$ is the marginal probability distribution on $\mathcal{X}$.

Definition 2.1.2. A *task* $\mathcal{T}$ for some given domain $\mathcal{D}$ is a pair $(\mathcal{Y}, P(y|x))$, where $\mathcal{Y}$ is a set of labels and $P(y|x)$ is a conditional probability distribution on $\mathcal{Y}$ given $x \in \mathcal{X}$.

In deterministic cases conditional probability distribution $P(y|x)$ corresponds to predictive function $f : \mathcal{X} \to \mathcal{Y}$.

Definition 2.1.3. A *labeled dataset* for some given domain $\mathcal{D}$ and task $\mathcal{T}$ is a set of pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ for some $n \in \mathbb{N}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ which is called a label. An *unlabeled dataset* for some given domain $\mathcal{D}$ is a set $\{x_1, \ldots, x_n\}$ for some $n \in \mathbb{N}$, where $x_i \in \mathcal{X}$.

Labeled and unlabeled datasets are connected to supervised and unsupervised machine learning, respectively. The aim of unsupervised machine learning is to estimate the marginal probability distribution $P(x)$ from an unlabeled dataset. In supervised machine learning the conditional probability distribution $P(y|x)$ on $\mathcal{Y}$ given $x \in \mathcal{X}$ is estimated from a labeled dataset.

In this thesis, we consider two domains, a source domain $\mathcal{D}_s$ and a target domain $\mathcal{D}_t$. Each domain has its task, a source task $\mathcal{T}_s$ and a target task $\mathcal{T}_t$. If these domains are different, then they have different feature spaces ($\mathcal{X}_s$ and $\mathcal{X}_t$) or different marginal probability distributions ($P_s(x)$ and $P_t(x)$) or a combination of both [89].

Definition 2.1.4. Let $\{(\mathcal{D}_{s_1}, \mathcal{T}_{s_1}), \ldots, (\mathcal{D}_{s_n}, \mathcal{T}_{s_n})\}$ be a set of source domains and corresponding source tasks and $\mathcal{D}_t$ be a target domain which corresponds to a target task $\mathcal{T}_t$.

Transfer learning is an approach which enables us to estimate the conditional probability distribution $P(y|x)$ from $\mathcal{T}_t$ using information gained from $(\mathcal{D}_{s_1}, \mathcal{T}_{s_1}), \dots, (\mathcal{D}_{s_n}, \mathcal{T}_{s_n})$.

Different settings of transfer learning based on the existence of the source and target labels and various relations between domains and tasks are presented in Figure 2.1. This division is based on our review in the transfer learning field.

Note, that $\mathcal{D}_s \neq \mathcal{D}_t$ also implies that either $\mathcal{X}_s \neq \mathcal{X}_t$ or $P_s(x) \neq P_t(x)$. Analogously, $\mathcal{T}_s \neq \mathcal{T}_t$ also implies that either $\mathcal{Y}_s \neq \mathcal{Y}_t$ or $P_s(y|x) \neq P_t(y|x)$. In the papers, one can often see the notation $\mathcal{T}_s = \mathcal{T}_t$ when $\mathcal{X}_s \neq \mathcal{X}_t$ informally means that $\mathcal{Y}_s = \mathcal{Y}_t$. The conditional probabilities cannot be equal, since conditions are from different spaces, $\mathcal{X}_s \neq \mathcal{X}_t$. Formally, it means that there exists a relation between $\mathcal{X}_s$ and $\mathcal{X}_t$ such that $P_s(y|x_s) = P_t(y|x_t)$ whenever x_s and x_t are in the relation.

From the view of feature spaces relation, we can divide transfer learning into two categories - heterogeneous and homogeneous. Homogeneous transfer learning assumes different source and target domains, but the same feature spaces.

Definition 2.1.5. *Homogeneous transfer learning* is a transfer learning where $\mathcal{D}_{s_i} \neq \mathcal{D}_t$ and $\mathcal{X}_{s_i} = \mathcal{X}_t$ for all i.

On the contrary, in heterogeneous transfer learning one assumes different feature spaces.

Definition 2.1.6. *Heterogeneous transfer learning* is a transfer learning where $\mathcal{X}_{s_i} \neq \mathcal{X}_t$ for all i.

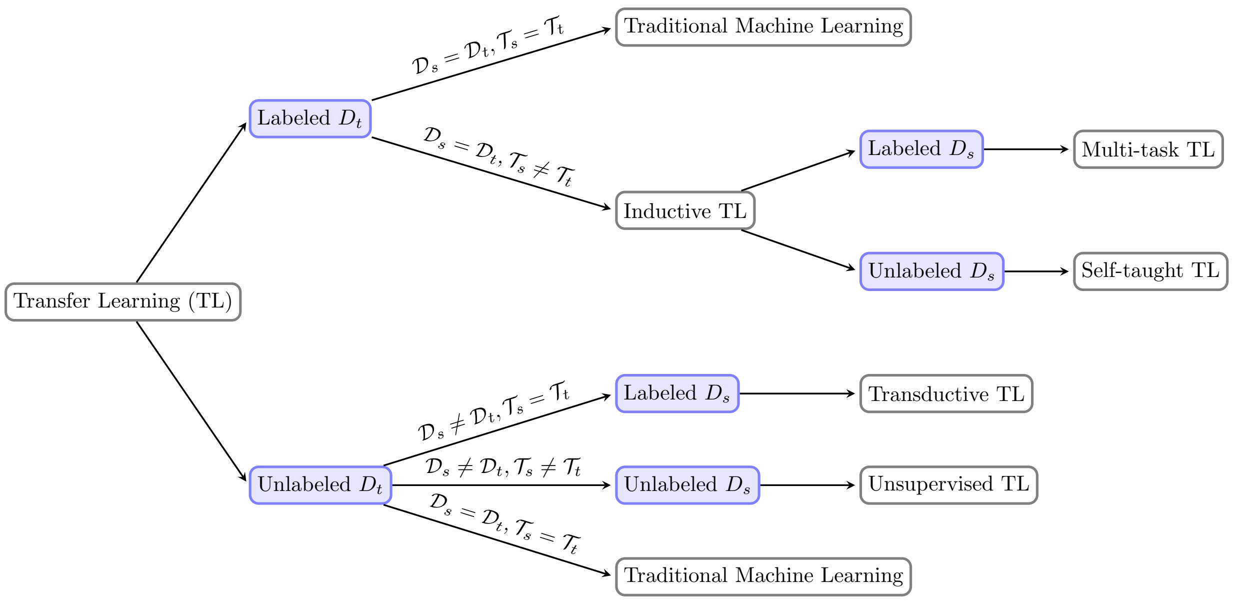

Transfer Learning (TL)
Labeled D_t
$\mathcal{D}_s = \mathcal{D}_t, \mathcal{T}_s = \mathcal{T}_t$
Traditional Machine Learning
$\mathcal{D}_s = \mathcal{D}_t, \mathcal{T}_s \neq \mathcal{T}_t$
Inductive TL
Labeled D_s
Multi-task TL
Unlabeled D_s
Self-taught TL
Unlabeled D_t
$\mathcal{D}_s \neq \mathcal{D}_t, \mathcal{T}_s = \mathcal{T}_t$
Labeled D_s
Transductive TL
$\mathcal{D}_s \neq \mathcal{D}_t, \mathcal{T}_s \neq \mathcal{T}_t$
Unlabeled D_s
Unsupervised TL
$\mathcal{D}_s = \mathcal{D}_t, \mathcal{T}_s = \mathcal{T}_t$
Traditional Machine Learning

Figure 2.1: Proposed overview of transfer learning settings considering the presence of the labels in the source and target datasets D_s and D_t, respectively.

A specific sub-field of transfer learning is domain adaptation [89], where the source and target tasks are the same, $\mathcal{T}_{s_i} = \mathcal{T}_t$. Based on the condition $\mathcal{D}_{s_i} \neq \mathcal{D}_t$, we categorise domain adaptation into homogeneous and heterogeneous as well.

Definition 2.1.7. *Homogeneous/heterogeneous domain adaptation* is homogeneous/heterogeneous transfer learning, where $\mathcal{T}_s = \mathcal{T}_t$.

2.1.1 Basic Approaches

In heterogeneous transfer learning, two basic transfer approaches are distinguished.

The first approach is called symmetric and transforms both the source and target feature spaces into a common latent feature space to unify the input spaces of the domains. It means that it discovers underlying meaningful structures between domains to find a common latent feature space [89]. Most existing heterogeneous transfer learning methods [110, 91, 129, 30] assume that both source and target data can be represented using a common latent feature space. Thus, we are looking for transformation mappings $M_{s\to\ell}$ and $M_{t\to\ell}$ to transform source domain data and target domain data to a new common latent feature space $\mathcal{X}_\ell$ as is shown in Figure 2.2 and 2.3.

Definition 2.1.8. *Symmetric heterogeneous transfer learning* is heterogeneous transfer learning, where both the source domain feature space $\mathcal{X}_s$ and the target domain feature space $\mathcal{X}_t$ are mapped into a common latent feature space $\mathcal{X}_\ell$.

$$\begin{tikzcd}
\mathcal{X}_s \arrow[dr, "M_{s\to\ell}"] & \\
& \mathcal{X}_\ell \\
\mathcal{X}_t \arrow[ur, "M_{t\to\ell}"] &
\end{tikzcd}$$

Figure 2.2: Symmetric transformation mapping of the source feature space $\mathcal{X}_s$ and target feature space $\mathcal{X}_t$ into a common latent feature space $\mathcal{X}_\ell$ using transformation mappings $M_{s\to\ell}$ and $M_{t\to\ell}$ (altered figure from [133]).

The second approach is asymmetric, which transforms the features of the source domain to match the features of the target domain or vice versa as shown on Figure 2.4, 2.5 and 2.6. It consists of finding a transformation mapping $M_{t\to s}$ between the feature spaces $\mathcal{X}_t$ and $\mathcal{X}_s$ or vice versa [61, 89, 24].

Definition 2.1.9. *Asymmetric heterogeneous transfer learning* is heterogeneous transfer learning, where the source domain feature space $\mathcal{X}_s$ is mapped directly to the target domain feature space $\mathcal{X}_t$ or vice versa.

If we focus on mapping from source feature space $\mathcal{X}_s$ to target feature space $\mathcal{X}_t$, it means that we can create training data in the target space to be used to learn a predictive model

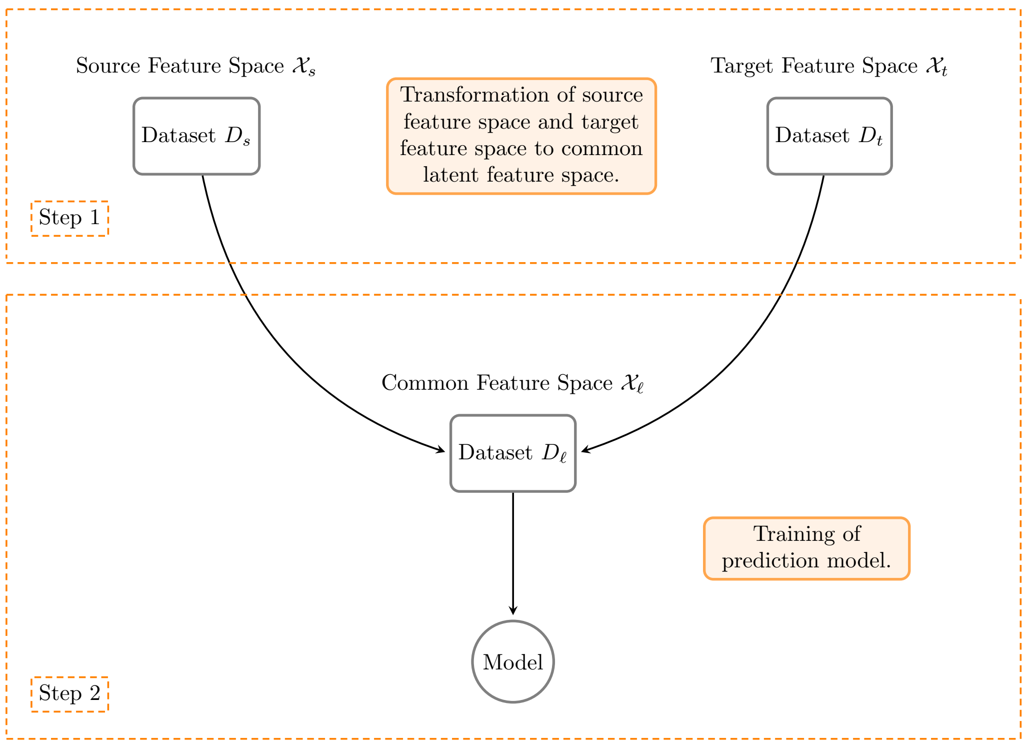

Source Feature Space $\mathcal{X}_s$
Dataset D_s
Transformation of source feature space and target feature space to common latent feature space.
Target Feature Space $\mathcal{X}_t$
Dataset D_t
Step 1
Common Feature Space $\mathcal{X}_\ell$
Dataset D_ℓ
Training of prediction model.
Model
Step 2

Figure 2.3: The symmetric approach consists of two steps. Step 1 consists of mapping dataset D_s and dataset D_t to a common latent feature space creating dataset D_ℓ. Step 2 consists of training the model based on data from the dataset D_ℓ.

$$\begin{tikzcd}
\mathcal{X}_s \arrow[r, "M_{s\to t}"] & \mathcal{X}_t
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{X}_s & \mathcal{X}_t \arrow[l, "M_{t\to s}"']
\end{tikzcd}$$

Figure 2.4: Asymmetric transformation mapping $M_{s\to t}$ of the source feature space $\mathcal{X}_s$ to the target feature space $\mathcal{X}_t$ or vice versa (altered figure from [133]).

on the target feature space $\mathcal{X}_t$. On the contrary, the mapping from target feature space $\mathcal{X}_t$ to source feature space $\mathcal{X}_s$ means that we want to use the predictive model already learned on source feature space $\mathcal{X}_s$ to predict labels of the target feature space $\mathcal{X}_t$. Note that if we consider the existence of more source domains, only the way we described first is suitable.

In the considered heterogeneous transfer learning setup, the main assumption is that the source and target domains are represented by different feature spaces, $\mathcal{X}_s \neq \mathcal{X}_t$. One can identify several situations of how these spaces can differ. Source and target domains can contain:

1. overlapping features with same joint distributions,

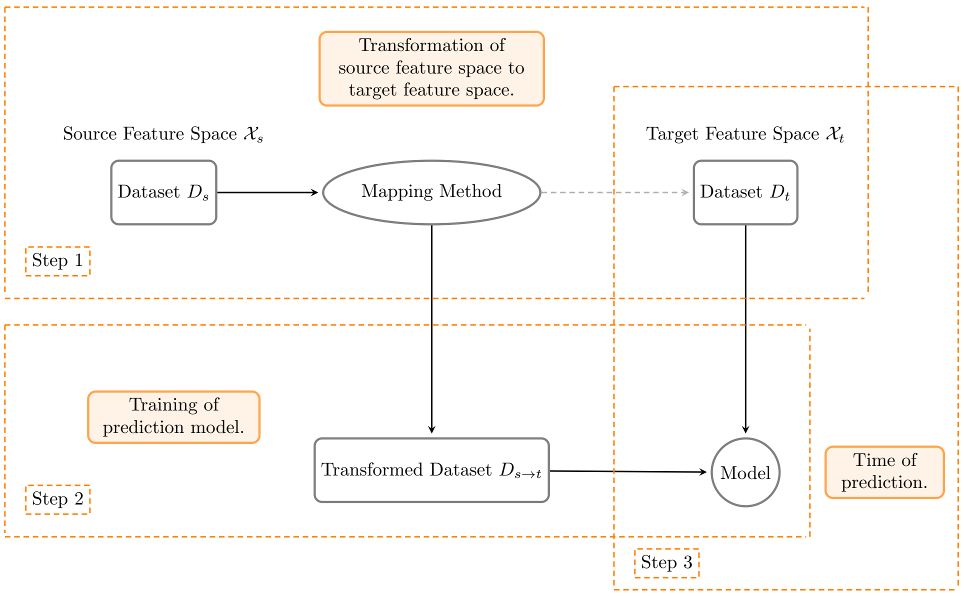

Transformation of source feature space to target feature space.
Source Feature Space $\mathcal{X}_s$
Target Feature Space $\mathcal{X}_t$
Dataset D_s
Mapping Method
Dataset D_t
Step 1
Training of prediction model.
Transformed Dataset $D_{s\to t}$
Model
Time of prediction.
Step 2
Step 3

Figure 2.5: Possible asymmetric heterogeneous transfer learning approaches: Step 1 consists of mapping dataset D_s to dataset D_t using mapping methods. Step 2 consists of training the model based on data from transformed dataset $D_{s\to t}$, step 3 is the phase where we test the model on dataset D_t from target feature space.

2. overlapping features with different marginal distributions,
3. no overlapping features.

These situations can be combined into various levels of complexity. As a real-world example of the first situation, consider a prediction model based on sensor data, where a sensor breakdown leads to missing data in one or more features, whereas the remaining features are identical. Usually, the prediction model itself is not able to handle this situation without a significant decrease in its performance. To solve this problem and avoid performance degradation means to somehow reach values of data that look like the original, i.e., source. Methods of data imputation can be helpful. This situation is very common in heterogeneous transfer learning.

A more complicated situation can be given by an example, where we consider a model trained to detect sentiment in movie reviews. We want to reuse this model to detect sentiment in book reviews. In this case, the model will suffer from decreasing performance because these two data domains correspond to various data distributions due to differences in words used, sentences, etc. in each domain. Once the distribution changes, most statistical models need to be rebuilt using new labeled training data from the new distribution. This can be both expensive and complex due to the need to collect new labeled data.

As a third situation, one can imagine source data represented as text documents and target data represented as images. Both have to belong to the same task; for example both

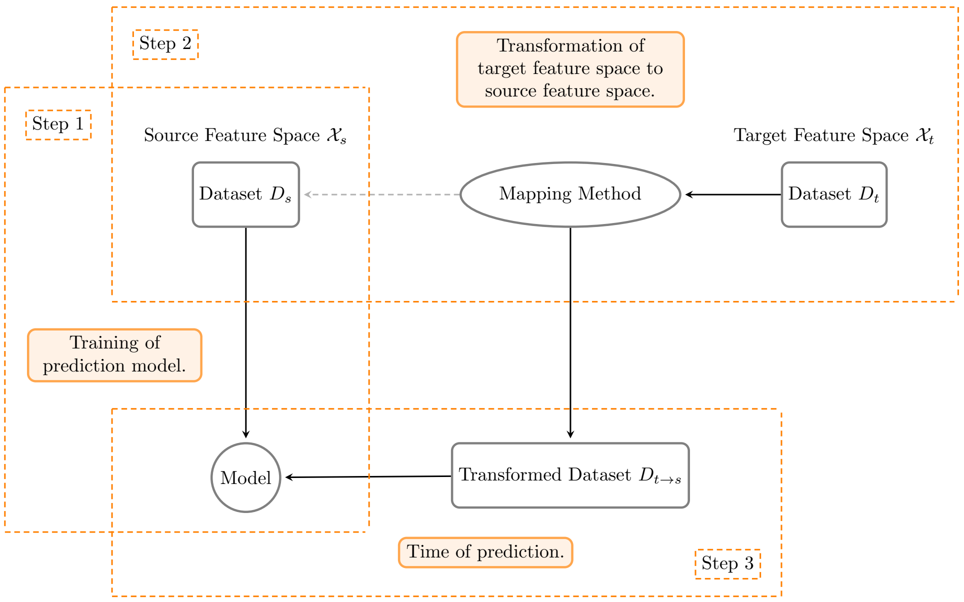

Step 2
Transformation of target feature space to source feature space.
Step 1
Source Feature Space $\mathcal{X}_s$
Target Feature Space $\mathcal{X}_t$
Dataset D_s
Mapping Method
Dataset D_t
Training of prediction model.
Model
Transformed Dataset $D_{t\rightarrow s}$
Time of prediction.
Step 3

Figure 2.6: Possible asymmetric heterogeneous transfer learning approaches: Step 1 consists of training the model based on data from dataset D_s from source feature space. In Step 2 we are looking for a mapping from dataset D_t to dataset D_s. Step 3 shows the model at time of prediction on the transformed dataset $D_{t\rightarrow s}$.

datasets are used for the classification of wild animals [96]. This example represents the most complex of all the mentioned situations.

2.2 Related Work

In this subsection we give an overview of related work in the domain of interest. In [A.4, A.5] we presented a comprehensive survey of asymmetric heterogeneous transfer learning methods that we have expanded with actual knowledge. Asymmetric heterogeneous transfer learning methods are widely used in domains such as image classification, cross-language classification, cross-project defect prediction, and activity recognition. We want to make readers aware that the individual methods are classified into these four main domains. We present a brief comparison in Table 2.1 of up to date methods with references to relevant research papers. Some complex and up-to-date surveys as [133], [27], and [148] exist. These surveys provide a very good summary with explanations of classic transfer learning methods, and they were a great inspiration for our overview.

Image Classification

Image classification is very popular in the machine learning field. However, it encounters many difficulties. An example of such difficulties can be labeling new images which require a lot of human labor and include ambiguous objects in images or objects having ambiguous explanations. Each image can also have a different exposition, brightness, intensity,

Table 2.1: Asymmetric heterogeneous transfer learning approaches. Source data are considered as labeled for all methods. Abbreviation: S = source domain, T = target domain.

Method	**Target Data**	**Transfer Direction**	**Type of Task**
ARC-t [61]	limited labels	$S \rightarrow T$	image classification
FSR (IFSR, UFSR, ELFSR) [34]	also unlabeled	$S \leftarrow T$	activity recognition, cross-language classification
HDP [84]	unlabeled	$S \rightarrow T$	defect prediction
HHTL [144]	unlabeled	$S \leftarrow T$	cross-language classification
van Kasteren's method [126, 125]	unlabeled	$S \rightarrow T$	activity recognition
MHTL [94]	unlabeled	$S \rightarrow T$	activity recognition
MMDT [46]	also unlabeled	$S \rightarrow T$	image classification
MOMAP [43]	limited labels	$S \rightarrow T$	activity recognition
SHDA-RF [116]	limited labels	$S \rightarrow T$	activity recognition
SHFR [145]	limited labels	$S \leftrightarrow T$	cross-language classification
SSKMDA [136]	unlabeled	$S \leftarrow T$	cross-language classification
TLRisk [24]	unlabeled	$S \rightarrow T$	cross-language classification, image classification

coloring, or angle of view. Some of these problems can be solved using transfer learning methods. Below, we present methods that are explicitly focused on asymmetric heterogeneous transfer learning. However, there exist a lot of related methods in the field of domain adaptation, which are presented in Chapter 4.

In [61], the problem of domain adaptation for transferring object models from one dataset to another was solved by introducing a novel approach in computer vision called Asymmetric Regularized Cross-domain transformation (ARC-t). The main idea is to learn an asymmetric non-linear transformation that maps data from one domain to another domain. There is a requirement that the source data, along with a part of the target data have to be labeled. The input to the algorithm consists of pairs of inter-domain examples that are known to be semantically similar or dissimilar, i.e. similarity based on meaning as opposed to form. This approach can also be used in the case that some classes in the target domain have missing labels. Authors of ARC-t [61] aim to generalize the model of [102], which uses symmetric transformations. Symmetric transformation means that the same rotation and scaling is applied to both domains without any separation of the domains. In asymmetric transformation, the operations are applied separately to each domain. ARC-t shows how non-linear transformations can be learned by kernelizing the formulation for a particular class of regularizers. The transformation matrix is learned in a non-linear Gaussian radial basis function kernel space. The resulting algorithm is based on squared Frobenius regularization [60] and similarity constraints. Similarity constraints are created for all pairs of data in the same class by using a similarity function. It helps us decide which pairs of data are similar. The ARC-t method was tested by authors in two experiments. The first experiment included 31 image classes in the source and target training data. In the second experiment, only 16 image classes were included in the target training data, and all 31 image classes were represented in the source training data. Within kernel canonical correlation analysis [108] the source and target domains were projected into common latent space using symmetric transformation. This preprocessing step is needed because the method needs to bring the source and target domains together to test against other baseline approaches. During testing, the method showed certain advantages compared to existing baseline approaches such as k-nearest neighbors, Support Vector Machine, domain and cross-domain metric learning, and the feature augmentation method proposed by [25].

Afterward, ARC-t was bound by the Max-Margin Domain Transforms (MMDT) [46] method, which eliminates two drawbacks of ARC-t. First, the feature learning did not optimize the objective function of a strong, discriminative classifier directly. Second, it did not scale well to domains with large numbers of data points due to the high number of constraints, which is proportional to the product of the number of labeled data points in the source and target. The main idea behind MMDT is to jointly learn affine hyperplanes that separate the classes consisting of source and target instances. This is done while learning the feature transformation to map the data points of the target to the source. Feature transformation projects the points onto the right side of each source hyperplane. In comparison, ARC-t uses similarity constraints to map the data points of the same category close to each other, followed by a classification step. MMDT proposes to project the target data points to the right side of each source learned hyperplane, leading for better classification accuracy [46].

Cross-Language Classification

Another typical task in machine learning is language transformation between multilingual documents or web pages, specifically the transformation from one language to another. This task can be solved with some limitations using automatic translators mentioned in [132]. Transfer learning is aiming to solve the task without these tools, only using the transfer of knowledge between related feature spaces. Most methods learn a feature mapping between source and target domains based on data correspondences [61, 24]. Correspondence means that there exist some relationships between data from different feature spaces.

In [144], a framework based on a method of hybrid heterogeneous transfer learning (HHTL) is presented. The HHTL method expects plenty of labeled source data and plenty of unlabeled target data. The HHTL framework consists of two components. The first component learns a heterogeneous feature mapping between the labeled source and unlabeled target domain. As a second component, latent representations are discovered to reduce the data distribution bias between the transformed unlabeled target domain and the labeled source domain. HHTL simultaneously corrects the data distribution bias on the mapped feature space. This framework was tested by authors on multilingual text mining tasks and also in the application of text sentiment classification.

Dai et al. in [24] propose a translated learning to learn feature mapping based on the construction of feature correspondences between different feature spaces. The translated learning uses a language model. The language model learns the probability of word occurrence over sequences of words [63]. In [24], the language model links the class labels to the features in the source spaces and maps their translation to features in the target spaces. This novel framework is called TLRisk, where training data and test data can come from totally different feature spaces. The task is to translate all the data from source feature space into a target feature space. Authors of TLRisk assume that there is no correspondence between instances in these different feature spaces. The language model proposed in TLRisk consists of feature translation and nearest-neighbor learning and is represented as a chain of links that is modeled using a Markov chain and risk minimization. For the development of such a model, there is a need for some co-occurrence data across the source and target spaces. The performance of the TLRisk framework was shown on text-aided image classification and cross-language classification (English to German classification). Note that translated learning is focused only on the third situation, where we do not assume overlapping features, as mentioned in Subsection 2.1.1.

Zhou et al. in [145] proposed a domain adaptation method where data originates from heterogeneous feature spaces of different dimensions. This method includes a transformation matrix which represents direct linear mapping from source to target domain. The mapping is compatible with the two following constraints: sparse feature representation and class-invariant transformation. Sparse feature representation means that a feature in one domain can be represented by only several features in a different domain. Class-invariant transformation means that the feature mapping of some feature is invariant to different classes. The Error Correcting Output Codes (ECOC) scheme was used to generate binary classifiers for the multi-class classification problem [29]. With ECOC, the solution presented in [145], called Sparse Heterogeneous Feature Representation (SHFR), can learn a transformation matrix. Part of the learning process of the transformation matrix is the adoption of a multi-task learning method based on [4]. Multi-task learning is based on learning more tasks simultaneously [89]. The SHFR method (also in combination with ECOC) was tested against methods such as DAMA [24], ARC-t [61], and HFA [30]. The proposed SHFR outperformed other methods for all tests.

Xiao and Guio in [136] proposed a novel feature space independent semi-supervised kernel matching method called Semi-Supervised Kernel Matching Domain Adaptation (SSKMDA). Their approach learns a prediction function on the labeled source data while mapping the target data points to similar source data points by matching the target kernel matrix to a submatrix of the source kernel matrix based on a Hilbert Schmidt Independence Criterion [42]. Usage of this criterion may be seen in newer transfer learning works such as [137]. The SSKMDA method was evaluated not only on cross-language text classification tasks but also on cross-domain sentiment classification tasks in both heterogeneous and homogeneous settings [27]. The method was compared and outperformed the HeMap [110], DAMA [24], ARC-t [61], and HFA [30].

Cross-Project Defect Prediction

Cross-project defect prediction is another important application area in transfer learning and software engineering. The aim is to build a prediction model using defect data collected from a software project and to predict defects for new software projects [85, 45, 76, 93]. However, measured data between projects must have the same metric set with identical meanings between projects. When a new project is developed, it contains little or no historical defect data for the training of a predictive model. We can use transfer learning to apply data or model from other source projects to detect defects in the new target project (cross-project defect prediction).

Nam and Kim [84] proposed the Heterogeneous Defect Prediction method (HDP) to be used in cross-project defect prediction. The HDP approach uses two steps: metric selection and metric matching. Metric selection is applied first to the source data to eliminate irrelevant features. This selection is made using standard feature selection methods such as gain ratio, $\mathcal{X}^2$, Relief-F, and significance feature evaluation. Then metrics between the source and target are matched based on similarities of distribution or correlation. A "matching score" between the source and target metric pair is calculated. The scoring techniques include percentile-based matching (called PAnalyzer), p-value from the Kolmogorov-Smirnov Test (called KSAnalyzer) and the Spearman's rank correlation coefficient to measure correlation between the source and target metrics (called SCoAnalyzer). After this, a standard classification algorithm may be trained on the transformed source to detect defects for target projects. The HDP method uses logistic regression as the base learner. Nam and Kim [84] used five groups of datasets from 34 software projects for their experiments.

Activity Recognition
We are going to state several examples from the activity recognition area where heterogeneous transfer learning finds numerous applications. In this area, the activities of daily living are monitored through various sensors. This monitoring is crucial for the future of elderly care. There is motivation to use existing data from other houses (each house has different dispositions, uses different sensors, etc.) in order to learn the parameters of a model for a new house. The reason is that activity recognition models often rely on labeled examples of activities for learning, which are missing in a new house, which is known as the cold-start problem. Activity recognition and discovery models usually include information based on structural, temporal, and also spatial features of the activities [17, 94].

In [94], the method called Multi Home Transfer Learning (MHTL) is proposed to map sensors from source to target domain based on their location in the apartment or on their function (e.g. health monitoring, daily activities monitoring, etc.). The MHTL composes of three phases – activity extraction, mapping, and label assignment. The activity extraction phase consists of various features from sensors, such as structural, spatial, and temporal. The MHTL is a good example of the utilization of meta-features. Meta-features are features that describe the properties of the original features. Meta-features are common for all data and they allow us to have a single common feature space that can be used for all houses [125]. In MHTL, meta-features are first manually introduced into the feature space (for every source-target pair). This feature space is then automatically mapped from the source to the target domain. As other works using meta-features see [10, 125].

Kasteren et al. in [125] propose a method for applying transfer learning to activity recognition. Each sensor is described using one or more meta-features, for example, the sensor on the washing machine might have one meta-feature specifying that the sensor is located in the bathroom and another that the sensor is attached to the water pipes. The mapping of features occurs while learning the prior distribution over the model parameters, which provides an initial estimate of the model parameters for target task and is learned from the source tasks. The Hidden Markov model is used to recognize activities from the sensor data and an expectation-maximization algorithm to learn the model parameters of the target house. During the training, the model is able to use both labeled and unlabeled data. For the evaluation of the performance of the proposed method, Kasteren et al. used three real-world activity recognition datasets.

Harel et al. designed a Multiple Outlook Mapping algorithm (MOMAP) [43]. MOMAP computes optimal affine mappings from different source feature spaces (in their terminology outlooks) to a target feature space by matching moments of empirical distributions. These spaces are not related through corresponding instances, but only through the common task. Before mapping in the MOMAP algorithm, the scaling of features of all spaces is required. The scaling aims to normalize the features of all spaces to the same range. Then mapping is performed by translating the means of each class to zero. Next, the rotation of the classes to fit each other is done using a rotation matrix. Finally, we have to translate the means of the mapped space to the final space. The framework performance is demonstrated on activity recognition data taken from wearable sensors. The task consists of the classification of 5 different activities, where the source domain contains different but related sensor readings as compared to the target. The method was compared against Support Vector Machine (SVM) as a baseline method. MOMAP classifier outperforms the SVM classifier in every test.

Feuz et al. proposed a novel heterogeneous transfer learning technique called Feature-Space Remapping (FSR) [34]. FSR can handle the different feature spaces without the use of co-occurrence data (correlated data). FSR maps the data to a different feature space using part of the labeled data from the target domain to infer relations to the source domain. FSR requires a one-time manual specification of meta-features and then can be applied to map multiple source and target domains. The features from target feature space are mapped to a different source feature space. To achieve feature mapping, they learn a mapping from each dimension in the target feature space to a corresponding dimension in the source feature space. FSR computes the average similarity between the source and target meta-feature values for each pair between features from source and target feature spaces. The similarity is calculated between two meta-features as the absolute value of the difference between meta-feature values divided by the maximum possible difference between the meta-features. As a product, we get many-to-one mapping. FSR method was presented in three variants: Informed Feature-Space Remapping (IFSR) and Uninformed Feature-Space Remapping (UFSR) depending on the presence of target labels and Ensemble Learning via Feature-Space Remapping (ELFSR) to accept data from multiple source domains [34]. IFSR is used when limited target labels are provided. UFSR is used when no labeled target training data is provided. ELFSR extends these models by accepting data from multiple source domains and creates an ensemble learning scenario. The performance of FSR and its variants were evaluated by authors in the activity recognition domain and the document classification domain.

Sukhija et al. in [116] proposed Supervised Heterogeneous Domain Adaptation via Random Forest (SHDA-RF). This method was devoted to activity recognition tasks but can be used for any standard heterogeneous domain adaptation. The proposed algorithm assumes that the source and target domains have different features that characterise data partitions with similar label distributions. With this assumption, the shared label distributions can act as a pivot to learn a sparse feature mapping between the feature spaces of the source and target domains. This sparse feature mapping represents target features as linear combinations of source features. The goal of this is to create a mapping of the source domain features to the target domain features with the shared labels distribution acting as the bridge. To bridge the source and target domain, the random forest models are used. The individual decision trees of the forest are constructed using random subsets of features. Every path in a decision tree leading to a partition of data is associated with a certain shared label distribution [116, 27].

Deep Transfer Learning

Finally, we consider it important to point out deep transfer learning [118], which is a new trendy area of transfer learning. It studies how to utilize knowledge from different fields, in our case datasets, using deep neural networks. In some domains, it is tough to obtain large and well-annotated datasets that are needed to train deep learning models and so the usage of these methods is limited or even impossible due to their dependence on massive training data.

Deep transfer learning deals with a combination of knowledge from different tasks and domains by using deep neural networks. Based on the techniques used in deep transfer learning, the deep transfer learning approaches can be divided into four categories as follows [118]:

- instance-based approaches utilize instances in source domain by appropriate weight,
- mapping-based approaches map instances from two domains into a new data space with better similarity,
- network-based approaches reuse the part of the network pre-trained in the source domain,
- adversarial-based approaches use an adversarial way to find transferable features suitable for both domains.

For more details about each category, see [118]. We found adversarial-based approaches as the most suitable for our task.

Adversarial-based deep transfer learning uses an adversarial learning strategy to find transferable features that are suitable for both the source and target domains. The approach is inspired by generative adversarial networks introduced by Goodfellow et al. in [41]. The basic idea behind generative adversarial networks is to set up a minimax game between two players, which will be described in more detail in Section 3.4.

We see the idea of deep transfer learning as very promising, because neural networks are very flexible in their ability to represent the desired mapping between different feature spaces.

2.3 Studied Tasks

In this thesis, our attention is paid to two tasks:

1. missing features reconstruction,
2. unsupervised domain adaptation.

In Table 2.2, we overview these tasks and our contributions.

Missing Features Reconstruction

Let us start with the first task - missing features reconstruction. This task covers the situation, when the overlapping features have the same joint distribution and some features are missing. Consider using an already trained prediction model in a situation when new data intended for prediction arrive. However, these data may differ from the data used for

Table 2.2: Thesis main chapters overview.

	Chapter 3	Chapter 4
Problem instance	missing features reconstruction	domain adaptation
Main challenge	design of universal method	eliminate drawbacks of existing methods
Type of contribution	methodology, analysis, model	model
Type of learning	supervised	unsupervised
Publications	[A.1, A.3]	[A.2]

training of the prediction model. These difference can be caused by several reasons, one of them can be a different data source or we can encounter a damaged dataset. If we consider the situation with a damaged dataset devoted to prediction, this damage is represented by missing data within a dataset. It can be either on the level of instances (e.g., missing entire entries, some values missing, corrupted data, etc.) or on the level of features (e.g., entire features missing). Missing entire features have a negative impact on the task being solved and may have fatal consequences which could mean that the task cannot be performed. If we are using a dataset for classification tasks, the ability to classify accurately decreases with an increasing level of damage. We find this problem to be essential.

The solved task can be summarized as follows. Prediction model is trained with satisfying performance using a dataset without missing features. We want to predict the labels of the other unseen dataset, which is unlabeled. However, this unlabeled dataset has entire missing features. When we are training the prediction model, we can modify the feature set. For example, we can ignore the missing features or delete them. Compared to that, at the time of prediction, it is often impossible to change the model, so it is necessary to map the target data representation to the source data representation.

Missing values in datasets can have various causes, such as a faulty manual data entry process, equipment errors, or incorrect measurements. Based on [32], there are four major approaches dealing with missing data:

1. discard the examples that contain missing values,
2. ignore missing values,
3. encode missing values with some dummy values such as zeros,
4. perform reconstruction of missing values.

The first approach is the simplest way how to deal with missing values. However, this solution is useful when only a few records contain missing data and when analysis of the remaining complete data will not lead to serious bias during prediction. Also if we face the problem of missing entire features in a production dataset, it is not possible to discard examples containing missing data.

Ignoring missing data means that all the missing data remains unreplaced. However, only some types of prediction models are able to handle this treatment (e.g. random forest).

The third approach proposes encoding missing data into a new value, which can be represented by a dummy value (i.e. trivial) like zero. However, this trivial imputation often leads to various problems during prediction [105].

If we consider an already trained prediction model and a significant number of missing data, it is beneficial to analyse methods of missing data reconstruction. By reconstruction, we mean imputation of missing values.

While handling missing data, it may be important to know missing data mechanisms - why and how they are missing. Through this thesis, we consider the missing completely at random (MCAR) mechanism of missingness. It means that the distribution of an example having a missing value for a feature does not depend on either the observed data or the missing data. The missing values can be predicted by using the values in non-missing features. Further information about mechanisms of missingness can be found in [97].

In data imputation, there is no best way how to deal with missing data; no general method exists. To beat this transfer learning application domain, we analyzed non-generative and generative methods which may be utilized for missing features reconstruction. As non-generative methods, we see imputation methods which impute missing value non-randomly based on values in non-missing features. We choose k-nearest neighbour, linear regression, imputation by autoencoders, and imputation by gradient boosted trees as representative methods. These methods are experimentally explored. Multiple imputation by chained equations and imputation by generative adversarial networks were chosen as representative generative methods. The generative methods enable one to sample from the distribution conditioned on the observed features and thus get information about the uncertainty in imputed values.

More details on missing features reconstruction, together with state-of-the-art methods, can be found in separate Chapter 3.

Unsupervised Domain Adaptation

Unsupervised domain adaptation was chosen as the second task. Domain adaptation is a specific sub-field of heterogeneous transfer learning, as stated in Definition 2.1.7, and is often represented by different image domains. A typical example of image domain differences is the domain shift [92], which means the distribution differences caused by change of origin conditions, e.g., background, location, pose changes, etc. Domain differences might be more severe when, for example, the source and target domains contain images of different types, such as photos, NIR images, paintings or sketches [23].

We want to reuse an already trained prediction model on some image domain to predict on another image domain. These image domains differ in feature representations, but we assume that both domains are rich with data. Also both domains can contain a sufficient amount of labels. The challenging part comes when pairs of corresponding images between domains do not exist. Due to this, there is no information how the images should be mapped to each other. This problem is called unsupervised domain adaptation and can be solved by transfer learning approaches which enable us to modify the domain in such a manner that allows it to be represented in the same way as the other domain, taking advantage of an already trained prediction model.

Transfer learning methods focused on domain adaptation can be categorized based on the use of neural networks. Csurka in [23] proposed two categories:

- shallow methods - they are not based on neural networks but rather on statistical theory,
- deep methods - they are based on neural networks augmented for domain adaptation.

There have been comprehensive surveys for both shallow and deep methods, see e.g. [23, 131, 135]. Based on [118], a promising way to solve this task is based on deep methods, which leverage deep networks to learn more transferable representations by embedding domain adaptation in the pipeline of deep learning.

In this thesis, we pay attention to deep methods and the possibility of their use in the mentioned tasks. Unsupervised domain adaptation is described in detail together with state-of-the-art methods in separate Chapter 4.

Chapter 3

Missing Features Reconstruction

This chapter is based on the following publications:

- Friedjungová, M.; Vašata, D.; Balatsko, M.; Jiřina, M. Missing Features Reconstruction Using a Wasserstein Generative Adversarial Imputation Network. In *Computational Science - ICCS 2020.* Springer International Publishing, Cham. 2020.

- Friedjungová, M.; Vašata, D.; Jiřina, M. Missing Features Reconstruction and Its Impact on Classification Accuracy. In *Computational Science - ICCS 2019.* Springer International Publishing, Cham. 2019.

- Friedjungová, M. *Asymmetric Heterogeneous Transfer Learning.* Ph.D. Minimum Thesis, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic, 2018.

In this chapter we study missing features reconstruction. The chapter is organized as follows. In Section 3.1, we introduce the reader to the missing features problem in the context of transfer learning. In Section 3.2, related work in this field is reviewed briefly. Then in Section 3.3, we propose linear and information imputability to provide ordering of several missing features to be imputed. This ordering helps obtain a more accurate imputation. Imputation methods researched in order to solve the task of missing features reconstruction are presented in Section 3.4 together with an introduction of our own imputation model called Wasserstein Generative Adversarial Imputation Network. Section 3.6 is devoted to the description of experiments performed, including the evaluation of their results. Finally, results are discussed in Section 3.7.

3.1 Missing Features Problem

When working with real-world datasets one of the standard problems that needs to be solved as part of the data preprocessing phase is dealing with missing data. The missingness of data can be represented by either individual missing data located only in some instances or by the absence of entire features.

To our best knowledge, not much attention is paid to the second situation where entire features are missing. There are no clear answers to questions such as how to face the situation, how the standard imputation methods will perform or if there is a need to approach this challenge in a different way. We are especially interested in how missing features impact the accuracy of a classification model, what possibilities of reconstructing missing entire features exist and how the model performs with imputed data.

We study these issues by experimentally comparing several state-of-the-art imputation methods in real-world scenarios where one needs to impute (i.e., reconstruct) entire features. One of the goals is to research the existence and possible impact of universal imputation models. Due to this, our attention is paid to more complex imputation models represented by autoencoders and generative adversarial networks. These models have a common advantage in that one does not need to know which features are missing in advance. On the contrary, regular imputation methods such as imputation by linear regression or by gradient boosted trees need to be trained for each combination of missing features separately.

A typical example where a universal imputation model is needed is the prediction of a classification model from sensor data, where a sensor breakdown leads to missing data in one or more features. Usually, the classification model itself is not able to handle the situation with missing features without a significant decrease in its performance. Furthermore, one typically does not know in advance which sensor is going to be broken. The best approach would be to retrain the classification model using data without missing features. However, in a production setting model retraining is impossible as the existing model is already deployed.

We consider a situation where the prediction model is trained on a complete preprocessed dataset with numeric features, and we study its accuracy on new unseen data with imputed missing features. The amount of missing data varies between 10% and 50%. Experiments are performed on ten real and two artificial datasets. The impact of imputation is measured by the classification accuracy of the best performing classification model out of six commonly used: logistic regression, multi-layer perceptron, k-nearest neighbors, naive Bayes, extreme gradient boosted trees [13], and random forest.

In terms of imputation methods, we compare non-generative models (k-nearest neighbors - k-NN, linear regression, extreme gradient boosted trees, and denoising autoencoder (DAE) [128]), to generative models (Multiple Imputation by Chained Equations (MICE) [7], Generative Adversarial Imputation Network (GAIN) [139], and Variational Autoencoder with Arbitrary Conditioning (VAEAC) [50]).

Moreover, we introduce our Wasserstein Generative Adversarial Imputation Network (WGAIN), a Wasserstein based modification of GAIN, also published in [A.1]. WGAIN is a generative imputation model and generally outperforms other presented models on the tested datasets.

In Section 3.3, we present our ordering methods: linear and information imputability, which are useful when we need to set up some ordering of several missing features to be imputed. There are two ways of accomplishing this task. The first is to impute all features simultaneously which can be done using a k-NN imputation model or using models based on neural networks. The second, which is usable for all other methods, is to apply the model sequentially one missing feature after another. However, to do this, our preliminary experiments showed that it is important to choose some order in which the features will be imputed. These ordering methods were published in [A.3].

3.2 Related Work

Currently, there exist many surveys presenting a summarization of methods for missing values imputation. For example, you can find them in [83, 52, 112, 142, 32, 1]. Most of them are more than five years old and focus on traditional imputation methods. However, one can see approaches based on deep learning as the state-of-the-art methods. Firstly, let us present papers that do not involve deep learning. After that we will review approaches based on deep learning.

A very good review of methods for imputation of missing values was provided by [32]. This study is focused on discrete values only with up to 50% missingness. They experimentally evaluated imputation methods such as hot-deck, naive Bayes, polynomial multiple regression and mean imputation, on 15 datasets used in 6 classifiers. Their results show that all imputation methods except for mean imputation improve classification error when missingness is more than 10%. The decision tree and naive Bayes classifiers were found to be missing data resistant, however other classifiers benefit from the imputed data.

In [142] performance of imputation methods was evaluated on datasets under different scales of missingness (up to 50%). Two scenarios were tested: values are missing only during the prediction phase, and values are missing during both the training and prediction phases. In this study, three classifiers were used: a decision tree, k-NN and a Bayesian network. Imputation by mean, k-NN, linear regression and ensemble were used as imputation methods. The experimental results show that the presence of missing values always leads to performance reduction of the classifier, no matter which imputation method is used to deal with missing values. However, if there are no missing data in the training phase, imputation methods are highly recommended at the time of prediction to eliminate the decrease in classification accuracy.

An iterative k-NN imputation method was introduced by Zhu et al. in [147]. The method combines weighted k-NN imputation and the grey relational analysis [73] in the domain of trash pickup logistics management systems. The grey relational analysis is a technique of measurement proposed to determine the correlation between referential sequences and a set of compared observations called grey relational grade, which is calculated instead of similarity measures such as Euclidean distance and others. The proposed method was compared with other imputation methods. It reaches higher performance than other methods in imputation accuracy and converges faster.

Arroyo et al. in [6] present imputation of missing values of Ozone in real-life datasets using imputation methods such as multiple linear and nonlinear regression, multi-layer perceptron, and radial basis function networks. This paper proves the usefulness of artificial neural networks.

Another state-of-the-art method is Multiple Imputation by Chained Equations (MICE) [124, 123], sometimes called Multivariate Imputation by Chained Equations. The method is based on multiple imputation which involves filling in the missing values multiple times, creating multiple "complete" datasets. To obtain single imputation the datasets should be pooled, e.g. by mean. This approach brings competitive results as can be seen in [50, 86, 115] and also in our research [A.1, A.3, A.6].

Approaches based on deep learning have been under active development for the last few years. They use many variants of neural networks starting from multi-layer perceptron, e.g., in [6, 112], which were also mentioned in the papers surveyed above. A more advanced approach is based on the autoencoder as a specific kind of neural network aiming to reconstruct inputs on its outputs. One of the most commonly used models is the denoising autoencoder (DAE) [128], e.g., [39, 18, 8, 31, 134]. Typically, autoencoders are used in a discriminative way, meaning they impute a single value, which is deterministic once the network is trained. An example of their usage in a generative way can be found in [39].

On the other hand, the most recent research focuses on generative models which are able to sample from the distribution conditioned on the observed features and thus get information about the uncertainty in imputed values. There are two groups of deep learning generative models. In the first group, there are models based on the variational autoencoder (VAE) [58] and its conditional alternations, see [140, 79, 75, 113]. In this group, some of the most successful imputation models are VAEAC [50] and HI-VAE [86]. The second group contains models based on the generative adversarial network (GAN) [41]. Notably, one can encounter them in image reconstruction tasks (also called image inpainting), see [66, 14, 90].

Let us start with the representatives of the first group. HI-VAE [86], i.e., heterogeneous-incomplete VAE framework is suitable for fitting incomplete heterogeneous data. It means that HI-VAE includes likelihood models for numerical and nominal features, and allows to estimate and potentially impute missing data accurately. The framework is based on the VAE architecture. The authors also assume the Gaussian prior on latent space and a generative model that fully factorizes for every heterogenous dimension in the data. Among other methods such as imputation by mean value and general latent feature model [121], HI-VAE was also compared to the already mentioned MICE and GAIN [139], which is described below. In most cases, HI-VAE achieved competitive results.

Another model based on VAE is VAEAC [50] which uses VAE conditioned on an arbitrary subset of observed features and then samples the remaining features as a whole. The conditioning features affect the prior on the latent Gaussian variables which are used to generate unobserved features. The model is trained using stochastic gradient variational Bayes [58]. The power of VAEAC was shown on data imputation, as well as on image inpainting problems and compared to MICE, GAIN and MissForest (described in [115]).

Let us continue with the second group based on GANs. One of the most prominent methods in this group is the Generative Adversarial Imputation Network (GAIN) [139]. The basic idea of GAN used in imputation is the use of the generator-discriminator mechanism to achieve the learning of the desired distribution. The generator observes some components of a real data vector, imputes the missing components conditioned on what is observed, and outputs a completed vector. The discriminator then takes a completed vector and attempts to determine which components were observed and which were imputed. The invention of GAIN lies in additional information in the form of a hint vector provided to the discriminator. The hint vector contains information about the missing data in the original sample and so the discriminator is able to focus only on particular components. This hint also helps the generator learn the data distribution according to the true one. The GAIN forms the base for our modification of the imputation method based on Wasserstein GAN [5], which is introduced in Section 3.5.

Only recently, GAIN was outperformed by the previously mentioned VAEAC and HI-VAE. However, for numeric features, HI-VAE achieves a comparable error to the rest of the methods [86]. Therefore we have chosen only VAEAC for the experimental comparison. Anyhow, the fact that the development of new methods is so fast today has to be considered. The reader can come across methods such as MisGAN [14], MIWAE [78], combination of generative imputation and stochastic prediction [53], and many others. Methods we focus on represent state-of-the-art methods.

We note that there is a lack of a comprehensive comparison of contemporary methods (e.g., MICE versus autoencoders or generative adversarial networks) or methods which can be used for missing feature reconstruction in general. This can be caused by the fact that when only few data are missing, satisfying results are reached using ordinary methods such as k-NN imputation [51] or MICE [122, 105]. In recent research neither of the described methods performs significantly better than the others.

3.3 Multiple Features Imputability

When imputing several missing features (i.e. entire missing vectors), there are two ways of reaching successful results using the previously mentioned methods. The first is to impute all features simultaneously which can be done using k-NN and models based on neural networks. The second, which is usable for all other methods, is to apply the model sequentially one missing feature after another. However, to do this, it is important to choose some order in which the features will be imputed. Based on preliminary experiments we focus on an ordering where the most easy to impute features are treated first.

In the case of MICE such a sequential imputation is not needed. The reason is that MICE is already prepared for multiple features imputation using an internal chained equation approach [122, 105].

In the following subsections we present our two approaches to ordering missing features for imputation - linear and information imputability. The proposed approaches are also published in [A.3].

3.3.1 Linear Imputability

A simple way of measuring how easy the imputation of a feature will be is to use the multiple correlation coefficient [3]. The multiple correlation coefficient $\rho_{X,\boldsymbol{X}'}$ between a random variable X and a random vector $\boldsymbol{X}' = (X'_1, \dots, X'_n)^T$ is the highest correlation coefficient between X and a linear combination $\alpha_1 X'_1 + \ldots + \alpha_n X'_n = \boldsymbol{\alpha}^T \boldsymbol{X}'$ of random variables $X'_1, \dots, X'_n$,

$$\rho_{X,\boldsymbol{X}'} = \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \rho_{X,\boldsymbol{\alpha}^T \boldsymbol{X}'}.$$

It takes values between 0 and 1, where $\rho_{X,\boldsymbol{X}'} = 1$ means that the prediction by linear regression of X based on $\boldsymbol{X}'$ can be done perfectly and $\rho_{X,\boldsymbol{X}'} = 0$ means that the linear regression will not be successful at all.

When $X_1, \dots, X_p$ are the p features, we call the multiple correlation coefficient $\rho_{X_i,\boldsymbol{X}_{-(i)}}$ between X_i and a random vector of other features $\boldsymbol{X}_{-(i)} = (X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_p)^T$ the *linear imputability* of feature X_i.

The estimation of the linear imputability is based on the following expression

$$\rho^2_{X_i,\boldsymbol{X}_{-(i)}} = \frac{\operatorname{cov}(X_i, \boldsymbol{X}_{-(i)})^T \big(\operatorname{cov}(\boldsymbol{X}_{-(i)})\big)^{-1} \operatorname{cov}(X_i, \boldsymbol{X}_{-(i)})}{\operatorname{var}(X_i)},$$

where $\operatorname{cov}(X_i, \boldsymbol{X}_{-(i)})$ is a vector of covariances between X_i and the remaining features $X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_p$, and $\operatorname{cov}(\boldsymbol{X}_{-(i)})$ is a $(p-1) \times (p-1)$ variance-covariance matrix of covariances between remaining features.

If we want to impute $k+1$ features, say $X_i, X_{i+1}, \dots, X_{i+k}$, in the first step we choose $X_j, i \leq j \leq i+k$ such that $\rho_{X_j,\boldsymbol{X}_{-(i,\dots,i+k)}}$ is the largest, where $\boldsymbol{X}_{-(i,\dots,i+k)} = (X_1, \dots, X_{i-1}, X_{i+k+1}, \dots, X_p)^T$ is a vector of the remaining features. Then, in the next step, we repeat the process where X_j is taken as a known feature. Thus we choose

$X_l, i \leq l \leq i+k, l \neq j$ such that its linear imputability with respect to random vector $\boldsymbol{X}_{-(i,\dots,j-1,j+1,\dots,i+k)}$ is the largest. We continue this way until all missing features are imputed.

3.3.2 Information Imputability

Linear imputability is a simple measure of how the linear regression imputation will perform. However, when one uses more sophisticated imputation models based on multi-layer perceptron or extreme gradient boosted trees that can handle non-linear dependencies, linear imputability may not be suitable.

Hence we propose another way of measuring imputability which is based on a particular result from Information theory. If a feature X_j is predicted by an estimator $\hat{X}_j$ based on other features represented by a vector $\boldsymbol{X}_{-(j)}$, i.e. $\hat{X}_j \equiv \hat{X}_j\big(\boldsymbol{X}_{-(j)}\big)$, then it can be shown (see [22]) that

$$\mathrm{E}\left(X_j - \hat{X}_j\right)^2 \geq \frac{1}{2\pi\mathrm{e}}\mathrm{e}^{2H(X_j|\boldsymbol{X}_{-(j)})},$$

where $H(X_j|\boldsymbol{X}_{-(j)})$ is the conditional (differential) entropy of X_j given $\boldsymbol{X}_{-(j)}$.

Hence the lower bound of the expected prediction error is determined by the conditional entropy $H(X_j|\boldsymbol{X}_{-(j)})$. The greater the entropy is the worse predictions one can achieve at best when estimating X_j from other features. Thus one may measure imputability through the value of conditional entropy multiplied by -1 in order to have larger values which correspond to better imputability. Hence we define the *information imputability* as a value of $-H(X_j|\boldsymbol{X}_{-(j)})$.

The process of multiple feature imputation is now exactly the same as it was using linear imputability. One first imputes the feature with the largest information imputability. The only difference is that in the second and all subsequent steps only the known features should be used for conditioning (i.e. excluding features imputed in the previous steps) since one is not able to get any new information no matter what model will be used for the imputation.

The problem that strongly limits the practical usage of information imputability is the estimation of conditional entropy. Even the most recently proposed estimators in [74, 114] suffer from the curse of dimensionality. This is due to the fact that all these estimators are based on the k-NN approach introduced by Kozachenko and Leonenko in [59]. As our numerical experiments indicate, the method is limited to approximately five features depending on the underlying joint distribution.

3.4 Used Imputation Methods

Plenty of methods for missing data imputation have been designed. They perform differently on various datasets and in practice the most suitable imputation method for a given dataset is usually chosen according to the evaluation of the average performance of each method in the phase of training [103].

In this section the methods used to solve the scenario of missing features reconstruction are described. We distinguish between non-generative and generative methods.

3.4.1 Non-Generative Methods

As non-generative methods we see imputation methods which impute missing values non-randomly based on values in non-missing features. They impute the simple value deterministically, i.e. they are imputing one value for each missing value. Based on the reviewed papers, the following methods were chosen for experiments.

k-Nearest Neighbors Imputation

k-nearest neighbors (k-NN) as an imputation method [51] takes a weighted average of the values from the k closest samples. The closest samples are found using Euclidean distance or other distances such as Manhattan, Mahalanobis, Pearson, etc. k-NN typically uses a weighted mean or a mode to fill in missing values. This method is very simple and can be used with mixed continuous and categorical data. There exist two big disadvantages to using k-NN as an imputation method - it becomes time-consuming when analyzing large datasets because it searches for the most similar instances through the whole dataset and it suffers from the curse of dimensionality.

Linear Regression Imputation

Another representative method is linear regression with ordinary least squares for estimating the unknown parameters in the linear regression model. It assumes that there is approximately a linear relationship between non missing features and missing features, through which we fit a linear regression model to predict missing values [70]. As a drawback, we face the need to train independent imputation models for each combination of missing features. However, it is not a big issue since the training of linear regression models is very fast.

Extreme Gradient Boosted Trees Imputation

The idea of gradient boosting originates from [35]. Boosting alone is an ensemble technique in which predictors are not made independently, but sequentially. It means that the predictors learn from mistakes of previous predictors. Gradient boosting can be applied on various models, in this work, we consider the very popular extreme gradient boosted trees (see [13] for a more detailed description).

Imputation by Denosing Autoencoder

An autoencoder (AE) [40] is a neural network which consists of the following parts: *encoder*, which takes the input $\boldsymbol{x} \in \mathbb{R}^d$ and maps it to the latent representation given by *code* $\boldsymbol{c} \in \mathbb{R}^k$ (the middle layer of the network), and the *decoder* which maps $\boldsymbol{c} \in \mathbb{R}^k$ back to $\boldsymbol{x'} \in \mathbb{R}^d$, so that $\boldsymbol{x}$ and $\boldsymbol{x}'$ are close to each other.

To achieve this behavior one minimizes the reconstruction error typically given as a squared loss:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x'}) = \|\boldsymbol{x} - \boldsymbol{x'}\|^2.$$

The denoising autoencoder (DAE) [128] is a modification of the AE, which receives noised data on the input and has to reconstruct the original data on the output. When DAE is trained for imputation, the input is modified such that randomly chosen features are zero, i.e. the input $\tilde{x}$ is

$$\widetilde{\boldsymbol{x}} = \boldsymbol{x} \odot \boldsymbol{\sigma},$$

where $\boldsymbol{x}$ is the original input which has to be reconstructed on the output, $\odot$ denotes element-wise multiplication, and components of a random binary vector $\boldsymbol{\sigma}$ have a Bernoulli distribution with a parameter corresponding to the inverse degree of training missingness.

DAE architectures could be divided into two types: compressing and overcomplete. In a compressing DAE, the code has a smaller dimension than the input. Usually, the compressing DAE encoder part is a fully connected network, where layers are built decrementally, each subsequent layer has h fewer neurons than the previous one. The decoder is built symmetrically as a fully connected network, where each subsequent layer has h more neurons than the previous one. h is an architecture hyper-parameter, which represents a compressing increment. Another important architecture hyper-parameter is the encoder/decoder layers count. Figure 3.1 shows the schematic structure.

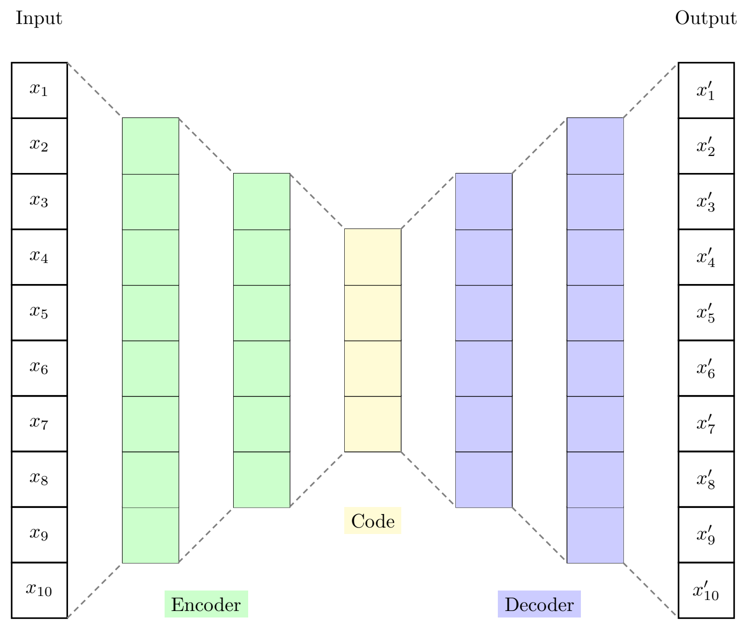


Figure 3.1: Compressing autoencoder with two encoder/decoder layers and $h = 2$.

The overcomplete DAE, unlike the compressing DAE, has a code of a higher dimension than the input layer, which allows for a more informative latent representation. It also assumes symmetrical encoder and decoder parts, which are constructed in a similar way to the compressing DAE, but here the encoder incrementally increases layer size starting from the input and the decoder compresses the code back to input dimensionality. Due to the overcomplete architecture, without any regularization it may suffer from overfitting. Figure 3.2 shows the schematic structure.

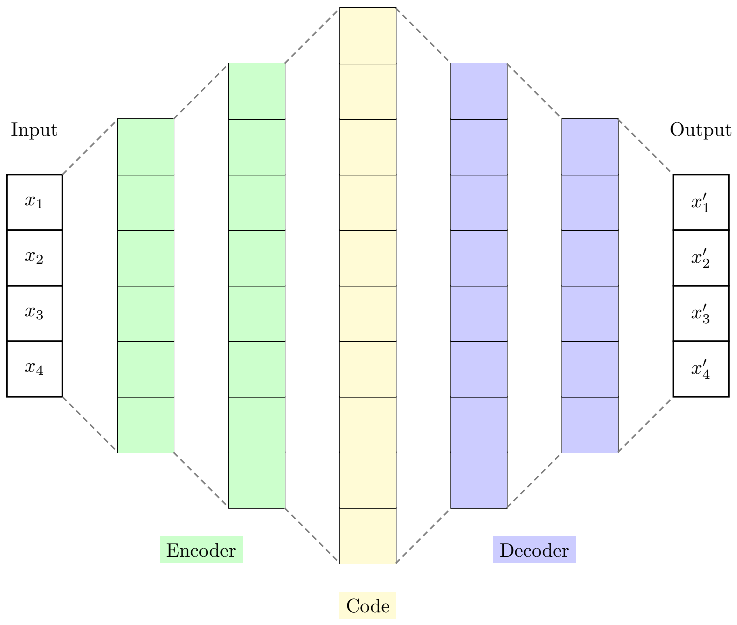


Figure 3.2: Overcomplete autoencoder with two encoder/decoder layers and $h = -2$.

3.4.2 Generative Methods

Generative methods encompass models that are able to impute missing values by sampling from a learned data distribution. All generative methods can be used in multiple imputation settings [122]. Based on a review of papers, the following methods were chosen for our experiments.

Multiple Imputation by Chained Equations

The Multiple Imputation by Chained Equations (MICE) [124, 123] does not simply aim to impute missing values using the most fitting value, it also tries to preserve some of the randomnesses of the single data distribution. This is being accomplished by performing several imputations, i.e., multiple imputation [98], each one based on randomly imputed values from the prior distribution. The chained equation process consists of four steps:

1. Initial simple imputation, e.g. by mean value, is performed for every missing feature in the damaged dataset.

2. The value imputed in the previous step for one selected feature is set back to the missing value.

3. Missing value created in step 2 is imputed by the result of a stochastic Bayesian regression model learned using the other features.

4. Steps 2 and 3 are repeated for each feature with originally missing data. The cycling through each of the missing features constitutes one iteration. At the end of one iteration all of the missing features have been replaced with predictions from regression models.

This process is repeated in several iterations (the usual number is ten [7]). However, it can be changed in order to be suitable for particular data. A more detailed description can be found in [7]. The MICE comes up with very good results and is currently one of the best-performing methods [122].

Generative Adversarial Imputation Network

Before we examine another imputation model, let us describe the necessary background of the generative adversarial network (GAN) [41]. Currently, GANs are widely used for generating new images or for their enhancements. To our best knowledge, the idea of using GANs for handling missing data is relatively new [66, 14, 139, 107]. It is a very promising approach that we want to focus on more.

A standard GAN presented in [41] consists of two neural networks called generator G and discriminator D, where both are trained in an adversarial mode. Adversarial training lies in a minimax two-player game, in which the networks are trying to beat each other. G is trained to generate new unseen data, while D evaluates them for authenticity. In other words, D decides whether the data instance belongs to the original training dataset or is generated (i.e. fake). The scheme of adversarial training is depicted in Figure 3.3.

To learn the generator's distribution p_g over data $\boldsymbol{x}$, we define a prior on input noise variables $p_z(\boldsymbol{z})$, and initialize a mapping to data space $G(\boldsymbol{z})$, as stated in [41]. The discriminator's output $D(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ came from the data rather than p_g. It is trained to maximize the probability of the correct distinction of both training examples and samples from generator G. Simultaneously, G is trained to minimize $\log(1-D(G(\boldsymbol{z})))$. Hence, both networks optimize the following value function $V(G,D)$:

$$V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].$$

Considering the imputation use-case, the goal of G is to accurately impute missing features, and the goal of D is to distinguish between the observed and imputed features. The D is trained to minimize the classification loss when classifying which feature was observed and which has been imputed.

An important GAN-based model for imputation is the Generative Adversarial Imputation Network (GAIN) [139]. GAIN builds on and adapts the standard GAN architecture extended in a way where the generator G takes the vector of real data combined with the

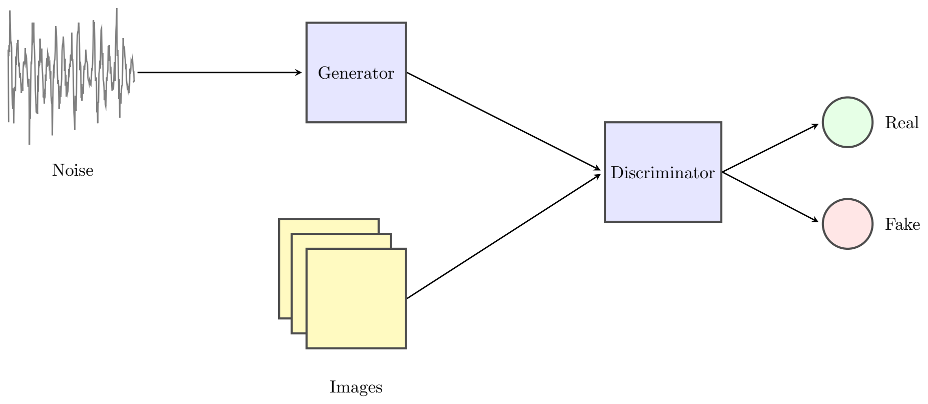


Figure 3.3: Standard GAN.

vector of random values and mask indicating which values in real data are actually missing. The imputed data is fed back to the discriminator D together with additional information in the form of hints narrowing its focus to randomly chosen features only and its aim is to figure out which of these features were originally missing.

Original data $\boldsymbol{X}$, a binary mask $\boldsymbol{M}$ and a random noise vector $\boldsymbol{Z}$ are taken as inputs of G. The binary mask $\boldsymbol{M}$ indicates which components of $\boldsymbol{X}$ are actually missing. The output of G is a vector $\boldsymbol{X'} = G(\boldsymbol{X}, \boldsymbol{M}, \boldsymbol{Z})$ whose components provide values to be imputed. This output is then combined with the original data vector to produce vector $\tilde{\boldsymbol{X}}$ that has non-missing components from $\boldsymbol{X}$ and missing components from $\boldsymbol{X'}$. There is a difference compared to the standard GAN when the output of the generator is completely real or completely fake. In GAIN, the output of G consists of some components that are observed and some that are imputed. The discriminator D then distinguishes which components of $\tilde{\boldsymbol{X}}$ are observed and which are imputed, i.e. its aim is to guess $\boldsymbol{M}$. The output of the discriminator is not a single value, but it is a vector of probabilities of components, i.e. estimation of a binary mask. It means that D is trained to maximize the probability of correctly predicting $\boldsymbol{M}$ and G is trained to minimize the probability of D predicting $\boldsymbol{M}$. To improve the process of learning D, additional information in the form of a hint vector $\boldsymbol{H}$ is provided. The hint vector $\boldsymbol{H}$ is derived from the original mask $\boldsymbol{M}$ of the damaged components such that most of the values are preserved and others randomly sampled are set to be 0.5. In essence, the discriminator is provided with an incomplete mask and its aim is to fill in a decision about the few randomly sampled components. In [139], GAIN significantly outperformed state-of-the-art imputation methods.

Variational Autoencoder with Arbitrary Conditioning

Variational autoencoder (VAE) [58] models inherit the autoencoder architecture, but make strong assumptions concerning the normal distribution of latent space. The idea behind it is that we can generate new data by decoding values that are randomly sampled from the latent space. However, the quality and relevance of the generated data depend on the restrictions of the latent space. This is achieved through the encoder which yields in a vector of means $\boldsymbol{\mu}$ and a vector of standard deviations $\boldsymbol{\sigma}$. To ensure that the distribution on the encoder's output $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ approximates the standard normal distribution with independent components, the Kullback–Leibler divergence [62] is used as another loss function. In VAE, the loss function is then composed as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x'}) = \|\boldsymbol{x} - \boldsymbol{x'}\|^2 + D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})||\mathcal{N}(\mathbf{0}, \mathbf{1})),$$

where $\boldsymbol{x}$ represents the input of the encoder and $\boldsymbol{x'}$ stands for the predicted output of the decoder. See Figure 3.4 for the structure of a VAE architecture.

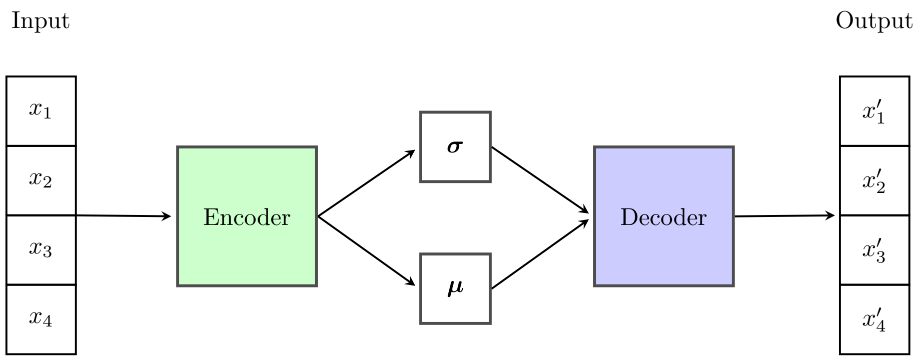


Figure 3.4: VAE architecture schematic structure.

Ivanov et al. in [50] introduced an imputation model based on VAE that can be conditioned on an arbitrary subset of observed features and then samples the remaining features in "one shot". The model is called Variational Autoencoder with Arbitrary Conditioning (VAEAC) and tackles the problem of learning all conditional distributions of the form $p(\boldsymbol{x}_I|\boldsymbol{x}_{U\setminus I})$, where U is the set of all features and I is its arbitrary subset [50]. The conditioning features affect the prior on the latent Gaussian variables which are used to generate unobserved features. The training of this network is performed by Stochastic Gradient Variational Bayes (SGVB) [58]. The VAEAC is recommended to be used in scenarios of feature imputation or image inpainting, in which the goal is to fill in an unobserved part of an image with artificial content in a realistic way.

Wasserstein Generative Adversarial Imputation Network
As the last model, we introduce in a separate Section 3.5 our imputation model called Wasserstein Generative Adversarial Imputation Network (WGAIN). The model has some benefits over other compared methods, shown experimentally in Section 3.6.

3.5 Wasserstein Generative Adversarial Imputation Network

In this section, we introduce our imputation model, the WGAIN published in [A.1], as a modification of GAIN adapting the discriminative approach from Wasserstein GAN [5].

Let us denote $\mathcal{X} = \mathbb{R}^d$ the d-dimensional feature space and let $\boldsymbol{X} = (X_1, \dots, X_d)$ be a random vector with values in $\mathcal{X}$ whose distribution is denoted by $\mathrm{P}(\boldsymbol{X})$. Let the mask be a random binary vector $\boldsymbol{M}$, i.e., random vector with values in $\{0,1\}^d$. The mask corresponds to unobserved values of $\boldsymbol{X}$ so that the value 0 of its jth component means that the jth feature of X_j is missing and the value 1 means that the jth feature of X_j is not missing. The distribution of $\boldsymbol{M}$ corresponds to the distribution of missingness in the data. Let us further denote by $\tilde{\boldsymbol{X}}$ the vector $\boldsymbol{X}$ having zeros in place of missing values given by

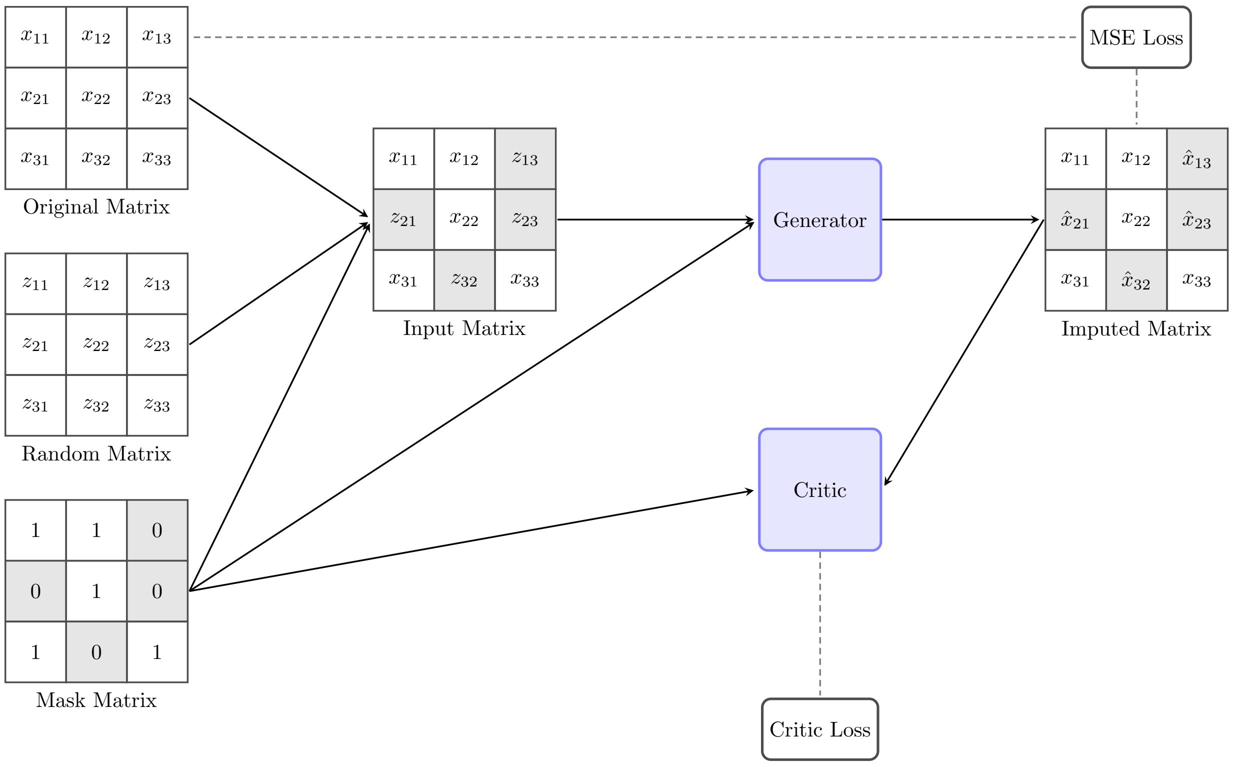
$$\tilde{\boldsymbol{X}} = \boldsymbol{X} \odot \boldsymbol{M},$$

where $\odot$ denotes element-wise multiplication. Our aim is to impute missing values in $\tilde{\boldsymbol{X}}$ based on information from non-missing features of $\tilde{\boldsymbol{X}}$ and $\boldsymbol{M}$. This is done in a generative way and it means that we want to learn the conditional distribution $\mathrm{P}(\boldsymbol{X} | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$ of $\boldsymbol{X}$ given $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}$ and $\boldsymbol{M} = \boldsymbol{m}$. To do this let $\boldsymbol{Z}$ be a random vector with independent identically distributed components having normal distribution $\mathcal{N}(0, \sigma^2)$ with variance σ^2 and define

$$\tilde{\boldsymbol{X}}_{\boldsymbol{Z}} = \boldsymbol{Z} \odot (1 - \boldsymbol{M}) + \boldsymbol{X} \odot \boldsymbol{M},$$

i.e. $\tilde{\boldsymbol{X}}_{\boldsymbol{Z}}$ is $\tilde{\boldsymbol{X}}$ with missing components replaced by normal random variables.

The WGAIN model consists of two parts, the generator g and the critic f, both represented by deep neural networks. The generator g is constructed as a mapping $g : \mathcal{X} \times \{0,1\}^d \to$



x_{11} x_{12} x_{13}
x_{21} x_{22} x_{23}
x_{31} x_{32} x_{33}
Original Matrix
z_{11} z_{12} z_{13}
z_{21} z_{22} z_{23}
z_{31} z_{32} z_{33}
Random Matrix
1 1 0
0 1 0
1 0 1
Mask Matrix
x_{11} x_{12} z_{13}
z_{21} x_{22} z_{23}
x_{31} z_{32} x_{33}
Input Matrix
Generator
Critic
Critic Loss
MSE Loss
x_{11} x_{12} $\hat{x}_{13}$
$\hat{x}_{21}$ x_{22} $\hat{x}_{23}$
x_{31} $\hat{x}_{32}$ x_{33}
Imputed Matrix

Figure 3.5: WGAIN structure and mini-batch data flow.

$\mathcal{X}$ so that

$$\hat{\boldsymbol{X}}_{\boldsymbol{Z}} = g(\tilde{\boldsymbol{x}}_{\boldsymbol{Z}}, \boldsymbol{m}) \odot (1 - \boldsymbol{m}) + \tilde{\boldsymbol{x}} \odot \boldsymbol{m}$$

is a random vector whose conditional distribution $\mathrm{P}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}} | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$, determined by the distribution $\mathrm{P}(\boldsymbol{Z})$ of $\boldsymbol{Z}$, should be close to the conditional distribution $\mathrm{P}(\boldsymbol{X} | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$. Note that $g(\tilde{\boldsymbol{x}}_{\boldsymbol{Z}}, \boldsymbol{m})$ is a random vector corresponding to $\tilde{\boldsymbol{x}}$ with all missing components imputed.

In order to train it, we employ the standard squared loss function

$$\mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{x}}_{\boldsymbol{z}}, \boldsymbol{x}) = \|\hat{\boldsymbol{x}}_{\boldsymbol{z}} - \boldsymbol{x}\|^2,$$

forcing the output $\hat{\boldsymbol{X}}_{\boldsymbol{Z}}$ to be close to the original data $\boldsymbol{X}$. To improve the performance of the generator, one may introduce a discriminator trying to find out which components of $\hat{\boldsymbol{X}}_{\boldsymbol{Z}}$ were imputed and use the discriminator for adversarial training. This approach was introduced in GAIN [139] and is the base of WGAIN.

Here, we present a similar mechanism of improving the performance of the generator. It is based on the Earth-Mover's (EM) distance [127, 99] between two probability distributions $\mathrm{P}(X), \mathrm{P}(Y)$ defined by

$$W\big(\mathrm{P}(X), \mathrm{P}(Y)\big) = \inf_{\gamma \in \boldsymbol{\Pi}(\mathrm{P}(X), \mathrm{P}(Y))} \mathrm{E}_{(X,Y)\sim\gamma} \|X - Y\|,$$

where $\boldsymbol{\Pi}(\mathrm{P}(X), \mathrm{P}(Y))$ denotes the set of all joint distributions (X, Y) whose marginals are respectively $\mathrm{P}(X)$ and $\mathrm{P}(Y)$. The term $\mathrm{E}_{(X,Y)\sim\gamma}\|X - Y\|$ might be understood as a measure of how much probability mass has to be transported in order to transform the distributions $\mathrm{P}(X)$ into the distribution $\mathrm{P}(Y)$ when the joint distribution is γ. The EM distance can thus be seen as the cost of the optimal transport plan, see [5] and references therein for more details. The EM distance is usually expressed using the Kantorovich-Rubinstein duality as

$$W\big(\mathrm{P}(X), \mathrm{P}(Y)\big) = \sup_{\|f\|_L \le 1} \mathrm{E}_{X\sim\mathrm{P}(X)} f(X) - \mathrm{E}_{Y\sim\mathrm{P}(Y)} f(Y), \tag{3.1}$$

where $\|f\|_L$ means that f is Lipschitz continuous with Lipschitz constant 1 which might be changed to any constant K since it just multiplies $W\big(\mathrm{P}(X), \mathrm{P}(Y)\big)$ by the same constant.

In Wasserstein GAN one approximates (3.1) by training the neural network $f_{\boldsymbol{w}}$ parametrized with weights $\boldsymbol{w}$ in some compact space $\mathcal{W}$, thus enforcing Lipschitz continuity. The function $f_{\boldsymbol{w}}$ is called the *critic* and is trained to maximize the expectations difference in (3.1). For a single dimensional generator g trying to transform random variable Z so that it has the distribution $\mathrm{P}(X)$ one maximizes

$$\max_{\boldsymbol{w}\in\mathcal{W}} \mathrm{E}_{X\sim\mathrm{P}(X)} f_{\boldsymbol{w}}(X) - \mathrm{E}_{Z\sim\mathrm{P}(Z)} f_{\boldsymbol{w}}(g(Z)).$$

In our case we want to minimize the EM distance between $\mathrm{P}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}} | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$ and $\mathrm{P}(\boldsymbol{X} | \tilde{\boldsymbol{X}} = \tilde{\boldsymbol{x}}, \boldsymbol{M} = \boldsymbol{m})$. Hence, we take the mask $\boldsymbol{M}$ as the second argument of the critic as additional information to the first argument given by $\boldsymbol{X}$ with correct features behind the mask $\boldsymbol{M}$. The critic is therefore a mapping $f_{\boldsymbol{w}} : \mathcal{X} \times \{0,1\}^d \to \mathbb{R}$ trained to maximize

$$\max_{\boldsymbol{w} \in \mathcal{W}} \mathrm{E}_{\boldsymbol{X} \sim \mathrm{P}(\boldsymbol{X})} f_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{M}) - \mathrm{E}_{\boldsymbol{Z} \sim \mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}),$$

which is usually estimated by sample means from mini-batches. The overall structure of WGAIN is depicted in Figure 3.5.

3.5.1 Training

The critic $f_{\boldsymbol{w}}$ is used in adversarial training of both the generator g and the critic itself. There the generator and the critic play an iterative two-player minimax game where the critic wants to recognize the imputed values from the real ones and the goal of the generator is to trick the critic so it cannot recognize them. Moreover, the generator's output is tightened to the correct output by the squared loss function $\mathcal{L}_{\mathrm{MSE}}$.

Putting it all together, we have two objective functions to minimize. The first corresponds to training of the critic given by

$$J(f_{\boldsymbol{w}}) = \lambda_{f_{\boldsymbol{w}}} \Big(\mathrm{E}_{\boldsymbol{Z} \sim \mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}) - \mathrm{E}_{\boldsymbol{X} \sim \mathrm{P}(\boldsymbol{X})} f_{\boldsymbol{w}}(\boldsymbol{X}, \boldsymbol{M}) \Big),$$

where the weight $\lambda_{f_{\boldsymbol{w}}}$ enables one to increase or decrease the influence of the corresponding gradient. Second is the objective for the generator,

$$J(g) = -\lambda_g \, \mathrm{E}_{\boldsymbol{Z} \sim \mathrm{P}(\boldsymbol{Z})} f_{\boldsymbol{w}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{M}) + \lambda_{\mathrm{MSE}} \, \mathrm{E}_{\boldsymbol{X} \sim \mathrm{P}(\boldsymbol{X}), \boldsymbol{Z} \sim \mathrm{P}(\boldsymbol{Z})} \mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{X}}_{\boldsymbol{Z}}, \boldsymbol{X}),$$

where λ_g and λ_{MSE} are weights enabling one to strengthen or weaken the influence of the squared loss function. The optimization is done via alternating gradient descent, where the first step is updating the critic $f_{\boldsymbol{w}}$ and the second step is updating the generator g. Hence, when perfectly trained, the discriminator gives negative values for cases with imputed features and positive values for cases with true features. On the other hand, the generator entering the critic will be pushed to obtain large positive values of the critic as it gives to real values.

The pseudo-code of the WGAIN training is given in Algorithm 1.

3.6 Experiments

An experimental validation of WGAIN and comparison with other methods using ten real and two artificial publicly available datasets is given below. These datasets contain numeric data only and are devoted to the classification task. Their overview, together with the corresponding best performing classification models, is given in Table 3.2.

During the experiments, all datasets were divided as follows: 70% of data was used to train all classification and imputation models and 30% was used as a test set to evaluate imputation performance. The imputation models were trained to impute in scenarios

Algorithm 1: WGAIN training pseudo-code

```
Input: $\alpha$ - the learning rate; $w_{\max}$ - maximal norm used in clipping; $m$ - the
       mini-batch size; $\lambda_{f_{\boldsymbol{w}}}, \lambda_g, \lambda_{\text{MSE}}$ - weights of the objectives
Draw $m$ samples $\{\boldsymbol{x}_j\}_{j=1}^m$ from the dataset;
Draw $m$ samples $\{\boldsymbol{m}_j\}_{j=1}^m$ from the mask distribution;
Draw $m$ samples $\{\boldsymbol{z}_j\}_{j=1}^m$ from the normal distribution of $\boldsymbol{Z}$;
while not converged do
    $\tilde{\boldsymbol{x}}_{\boldsymbol{z}_j} \leftarrow \boldsymbol{z}_j \odot (1 - \boldsymbol{m}_j) + \boldsymbol{x}_j \odot \boldsymbol{m}_j$;
    $\hat{\boldsymbol{x}}_{\boldsymbol{z}_j} \leftarrow g(\tilde{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j) \odot (1 - \boldsymbol{m}_j) + \boldsymbol{x}_j \odot \boldsymbol{m}_j$;
    Update weights $\boldsymbol{w}$ of $f_{\boldsymbol{w}}$ using RMSProp with learning rate $\alpha$ and gradient
    $\nabla J(f_{\boldsymbol{w}}) = \lambda_{f_{\boldsymbol{w}}} \nabla \left[\frac{1}{m}\sum_{j=1}^m f_{\boldsymbol{w}}\big(\hat{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j\big) - \frac{1}{m}\sum_{j=1}^m f_{\boldsymbol{w}}\big(\boldsymbol{x}_j, \boldsymbol{m}_j\big)\right]$;
    Clip the norm of $\boldsymbol{w}$ by $\boldsymbol{w}_{\max}$;
    Update weights of $g$ using RMSProp with learning rate $\alpha$ and gradient
    $\nabla J(g) = \nabla \left[-\lambda_g \frac{1}{m}\sum_{j=1}^m f_{\boldsymbol{w}}\big(\hat{\boldsymbol{x}}_{\boldsymbol{z}_j}, \boldsymbol{m}_j\big) + \lambda_{\text{MSE}} \frac{1}{m}\sum_{j=1}^m \|\hat{\boldsymbol{x}}_{\boldsymbol{z}_j} - \boldsymbol{x}_j\|^2\right]$;
end
```

where randomly selected combinations of multiple features were missing. The amount of missingness varies from 10% to 50% of missing features. Finally, evaluation of the accuracy of the classification model combined with all imputation methods is performed on the test dataset. The whole process is depicted in Figure 3.6.

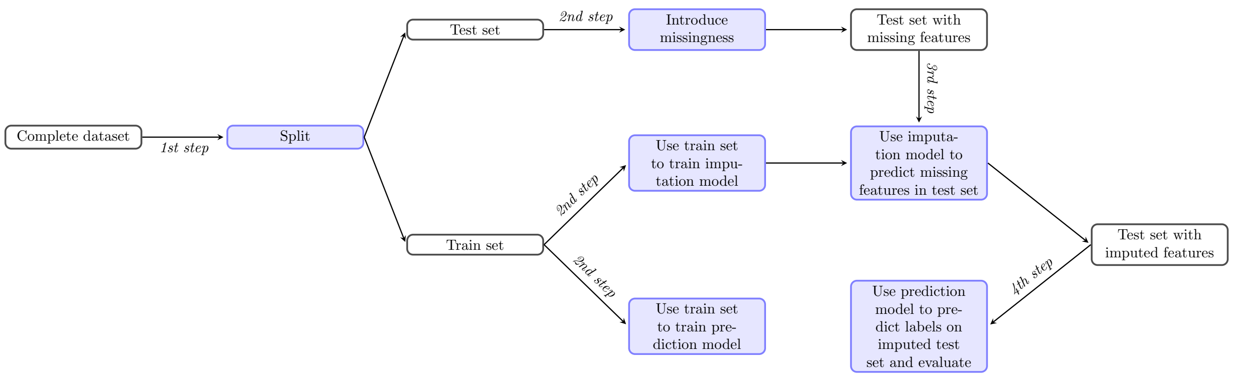

Complete dataset
1st step
Split
Test set
2nd step
Introduce missingness
Test set with missing features
3rd step
Use train set to train imputation model
Use imputation model to predict missing features in test set
Train set
2nd step
2nd step
Use train set to train prediction model
Test set with imputed features
4th step
Use prediction model to predict labels on imputed test set and evaluate

Figure 3.6: Setup of performed experiments.

3.6.1 Imputation Models and Their Parameters

Let us start with the presented WGAIN model. The generator and the critic architectures were the same for all datasets and are described in Table 3.1.

During the training, the following hyper-parameters were found by random search:

- The original data $\boldsymbol{X}$ are sampled in mini-batches of size $m = 128$.
- The missingness is introduced using the mask $\boldsymbol{M}$ with the following distribution: for each training point, the portion of missingness is sampled from a uniform distribution between 0 and maximum missing rate, which was chosen to be 0.3. Then the binary elements of $\boldsymbol{M}$ were independently sampled with this portion of missingness, i.e., its item is 0 with a probability which was previously sampled.
- The components of random vector $\boldsymbol{Z}$ are i.i.d. with normal distribution having mean 0 and standard deviation 0.01.
- The weights of the objectives functions $J(f_{\boldsymbol{w}})$ and $J(g)$ are $\lambda_{f_{\boldsymbol{w}}} = 10$, $\lambda_g = 2$, and $\lambda_{\text{MSE}} = 1$.
- Maximal norm used in clipping of the critic weights is $w_{\max} = 1$.
- We use RMSProp with learning rate $\alpha = 0.0001$ as optimizers.
- The number of training epochs is 8000.

Table 3.1: Architecture details of the WGAIN. Abbreviation: FC=fully connected layer.

Layer	Generator
	concatenate data and mask
1	FC-(1.5 input dimension), ReLU
2	FC-(1.25 input dimension), ReLU
3	FC-(input dimension), Linear
Layer	**Critic**
	concatenate data and mask
1	FC-(1.5 input dimension), ReLU
2	FC-(1.25 input dimension), ReLU
3	FC-(1), Linear

The GAIN implementation follows the original paper [139] and the hyper-parameters are the same as the described WGAIN with the following differences:

- The generator architecture differs only in the sizes of layers, which are all equal to the input dimension.
- The discriminator architecture is analogous to the generator architecture except for the sigmoid activation function on the last layer.
- The binary elements of $\boldsymbol{M}$ are independently sampled with a common portion of missingness, which is 0.2.
- The hint rate used for the hint matrix is 0.9.

- As an optimizer, we use Adam [57] with learning rate of 0.0001.
- The number of training epochs is 7000.

In the case of DAE, we follow the structure presented in [39]. For the hyper-parameters search, the hyperband [68] algorithm was used. The typical best setup is the following: ELU [16] as an activation function, three layers in both the encoder and decoder parts, the size of the code is twice the input dimension, and no regularization is used. DAE, GAIN, and WGAIN models were implemented using the `TensorFlow` library[1].

The implementation of VAEAC was based on the repository[2] corresponding to the original paper [50]. All hyper-parameters stayed in the default settings.

For the MICE method, we used the `scikit-learn` [3] library and the `IterativeImputer` class. In the default settings, the implementation uses Bayesian ridge regression as the internal imputation model.

Linear regression imputation was implemented using the `scikit-learn` library as well. We tested two scenarios within the case where multiple features were missing. The first scenario (linreg-li) is based on subsequent imputation using linear imputability, described in Section 3.3, and the second (linreg-iter) is based on an iterative approach analogous to the chained equation process described in Section 3.4.2. which corresponds to chained equations in MICE. The linreg-iter approach repeats two steps. First, every single missing value is imputed from known features only. Second, all the imputed values are iteratively re-imputed from other features (all features except the one being imputed) in random order 10 times.

The XGBT was implemented using the `xgboost` library[4] in two scenarios. The first (XGBT-li) is analogous to linreg-li and the second (XGBT-iter) to linreg-iter. Even though we carried out several experiments tuning hyper-parameters using the randomized search algorithm, due to computational demands and the need to be able to replicate the experiment easily, the results presented in this thesis correspond to the default hyper-parameters.

The k-NN imputation was implemented using the `fancyimpute` library[5]. A missing value is imputed by sampling the mean of the values of its neighbors weighted proportionally to their inverse distances. In the case where multiple features are missing, we impute all missing values at once (per row). Based on preliminary experiments, for the hyper-parameter k, we tested values $11, 13, 15, 17, 19, 21, 23, 25$. The best k for each dataset was chosen based on the RMSE value.

For each generative imputation model we performed several imputations and before evaluation took place, the results were pooled by the mean into a single dataset.

The multiple features subsequent imputation scenario using information imputability is not presented here since in preliminary experiments it does not bring any benefits over linear imputability.

[1]`TensorFlow` platform: `https://www.tensorflow.org`
[2]`VAEAC` implementation: `https://github.com/tigvarts/vaeac`
[3]`Scikit-learn` repository: `https://scikit-learn.org`
[4]`XGBoost` repository: `https://github.com/dmlc/xgboost`
[5]`Fancyimpute` repository: `https://github.com/iskandr/fancyimpute`

3.6.2 Evaluation

The impact of imputation is evaluated using the classification accuracy of the best performing classification model chosen from the six commonly used: logistic regression (LR), multi-layer perceptron (MLP), k-nearest neighbors (k-NN), naive Bayes (NB), extreme gradient boosted trees (XGBT) (for details see [13]), and random forest (RF). The best hyperparameters for each model were found using a randomized search algorithm. The accuracy of the best performing model for each dataset on the validation set is shown in Table 3.2. As a next step, the best performing classification models were combined with imputation methods. The accuracies of classification models on the imputed test dataset were then measured.

Since for the overall evaluation it is not correct to directly compare accuracies between different datasets, the recommended approach is to use rank comparison [28]. In order to do the comparison, the algorithms are ranked for each dataset separately, the best performing algorithm getting the rank of 1, the second-best rank 2, etc. To obtain the overall evaluation, we calculate the mean rank over the datasets. The results can be seen in Table 3.3. The original accuracies for each amount of missingness are presented in Appendix A.

When the degree of missingness varies from 10% to 20% the WGAIN performs the best, for 30% of missingness, the GAIN reached the same result as WGAIN. When the degree of missingness is equal to 40% the GAIN outperforms the WGAIN. Finally, for 50% XGBT-li outperforms the GAIN and WGAIN slightly. When looking at individual datasets, for degree of missingness from 10% to 30% (except the EEG dataset WGAIN), performance is always comparable to the best imputation models, as can be seen in Appendix A.

The results of the ranking evaluation can be statistically evaluated using the Friedman test [37, 36] and the corresponding posthoc tests [28]. The Friedman test is the non-parametric counterpart of the well-known ANOVA, that should be preferred when the

Table 3.2: Details of datasets with the corresponding best performing classification model chosen based on the validation set and its accuracy on the test set. The number of features (# f.) does not include the target label. The # r. stands for the number of records.

Dataset name	Origin	# f.	# r.	Model name	Model accuracy
Cancer [69]	real	9	683	RF	0.975
EEG [69]	real	14	14980	k-NN	0.952
MAGIC [69]	real	10	19020	XGBT	0.868
Ozone-1 [69]	real	72	1846	k-NN	0.977
Ozone-8 [69]	real	72	1848	LR	0.941
QSAR [69]	real	41	1055	MLP	0.868
Shuttle [69]	real	9	57998	RF	0.999
Spambase [69]	real	57	4597	MLP	0.940
Waveform [69]	real	21	5000	LR	0.869
Yeast [69]	real	8	1484	XGBT	0.578
Ringnorm [2]	artificial	20	7400	NB	0.979
Twonorm [2]	artificail	20	7400	MLP	0.979

Table 3.3: Mean ranks of accuracy for different degrees of missingness.

	Degree of missingness				
Method	10%	20%	30%	40%	50%
k-NN	6.17	5.58	5.92	4.58	5.17
MICE	5.00	6.67	6.25	6.96	6.33
linreg-li	5.25	5.42	5.25	5.79	5.25
linreg-iter	5.25	5.50	5.33	5.71	5.25
XGBT-li	6.04	5.33	5.17	4.79	**4.42**
XGBT-iter	6.83	5.21	5.42	5.29	5.67
DAE	6.62	7.42	7.42	7.46	7.25
VAEAC	5.62	5.17	5.58	5.75	5.83
GAIN	4.50	4.50	**4.33**	**3.58**	4.67
WGAIN	**3.71**	**4.21**	**4.33**	5.08	5.17

comparison is performed on different datasets. Let r_i^j be the rank of the j-th of k algorithms on the i-th of N datasets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms perform the same and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

is asymptotically distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of thumb, $N > 10$, $k > 5$). We also used its variant with the F_F statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

Resulting p-values of Friedman χ_F^2 and F_F tests are shown in Table 3.4. One can see that except 40% of missing data the null-hypothesis, that all methods perform the same, cannot be rejected at a 10% significance level. For 40% when the Nemenyi post-hoc test is applied the only significant difference in performance is between DAE and GAIN at a 10% significance level.

Table 3.4: P-values of Friedman χ_F^2 and F_F test on mean ranks of accuracy.

	Degree of missingness				
Test	10%	20%	30%	40%	50%
χ_F^2 test	0.278	0.319	0.382	0.094	0.534
F_F test	0.277	0.322	0.388	0.085	0.546

3.7 Discussion

In this section, we discuss the results of non-generative and generative models used for missing features reconstruction. Our goal was to experimentally research the impact of imputation methods and in particular explore universal imputation methods. Moreover, we proposed the WGAIN as a new deep learning imputation model, which was compared to other imputation models.

In general, the more features are missing, the more difficult it is to reconstruct them. Our presented research does not focus on relations between individual features and their individual impact on classification accuracy. Instead, we are interested in fixing the general situation of missing features without any detailed knowledge about them.

There are two scenarios of missing features. The first one assumes that it is known in advance which features are going to be missing. In such a case, the optimal solution is to train an imputation model to impute these missing features as best as possible. A model trained in this way can be very powerful. However, its usage is limited only to one specific situation, i.e., one specific combination of missing features. In the second scenario it is not known which features are going to be missing. In such a case, there is a need for a universal imputation model which will be able to sufficiently reconstruct any combination of missing features.

This in particular means the ability to solve the reconstruction of various amounts of missing features. This approach is also very helpful when we need to respond to missing features immediately. However, due to its generality it may suffer in terms of accuracy of the imputed data.

In experimental research, we compared methods suitable for both scenarios, when 10%, 20%, 30%, 40%, and 50% of features were missing. We considered the DAE, VAEAC, GAIN, and newly proposed WGAIN as the general imputation models. For each method only one model was trained and applied to impute in all mentioned amounts of missing features. The remaining imputation models - k-NN, linreg-li, linreg-iter, MICE, XGBT-li, and XGBT-iter are trained to solve a specific combination of missing features. It means that for every set of missing features a new imputation model has to be trained. In practice, we have to train and keep more models if it is not known which features are going to be missing in advance. Such an approach can be computationally demanding or even impossible.

Considering the scenario when we do not know which features exactly are going to be missing in advance, we look for a general imputation model which performs well in each considered situation. As can be seen in Table 3.3, where the mean ranks of classification accuracy are used to evaluate imputation models, the WGAIN is the best in the case of up to 30% of missing features. In the case above 30%, its results are at least competitive. In second place, we have the GAIN, which has the same base. However, WGAIN differs in that it minimizes an approximation of the Earth Mover's distance, and the corresponding discriminator's critic of the Wasserstein approach does not suffer from vanishing gradients which helps in the case of Wasserstein GAN and was expected to be beneficial in WGAIN as well. This enables the model to capture the desired distribution better. Surprisingly, DAE and VAEAC did not provide any competitive results.

Very good results are also obtained by XGBT in both li and iter versions. However, this method needs to be trained to impute a known combination of missing features. So it is not a universal imputation model.

If we focus on results of the proposed linear imputability and basic iterative approach, one can see in Table 3.3, there is no fundamental difference between imputation using linear imputability to sort missing features and the iterative approach without any emphasis on the order of missing features. However, if we explore the results of XGBT, imputation using linear imputability reached better results when we faced more than 40% of missing features.

In conclusion, we proposed a new universal imputation model, the WGAIN, which provides the best results in the case of up to 30% of missing features, and in the remaining cases, its results are at least competitive. This model does not need to know which features are missing in advance, once it is trained it is able to solve the missing features reconstruction problem.

Chapter **4**

Unsupervised Domain Adaptation

This chapter is based on the following publication:

- Friedjungová, M.; Vašata, D.; Chobola, T.; Jiřina, M. Unsupervised Latent Space Translation Network. Accepted to *ESANN 2020*.

In this chapter, we deal with the unsupervised domain adaptation task. The chapter is organised as follows. In Section 4.1, we introduce the reader to another transfer learning task - domain adaptation. In Section 4.2, we review related work in this field. Then in Section 4.3, we present a novel Latent Space Translation Network solving the proposed task with focus on an unsupervised mode where during the training there is no need for labeled or paired data. Section 4.4 is devoted to the description of experiments performed, including the evaluation of the proposed network, which greatly outperforms competing approaches. Finally, the results are discussed in Section 4.5.

4.1 Domain Adaptation Problem

One task that is often discussed in computer vision is the mapping of an image from one domain to a corresponding image in another domain known as domain adaptation or image-to-image translation. Recently, domain adaptation enjoys great attention from machine learning researchers and many notable algorithms were presented (see Section 4.2). Huge progress was achieved using methods based on adversarial training, specifically using generative adversarial networks [41, 23, 131]. The amount of new domain adaptation models is increasing rapidly and so it is somewhat difficult to keep up with state-of-the-art methods. There are still challenging problems to solve though.

The problem of mapping images between different domains can be tackled in both a supervised and an unsupervised manner. In the supervised approach, one needs pairs of corresponding images in both domains in order to learn the domain adaptation model. In real-world datasets, these pairs often have to be created manually, which is both a money and time-consuming problem. On the other hand, an unsupervised approach is used when working with independent unpaired sets of images. The goal is to relate these domains, which means finding a corresponding mapping between feature spaces $\mathcal{X}_s$ and $\mathcal{X}_t$. The difficulty lies in the fact that there are no paired examples demonstrating how the images from different domains should be mapped to each other.

There is a number of terminology inconsistencies in literature focusing on transfer learning. We define domain adaptation as a sub-field of transfer learning as in Definition 2.1.7. In this chapter, we consider a heterogeneous domain adaptation scenario, where two image domains differ in feature representations, $\mathcal{X}_s \neq \mathcal{X}_t$, but the prediction task is the same, $\mathcal{T}_s = \mathcal{T}_t$. Specifically, we assume that both domains are rich with data. Each domain is equipped with labels.

As a result of our research, we propose a novel Latent Space Translation Network (LSTNet) based on shared latent space representation and adversarial training inspired by the Unsupervised Image-to-image Translation (UNIT) framework [71]. In contrary to UNIT (see details in Section 4.3), we do not use a variational autoencoder (VAE) as a model component. Instead we introduce another adversarial discriminator which attempts to guess from a latent space representation of an image which domain it is from. In order to train the proposed network, there is no need for labeled or paired data, so it is enough to have an independent unlabeled dataset in each domain. Our approach greatly outperformed other competing methods as can be seen in Section 4.4 and 4.5.

4.2 Related Work

In this section, we present state-of-the-art domain adaptation methods which mainly focus on an unsupervised setting, which is popular in recent research [9, 12, 71, 47, 72]. The supervised setting has its limitations in that it needs paired data between presented domains.

One of the more significant models is CoGAN [72]. CoGAN architecture applies a generative adversarial network (GAN) to the domain transfer problem by training two coupled GANs to generate the source and target images, respectively. The approach achieves a domain invariant feature space by tying the high-level layer parameters of the two GANs. They learn a joint distribution without any pair of corresponding images with just samples drawn individually from both domains. The same noise input can generate a corresponding pair of images from these two domains. CoGAN was experimentally compared to a conditional GAN [72, 80] to demonstrate its advantage.

Another unsupervised model based on GAN loss is the Adversarial Discriminative Domain Adaptation framework (ADDA) [120], which combines discriminative modeling, untied weight sharing, and adversarial loss. The ADDA framework works in an unsupervised setup, i.e. no paired data are needed. First, it learns a discriminative representation using the labels in the source domain and then a separate encoding that maps the target data to the same space using an asymmetric mapping learned through domain-adversarial loss. ADDA provided a competitive result to CoGAN and achieved state-of-the-art results for its time on benchmark digits datasets such as MNIST [65, 64], USPS [48], and SVHN [87]. Additionally, they also evaluate on the NYUD [111] dataset to study adaptation across modalities.

Domain Transfer Network (DTN) [117] is also an unsupervised GAN-based network, which outperformed the ADDA framework [120] on basic adaptations of MNIST and SVHN datasets. However, they tackle the problem of emoji generation as the main challenge. Authors claim that the DTN framework is able to produce face emoji that are visually appealing and capture much more facial characteristics than emoji created by human annotators. This domain of application is also known as neural style transfer [38] where new images are synthesized by minimizing the content loss with respect to one input sample and the style loss with respect to one or more input samples. The content loss is typically represented as an image encoding by training a neural network. The style loss then compares the statistics of activations in various network layers. The aim of the style transfer is to replicate the style of one or several images. However, in DTN, the style loss is not presented in order to make the method more general.

Another unsupervised domain adaptation approach is based on CycleGAN [146]. This approach is revolutionary because of the introduction of cycle-consistency. CycleGAN uses two generators and two discriminators. The first generator is used to convert images from the source domain to the target domain. The second generator works in the opposite direction. Each generator has a corresponding discriminator which decides whether the image is synthetic or real. There exist two essential losses, the adversarial and the cycle-consistency. The adversarial loss is the same as we are familiar with from GANs, mentioned in Section 3.4. To further improve performance of the model, cycle-consistency is introduced. Cycle-consistency assumes that if an image is converted from one domain to another domain and back, a similar image should be obtained. Let us consider the generators $G_1 : \mathcal{X}_1 \rightarrow \mathcal{X}_2$ and $G_2 : \mathcal{X}_2 \rightarrow \mathcal{X}_1$, where $\mathcal{X}_1$ stands for source domain and $\mathcal{X}_2$ for target domain. Every generator is coupled with a discriminator, D_1 which distinguishes $\mathcal{X}_2$ from $G_1(\mathcal{X}_1)$ and D_2 which distinguishes $\mathcal{X}_1$ from $G_2(\mathcal{X}_2)$. Furthermore, cycle-consistency loss enforces $G_2(G_1(x_1)) \approx x_1$ and $G_1(G_2(x_2)) \approx x_2$. The results of CycleGAN were demonstrated on various domain translation tasks such as season transfer, collection style transfer or photo generation from paintings [146]. According to the authors, CycleGAN works well on tasks where color or texture adaptations have to be tackled. Its limitation is in tasks where geometric changes are needed.

The idea of cycle-consistency has gained great popularity in further development of domain adaptation solutions. Other examples include Discovery GAN (DiscoGAN) [55] and DualGAN [138]. The architecture of both is similar to CycleGAN. However, both, DiscoGAN and DualGAN call ”cycle-consistency loss” a ”reconstruction loss”. DiscoGAN uses cross-entropy for GAN loss. Its performance was demonstrated on many experiments using synthetic and real data in the style transfer domain. The network achieved competitive results in both, texture and geometrical properties of images, while no paired data was present.

In DualGAN [138], the cycle-consistency idea is linked to dual learning [44], which was first presented in the language translation domain. The idea behind dual learning is basically the same as in adversarial training, when we consider two agents competing against each other. The network is also trained in parallel. Nevertheless, in DualGAN the critic from Wasserstein GAN (WGAN) [5] is presented instead of the cross-entropy loss used in standard GAN [41].

In [101], Russo et al. present a symmetric mapping among domains. They jointly optimized bi-directional image transformations combining them with target self-labeling. They also introduced a semantic constraint on the source images - a class consistency loss that aligns the generators to preserve the class identity of an image passing through both domain mappings. The designed Symmetric Bi-directional ADAptive Generative Adversarial Network (SBADA-GAN) uses two classifiers at the end to predict the labels of the original target domain and on its source-like transformed version. They achieved the same results as in ADDA while transferring SVHN to MNIST. However, SBADA-GAN outperformed other methods in bidirectional mapping between MNIST and USPS data.

The current results in SVHN to MNIST adaptation have been surpassed by the GenToAdapt [104] method. GenToAdapt lies in a joint adversarial-discriminative unsupervised approach that transfers the information of the target distribution to a learned embedding using a generator-discriminator pair. The training of the proposed method consists of two parallel streams. In the first stream, a multi-class classifier is used to predict source labels from an embedding of the feature extraction network (encoder). The embedding is also used in a second stream by the generator to generate a realistic source image. The realistic properties of the images from the generator are controlled by the discriminator. The second stream is given by the adversarial branch which contains a generator and discriminator represented as an auxiliary classifier GAN (ACGAN) [88]. ACGAN is a variant of the conditional GAN where the discriminator is modeled as a multi-class classifier instead of providing conditioning information at the input. GenToAdapt demonstrated superiority over existing methods on tasks such as digit classification, object recognition and domain adaptation from synthetic to real data.

So far we focused on image-to-image translation between two domains in particular because of method limitations in scalability and robustness in handling more domains. Another challenging problem is multi-source domain adaptation which lies in adapting more than two domains. This task is tackled for example by StarGAN [15]. StarGAN allows simultaneous training of multiple datasets with different domains within a single network. The authors also used ACGAN [88] which allows a single discriminator to control multiple domains and which they extended to generate probability distributions over labels for all datasets. When learning from multiple datasets, we have to incorporate datasets containing different types of labels. The problem is that label information is often only partially known to each dataset. StarGAN tackles this problem using a generator taught to ignore the unspecified labels, which are zero vectors, and focus on the explicitly given label. As a network architecture, the CycleGAN [146] was adapted as the generator network of StarGAN and the GAN for the discriminator, which discriminates whether local image patches are real or fake [49, 67]. StarGAN effectiveness was demonstrated on facial attribute transfer and facial expression synthesis tasks and it outperformed the above mentioned CycleGAN [146].

In 2019 a combination of the previously mentioned SBADA-GAN with Mean Teacher [119] was presented in [33]. They replaced the target classifier of SBADA-GAN with a Mean Teacher strategy and developed a bidirectional class cycle-consistency strategy to preserve the class identity of the transformed images. In a nutshell, Mean Teacher is an approach for semi-supervised learning, which contains several steps. Consider an architecture solving some supervised machine learning task which is called the student model. Then a copy of this architecture is created and called the teacher model. In every training step the same mini-batches are used as input for both models. However, some random noise is added to these inputs separately. A consistency cost between outputs from both models is added. Weights of the student model are updated in the standard way, the weights of the teacher model are updated towards the student model weights as the exponential moving average of the student model weights. Mean Teacher encourages the students to give consistent predictions under different noises, which improves the robustness of the students. Another innovation of this new SBADA-GAN with Mean Teacher is enforcing Lipschitz constraints on discriminators in order to improve the stability of the training process. These innovations combined bring very competitive results to other up to date methods in domain adaptation between MNIST, USPS, and SVHN datasets. SBADA-GAN with Mean Teacher outperformed methods such as ADDA, GenToAdapt, and the original

SBADA-GAN. However, both SBADA-GAN and SBADA-GAN with Mean Teacher need labeled data to be trained.

Not to be forgotten, the domain adaptation field is very close to optimal transportation theory as was presented in [95]. Optimal transportation theory was first introduced in 1781 in [81] to study the problem of resource allocation, i.e. transport plan. Centuries later, optimal transport was used for the first time in [21, 20] for domain adaptation to compare probability distributions in a geometrically sound manner and learn the transformation between domains. These methods of optimal transport lie in the reduction of divergence between probability distributions of two domains when taking the geometry into account. In works such as [109, 95, 127], Wasserstein distance was introduced as a loss to promote similarities between data distributions in a deep learning setting.

In [19], the joint distribution optimal transport (JDOT) method directly learns a classifier embedded in the cost function. The idea behind it is to align the joint features distribution instead of only the features distribution. A deep learning modification of JDOT called DeepJDOT [9] was introduced and yields state-of-the-art results. The method consists of two parts. The first part is composed of an embedding function, which maps the input into the latent space, and the second is represented by a classifier, which maps the latent space to the label space on the target domain. DeepJDOT jointly optimizes this latent space and the classifier to provide a method that performs well on the target domain. In contrary to JDOT, deep modification minimizes the Wasserstein distance between the embedded space distributions, rather than between the original input spaces. The method is an iterative process where pseudo-labels are learned by a classifier on source domain first for every target point. Then, every source point has to be transported to a target point. During the process not only the total distance traveled is minimalized, but also the number of modifications of labels during the transport, i.e. between the source label and target pseudo-label. DeepJDOT was demonstrated on public benchmark domain adaptation datasets such as MNIST, USPS and SVHN, and compared with some of the methods already mentioned, such as ADDA, CoGAN, GenToAdapt. In most cases the DeepJDOT performs the best.

The last mentioned model based on GANs and VAEs, with cycle-consistency constraints is the UNsupervised Image-to-image Translation (UNIT) framework [71]. As already stated in the framework name, UNIT focuses on unsupervised adaptation of two domains. Basically, the method is based on CoGAN, however, the generator is replaced by VAE. So, one would say that each domain is modeled using a VAE-GAN. The adversarial training objective interacts with a weight-sharing constraint, as presented in CoGAN, which enforces a shared latent space to generate corresponding images in two domains, while the VAEs relate translated images with input images in the respective domains. The UNIT framework consists of six subnetworks including:

- two image encoders, each maps its domain to latent code,
- two image generators, which map latent codes to images,

- two domain adversarial discriminators, each for the respective domain to evaluate whether the translated images are real or fake.

UNIT framework learns translation in both directions jointly. Authors used MNIST-USPS datasets for domain adaptation in classification tasks and outperformed CoGAN. However, according to the authors, the UNIT framework has two limitations. First, the translation model is unimodal due to the Gaussian latent space assumption. Second, training could be unstable due to the saddle point searching problem [71].

4.3 Latent Space Translation Network

To solve the unsupervised domain adaptation task we present a Latent Space Translation Network (LSTNet). LSTNet is based on shared latent space representation and adversarial training inspired by the UNIT framework. However, we do not use a variational autoencoder as a component and instead we introduce another adversarial discriminator which attempts to guess from a latent space representation of an image which domain it is from. This approach enforces the encoders from source domains to latent space representations to yield similar distributions for both domains, which however do not need to be Gaussian as is necessary in variational autoencoders. For this to work one needs a shared latent space assumption, which means that a pair of corresponding images in the two domains can be mapped to the same latent representation in a shared latent space, as was introduced in [72].

Let us denote by $\mathcal{X}_1$ the feature space of the source image domain with associated labels in some label space $\mathcal{Y}$. Similarly, let $\mathcal{X}_2$ be the feature space of the target image domain, but with unknown labels. The goal of domain adaptation is to learn conditional probability $P(y|x)$ in the target domain by leveraging information from the source domain. Therefore, we consider a source domain dataset $\{(x_1^{(i)}, y^{(i)}) \in \mathcal{X}_1 \times \mathcal{Y} \mid i = 1, \ldots, n_1\}$ consisting of image-label pairs and a target domain dataset $\{x_2^{(j)} \in \mathcal{X}_2 \mid j = 1, ..., n_2\}$ with no labels. In the unsupervised approach to domain adaptation one starts by learning the mapping $g : \mathcal{X}_2 \to \mathcal{X}_1$ based on independent datasets in $\mathcal{X}_1$ and $\mathcal{X}_2$. Then the desired prediction function is given by the composition of g with a predictive function h in $\mathcal{X}_1$, $f = h \circ g$, which can be estimated because we have labels in the source domain dataset.

Let us now focus on finding a suitable function g. Using the latent space assumption one can construct such a function as a composition of an encoder function $E_2 : \mathcal{X}_2 \to \mathcal{L}$, mapping images from target space $\mathcal{X}_2$ to shared latent space $\mathcal{L}$ and a generator function $G_1 : \mathcal{L} \to \mathcal{X}_1$ mapping points in the shared latent space $\mathcal{L}$ to source space $\mathcal{X}_1$. In order to be able to train these functions in an unsupervised manner it is useful to have an encoder $E_1 : \mathcal{X}_1 \to \mathcal{L}$ and a generator $G_2 : \mathcal{L} \to \mathcal{X}_2$ which enables one to apply cycle consistency constraints [146], given by $x_1 = G_1(E_1(x_1))$, $x_2 = G_2(E_2(x_2))$, $x_1 = G_1(E_2(G_2(E_1(x_1))))$, $x_2 = G_2(E_1(G_1(E_2(x_2))))$.

Similarly to UNIT [71] the Latent Space Translation Network (LSTNet) itself consists of six subnetworks including two domain encoders E_1, E_2, two image generators G_1, G_2

and two domain adversarial discriminators D_1, D_2. The encoder is responsible for mapping an input image to a code in latent space $\mathcal{L}$, which is taken by the generator to reconstruct the image. Discriminators are trained to differentiate between real and fake images for each domain, whereas the generators are trained to fool them.

Since we assume that there is one-to-one correspondence between images in both domains we may expect that the probability distributions $\mathrm{P}(E_1(x_1))$ and $\mathrm{P}(E_2(x_2))$ of points in the shared latent space $\mathcal{L}$ mapped from $\mathcal{X}_1$ and from $\mathcal{X}_2$ are similar. To achieve this we introduce another adversarial discriminator D_l trying to differentiate between points from source and target domains based on their latent space representations. This eliminates one of the drawbacks of the UNIT framework which is the Gaussian latent space assumption enforced by the VAE component.

Furthermore, in order to support the latent space assumption, we assume shared intermediate representations of both encoders E_1, E_2 and generators G_1, G_2. Hence, we have $E_1 = E_s \circ E_1^*$ and $E_2 = E_s \circ E_2^*$, where E_s is the shared component of both encoders E_1, E_2 and E_1^* and E_2^* are the custom components of E_1 and E_2, respectively. A similar composition holds for the generators, i.e. $G_1 = G_1^* \circ G_s$ and $G_2 = G_2^* \circ G_s$, where G_s is the shared component of both generators G_1, G_2, and G_1^* and G_2^* are the custom components of G_1 and G_2, respectively. A schematic depiction of the entire network is given in Figure 4.1.

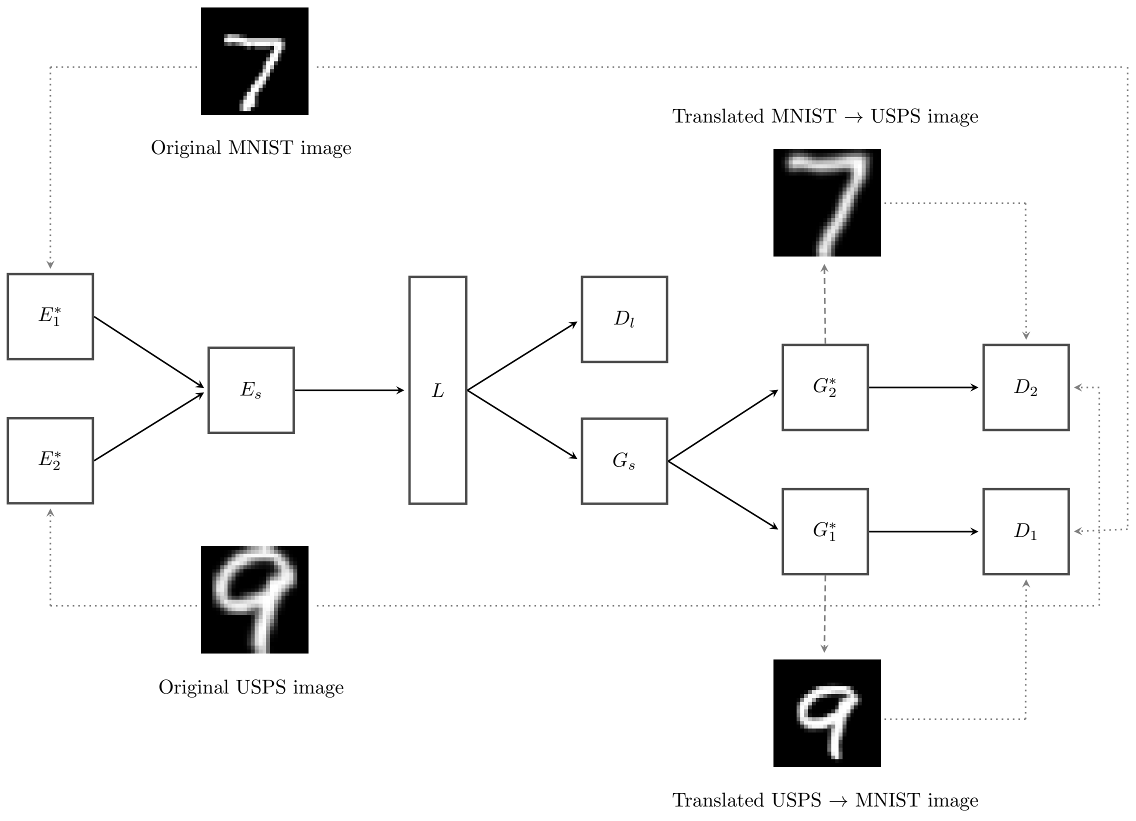


Figure 4.1: Architecture of proposed LSTNet with MNIST-USPS example.

4.3.1 Training

For the training we may identify three subnetworks: $\mathrm{AN}_1 = (E_2, G_1, D_1)$, $\mathrm{AN}_2 = (E_1, G_2, D_2)$, $\mathrm{AN}_l = (E_1, E_2, D_l)$. AN_1 is responsible for distinguishing real images sampled from $\mathcal{X}_1$ from images sampled from $\mathcal{X}_2$ and translated to $\mathcal{X}_1$ by the mapping $G_1 \circ E_2$. Analogously, AN_2 is responsible for distinguishing real images sampled from $\mathcal{X}_2$ from images sampled from $\mathcal{X}_1$ and translated to $\mathcal{X}_2$ by the mapping $G_2 \circ E_1$. AN_l is trying to find out which source domain the current point in the latent space corresponds to. Therefore, it should output 1 (true) for images sampled from $\mathcal{X}_1$ mapped by E_1 into $\mathcal{L}$ and 2 (false) for images sampled from $\mathcal{X}_2$ mapped by E_2.

It should be mentioned that in our case none of the three subnetworks is a proper GAN. This is because they are not generative in the common sense - we never let the generators transform random inputs into images.

The learning consists of simultaneous optimization of objective functions corresponding to adversarial training of the three networks AN_1, AN_2, AN_l and objective functions corresponding to four cycle consistency conditions: $\mathrm{id}_{\mathcal{X}_1} = G_1 \circ E_1$, $\mathrm{id}_{\mathcal{X}_2} = G_2 \circ E_2$, $\mathrm{id}_{\mathcal{X}_1} = G_1 \circ E_2 \circ G_2 \circ E_1$, and $\mathrm{id}_{\mathcal{X}_2} = G_2 \circ E_1 \circ G_1 \circ E_2$. Hence, we want to minimize the weighted sum of particular objectives

$$\begin{aligned} J(E_1, E_2, G_1, G_2, D_1, D_2, D_l) &= w_1 J_{\mathrm{AN}_1}(E_2, G_1, D_1) + w_2 J_{\mathrm{AN}_2}(E_1, G_2, D_2) \\ &\quad + w_l J_{\mathrm{AN}_l}(E_1, E_2, D_l) + w_3 J_{\mathrm{CC}_1}(E_1, G_1) + w_4 J_{\mathrm{CC}_2}(E_2, G_2) \\ &\quad + w_5 J_{\mathrm{CC}_3}(E_1, E_2, G_1, G_2) + w_6 J_{\mathrm{CC}_4}(E_1, E_2, G_1, G_2), \end{aligned} \tag{4.1}$$

where standard objective functions for adversarial networks are

$$\begin{aligned} J_{\mathrm{AN}_1}(E_2, G_1, D_1) &= \mathrm{E}_{x_1 \sim \mathrm{P}_{\mathcal{X}_1}} \log D_1(x_1) + \mathrm{E}_{x_2 \sim \mathrm{P}_{\mathcal{X}_2}} \log \big(1 - D_1(G_1(E_2(x_2)))\big), \\ J_{\mathrm{AN}_2}(E_1, G_2, D_2) &= \mathrm{E}_{x_2 \sim \mathrm{P}_{\mathcal{X}_2}} \log D_2(x_2) + \mathrm{E}_{x_1 \sim \mathrm{P}_{\mathcal{X}_1}} \log \big(1 - D_2(G_2(E_1(x_1)))\big), \\ J_{\mathrm{AN}_l}(E_1, E_2, D_l) &= \mathrm{E}_{x_1 \sim \mathrm{P}_{\mathcal{X}_1}} \log D_l(E_1(x_1)) + \mathrm{E}_{x_2 \sim \mathrm{P}_{\mathcal{X}_2}} \log \big(1 - D_l(E_2(x_2))\big) \end{aligned}$$

and objective functions for cycle consistency conditions are given by MAE:

$$\begin{aligned} J_{\mathrm{CC}_1}(E_1, G_1) &= \mathrm{E}_{x_1 \sim \mathrm{P}_{\mathcal{X}_1}} \|x_1 - G_1(E_1(x_1))\|_1, \\ J_{\mathrm{CC}_3}(E_1, E_2, G_1, G_2) &= \mathrm{E}_{x_1 \sim \mathrm{P}_{\mathcal{X}_1}} \|x_1 - G_1(E_2(G_2(E_1(x_1))))\|_1, \end{aligned}$$

and analogously for $J_{\mathrm{CC}_2}(E_2, G_2)$ and $J_{\mathrm{CC}_4}(E_1, E_2, G_1, G_2)$.

The training represents a two team adversarial game, where the first team consists of encoders and generators, and the second team consists of discriminators. The optimization is done via alternating gradient descent, where the first step is updating the discriminators D_1, D_2, and D_l, and the second step is updating the encoders E_1, E_2 and generators G_1, G_2.

4.4 Experiments

To evaluate the proposed LSTNet we performed several experiments on publicly available benchmark datasets. The LSTNet was evaluated in two ways. The first evaluation considers classification accuracy of the prediction model trained on the labeled source domain to classify samples in the unlabeled target domain. The second evaluation is based on a qualitative comparison by visual perception of various image-to-image translations.

4.4.1 Datasets Description

There are some benchmark datasets to evaluate domain adaptation algorithms. We performed the domain adaptation experiments using the most common - MNIST [65, 64] and USPS [48], both described in Table 4.1 and examples of data can be seen in Figures 4.2 and 4.3, respectively. Both datasets are devoted to digit classification and were used in previous related studies such as [9, 71, 72]. Due to their popularity, we are able to compare the performance of the proposed LSTNet to other methods.

We also perform qualitative evaluation by visual perception. For this evaluation, we chose a dataset devoted to season transfer, which consists of winter and summer photos of Yosemite park (summer2winter), and two datasets devoted to object transfiguration, specifically apples with oranges (apple2orange) and wild horses with zebras (horse2zebra). Object transfiguration means that the model is trained to translate one object class to another. The desired output is not well-defined. The apple2orange and horse2zebra datasets originate from the ImageNet [100] database, where part of the data was selected according to the appropriate classes, i.e., only images with labels orange, apple, horse, and zebra were chosen. In Figures 4.5, 4.4, and 4.6 we show samples of the images. These datasets were also used in CycleGAN experiments presented in [146].

During the experiments, the presented datasets were not preprocessed (except geometric data augmentation in the case of MNIST-USPS) and their size stayed unchanged as well. The description of datasets can be found in Table 4.1.

Table 4.1: Description of datasets used for domain adaptation.

Dataset name	Train set	Test set	Size	Color	Type	Example
MNIST [65, 64]	60000	10000	28×28	grayscale	digits	Fig. 4.2
USPS [48]	7291	2007	16×16	grayscale	digits	Fig. 4.3
apple2orange [100]	995, 1019	266, 248	256×256	RGB	fruits	Fig. 4.4
summer2winter [146]	1231, 962	309, 238	256×256	RGB	nature	Fig. 4.5
horse2zebra [100]	1067, 1334	120, 140	256×256	RGB	animals	Fig. 4.6

4.4.2 Experimental Results

There exist several measures for evaluating results and quality of GAN-based models, however, there is no consensus on which measure best captures the strengths and limitations of these models and can be used as a fair model comparison [11]. Our model and its results were evaluated in two scenarios. Firstly, we used classification accuracy, which requires the presence of labels in both domains. Due to this we used labeled benchmark datasets MNIST and USPS, described in Section 4.4.1. As a second evaluation scenario, we used qualitative evaluation by visual perception on an image-to-image translation task without

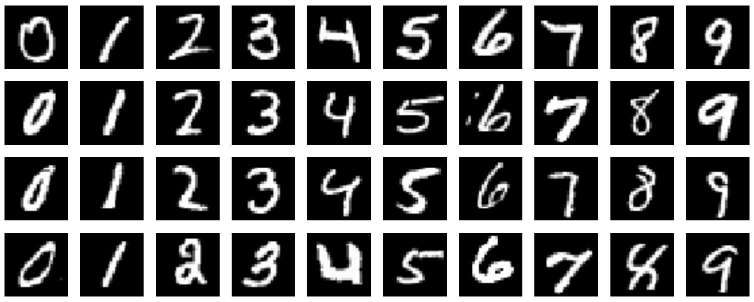


Figure 4.2: MNIST examples [65].

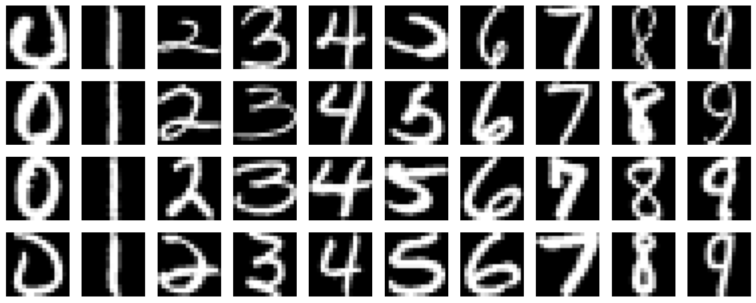


Figure 4.3: USPS examples [48].

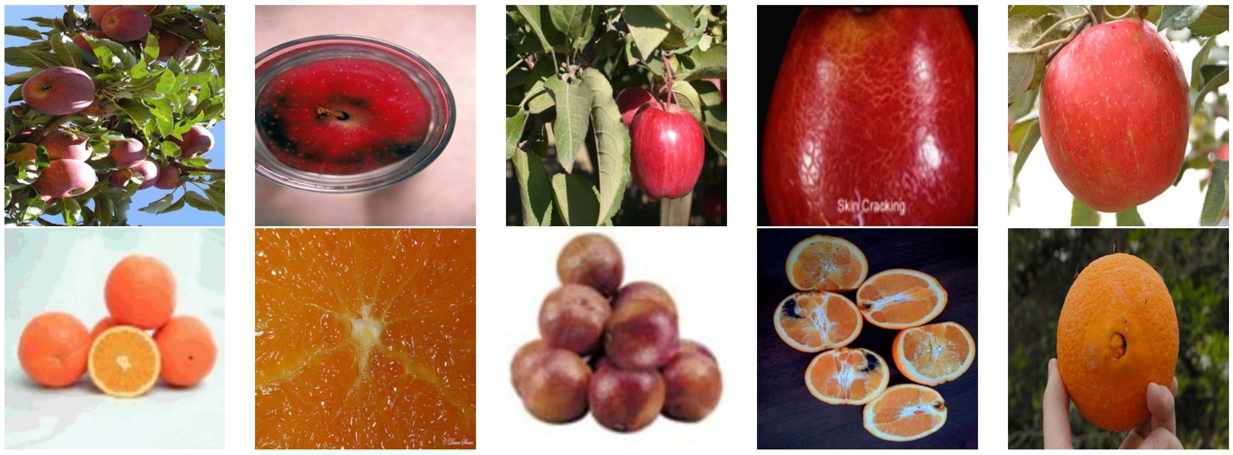


Figure 4.4: Example from the apple2orange dataset [100]. The first row depicts apples, the second row oranges.

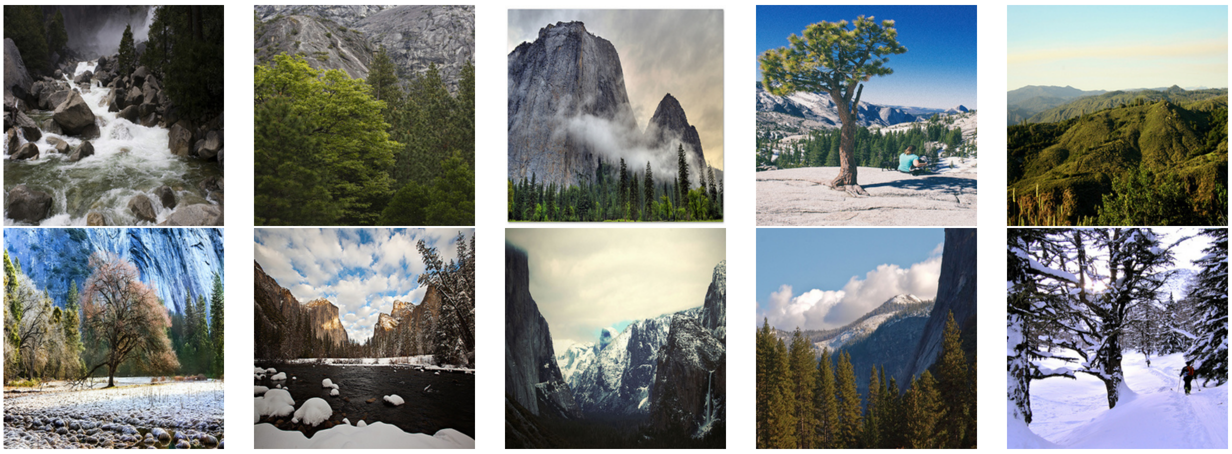


Figure 4.5: Example from the summer2winter dataset [146]. The first row depicts summer photos, the second row winter photos.

Figure 4.6: Example from the horse2zebra dataset [100]. The first row depicts horses, the second row zebras.

the presence of labeled or even paired data. This type of evaluation is commonly used in domain adaptation [15, 49, 71, 56, 138, 146].

MNIST - USPS

Here we focus on LSTNet used on MNIST and USPS data. In the first step, the LSTNet was trained using images from both domains without knowledge of labels. As an optimizer we used Adam [57] with a learning rate of 0.0001 and moment estimates exponential decays 0.8 and 0.999. Mini-batches were of size 64 images from each domain. The number of training epochs was 50. We also used data augmentation with randomly rotated training images by a maximum of 10 degrees, rescaled by a random number in the range of [0.9, 1.1], and shifted randomly by a maximum of 2 pixels in each direction. The weights corresponding to the objective function (4.1) were chosen to be $w_1, w_2 = 20$, $w_l = 30$, and $w_3, w_4, w_5, w_6 = 100$.

The weights were chosen so that the parts in the overall loss function (4.1) have similar values. A description of the architecture details is given in Table 4.2.

In the second step, the classification model was trained on the MNIST training dataset in a supervised manner (accuracy achieved on the test set was 0.9941). Then the USPS test dataset was translated into the MNIST domain using a previously trained translation network. The classification model was tested on this translated dataset and achieved an accuracy of 0.9701. Similarly, we trained a classifier on USPS (accuracy 0.9751) and then evaluated it on the translated MNIST test dataset (accuracy 0.9761).

A comparison of our results and results presented in [72, 71, 120, 9] is given in Table 4.3. We achieved significantly better results in both the USPS to MNIST and the MNIST to USPS translations than the other methods. The results are discussed in Section 4.5. Note that the training of our model is completely independent of labeled or paired data. On the other hand, methods we compare with, except for UNIT, use labels while training their networks.

Image-to-Image Translation Data

Here we performed LSTNet on three datasets: apple2orange, summer2winter, and horse2zebra. The training process is analogous to the previous MNIST-USPS case, with the following differences. Mini-batches were of size 16 images from each domain. We did not use any data augmentation. A description of the architecture details is given in Table 4.4.

The examples of results of translation via LSTNet can be seen in the following figures: for apple to orange see Figures 4.7 and 4.8 respectively, for summer to winter nature see Figures 4.9 and 4.10, and for wild horses translated into zebras see Figure 4.11 and 4.12 vice versa.

4.4.3 Latent Space Representation

The main component of our model is the discriminator on latent space as described in Section 4.3, which enforces the latent representations of both domains to have similar distributions. Let us use the t-SNE projection [77] to two-dimensional space to observe the latent representations in domain adaptation of MNIST and USPS. The latent space visualizations are shown in Figure 4.13, where only the original source domains are depicted, and in Figure 4.14, where also the labels of the original figures of digits are shown. We used 2000 test samples for each domain to create visualizations. It can be seen that the distributions of data do not align perfectly. On the other hand, both distributions seem to be close to each other and partially overlapped. It should also be noted that the training accuracy of the latent space discriminator was close to 0.5, so it was not able to distinguish the source domains. The higher capacity of the discriminator may probably lead to even better alignment of the distributions in the latent space.

Table 4.2: Architecture details of the LSTNet on MNIST-USPS task. Abbreviation: DCONV=transposed convolutional layer, FC=fully connected layer, N=neurons, K=kernel size, S=stride size, V= "valid" padding instead of default "same" padding. Slash is used to distinguish the first and second domain. Shared means that the layer with its parameters is used in the shared part E_s of the encoders and analogously for the generators.

Layer	Encoders	Shared
1	CONV-(N64, K7, S1), BatchNorm, LeakyReLU	No
2	CONV-(N128, K5, S2), BatchNorm, LeakyReLU	No
3	CONV-(N256, K3, S2/S1), BatchNorm, LeakyReLU	No
4	CONV-(N512, K3/K2-V, S1), BatchNorm, LeakyReLU	No
5	CONV-(N256, K3, S1), BatchNorm, LeakyReLU	Yes
6	CONV-(N128, K3, S3), BatchNorm, LeakyReLU	Yes
Layer	Generators	Shared
1	DCONV-(N128, K3, S1), BatchNorm, LeakyReLU	Yes
2	DCONV-(N256, K3, S1), BatchNorm, LeakyReLU	Yes
3	DCONV-(N512, K3/K2-V, S1), BatchNorm, LeakyReLU	No
4	DCONV-(N256, K3, S2), BatchNorm, LeakyReLU	No
5	DCONV-(N128, K5, S2/S1), BatchNorm, LeakyReLU	No
6	DCONV-(N64, K7, S1), BatchNorm, LeakyReLU	No
7	DCONV-(N1, K1, S1), TanH	No
Layer	Discriminators	Shared
1	CONV-(N64, K3, S1), LeakyReLU, MaxPooling-(K2, S1)	No
2	CONV-(N128, K3, S1), LeakyReLU, MaxPooling-(K2, S2/S1)	No
3	CONV-(N256, K5, S1), LeakyReLU, MaxPooling-(K2, S2)	No
4	CONV-(N512, K3/K2-V, S1), LeakyReLU, MaxPooling-(K2, S2)	No
5	FC-(N1), Sigmoid	No
Layer	Latent Discriminator	Shared
1	CONV-(N256, K3, S1), LeakyReLU, MaxPooling-(K2, S1)	No
2	CONV-(N512, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
3	CONV-(N256, K3, S1), LeakyReLU, MaxPooling-(K2, S1)	No
4	FC-(N1), Sigmoid	No

Table 4.3: Comparison of accuracies of methods used in unsupervised domain adaptation.

Method	USPS $\rightarrow$ MNIST	MNIST $\rightarrow$ USPS
ADDA [120]	0.901	0.894
CoGAN [72]	0.9315	0.9565
GenToAdapt [104]	0.908	0.953
UNIT [71]	0.9358	0.9597
DeepJDOT [9]	0.964	0.957
Proposed LSTNet	**0.9701**	**0.9761**

Table 4.4: Architecture details of the LSTNet on image-to-image translation data. Abbreviation: DCONV=transposed convolutional layer, FC=fully connected layer, N=neurons, K=kernel size, S=stride size, V= ”valid” padding instead of default ”same” padding. Shared means that the layer with its parameters is used in the shared part E_s of the encoders and analogously for the generators.

Layer	Encoders	Shared
1	CONV-(N64, K7, S1), BatchNorm, LeakyReLU	No
2	CONV-(N128, K3, S2), BatchNorm, LeakyReLU	No
3	CONV-(N256, K3, S2), BatchNorm, LeakyReLU	No
4	CONV-(N256, K3, S1), BatchNorm, LeakyReLU	No
5	CONV-(N256, K3, S1), BatchNorm, LeakyReLU	No
6	CONV-(N256, K3, S1), BatchNorm, LeakyReLU	Yes
Layer	Generators	Shared
1	DCONV-(N256, K3, S1), BatchNorm, LeakyReLU	Yes
2	DCONV-(N256, K3, S1), BatchNorm, LeakyReLU	No
3	DCONV-(N256, K3, S1), BatchNorm, LeakyReLU	No
4	DCONV-(N256, K3, S2), BatchNorm, LeakyReLU	No
5	DCONV-(N128, K3, S2), BatchNorm, LeakyReLU	No
6	DCONV-(N64, K3, S1), BatchNorm, LeakyReLU	No
7	DCONV-(N3, K1, S1), TanH	No
Layer	Discriminators	Shared
1	CONV-(N64, K5, S1), LeakyReLU, MaxPooling-(K2, S2)	No
2	CONV-(N128, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
3	CONV-(N256, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
4	CONV-(N512, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
5	FC-(N1), Sigmoid	No
Layer	Latent Discriminator	Shared
1	CONV-(N64, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
2	CONV-(N128, K3, S1), LeakyReLU, MaxPooling-(K2, S2)	No
3	CONV-(N256, K3, S1), LeakyReLU, MaxPooling-(K2, S1)	No
4	FC-(N1), Sigmoid	No

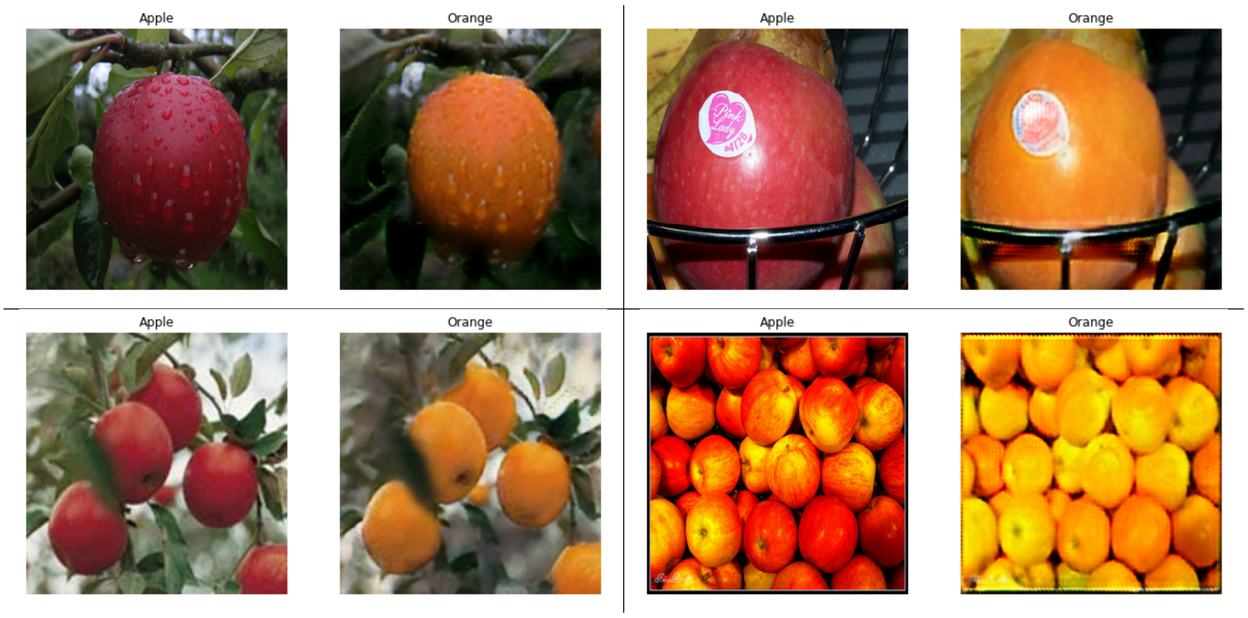


Figure 4.7: Examples of four apple to orange translations. The original apple images on the left were translated to oranges on the right.

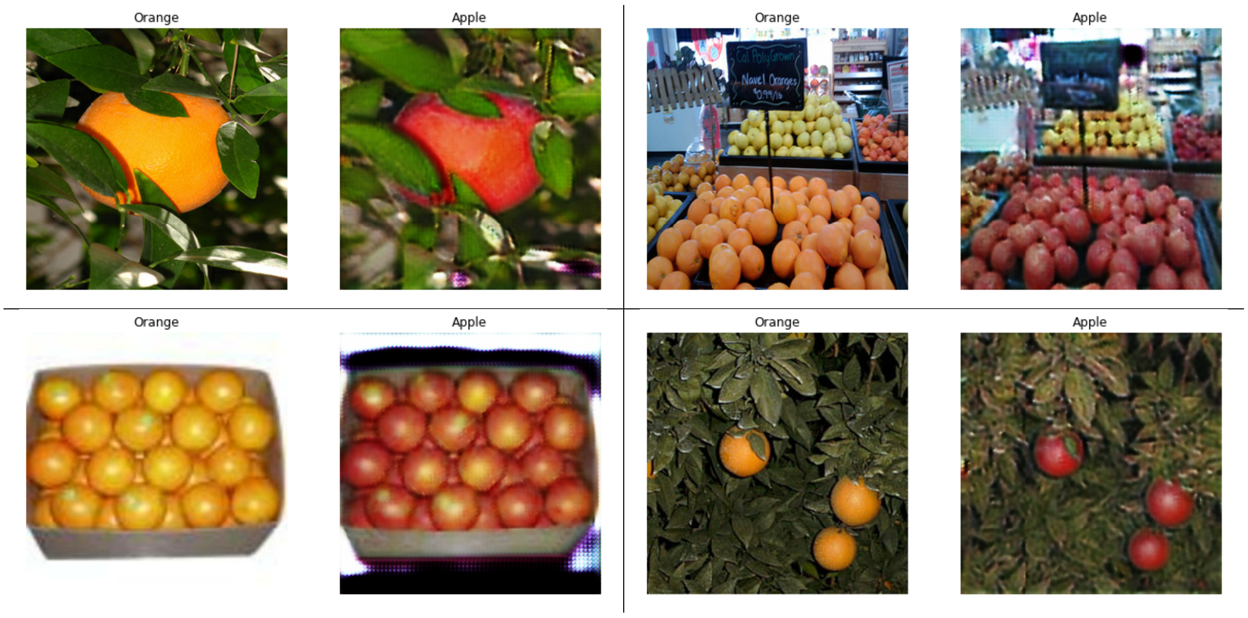


Figure 4.8: Examples of four orange to apple translations. The original orange images on the left were translated to apples on the right.

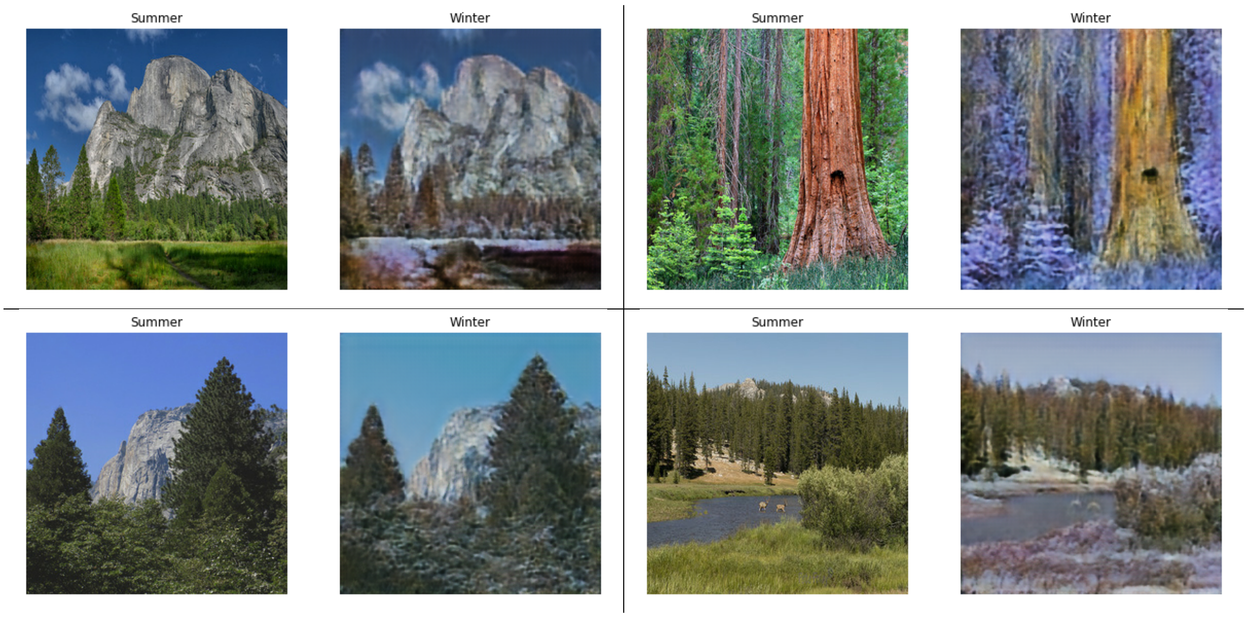


Figure 4.9: Examples of four summer to winter translations. The original summer images on the left were translated to winter scenes on the right.

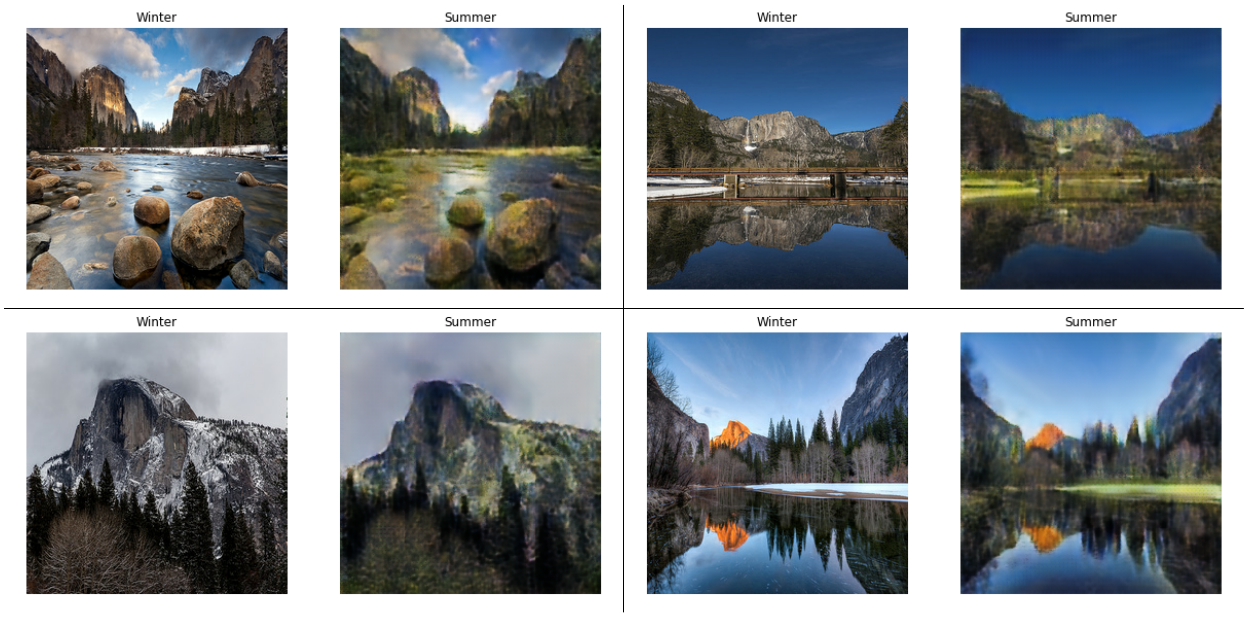


Figure 4.10: Examples of four winter to summer translations. The original winter images on the left were translated to summer scenes on the right.

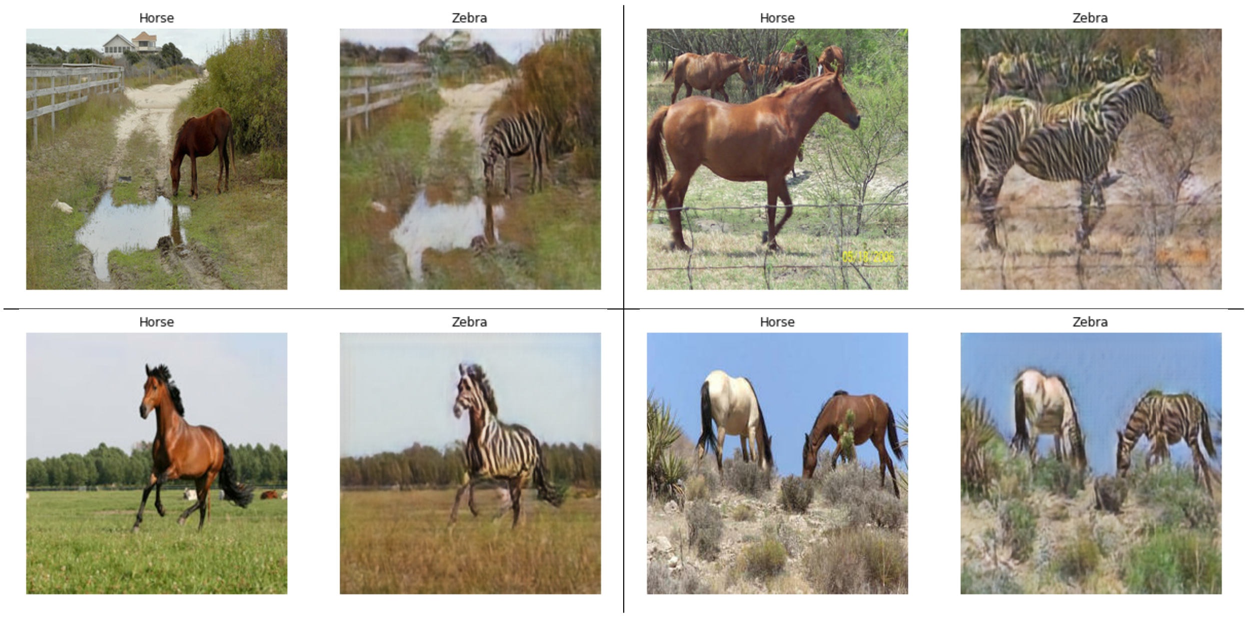


Figure 4.11: Examples of four horse to zebra translations. The original images of horses on the left were translated to zebras on the right.

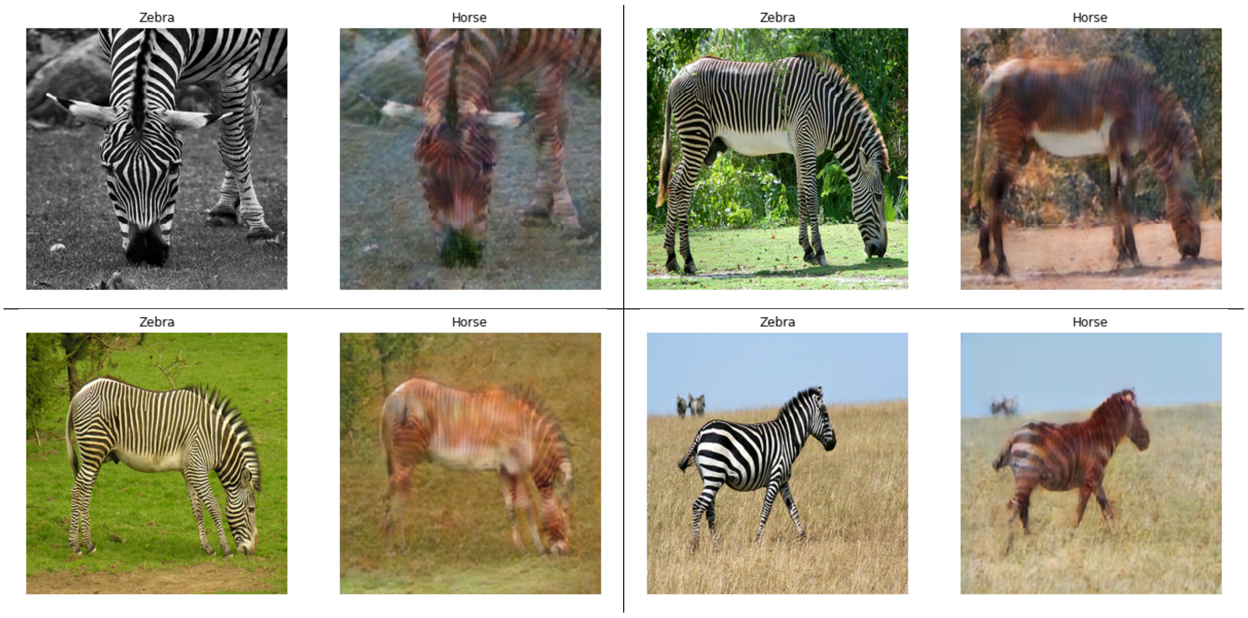


Figure 4.12: Examples of four zebra to horse translations. The original images of zebras on the left were translated to horses on the right.

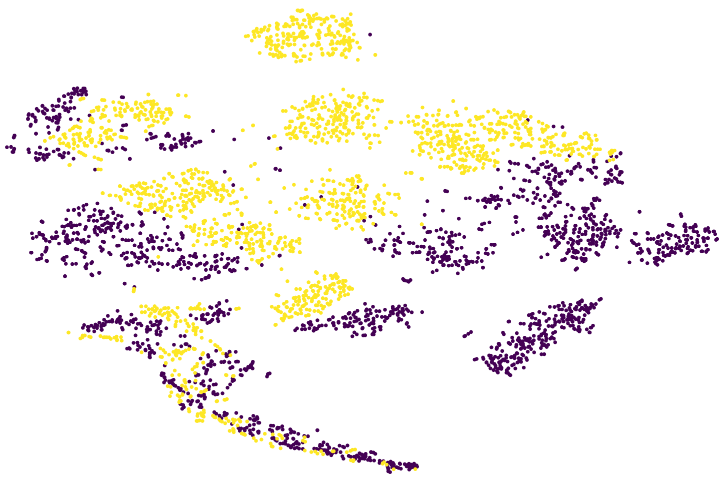


Figure 4.13: Two-dimensional t-SNE latent space visualization, where the colors differ between domains. USPS is purple, MNIST is yellow.

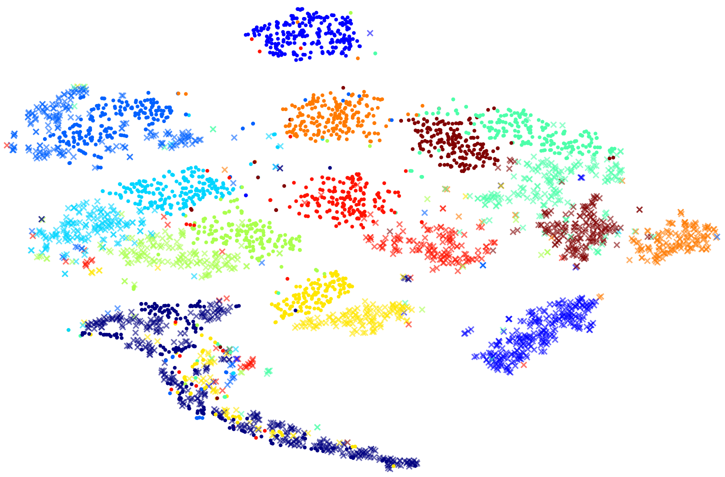


Figure 4.14: Latent space visualization, where the colors differ between classes. USPS is represented by crosses, MNIST by points.

4.5 Discussion

In this chapter we proposed our own contribution to the domain adaptation problem. It is given by the Latent Space Translation Network which is a novel framework based on a shared latent space representation and adversarial training. We introduce an additional adversarial discriminator on the latent representation which forces the latent space distributions from both domains to be similar. We experimentally showed an interesting performance enhancement of the proposed network in the domain adaptation of MNIST and USPS datasets and we also showed its ability to perform on three other image-to-image translation datasets.

Contrary to almost all state-of-the-art methods (except UNIT), our model is completely unsupervised, it means that it does not need labeled or even paired data to solve the domain adaptation task. It should be emphasised that even in this unsupervised set up, the model outperforms the others (in the MNIST-USPS task) which used knowledge of labels during training. It means that the model mapping preserves the correct categories of data, even though it is not explicitly trained to do this. This behaviour is primarily caused by cycle-consistency and the adversarial discriminator on the latent representation.

Note, that although our model has state-of-the-art results on the MNIST-USPS task, it may not perform well in a situation where one of the domains contains much more information than the other. As an example imagine a less informative MNIST dataset and a more informative dataset of photos of real street numbers. The issue is that our model is not generative, so the mapping from a less informative domain to a rich informative domain often leads to mode collapse. We want to address this issue in future work.

Chapter **5**

Conclusion

In this chapter we conclude the thesis by summarising every chapter and its main contributions. We also suggest possible future work based on our experience gained through this thesis.

5.1 Thesis Summary

This doctoral thesis is divided into five chapters. Let us summarize the content and contributions of each one.

Initially, in Chapter 1 we introduced the reader to the motivation behind our efforts together with the goals we wished to achieve. The main topic of this thesis is asymmetric heterogeneous transfer learning. In particular, we focus on two tasks - missing features reconstruction and unsupervised domain adaptation. An outline and list of contributions of this thesis were presented.

Next, in Chapter 2 we provided the necessary theoretical background of transfer learning. We presented a comprehensive overview of state-of-the-art methods in asymmetric heterogeneous transfer learning, which we focus on primarily. Based on our review in this field we also proposed a division of different transfer learning settings. At the end of the chapter we introduced the two tasks that we deal with in more detail.

Chapter 3 is devoted to missing features reconstruction. We introduced the problem together with an overview of data imputation methods. We presented two contributions to the missing features reconstruction task. First, we proposed linear and information imputability that might be used to order missing features to be imputed. This ordering leads to a more accurate imputation. Second, we proposed a novel imputation model called Wasserstein Generative Adversarial Imputation Network (WGAIN). This model is universal in a sense that we do not need to know which features are going to be missing in advance. Our model was compared to other state-of-the-art methods and performed competitively and in some cases it was even the best.

In Chapter 4 we dealt with unsupervised domain adaptation. We overviewed state-of-the-art methods with focus on unsupervised domain adaptation. We proposed a novel

Latent Space Translation Network (LSTNet). Our network is able to solve the task with no paired data and even no labeled data. In the presented experiments our network clearly outperforms state-of-the-art approaches.

5.2 Contributions of the Dissertation Thesis

All goals of this thesis were achieved. Let us briefly highlight the main contributions of the presented dissertation thesis as follows:

1. We provide comprehensive definitions of transfer learning concepts together with an overview of methods in the less common asymmetric heterogeneous field.

2. We present our own imputation model, Wasserstein Generative Adversarial Imputation Network, to tackle the problem of missing features reconstruction. This model is able to work in a general set up when it is not known which features are missing in advance.

3. We propose linear and information imputability to order missing features. Thanks to this ordering, sequential imputation methods are able to impute data more accurately.

4. We present the novel Latent Space Translation Network, which is able to solve the unsupervised domain adaptation task, where no paired or even labeled data are available.

All contributions of this thesis are properly published as can be seen in the list of reviewed publications of the author on page 85.

5.3 Limitations and Perspectives for Future Work

In this thesis we have examined asymmetric heterogeneous transfer learning in two application domains. Even though we proposed a satisfying solution, there are still possibilities for improvements. For future work we suggest exploring the following:

- In missing features reconstruction, it would be interesting to focus on the image inpainting task. The proposed WGAIN could be tested on this task and compared to specialised methods such as Context Encoders.

- Introduce Mean Teacher or another curriculum learning strategy to the LSTNet training process. It may be beneficial in a similar way as when Mean Teacher was beneficial to SBADA-GAN [33].

- Replace the discriminators in the LSTNet with Wasserstein critics, which may lead to an improvement of the training process.

- The LSTNet could be further improved in a way that it would be able to react on domains with unequal information. A the moment, the mapping from a less informative domain to a rich informative domain might lead to mode collapse. This issue can be solved by introducing generativeness.

Bibliography

[1] E. Acuña and C. Rodriguez. The treatment of missing values and its effect on classifier accuracy. In *Classification, Clustering, and Data Mining Applications*, pages 639–647, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[3] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Statistics. Wiley, 3rd edition, 2003.

[4] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[5] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] Á. Arroyo, Á. Herrero, V. Tricio, E. Corchado, and M. Woźniak. Neural models for imputation of missing ozone data in air-quality datasets. *Complexity*, 2018.

[7] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, 2011.

[8] B. K. Beaulieu-Jones and J. H. Moore. Missing data imputation in the electronic health record using deeply learned autoencoders. In *Pacific Symposium on Biocomputing 2017*, pages 207–218. World Scientific, 2017.

[9] B. Bhushan Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation.

In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–463, 2018.

[10] U. Blanke and B. Schiele. Remember and transfer what you have learned-recognizing composite activities based on activity spotting. In *International Symposium on Wearable Computers (ISWC) 2010*, pages 1–8. IEEE, 2010.

[11] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[12] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

[13] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[14] S. Cheng-Xian Li, B. Jiang, and B. Marlin. Misgan: Learning from incomplete data with generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[15] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[17] D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36(3):537–556, 2013.

[18] A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu. Missing data imputation via denoising autoencoders: the untold story. In *International Symposium on Intelligent Data Analysis*, pages 87–98. Springer, 2018.

[19] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017.

[20] N. Courty, R. Flamary, and D. Tuia. Domain adaptation with regularized optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–289. Springer, 2014.

[21] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

[22] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 2nd edition, 2006.

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

[24] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, and Y. Yu. Translated learning: Transfer learning across different feature spaces. In *Advances in Neural Information Processing Systems 21*, pages 353–360. Curran Associates, Inc., 2009.

[25] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[26] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

[27] O. Day and T. M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):29, Sep 2017.

[28] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[29] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[30] L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. 2012.

[31] Y. Duan, Y. Lv, Y.-L. Liu, and F.-Y. Wang. An efficient realization of deep learning for traffic data imputation. *Transportation research part C: emerging technologies*, 72:168–181, 2016.

[32] A. Farhangfar, L. A. Kurgan, and J. G. Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41:3692–3705, 2008.

[33] C. Feng, Z. He, J. Wang, Q. Lin, Z. Zhu, J. Lu, and S. Xie. Domain adaptation with sbada-gan and mean teacher. *Neurocomputing*, 2019.

[34] K. D. Feuz and D. J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). In *ACM Transactions on Intelligent Systems and Technology*. ACM, 2015.

[35] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[36] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[37] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 03 1940.

[38] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[39] L. Gondara and K. Wang. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 260–272. Springer, 2018.

[40] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[41] 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 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[42] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[43] M. Harel and S. Mannor. Learning from multiple outlooks. In *Proceedings of the 28th international conference on machine learning*, 2011.

[44] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma. Dual learning for machine translation. In *Advances in neural information processing systems*, pages 820–828, 2016.

[45] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang. An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering*, 19(2):167–199, 2012.

[46] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko. Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*, 2013.

[47] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[48] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[49] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[50] O. Ivanov, M. Figurnov, and D. Vetrov. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2019.

[51] P. Jonsson and C. Wohlin. An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 108–118. IEEE, 2004.

[52] I. Jordanov, N. Petrov, and A. Petrozziello. Classifiers accuracy improvement based on missing data imputation. *Journal of Artificial Intelligence and Soft Computing Research*, 8(1):31–48, 2018.

[53] M. Kachuee, K. Karkkainen, O. Goldstein, S. Darabi, and M. Sarrafzadeh. Generative imputation and stochastic prediction. *arXiv preprint arXiv:1905.09340*, 2019.

[54] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[55] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017.

[56] Y.-B. Kim, K. Stratos, and D. Kim. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, 2017.

[57] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd international conference for learning representations, San Diego*, 2015.

[58] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[59] L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

[60] B. Kulis et al. Metric learning: a survey. In *Foundations and Trends in Machine Learning*. Now Publishers, Inc., 2013.

[61] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792. IEEE, 2011.

[62] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[63] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, 2001.

[64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[65] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. 2010.

[66] D. Lee, J. Kim, W.-J. Moon, and J. C. Ye. Collagan: Collaborative gan for missing image data imputation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2487–2496, 2019.

[67] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016.

[68] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1), 2017.

[69] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[70] R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

[71] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

[72] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[73] S. Liu, Y. Yang, and J. Forrest. *Grey data analysis*. Springer, 2017.

[74] D. Lombardi and S. Pant. Nonparametric k-nearest-neighbor entropy estimator. *Phys. Rev. E*, 93:013310, Jan 2016.

[75] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, 17(9):1967, 2017.

[76] Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer learning for cross-company software defect prediction. volume 54, pages 248–256. Elsevier, 2012.

[77] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[78] P.-A. Mattei and J. Frellsen. Miwae: Deep generative modelling and imputation of incomplete data. *arXiv preprint arXiv:1812.02633*, 2018.

[79] J. T. McCoy, S. Kroon, and L. Auret. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine*, 51(21):141–146, 2018.

[80] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[81] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[82] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[83] J. S. Murray et al. Multiple imputation: A review of practical and theoretical findings. *Statistical Science*, 33(2):142–159, 2018.

[84] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan. Heterogeneous defect prediction. In *IEEE Transactions on Software Engineering*, volume 44, pages 874–896. IEEE, 2017.

[85] J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In *2013 35th international conference on software engineering (ICSE)*, pages 382–391. IEEE, 2013.

[86] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, page 107501, 2020.

[87] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[88] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[89] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[90] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[91] P. Prettenhofer and B. Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1118–1127, 2010.

[92] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning.* The MIT Press, 2009.

[93] F. Rahman, D. Posnett, and P. Devanbu. Recalling the ”imprecision” of cross-project defect prediction. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, pages 1–11, 2012.

[94] P. Rashidi and D. J. Cook. Multi home transfer learning for resident activity discovery and recognition. *KDD Knowledge Discovery from Sensor Data*, 12:56–63, 2010.

[95] I. Redko, A. Habrard, and M. Sebban. Theoretical analysis of domain adaptation with optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 737–753. Springer, 2017.

[96] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[97] D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[98] D. B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[99] Y. Rubner, L. J. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA image understanding workshop*, volume 661, page 668, 1997.

[100] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[101] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8099–8108, 2018.

[102] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Computer Vision – ECCV 2010*, pages 213–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[103] C. M. Salgado, C. Azevedo, H. Proença, and S. M. Vieira. *Missing Data*, pages 143–162. Springer International Publishing, Cham, 2016.

[104] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2018.

[105] J. L. Schafer. *Analysis of Incomplete Multivariate Data.* Chapman and Hall/CRC, London, 1997.

[106] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002.

[107] C. Shang, A. Palmer, J. Sun, K.-S. Chen, J. Lu, and J. Bi. Vigan: Missing view imputation with generative adversarial networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 766–775. IEEE, 2017.

[108] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, New York, NY, USA, 2004.

[109] J. Shen, Y. Qu, W. Zhang, and Y. Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[110] X. Shi, Q. Liu, W. Fan, S. Y. Philip, and R. Zhu. Transfer learning on heterogeneous feature spaces via spectral transformation. In *2010 IEEE 10th International Conference on Data Mining (ICDM).* IEEE, 2010.

[111] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[112] E.-L. Silva-Ramírez, R. Pino-Mejías, and M. López-Coello. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65–74, 2015.

[113] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015.

[114] K. Sricharan, D. Wei, and A. O. Hero. Ensemble estimators for multivariate entropy estimation. *IEEE Transactions on Information Theory*, 59(7):4374–4388, July 2013.

[115] D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[116] S. Sukhija, N. C. Krishnan, and G. Singh. Supervised heterogeneous domain adaptation via random forests. In *IJCAI*, pages 2039–2045, 2016.

[117] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[118] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. *arXiv preprint arXiv:1808.01974*, 2018.

[119] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[120] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[121] I. Valera, M. F. Pradier, and Z. Ghahramani. General latent feature modeling for data exploration tasks. *arXiv preprint arXiv:1707.08352*, 2017.

[122] S. van Buuren. *Flexible imputation of missing data.* Chapman and Hall/CRC, 2018.

[123] S. van Buuren and C. G. M. K. Groothuis-Oudshoorn. Multivariate imputation by chained equations: Mice v1. 0 user's manual. 2000.

[124] S. van Buuren and C. G. M. K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68, 2010.

[125] T. Van Kasteren, G. Englebienne, and B. J. Kröse. Transferring knowledge of activity recognition across sensor networks. In *International Conference on Pervasive Computing*, pages 283–300. Springer Berlin Heidelberg, 2010.

[126] T. van Kasteren, G. Englebienne, B. J. Kröse, et al. Recognizing activities in multiple contexts using transfer learning. In *AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems*, pages 142–149, 2008.

[127] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[128] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. *Extracting and composing robust features with denoising autoencoders.* In Proceedings of the 25th international conference on Machine learning ACM, 2008.

[129] C. Wang and S. Mahadevan. Heterogeneous domain adaptation using magnifold alignment. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 2011.

[130] D. Wang and T. F. Zheng. Transfer learning for speech and language processing. *arXiv preprint arXiv:1511.06066*, 2015.

[131] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[132] B. Wei and C. Pal. Cross lingual adaptation: an experiment on sentiment classifications. In *Proceedings of the ACL 2010 conference short papers*, pages 258–262, 2010.

[133] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[134] L. Z. Wong, H. Chen, S. Lin, and D. C. Chen. Imputing missing values in sensor networks using sparse data representations. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, pages 227–230, New York, NY, USA, 2014. ACM.

[135] M. K. Wouter and L. Marco. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[136] M. Xiao and Y. Guo. Feature space independent semi-supervised domain adaptation via kernel matching. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):54–66, 2014.

[137] K. Yan, L. Kou, and D. Zhang. Learning domain-invariant subspace using domain features and independence maximization. *IEEE transactions on cybernetics*, 48(1):288–299, 2017.

[138] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[139] J. Yoon, J. Jordon, and M. Van Der Schaar. GAIN: Missing data imputation using generative adversarial nets. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5689–5698. PMLR, 10–15 Jul 2018.

[140] A. Zadeh, Y.-C. Lim, P. P. Liang, and L.-P. Morency. Variational auto-decoder. *arXiv preprint arXiv:1903.00840*, 2019.

[141] J. Zhang, W. Li, P. Ogunbona, and D. Xu. Recent advances in transfer learning for cross-dataset visual recognition: A problem-oriented perspective. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[142] Q. Zhang, A. Rahman, and C. D'este. Impute vs. ignore: Missing values for prediction. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[143] J. T. Zhou, S. J. Pan, I. W. Tsang, and S.-S. Ho. Transfer learning for cross-language text categorization through active correspondences construction. In *AAAI*, pages 2400–2406, 2016.

[144] J. T. Zhou, S. J. Pan, I. W. Tsang, and Y. Yan. Hybrid heterogeneous transfer learning through deep learning. In *28th AAAI Conference on Artificial Intelligence*, 2014.

[145] J. T. Zhou, I. W. Tsang, S. J. Pan, and M. Tan. Heterogeneous domain adaptation for multiple classes. In *International Conference on Artificial Intelligence and Statistics*, 2014.

[146] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[147] M. Zhu and X. Cheng. Iterative knn imputation based on gra for missing values in tplms. In *Computer Science and Network Technology (ICCSNT), 2015 4th International Conference on*, volume 1, pages 94–99. IEEE, 2015.

[148] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *arXiv preprint arXiv:1911.02685*, 2019.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] Friedjungová, M.; Vašata, D.; Balatsko, M.; Jiřina, M. Missing Features Reconstruction Using a Wasserstein Generative Adversarial Imputation Network. In *Computational Science - ICCS 2020*. Springer International Publishing, Cham, 2020.

[A.2] Friedjungová, M.; Vašata, D.; Chobola, T.; Jiřina, M. Unsupervised Latent Space Translation Network. Accepted to *ESANN 2020*.

[A.3] Friedjungová, M.; Vašata, D.; Jiřina, M. Missing Features Reconstruction and Its Impact on Classification Accuracy. In: *Computational Science - ICCS 2019*. Springer International Publishing, Cham, 2019.

The paper has been cited in:

- McCombe, N.; Ding, X.; Prasad, G.; Finn, D.P.; Todd, S.; McClean, P.; Wong-Lin, K. F. Predicting feature imputability in the absence of ground truth. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020.

[A.4] Friedjungová, M.; Jiřina, M. An Overview of Transfer Learning Focused on Asymmetric Heterogeneous Approaches. In: *Data Management Technologies and Applications*. Springer International Publishing, Cham, 2018.

The paper has been cited in:

- Huang, J.; Zhou, Z.; Shang, J.; Niu, C. Heterogeneous domain adaptation with label and structural consistency. In *Multimedia Tools and Applications*. Springer International Publishing, Cham. 2020.
- Potnis, I. R. Sharing Learned Models Between Heterogeneous Robots: An Image Driven Interpretation. *Faculty of the Graduate School of the University of Maryland, Baltimore County*. 2018.

[A.5] Friedjungová, M.; Jiřina, M. Asymmetric Heterogeneous Transfer Learning: A Survey. In: *Proceedings of the 6th International Conference on Data Science, Technology and Applications.* SciTePress, 2017.

The paper has been cited in:

- Al-Helali, B.; Chen, Q.; Xue, B.; Zhang, M. Multi-tree genetic programming for feature construction-based domain adaptation in symbolic regression with incomplete data. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference.* ACM New York, NY, USA. 2020.
- Lu, Y.; Luo, L.; Huang, D.; Wang, Y.; Chen, L. Knowledge Transfer in Vision Recognition: A Survey. In *ACM Computing Surveys (CSUR).* ACM New York, NY, USA. 2020.
- Wang, J.; Chen, Y.; Feng, W.; Yu, H.; Huang, M.; Yang, Q. Transfer Learning with Dynamic Distribution Adaptation. In *ACM Transactions on Intelligent Systems and Technology (ACM TIST).* ACM New York, NY, USA. 2020.
- Huang, J.; Zhou, Z.; Shang, J.; Niu, C. Heterogeneous domain adaptation with label and structural consistency. In *Multimedia Tools and Applications.* Springer International Publishing, Cham. 2020.
- Xu, P.; Deng, Z.; Wang, J.; Wang, S. Joint Information Preservation for Heterogeneous Domain Adaptation. In *Journal of Frontiers of Computer Science and Technology.* 2019.
- Lu, Y.; Luo, L.; Huang, D.; Saidi, A.; Wang, Y.; Chen, L. Knowledge Transfer in Vision Recognition: A Survey. In *HAL archives.* 2019.

Remaining Publications of the Author Relevant to the Thesis

[A.6] Friedjungová, M. *Asymmetric Heterogeneous Transfer Learning.* Ph.D. Minimum Thesis, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic, 2018.

Remaining Publications of the Author

[A.7] Kubernátová, P.; Friedjungová, M.; van Duijn, M. Constructing a Data Visualization Recommender System. In: *Data Management Technologies and Applications.* Springer International Publishing, Cham, 2019.

The paper has been cited in:

- Ehsan, H. View recommendation for visual data exploration. *Dissertation Thesis, The University of Queensland.* 2019.

[A.8] Kubernátová, P.; Friedjungová, M.; van Duijn, M. Knowledge at First Glance: A Model for a Data Visualization Recommender System Suited for Non-expert Users.. In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications.* SciTePress, 2018.

Appendix A

Appendix

Table A.1: Ranks of accuracies of the imputation methods for 10% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	10	5.5	3	3	7.5	9	5.5	3	7.5	1
EEG	1	7	5.5	5.5	2	3	9	10	8	4
MAGIC	1	7	8.5	8.5	2.5	2.5	10	4	5	6
Ozone-1	9.5	4	2.5	2.5	8	9.5	1	6	7	5
Ozone-8	10	1	3.5	3.5	7	7	9	7	3.5	3.5
QSAR	8.5	4.5	1.5	1.5	8.5	10	6	4.5	7	3
Shuttle	4	2	2	2	8.5	8.5	10	6	7	5
Spambase	1	8	9.5	9.5	3	4	5	6	2	7
Waveform	3	2	4.5	4.5	8	9	10	6	1	7
Yeast	6	7	9.5	9.5	3.5	3.5	5	8	2	1
Ringnorm	10	9	6.5	6.5	5	8	4	3	2	1
Twonorm	10	3	6.5	6.5	9	8	5	4	2	1

Table A.2: Means of accuracies for 10% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	0.970	0.974	0.975	0.975	0.974	0.973	0.974	0.975	0.974	**0.975**
EEG	**0.923**	0.905	0.905	0.905	0.922	0.913	0.899	0.637	0.903	0.905
MAGIC	**0.856**	0.847	0.846	0.846	0.855	0.855	0.846	0.853	0.852	0.851
Ozone-1	0.975	0.976	0.977	0.977	0.975	0.975	**0.977**	0.976	0.976	0.976
Ozone-8	0.940	**0.941**	0.941	0.941	0.941	0.941	0.940	0.941	0.941	0.941
QSAR	0.861	0.862	**0.863**	**0.863**	0.861	0.860	0.862	0.862	0.861	0.863
Shuttle	0.999	**1.000**	**1.000**	**1.000**	0.998	0.998	0.994	0.999	0.999	0.999
Spambase	**0.936**	0.928	0.925	0.925	0.933	0.933	0.931	0.930	0.934	0.930
Waveform	0.860	0.860	0.860	0.860	0.859	0.859	0.859	0.860	**0.860**	0.859
Yeast	0.552	0.551	0.548	0.548	0.554	0.554	0.553	0.550	0.554	**0.556**
Ringnorm	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	**0.968**
Twonorm	0.971	0.972	0.972	0.972	0.971	0.971	0.972	0.972	0.972	**0.972**

Table A.3: Ranks of accuracies of the imputation methods for 20% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	9	2	4.5	4.5	10	8	3	6	7	1
EEG	1	7	5	6	2	3	9	10	8	4
MAGIC	1	9	7.5	7.5	2	3	10	5	4	6
Ozone-1	10	9	5	5	8	3	1	2	7	5
Ozone-8	9	3	1.5	1.5	7	4.5	10	8	6	4.5
QSAR	6	9	7.5	7.5	1	4	10	5	3	2
Shuttle	3	4	1.5	1.5	7	8	10	5	9	6
Spambase	1	8	9.5	9.5	4	3	6	5	2	7
Waveform	9	5	2.5	2.5	6	8	10	4	1	7
Yeast	5	8	9.5	9.5	3	1	7	4	2	6
Ringnorm	10	9	6.5	6.5	4	8	5	2	3	1
Twonorm	3	7	4.5	4.5	10	9	8	6	2	1

Table A.4: Means of accuracies for 20% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	0.968	0.972	0.971	0.971	0.968	0.970	0.971	0.971	0.970	**0.972**
EEG	**0.883**	0.820	0.820	0.820	0.880	0.847	0.804	0.538	0.817	0.821
MAGIC	**0.843**	0.822	0.823	0.823	0.841	0.840	0.817	0.834	0.835	0.833
Ozone-1	0.976	0.976	0.976	0.976	0.976	0.977	**0.977**	0.977	0.976	0.976
Ozone-8	0.940	0.941	**0.941**	**0.941**	0.940	0.941	0.940	0.940	0.940	0.941
QSAR	0.857	0.856	0.856	0.856	**0.860**	0.859	0.856	0.858	0.859	0.860
Shuttle	0.999	0.998	**0.999**	**0.999**	0.996	0.995	0.972	0.998	0.991	0.997
Spambase	**0.930**	0.914	0.912	0.912	0.924	0.924	0.918	0.918	0.928	0.917
Waveform	0.847	0.849	0.850	0.850	0.848	0.847	0.846	0.849	**0.850**	0.847
Yeast	0.517	0.514	0.513	0.513	0.518	**0.520**	0.516	0.518	0.519	0.516
Ringnorm	0.943	0.944	0.944	0.944	0.944	0.944	0.944	0.945	0.945	**0.948**
Twonorm	0.963	0.963	0.963	0.963	0.963	0.963	0.963	0.963	0.964	**0.964**

Table A.5: Ranks of accuracies of the imputation methods for 30% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	9	1	4	4	10	8	2	7	4	6
EEG	1	6	4	5	2	3	9	10	8	7
MAGIC	1	9	7.5	7.5	2	3	10	5	4	6
Ozone-1	10	7	5.5	5.5	8	9	1	3	4	2
Ozone-8	10	4	1.5	1.5	6	3	9	7	8	5
QSAR	3	10	7.5	7.5	4	2	9	1	6	5
Shuttle	3	7	1.5	1.5	5	8	10	6	9	4
Spambase	1	7	8.5	8.5	3	5	10	6	2	4
Waveform	8	2	3.5	3.5	7	9	10	6	1	5
Yeast	5	8	9.5	9.5	3	1	7	4	2	6
Ringnorm	10	9	6.5	6.5	4	8	5	3	2	1
Twonorm	10	5	3.5	3.5	8	6	7	9	2	1

Table A.6: Means of accuracies for 30% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	0.967	**0.970**	0.969	0.969	0.964	0.968	0.969	0.969	0.969	0.969
EEG	**0.865**	0.789	0.792	0.792	0.862	0.823	0.772	0.537	0.784	0.788
MAGIC	**0.822**	0.779	0.780	0.780	0.818	0.817	0.774	0.804	0.810	0.803
Ozone-1	0.974	0.976	0.976	0.976	0.975	0.975	**0.977**	0.976	0.976	0.977
Ozone-8	0.940	0.940	**0.941**	**0.941**	0.940	0.941	0.940	0.940	0.940	0.940
QSAR	0.857	0.845	0.848	0.848	0.856	0.858	0.848	**0.858**	0.850	0.851
Shuttle	0.999	0.991	**0.999**	**0.999**	0.994	0.989	0.941	0.994	0.987	0.996
Spambase	**0.926**	0.906	0.902	0.902	0.914	0.910	0.901	0.908	0.921	0.911
Waveform	0.835	0.839	0.838	0.838	0.835	0.833	0.827	0.836	**0.839**	0.837
Yeast	0.517	0.514	0.513	0.513	0.518	**0.520**	0.516	0.517	0.519	0.516
Ringnorm	0.905	0.905	0.906	0.906	0.907	0.906	0.906	0.909	0.909	**0.917**
Twonorm	0.951	0.952	0.952	0.952	0.952	0.952	0.952	0.951	0.952	**0.952**

Table A.7: Ranks of accuracies of the imputation methods for 40% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	8	3.5	5.5	5.5	10	9	3.5	2	1	7
EEG	1	8	7	6	2	3	9	10	5	4
MAGIC	1	9	7.5	7.5	2	3	10	5	4	6
Ozone-1	10	4	6.5	6.5	9	8	1	5	2	3
Ozone-8	9	5	1.5	1.5	3.5	3.5	10	6	8	7
QSAR	1	10	4.5	4.5	3	2	9	7	6	8
Shuttle	1	8	6.5	6.5	2	3	10	4	9	5
Spambase	1	8	9.5	9.5	4	3	6	7	2	5
Waveform	8	4	2.5	2.5	6	9	10	5	1	7
Yeast	3	10	8.5	8.5	4	1	7	5	2	6
Ringnorm	5	9	6.5	6.5	4	10	8	3	2	1
Twonorm	7	5	3.5	3.5	8	9	6	10	1	2

Table A.8: Means of accuracies for 40% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	0.962	0.965	0.964	0.964	0.958	0.961	0.965	0.965	**0.966**	0.962
EEG	**0.804**	0.704	0.707	0.707	0.803	0.749	0.691	0.535	0.708	0.710
MAGIC	**0.814**	0.758	0.759	0.759	0.811	0.810	0.751	0.787	0.796	0.783
Ozone-1	0.974	0.976	0.976	0.976	0.975	0.975	**0.978**	0.976	0.977	0.976
Ozone-8	0.939	0.940	**0.940**	**0.940**	0.940	0.940	0.939	0.940	0.940	0.940
QSAR	**0.854**	0.842	0.845	0.845	0.852	0.852	0.842	0.843	0.843	0.842
Shuttle	**0.988**	0.933	0.935	0.935	0.976	0.971	0.895	0.958	0.923	0.946
Spambase	**0.915**	0.892	0.888	0.888	0.899	0.901	0.894	0.893	0.905	0.894
Waveform	0.825	0.830	0.830	0.830	0.826	0.823	0.817	0.829	**0.830**	0.825
Yeast	0.479	0.470	0.471	0.471	0.478	**0.481**	0.471	0.476	0.480	0.475
Ringnorm	0.849	0.847	0.849	0.849	0.850	0.847	0.848	0.854	0.856	**0.872**
Twonorm	0.938	0.939	0.939	0.939	0.938	0.938	0.939	0.938	**0.940**	0.940

Table A.9: Ranks of accuracies of the imputation methods for 50% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	9	3	6.5	6.5	10	8	2	1	4	5
EEG	1	6	7.5	7.5	2	3	9	10	4	5
MAGIC	1	9	7.5	7.5	2	3	10	6	4	5
Ozone-1	10	8	5.5	5.5	7	9	1	3	4	2
Ozone-8	8	5	2.5	2.5	1	4	10	6	9	7
QSAR	3	7	4.5	4.5	1	2	6	8	9	10
Shuttle	1	9	3.5	3.5	2	6	10	5	8	7
Spambase	1	5	7.5	7.5	4	6	10	9	2	3
Waveform	6	1	2.5	2.5	8	9	10	5	4	7
Yeast	10	8	6.5	6.5	5	1	4	3	2	9
Ringnorm	3	10	6.5	6.5	5	9	8	4	2	1
Twonorm	9	5	2.5	2.5	6	8	7	10	4	1

Table A.10: Means of accuracies for 50% of missing features.

	k-NN	MICE	linreg-li	linreg-iter	XGBT-li	XGBT-iter	DAE	VAEAC	GAIN	WGAIN
Cancer	0.960	0.963	0.962	0.962	0.957	0.960	0.964	**0.964**	0.963	0.963
EEG	**0.769**	0.659	0.657	0.657	0.759	0.702	0.648	0.526	0.666	0.665
MAGIC	**0.799**	0.735	0.738	0.738	0.794	0.793	0.718	0.757	0.775	0.760
Ozone-1	0.974	0.975	0.976	0.976	0.975	0.975	**0.978**	0.978	0.978	0.978
Ozone-8	0.939	0.940	0.940	0.940	**0.940**	0.940	0.939	0.940	0.939	0.940
QSAR	0.842	0.824	0.832	0.832	**0.846**	0.843	0.825	0.821	0.819	0.816
Shuttle	**0.998**	0.974	0.989	0.989	0.992	0.988	0.932	0.988	0.979	0.981
Spambase	**0.909**	0.879	0.875	0.875	0.881	0.875	0.867	0.868	0.898	0.886
Waveform	0.807	**0.812**	0.811	0.811	0.805	0.801	0.796	0.808	0.811	0.806
Yeast	0.433	0.440	0.440	0.440	0.440	**0.447**	0.442	0.444	0.447	0.436
Ringnorm	0.778	0.767	0.769	0.769	0.774	0.767	0.768	0.777	0.781	**0.803**
Twonorm	0.918	0.920	0.921	0.921	0.920	0.919	0.920	0.918	0.921	**0.921**