On Domain Adaptation Invariance of Deep Classification Machines

a Masters's Thesis

by CAN DENIS SAVCI

supervised by
Prof. Dr. Tobias Damm (RPTU)
Dr. Alex Sarishvili (ITWM)
Julia Burr (ITWM)

December 2023
Department of Mathematics

Abstract

In the rapidly evolving landscape of Artificial Intelligence and Machine Learning applications excelling at difficult challenges within various fields, a theory providing solid explanations for the success becomes pivotal for the creation of novel and robust Machine Learning solutions for a large variety of tasks. Noticing a surge in empirical evidence for the value of Neural Networks in the realm of Machine Learning tasks, there remains a conspicuous gap in our comprehension of why Neural Networks exhibit such exceptional effectiveness.

This master's thesis is concerned with a novel hypothesis on Domain Adaptation, a well-known subfield of classification Machine Learning problems. The hypothesis aims to shed light on the power of deep classifiers claiming that deep Neural Networks inherently exhibit certain invariance to some domain shifts. This claim relies on classical theory together with novel ideas proposed in modern literature.

Our work first investigates the theoretical foundations of the hypothesis and puts the proposed explanations to the test. We illustrate the limits of the existing body of theoretical results. Afterward, this thesis empirically validates the practical value of the hypothesis.

While most of the hypothesis underlying claims can be proven in detail, our findings reveal its limitations and inadequacies in fully explaining empirically observed outcomes. The thesis shows that despite appearing empirically sound, the hypothesis falls short of providing a universally applicable and bulletproof explanation for the performance of practical deep Neural Network applications.

Contents

List of Symbols

$\mathbb{N}$	Natural numbers including zero
$\mathbb{Z}$	Integer number
$\mathbb{R}$	Real numbers
x^t	If x is a vector or matrix, x^t is the corresponding transposed vector or matrix
e	Euler's number
$\log$	Natural logarithm, logarithm to base e
$\log_2$	Logarithm to base 2
$\simeq (, \gtrsim)$	Asymptotically (greater or) equal
$\propto$	Linearly proportional to
$\|\cdot\|$	Euclidean vector norm
$\|\cdot\|_\infty$	supremum norm
$\mathbb{1}\{\cdot\}$	Boolean indicator function: $\mathbb{1}\{b\} = \begin{cases} 1 \text{ if } b \text{ is true} \\ 0 \text{ if } b \text{ is false} \end{cases}$
$\mathbb{1}_{\mathbb{R}_{\geq 0}}(\cdot)$	Heavyside function in $\mathbb{R}$: $\mathbb{1}_{\mathbb{R}_{\geq 0}}(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ else} \end{cases}$

Introduction and Overview

"Overall, elucidating the nature and mechanisms of AI systems [...] is a formidable challenge that has suddenly become important and urgent."
- Microsoft Research [BCE+23]

Deep Neural Networks - An Object of Particular Interest

Deep Neural Networks, that is Neural Networks with at least one hidden layer, are one of the main reasons people of all backgrounds are talking about Machine Learning and AI. With advancements like political deep fakes, powerful chess engines and GPT-4 which even seems able to reason like a human, cf. [BCE+23], many people have encountered Machine Learning or AI in their daily life. Furthermore, if we consider economic relevance, according to Bughin et al. on behalf of McKinsey & Company in [BC18], AI could boost the global economy by 16% (roughly $13 trillion) by 2030 compared to 2018, with ongoing growth of 1.2% each year, akin to past tech impacts like the invention of steam engines ($\sim$0.3% per year between 1850 and 1910) and robot introduction in production processes ($\sim$0.4% per year). Research by Rao and Verweij on behalf of Pricewaterhouse-Coopers GmbH (PwC) estimates similar effects ([RV17]).
Deep Neural Networks' omnipresence in media, their undeniable success in a wide variety of applications and their simple set-up make them an object of particular interest for foundational and explanatory research. Since their upcoming in the 1960s (e.g. [IL67],[Ama67]), the properties of deep Neural Networks have been investigated in the literature (as possible entry points refer to the reviews [BGKP22] and [TKOSK23]). Nevertheless, the current success cannot be explained using mathematical theory. Berner et al. provide a long yet not exhaustive list of open questions regarding the performance of Neural Networks in [BGKP22]. Why do deep architectures not overfit, while being powerful enough to fit most data perfectly? Why do classical Machine Learning heuristics like explicit regularization have seemingly no impact, but otherwise avoided heuristics like over-parametrization do even improve generalization? What is the role of the optimization, why is Stochastic Gradient Descent performing this well and why does it not get stuck in bad local minima? What is the role of depth?
Answering these questions will be the work of years of research and out of the scope of this thesis. As a small step in this direction, we will investigate one proposed explanatory approach that has the potential to aid in understanding the success of deep Neural Networks. This proposal explicitly makes claims about the effect of depth and has a particular emphasis on the subject of **Domain Adaptation**, a sub-class of Machine Learning problems.
The term "domain" refers to the distribution of data, including factors like the data origin and the environment or conditions under which the data was collected. Domain Adaptation thus cares about the challenges arising when a Machine Learning algorithm is applied to a domain that is different from the domain it was optimized or trained on. While it is a whole research field itself, understanding Domain Adaptation capabilities

also aids the development of Neural Networks. A well-known problem of very deep Neural Networks is so-called adversarial attacks, i.e. attacks that manipulate inputs on a scale not humanly recognizable, but resulting in fatally wrong outputs.
An easy-to-grasp example is "traffic sign attacks". By manipulation of pixels in the digital or the application of special stickers in a real-world setting, image recognition algorithms are purposefully tricked to classify traffic signs wrong. Within [PLZ23], Pavlitska et al. provided a brief insight into how severe even small changes can be. They conclude that sophisticated traffic sign detection can be misled by changes that are easily missed by humans. Controlling an autonomic car based on such traffic sign detection might fail with fatal consequences.

The History of Deep Neural Networks

The majority of research regarding Neural Networks and their applications is not even 10 years old. The underlying concepts however exist far longer. In 1943, W. McCulloch and W. Pitts proposed an abstract calculation procedure inspired by human brain neurons, which could perform simple logical calculations. This structure is referred to as McCulloch and Pitts Neuron ([MP43]). In 1958, F. Rosenblatt extended the idea of human-like neurons to define so-called perceptrons similar to the computation units used in Neural Networks to this day ([Ros58]). The next milestone was the formulation of backpropagation often attributed to P. Werbos and his publication [WJ74]. Backpropagation is an efficient way to apply the chain rule to calculate gradients of concatenated layers of perceptrons, i.e. deep Neural Networks. This makes gradient-based optimization possible and is nowadays still the main key to training Neural Networks. However, due to issues like vanishing gradients and no extraordinary performance in the beginning, it took almost 40 years until Geoffrey Hinton et al. reinvented backpropagation in 2012 and combined it with effective regularization methods to show that deep Neural Networks could beat state-of-the-art image recognition algorithms in varying tasks ([KSH12a]). In the same year, a large-scale object recognition network by Google Brain made headlines as it showed possible detection of human bodies, faces or cats in different image data sets ([LRM+12]). With these first publicly noticed and highly successful deep Neural Network applications, a deep learning boom started. This can be precisely tracked using the number of publications regarding the topic of deep learning per year. In the early 2000s, a few hundred research papers were published each year. In 2012 we observed almost a thousand publications before an exponential growth started peaking at approximately 100 000 publications in each year 2022 and 2023[1].
Since 2012, deep learning has shown success in a wide variety of applications. A non-exhaustive list of popular fields for application is disease classification ([ALS22]), image classification ([RW17]), object detection ([ZZXW19]), speech emotion recognition ([MCA+23]), text summarizing and sentiment analysis ([ZZY+22] and [LJQ20]), malware detection ([MS23]), face recognition ([WD21]), facial expression analysis ([MH20]) and many more.

[1] According to Scopus.com, accessed on 11.12.2023

As mentioned, this thesis is also concerned with Domain Adaptation problems where the data we train on (sampled from a so-called source domain or source distribution) has intrinsic differences to the data we want to apply the trained algorithm to (sampled from a so-called target domain or distribution). Looking into Domain Adaptation performance and performance-enhancing heuristics is often motivated by applications that lack labeled real-world data points, where one is left to train with synthetic data. In this case, one needs to know in which cases the trained Machine Learning algorithm performs well in the real world.
Theory on the performance drop due to the cross-domain setting was established in the 2000s by Blitzer et al. ([BCK+08]) and Mansour et al. ([MMR09]) in different learning frameworks.
Domain Adaptation research is often about constructing special algorithms and optimization regularizers that use a small quantity of unlabeled target data points in addition to the labeled (source domain) training data to enforce the final classifier to perform similarly well on both data distributions. This can be done by aligning the distributions of source and target for example concerning the indistinguishability of samples ([GUA+16]), conditional distribution matching of the data ([dCZWG20]) or information-theoretical similarity of distributions ([NTG+21]).

A Hypothesis Worth a Closer Look

As seen, the existing body of literature on Neural Networks and their applications in various research domains is undeniably extensive and rich with numerous successful practical examples. They greatly extend the range of the Machine Learning toolbox. However, the literature's collected knowledge falls short when trying to convey a comprehensive understanding of the underlying mathematical principles that account for the success of Neural Networks. It lacks theoretical guarantees for completely new practical applications and an explanation for the remarkable performance already observed. In the long run, it is only reasonable to make numerous assertions on the fundamental theory of Neural Networks and test their validity.
For this purpose, this thesis investigates a promising **hypothesis provided by Dr. Alex Sarishvili** on behalf of Fraunhofer ITWM, department System Analysis, Prognosis and Control. The proposed hypothesis regarding classification problems can be qualitatively broken down to

Sarishvili's Hypothesis 0.1.

> *"Trained deep Neural Network generalization performance is invariant regarding multiple domain data scenarios and needs no special architecture or regularization terms to tackle Domain Adaptation problems."*

We investigate a possible explanation and the motivation behind this hypothesis as an interplay of several classical results and intuitions which are yet to be tested for their theoretical and practical utility. The core of the hypothesis is motivated by the manifold disentangling perspective of Neural Network training ([BWS15], cf. Section 4.1.2)

and the Information Bottleneck framework ([ST17], cf. Section 3.3.1) of deep Neural Networks. It aims to combine these ideas with classical Machine Learning results and information theoretical measures.
Additionally, the hypothesis aims to evaluate in how far special domain alignment heuristics such as Kullback-Leibler alignment ([NTG+21]) and network architectures like DANN ("domain adversarial training of Neural Networks", [GUA+16], Example 4.1) are necessary or merely redundant safety measures to aid Domain Adaptation performance.
The hypothesis suggests the investigation of Neural Network induced deep data representations (cf. Definition 2.5), i.e. intermediate results of the feed-forward evaluation of Neural Networks. Sarishvili proposes that by classical training, deep classification networks already establish deep representations of input data that primarily contain the decisive information for correct classification and not necessarily more. While learning important class-defining data features, Neural Networks simultaneously conveniently forget features dispensable for the classification task. The consequence would be an invariance to small domain shifts. If a deep classifier performs well on one domain and does not attribute importance to the feature differing from the other domain it should also perform well there. This automatic overlap of the domain representations theoretically makes additional architectures like DANN redundant.
The proposal further states that the interplay of performance and representation can be described by information theoretical measurements possibly leading to a novel optimization objective for training Neural Networks. If this reformulated optimization is equivalent to regular training, it yields a more understandable interpretation thereof and thus helps to understand the success of Neural Networks.

Proposal 0.2 (Domain Adaptation Invariance Graph)**.**

A precise and cohesive description of the previously explained idea can be illustrated as a graph of mathematical connections based on classical results and pure mathematical intuition. To provide bookmarks for the rest of the thesis we will add a small description of the connections which will be clearer when the definitions and theorems are investigated throughout this thesis.
To easily distinguish the two fields of mathematical theory, we identify typical **Machine Learning** properties with the color blue ●.
Information theoretical measures are going to be associated with the color green ●.
Grey outlines are reserved for the problem setting, i.e. the network's architecture and training.
Up and down arrows visualize claimed in- or decrease of the respective value in $\mathbb{R}$.

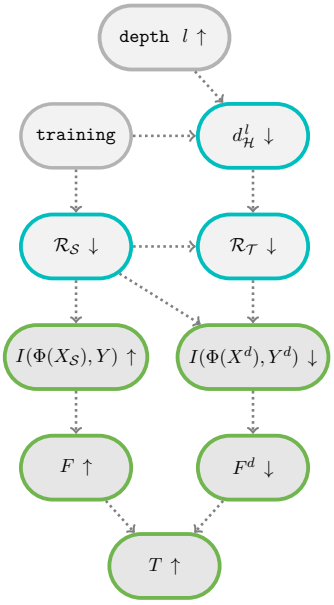


Figure 1: Scematic representation of the steps investigated to validate Sarishvili's Hypothesis 0.1.

The connections in the infographic in Fig. 4 can be described step by step. We will refer to single steps multiple times throughout this thesis. Note, that the necessary terms and definitions to describe the proposal are skipped now but are not expected to be known to the reader yet. For the sake of brevity here, their precise introduction is shifted to the upcoming theory sections where their properties and connections are also investigated.

(a) Classical **Machine Learning** theory is often concerned with learning from finite data sampled from unknown distributions. These distributions often define so-called data domains. The main novel claim is the conjecture of reduction in domain separability. The Machine Learning specific properties are claimed to be connected as follows.

  (i) When **training** a Neural Network on a sample from the training or **source domain** (emphasized by "$\mathcal{S}$"), the general **misclassification risk** $\mathcal{R}_{\mathcal{S}}$ of the network on the whole domain is reduced.

  (ii) During the **training** $d^l_{\mathcal{H}}$, measuring the **difference in representations** in the l-th layer between the domains, decreases. The magnitude of this effect depends on the layer **depth** l.

(iii) These two effects of training then imply that the **misclassification risk** $\mathcal{R}_{\mathcal{T}}$ is small on the testing or **target domain** (emphasized by "$\mathcal{T}$"), i.e. after domain shift.

(b) The **connection** between both fields relates misclassification risks to the information about classes Y and data domains Y^d that are preserved along network representations. Y^d is a binary labeling assigning the domain of origin to data. Both Y and Y^d are considered as random variables.

(i) Good Neural Network classification performance relates to source data representations $\Phi(X_{\mathcal{S}})$ and class labels Y with high mutual information.

(ii) If a deep classifier performs well on two domains $\mathcal{S}$ and $\mathcal{T}$, this means that the deep representations of data X^d originating from either domain $\Phi(X^d)$ have low mutual information with Y^d labeling from which domain the data is sampled.

(c) **Information Theory** on its own aims to describe probability distributions and their predictability. Thereby, it can measure the similarity of distributions by measuring how predictable one distribution is when another one is known.

(i) Fidelity F is a measure to evaluate whether representations are useful for classifications and still as low in complexity as possible. Fidelity relates to the performance of a classification task measured by the preserved information about the labels. It can be applied to both the data classification by labels (pairing $X_{\mathcal{S}}$ and Y) and the classification of data with respect to its domain of origin as an additional label (pairing X^d and Y^d).

(ii) Finally the two paths are merged in an optimization of representations, enforcing high classification fidelity F and low domain classification fidelity F^d.

This thesis investigates which of the connections and properties that the proposal relies on are provable and support Sarishvili's hypothesis and at which parts the proposal cannot achieve the transition from a meaningful intuition to a solid theoretical framework. We will investigate this graph in detail in the following chapters. We will examine which relationships are mathematically sound and provable, which are reasonable but do not suffice as an explanation, and which do not hold in general. Additionally, we study to what extent, if at all, the theorems provide practical guarantees.

Outline of the Thesis

Section 1 establishes a mathematical framework to investigate Machine Learning problems. It proceeds to present classical theoretical results, regarding claims Proposal 0.2 (a) (i) and (iii). Section 2 is concerned with the practical applicability of the preceding results and further provides deeper insight into modern literature's approaches of dealing with Neural Networks. After this purely Machine Learning focused part, Section 3 shifts to an information theoretical framework and investigates Proposal 0.2 claims (b) (i) and

(ii) and (c) (i) and (ii). Finally in Section 4 we investigate claim (a) (ii) empirically by experimentation and extend the idea of the claim to a more specific analysis of Neural Network data representations.

1. Insights into Domain Adaptation Invariance

> *"Every time a scientific paper presents a bit of data, it's accompanied by an error bar - a quiet but insistent reminder that no knowledge is complete or perfect. It's a calibration of how much we trust what we think we know. If the error bars are small, the accuracy of our empirical knowledge is high; if the error bars are large, then so is the uncertainty in our knowledge."*
> - Carl Sagan [Sag97]

The goal of this thesis is to thoroughly discuss Proposal 0.2. The first and most reasonable course of action is to investigate the different claims individually and, if possible, from a purely mathematical point of view in a precisely defined environment. In order to yield productive discussions later, this chapter establishes the theoretical foundation of this thesis. We establish Machine Learning in a mathematical sense and provide an overview of classical theory backing the first part of the proposal.

1.1. The Machine Learning Framework

The most common formulation of a Machine Learning problem is that one tries to learn properties of an unknown distribution by investigating data points distributed according to it. If we care about Domain Adaptation problems, this sample is not exactly from the desired distribution but a slightly shifted one. All this can be defined as follows.

Definition 1.1 (Data)**.**
Let $\mathcal{X} \subseteq \mathbb{R}^m$, $\mathcal{Y} \subseteq \mathbb{R}^d$ be non-empty real valued sets. We call elements of $\mathcal{Y}$ **labels** and elements of $\mathcal{X}$ **data points**. A **data domain** consists of a distribution over $\mathcal{X}$ for unlabeled or over $\mathcal{X}\times\mathcal{Y}$ for labeled data.
We typically write $\mathrm{P}_{\mathcal{S}}$ or $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$ for the **data distributions** underlying the unlabeled or labeled training data. If another additional distribution needs to be considered it is denoted as $\mathrm{P}_{\mathcal{T}}$ or $\mathrm{P}_{\mathcal{T},\mathcal{Y}}$.

Definition 1.2 (Source and Target Domain)**.**
If multiple distributions are in place, we call $\mathcal{S}=(\mathcal{X}, \mathrm{P}_{\mathcal{S}})$ or $\mathcal{S}=(\mathcal{X}\times\mathcal{Y}, \mathrm{P}_{\mathcal{S},\mathcal{Y}})$ the **source data domain** or **training domain** and $\mathcal{T}=(\mathcal{X}, \mathrm{P}_{\mathcal{T}})$ or $\mathcal{T}=(\mathcal{X}\times\mathcal{Y}, \mathrm{P}_{\mathcal{T},\mathcal{Y}})$ the **target data domain** or **testing domain**. Whether the considered data points are labeled or unlabeled will be clear from the respective context.

Multiple random variables sampled from such data distributions are collected in a sample.

Definition 1.3 (Samples)**.**
A **sample** $\mathrm{S}^n = (\mathrm{S}_1, \ldots, \mathrm{S}_n)$ of size $n \in \mathbb{N}_{>0}$ is a collection of random variables S_i on $\mathcal{X}$ or $\mathcal{X}\times\mathcal{Y}$. Realizations of S^n will be denoted by lowercase letters. One sample realization $\mathrm{s}^n = (\mathrm{s}_1, \ldots, \mathrm{s}_n)$ consists of i.i.d. realizations of the random variables in S^n.

Notation 1.4.
If we want to emphasize that all points in one sample S^n are some unlabeled data points from one of the distributions, we write $X_{\mathcal{S}}^n$ or $X_{\mathcal{T}}^n$ instead of S^n. Similarly, if all these points are labeled, we write $(X, Y)_{\mathcal{S}}^n$ or $(X, Y)_{\mathcal{T}}^n$.
In general these points will be random variables according to one of our distributions. Analogously, for realizations we will use lowercase letters $x_{\mathcal{S}}^n$, $x_{\mathcal{T}}^n$, $(x, y)_{\mathcal{S}}^n$ and $(x, y)_{\mathcal{T}}^n$. The additional mark $\mathcal{S}$ or $\mathcal{T}$ will be omitted if the underlying data distribution is clear from the context.

As mentioned, Machine Learning algorithms aim to learn useful information about the labeling of data distributed according to the true data distribution. Usually, this means guessing a map from a predefined function class that emulates the correspondence between data and labels. A common way to formulate this is as an optimization problem with an objective function. Many algorithms and theoretical results use an estimated sample loss.

Definition 1.5 (Loss and Risk)**.**
Let $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ be the set of measurable functions mapping $\mathcal{X}$ to $\mathcal{Y}$. A **loss** function

$$\mathcal{L} \colon \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \longrightarrow \mathbb{R}$$

assigns real values to a pair consisting of a map from $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ and a labeled data point.
The **risk** of f is defined for any distribution P over the space of labeled data $\mathcal{X} \times \mathcal{Y}$ as

$$\mathcal{R}_{\mathrm{P}}(f) = \mathbb{E}_{(X,Y)\sim \mathrm{P}} \left[\mathcal{L}\left(f, (X, Y)\right) \right].$$

For any labeled sample realization $\mathrm{S}^n = (X, Y)^n$ the random variable of the mean loss of the sample for $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ is

$$\overline{\mathcal{R}}_{\mathrm{S}^n}(f) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f, (X_i, Y_i)\right).$$

The realization of this random variable for $\mathrm{s}^n = (x, y)^n$ is called **empirical risk** of f,

$$\overline{\mathcal{R}}_{\mathrm{s}^n}(f) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f, (x_i, y_i)\right).$$

One may find it challenging to identify a good or even optimal choice for f in $\mathcal{M}(\mathcal{X}, \mathcal{Y})$, as the set of all measurable functions yields a vast search space. Further, it might be desirable to find a map from a certain family, e.g. linear functions, polynomials, or functions that Neural Networks can estimate. Either way, these considerations lead us to the concept of so-called hypothesis classes as a predefined restriction of the space of considered function.

Definition 1.6 (Hypothesis Class)**.**
A **hypothesis class** $\mathcal{H}$ is a subset of $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ which can be finitely parameterized. This means there exists a function $\Psi\colon \mathcal{X} \times \Theta \to \mathcal{Y}$ with $\Theta \subseteq \mathbb{R}^o$ and $o \in \mathbb{N}$, s.t.

$$\mathcal{H} = \big\{\Psi(\,\cdot\,, \vartheta) \,|\, \vartheta \in \Theta\big\}.$$

The elements $h \in \mathcal{H}$ are called hypotheses.

Example 1.7 (Linear Binary Classification)**.**
When considering binary classification, i.e. $\mathcal{Y} = \{0, 1\}$, hypotheses are typically compositions of functions of a certain function class with a threshold.
Consider an (affinely) linear function g and the heavyside function $\mathbb{1}_{\mathbb{R}_{\geq 0}}$ i.e. a threshold identifying if its input is non-negative. Then

$$h(\,\cdot\,) = \mathbb{1}_{\mathbb{R}_{\geq 0}}\big(g(\,\cdot\,)\big)$$

is called a linear classifier. Thus, the hypothesis class of linear classifiers consists of classifiers discriminating data by a separating hyperplane defined by $\{x \,|\, g(x) = 0\}$ for affinely linear g.
In practice, we sometimes need to exactly flip the labeling of all data points. In the case of linear classifiers, this is simple. The converse hypothesis $h'(\,\cdot\,) = 1 - h(\,\cdot\,) = \mathbb{1}_{\mathbb{R}_{\geq 0}}(-g(\,\cdot\,))$ is an element of $\mathcal{H}$ for every h as constructed above.

With data and risk defined, the foundation for the rest of the thesis is set. Below it is illustrated in which way many Machine Learning (ML) problems, including title-giving Domain Adaptation, can be described in this setting.

Remark 1.8 (Machine Learning Problems)**.**
The most prominent ML problems can be formulated inside our framework:

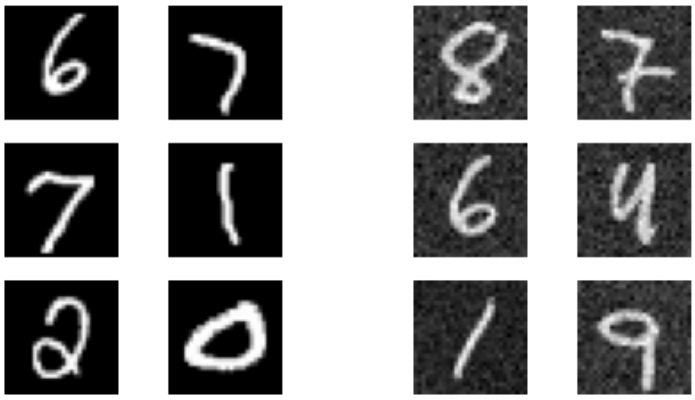
- *Classification tasks*: Consider a labeled sample $(X, Y)^n$, consisting of multidimensional data points $\mathcal{X} \subseteq \mathbb{R}^m$ and categorical labels $\mathcal{Y} = \{0, 1\}^d$, together with 0-1-loss
$$\mathcal{L}(f, (x, y)) = \mathbb{1}\big\{f(x) \neq y\big\} = \big\|f(x) - y\big\|_\infty$$
evaluating if $f(x)$ predicts the label y correctly or not.[2]
- *Regression tasks*: Consider again a labeled sample $(X, Y)^n$ with multidimensional data points $\mathcal{X} \subseteq \mathbb{R}^m$ and require interval or ordinal labels $\mathcal{Y} = \mathbb{R}^d$. An example of the loss is the squared error based on the Euclidean distance between true value y and prediction $f(x)$
$$\mathcal{L}(f, (x, y)) = \|f(x) - y\|^2$$
resulting in the empirical risk measured by the MSE (Mean Squared Error).

[2] $\mathbb{1}$ here is the Boolean indicator function, i.e. $\mathbb{1}\{b\} = 1$ if statement b is true, and $\mathbb{1}\{b\} = 0$ if b is false.

- *Clustering*: A simple example is KNN[3]. Consider an unlabeled sample X^n with $\mathcal{X} \subseteq \mathbb{R}^m$ and set $\mathcal{Y} \coloneqq 1, \ldots, k$ the cluster indicators. For proposed cluster centers $c_1, \ldots, c_k$, the standard loss is

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

Example 1.9 (Domain Adaptation)**.**
Classical Machine Learning theory is often concerned with evaluating the quality of empirical risk as an estimator for the true risk (cf. Theorem 1.21) by imposing the strict case that the training sample already only consists of points from the true data distribution, $\mathrm{S}^n = (X, Y)^n$. This leads to the typical training/test framework, which the reader should be familiar with already. We train the exact task, the learned hypothesis h should eventually be applied to.
With the choices in Definition 1.1 the extension called **Domain Adaptation** is possible as well. Domain Adaptation describes problems in which the training data used to estimate the risk is sampled from a distribution typically slightly different from the true data distribution. A simple example is working on training sets with no noise or



(a) Handwritten digits as data from the source domain.

(b) Handwritten digits with added background noise as target data.

Figure 2: A simple example of Domain Adaptation data.

different noise compared to some real-world data that the trained algorithm will be applied to. More abstractly spoken, it aims at minimizing the impact of changes in data features in $\mathcal{X}$ that are assumed not to be important for classification. Think about the background in most image data or noise in audio data which could and should be ignored by classifiers, remember the concept of adversarial attacks we mentioned in the introduction. An example is shown in Fig. 2.
Another example is a problem where the exact real data distribution is not accessible

[3]K-nearest neighbor clustering algorithm.

and the researcher is left to train on a slightly different one. An easy-to-grasp instance is e-mail data and the task of spam detection because it differs from one receiver or user to another. Similarly, some learning data might be acquired on hardware different from the hardware on which the final algorithm is to be applied. This can be the case due to small differences in material, climate, or machines. Just consider training on data from a production line located in northern Europe and testing on another one in the dry subtropics with different synthetic substance suppliers.

Remark 1.10 (Domain Adaptation Samples)**.**
Some common assumptions on Domain Adaptation training samples can be categorized as

- $\mathrm{S}^n = (X, Y)_{\mathcal{S}}^n$ for blind Domain Adaptation,
- $\mathrm{S}^n = \left((X, Y)_{\mathcal{S}}^k, X_{\mathcal{T}}^{n-k}\right)$ for unsupervised Domain Adaptation or
- $\mathrm{S}^n = \left((X_{\mathcal{S}}, Y)_{\mathcal{S}}^k, (X, Y)_{\mathcal{T}}^{n-k}\right)$ for supervised Domain Adaptation

where k is a natural number smaller n. In the source-only case in Sarishvili's Hypothesis 0.1 we do blind domain adaptation. The classifier has never seen data from the target domain in training and cannot be sensitized to its features. Most state-of-the-art algorithms are unsupervised. Here the classifier is only trained on the labeled source data but the exact parameterization is also influenced by examples from the target domain. An example will be presented later in Example 4.1 and will be the focus of one experiment in Section 4.

For completeness, the concept of learning itself also has to be defined.

Definition 1.11 (Learning Process)**.**
Let $\mathcal{H}$ be a hypothesis class. A **learning algorithm** is any stochastic or deterministic algorithm choosing a correspondence map for $\mathcal{X}$ and $\mathcal{Y}$ from $\mathcal{H}$ after observing a sample of size n, either labeled, unlabeled or mixed.

$$\mathcal{A}\colon (\mathcal{X} \times \mathcal{Y})^k \times \mathcal{X}^{n-k} \to \mathcal{H},\ \mathrm{s}^n \mapsto \mathcal{A}(\mathrm{s}^n)$$

Typically the fixed sample size n is just a formality and such algorithms work in the same way independently of the number of sampled data points.

To streamline the upcoming theory, this thesis focuses on the following restricted problem class, avoiding unnecessary technicalities.

Assumption 1.12.
For the remaining thesis, assume all Machine Learning problems to be *binary classifications* with 0-1-*loss* if not explicitly stated otherwise. This means

$$\mathcal{L}\big(f, (x, y)\big) = \mathbb{1}\big\{f(x) \neq y\big\} \text{ and } \mathcal{Y} = \{0, 1\}.$$

Further, we assume as in Example 1.7 that for each hypothesis $h \in \mathcal{H}$ the *converse hypothesis* $1 - h$ is in $\mathcal{H}$. Further we will assume, that all sample points are drawn independently and all samples are nonempty, i.e. $n > 0$.

Note, that Assumption 1.12 is a severe restriction and certainly does *not* cover all Machine Learning problems without loss of generality! However, results with relaxed assumptions either rely on the theory presented throughout this thesis or become too convoluted and complex to offer insights into the proposed connections of Proposal 0.2. Furthermore, current research has not yet explained the success of such seemingly "simple" models like binary classification comprehensively. Addressing more complex problems would require a solid mathematical foundation by bridging these knowledge gaps. Now, with a well-defined scenario set up, a measure to compare different hypothesis classes not only by quality but also by complexity is needed. A classical complexity definition measures the ability to fit any possible labels to a set of data points. This measure is called the Vapnik-Chervonenkis dimension (VC dimension, [Vap99]). Note, that the theory provided by Vapnik and Chervonenkis will also guide the whole discussion about generalization throughout this thesis. Upcoming studies in Section 2.2 and Section 2.4 will justify the restriction to the approach or choice of VC theory. It is a comparably old and rich field concerning generalization theory making the discussions as insightful as possible. Additionally, the concept of complexity, as defined below, is comparably intuitive. Other possible generalization theorems will be briefly discussed in Section 2.4 as well.

Definition 1.13 (Growth Function and VC Dimension; [Vap99])**.**
Let $\mathcal{H}$ denote a hypothesis set. Using $\mathcal{H}$, define the **partition function** φ_X on a nonempty set $X = (x_1, \dots, x_n) \subseteq \mathcal{X}$ as the binary representations of possible hypotheses on X

$$\varphi_X \colon \mathcal{H} \to \mathcal{Y}^n = \{0,1\}^n,\ h \mapsto \big(h(x_1), \dots, h(x_n)\big)\,.$$

The map φ induces a finite representation of $\mathcal{H}$ called the **effect space** of $\mathcal{H}$ on X.

$$\mathcal{H}_X \coloneqq \varphi_X(\mathcal{H}) \subset \mathcal{Y}^n$$

The **growth function** measures the potential ability of $\mathcal{H}$ to achieve different classifications on *one* set of n points in $\mathcal{X}$:

$$\mathcal{G}_{\mathcal{H}}(n) \coloneqq \sup_{X \in \mathcal{X}^n} |\mathcal{H}_X|$$

$\mathcal{H}$ is said to **shatter** a set X of n points if

$$\mathcal{H}_X = \mathcal{Y}^n.$$

This means all possible binary labelings can be achieved which is equivalent to

$$\mathcal{G}_{\mathcal{H}}(n) = |\mathcal{H}_X| = 2^n.$$

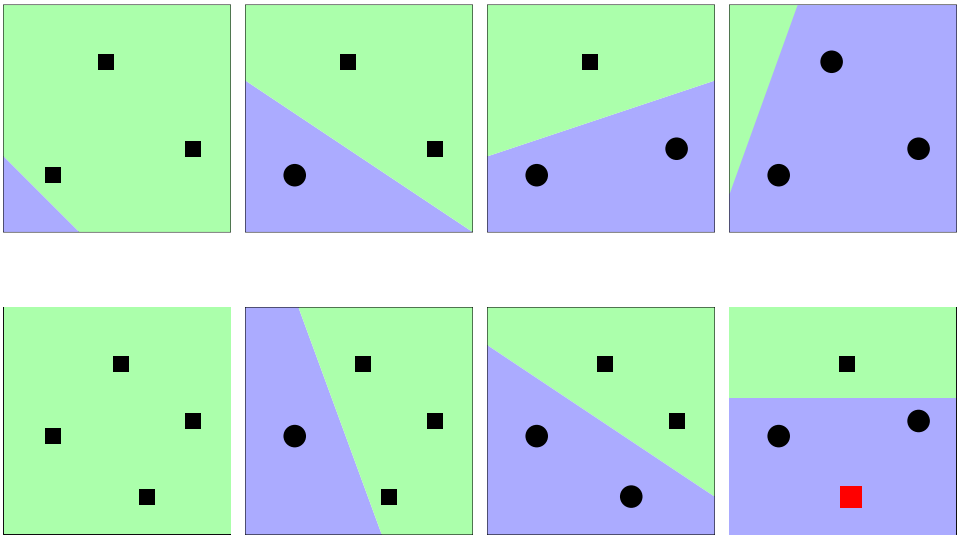


Figure 3: Hyperplane induced by example linear classifiers. Rectangle and circle symbolize the labels we try to fit or in this case, linearly separate. All label combinations are possible on three generically placed points in $\mathbb{R}^2$ and, up to symmetry, shown in the top row. On four points, we cannot achieve all label combinations. We cannot linearly separate the classes in the bottom right picture. The red rectangle symbolizes a misclassification.

Shattering means any label combinations can be achieved on the set X by a hypothesis in $\mathcal{H}$. The limit of the ability to shatter at least one possible set of a given size is the so-called **Vapnik-Chervonenkis dimension** (VC dimension)

$$\Delta_{\mathcal{H}} := \sup\{n \in \mathbb{N} \mid \mathcal{G}_{\mathcal{H}}(n) = 2^n\}.$$

A typical example to grasp the idea of VC dimension is linear classifiers on a two-dimensional space.

Example 1.14 (VC DIMENSION)**.** A linear classifier is graphically a hyperplane splitting the data space into one halfspace labeled 0 and one labeled 1. For data in $\mathbb{R}^2$ one can easily see that all labels on three points, that are not placed in a line, are achievable by linear classifiers.
If we introduce a fourth point such that the points are positioned in a general rectangle, we still can achieve almost all labeling. However, if two points on opposite rectangle corners are assigned one and the other two points the other label, we cannot find a hyperplane that separates the points according to these labels. This set of four points cannot be shattered by linear classifiers. This example is sketched in Fig. 3. We will see in Section 2.2.1 that no set of four points in $\mathbb{R}^2$ and more generally no set of $m + 2$ points in $\mathbb{R}^m$ can be shattered by linear classifiers.

Remark 1.15 (Properties of the VC Dimension)**.**
Some properties are immediate with the VC dimension.

(a) If $\Delta_{\mathcal{H}} = N \in \mathbb{N}_{>0}$, then $\mathcal{G}_{\mathcal{H}}(n) = 2^n$ holds for all $n \leq N$.

(b) Nevertheless, $\Delta_{\mathcal{H}} = N$ does not mean we can fit all labels on all data sets of cardinality $n \leq N$. Similarly, it does not mean any labeled sample of size larger N cannot be classified correctly by $\mathcal{H}$ hypotheses at all. We just cannot fit all possible labels.

(c) The complexity $\Delta_{\mathcal{H}}$ in general is no finite number. For example sine-classifiers

$$\mathcal{H} = \{h \mid \exists w \in \mathbb{R}^m \colon\ h(x) = \mathbb{1}_{\mathbb{R}_{\geq 0}}(\sin w^t x)\}$$

can be shown to shatter at least one set of n points in $\mathbb{R}^m$ for all $n \in \mathbb{N}$, i.e. $\Delta_{\mathcal{H}} = \infty$ (cf. Section 2.2.1).

The Vapnik-Chervonenkis framework laid out enables discussion of generalization guarantees.

1.2. Generalization

One of the most fundamental claims of Proposal 0.2 is claim (a) (i), the connection of the theory of domain distributions to empirical risks in applications. It is that with training, i.e. minimizing the loss on a finite sample, the expected misclassification error is also decreasing.

If we train a good classifier with zero training loss, we want to know whether this classifier will also perform well on unseen data. Another equivalent formulation is that the empirical misclassification error is a precise or possibly pessimistic estimate of the expected misclassification error, i.e. the risk of misclassifying unseen data. This concept is called *generalization*. While the success of Machine Learning applications of all kinds hints that good generalization is often achieved, proving it mathematically is no finished research topic. Classical results try to prove generalization by providing uniform bounds on the true risk across the whole data and hypothesis space. These also called worst-case bounds yield understandable explanations for why a classifier works or does not work. Possible explanations can be based on the already presented complexity of hypothesis classes. The bounds can also be used to characterize notions like over- and underfitting.

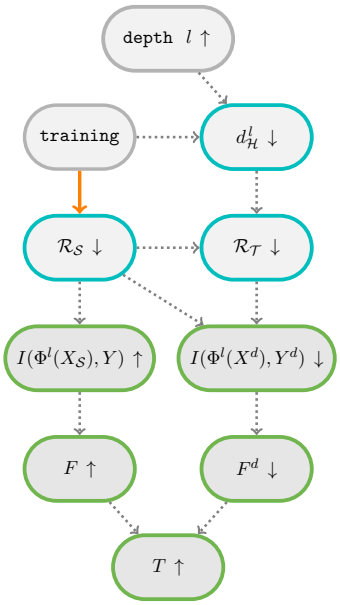


Figure 4: Here in Section 1.2 we look into generalization and Proposal 0.2 (a) (i). Orange arrows now and in later sections emphasize the claim which the respective section investigates.

However, as we will see the explanatory capability of uniform bounds could diminish when describing state-of-the-art Machine Learning algorithms and powerful Neural Network architectures on their own (cf. Section 2).

The upcoming theorems are dedicated to proving a uniform generalization bound (Theorem 1.21). For this purpose, some helpful lemmata investigating the gap between risk and empirical risk are needed. While found in many different variations, proving this main bound relies on a method called "symmetrization" of the generalization gap.
Note that the upcoming proof of this symmetrization method in Lemma 1.17 borrows crucial ideas and its structure from a proof of the Glivenko-Cantelli Theorem by Devroye et al. in their book [DGL96].
Since we consider many distributions of expressions dependent on a multitude of differ-

ent random variables, some calculations in the upcoming proofs can be difficult to read. To avoid unclarity we explicitly denote underlying random variables in the following way.

Notation 1.16.
Consider a sample $\mathrm{S}^n = (\mathrm{S}_1, \dots, \mathrm{S}_n)$ and a boolean expression $\mathbb{B}$ which maps sample realizations to True or False. Then we will often write

$$\mathrm{P}_{\mathrm{S}^n}\big(\mathbb{B}(\mathrm{S}_1, \dots, \mathrm{S}_n)\big) \coloneqq \mathrm{P}\big(\mathbb{B}(\mathrm{S}_1, \dots, \mathrm{S}_n) = \mathrm{True}\big)$$

to emphasize that the distribution is induced by the distributions of S_1 to S_n.

Now we prove symmetrization, a tool to make statements about the misclassification risk (Definition 1.5, Remark 1.8)

$$\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) = \mathbb{E}_{(X,Y)\sim \mathrm{P}_{\mathcal{S},\mathcal{Y}}}\Big[\mathbb{1}\{h(X) \neq Y\}\Big]$$

using empirical risk (Definition 1.5) without knowing the underlying data distribution at all. Note, that the symmetrization Lemma is a necessary step in the proofs of classical VC learning theory attributed to Vapnik and Chervonenkis ([VC71, Theorem 2]) and it was later collected as a sub-result in [DGL96, Theorems 12.4 and 12.5].

Lemma 1.17 (Symmetrization by Ghost-Samples; [VC71][DGL96])**.**
Let $\mathcal{H}$ be a hypothesis class and let $\mathcal{G}_{\mathcal{H}}$ denote the corresponding growth function. Further, let $(X,Y)^n$ and $(X',Y')^n$ be two independent identically distributed samples of size $n > 0$ both distributed according to $P_{\mathcal{S},\mathcal{Y}}$.
For all positive ε with $\varepsilon \geq \sqrt{\frac{2}{n}}$ it holds

$$\sup_{h\in\mathcal{H}} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right) \tag{1.1}$$

$$\leq 2\,\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right) \tag{1.2}$$

$$\leq 2\,\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right). \tag{1.3}$$

Proof. As mentioned, the upcoming proof is adapted from [DGL96, Theorem 12.4]. Both inequalities are proven by applying so-called symmetrization techniques which are used to establish many classical uniform bounds. We prove the claim for an arbitrary hypothesis $h^{(0)} \in \mathcal{H}$ in the left-hand side expression. This yields the claim after taking the supremum.

The first part of the proof is validating

$$\sup_{h\in\mathcal{H}} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right)$$

$$\leq 2\,\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right).$$

The goal of this first inequality is to get rid of the dependence on the true risk since it is in general unknown and hard to use for further calculation. The proof idea is called *symmetrization by ghost samples*.
Let $h^{(0)} \in \mathcal{H}$ be arbitrary. Using the independence of the samples of $(X,Y)^n$ and $(X',Y')^n$, one obtains

$$\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right)$$

$$\geq \mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)}) - \overline{\mathcal{R}}_{(X,Y)^n}(h^{(0)}) \geq \frac{\varepsilon}{2}\right)$$

$$= \mathrm{P}_{(X,Y)^n,(X',Y')^n}\Big(\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X,Y)^n}(h^{(0)})\right) -$$

$$\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)})\right) \geq \frac{\varepsilon}{2}\Big)$$

$$\geq \mathrm{P}_{(X,Y)^n,(X',Y')^n}\Big(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X,Y)^n}(h^{(0)}) \geq \varepsilon \text{ and}$$

$$\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)}) \leq \frac{\varepsilon}{2}\Big)$$

$$= \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X,Y)^n}(h^{(0)}) \geq \varepsilon\right)\mathrm{P}_{(X',Y')^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)}) \leq \frac{\varepsilon}{2}\right).$$

Since $\mathbb{E}_{(X',Y')^n}\left[\overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)})\right] = \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)})$ we can apply Chebychev's inequality (Appendix A, Theorem A.1) to bound the right factor from below, precisely we show

$$\mathrm{P}_{(X',Y')^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)}) \leq \frac{\varepsilon}{2}\right) \geq \frac{1}{2}.$$

For this purpose, consider the complementary probability and find an upper bound less or equal to $\frac{1}{2}$.

$$\mathrm{P}_{(X',Y')^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)}) > \frac{\varepsilon}{2}\right)$$

$$\leq \mathrm{P}_{(X',Y')^n}\left(\left|\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h^{(0)}) - \overline{\mathcal{R}}_{(X',Y')^n}(h^{(0)})\right| > \frac{\varepsilon}{2}\right)$$

$$\leq 4 \frac{\mathrm{Var}_{(X',Y')^n}\left(\overline{\mathcal{R}}_{(X',Y')^n}\left(h^{(0)}\right)\right)}{\varepsilon^2} = 4 \frac{\mathrm{Var}_{(X',Y')^n}\left(n\,\overline{\mathcal{R}}_{(X',Y')^n}\left(h^{(0)}\right)\right)}{n^2\varepsilon^2}$$

$$= 4 \sum_{i=1}^{n} \frac{\mathrm{Var}_{(X_i',Y_i')}\left(\mathbb{1}\{h(X_i') \neq Y_i'\}\right)}{n^2\varepsilon^2} = 4 \frac{\mathrm{Var}_{(X_1',Y_1')}\left(\mathbb{1}\{h(X_1') \neq Y_1'\}\right)}{n\varepsilon^2}$$

$$\leq \frac{1}{2}$$

The last inequality follows a consequence of the assumption $\varepsilon \geq \sqrt{\frac{2}{n}}$ and that the variance can be bound by $\frac{1}{4}$. The latter is a simple application of Popoviciu's variance inequality for bounded random variables, see Appendix A Theorem A.3. Thus, we can conclude

$$\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right)$$

$$\geq \frac{1}{2}\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left(h^{(0)}\right) - \overline{\mathcal{R}}_{(X,Y)^n}\left(h^{(0)}\right) \geq \varepsilon\right).$$

The hypothesis $h^{(0)}$ was chosen arbitrarily. Hence, taking the supremum proves the first part of the claim.

The second part of the claim,

$$\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right) \leq \mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right),$$

is rather involved to prove due to choosing the hypothesis inside the argument of the probability distribution, i.e. dependent on the sample. This pointwise or rather samplewise choice makes it hard to obtain bounds making use of the finite effect space $\mathcal{H}_X$ and the growth function of the hypothesis class. Therefore, the remaining proof aims to rewrite the argument by using a trick called *symmetrization by random signs* to loosen the dependence between sampled data points and hypotheses and to obtain a uniform bound.
The idea of this second symmetrization method is considering one pair of each sample, collected in $\mathrm{S} = (X_i, Y_i, X_i', Y_i')$, and using the equality

$$\mathrm{P}_{\mathrm{S}}\left(k = \mathbb{1}\{h(X_i') \neq Y_i'\} - \mathbb{1}\{h(X_i) \neq Y_i\}\right) = \mathrm{P}_{\mathrm{S}}\left(k = \mathbb{1}\{h(X_i) \neq Y_i\} - \mathbb{1}\{h(X_i') \neq Y_i'\}\right)$$

that holds for each integer k due to the independence of the four random variables in S and the symmetry of the expression in the pairs (X_i, Y_i) and (X_i', Y_i'). This is useful because it allows for the artificial introduction of a nested probability distribution that can be bound independently of the sample realizations. It counterintuitively makes the expression more complex first to obtain a simple bound in the end.
For the purpose laid out above, define random signs $\sigma = (\sigma_1, \ldots, \sigma_n)$ with uniform

distributions on $\{-1,+1\}^n$. Using the additivity of expectation to manually embed the expectation over $\sigma = (\sigma_1, \ldots, \sigma_n)$ into Eq. (1.2) concludes the second symmetrization and enables finishing the proof.

$$
\begin{aligned}
&\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right) \geq \frac{\varepsilon}{2}\right) \\
&= \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\} \geq \frac{\varepsilon}{2}\right\}\right] \\
&\overset{(*)}{=} \frac{1}{2^n}\sum_{\sigma\in\{-1,1\}^n}\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right\}\right] \\
&= \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{E}_\sigma\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right\}\right]\right] \\
&= \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathrm{P}_\sigma\left(\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right)\right]
\end{aligned}
$$

While equality $(*)$ is no result of complex mathematical theory, it seems counterintuitive and might not be trivial to the reader. A detailed calculation of this step is presented in the Appendix for convenience. The interested reader might refer to Appendix B, Example B.1.
To finish the proof of this Lemma the application of Hoeffding's inequality (Appendix A, Theorem A.2) is necessary. Thus we have to rewrite our symmetrized equation accordingly.

$$
\begin{aligned}
&\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathrm{P}_\sigma\left(\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right)\right] \\
&= \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathrm{P}_\sigma\left(\max_{h\in\mathcal{H}_{(X,X')^n}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right)\right] \\
&= \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathrm{P}_\sigma\left(\bigcup_{h\in\mathcal{H}_{(X,X')^n}}\left\{\frac{1}{n}\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\right\}\right)\right] \\
&\leq \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\sum_{h\in\mathcal{H}_{(X,X')^n}}\mathrm{P}_\sigma\left(\sum_{i=1}^{n}\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{n\,\varepsilon}{2}\right)\right]
\end{aligned}
$$

The final expression of the previous calculation finally allows for eliminating the hypothesis dependence by bounding every addend independently of h using Hoeffding's inequality.
For fixed $(X_i, Y_i), (X_i', Y_i')$ the expression $\sigma_i\,\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big)$ is a

random variable with values in $\{-1, 0, 1\}$ and zero mean. Thus,

$$\mathrm{P}_\sigma\left(\sum_{i=1}^{n}\sigma_i\,\left(\mathbb{1}\{h(X_i')\neq Y_i'\}-\mathbb{1}\{h(X_i)\neq Y_i\}\right)\geq\frac{n\,\varepsilon}{2}\right)\overset{(Thm.A.2)}{\leq}\exp\left(-\frac{2n^2\varepsilon^2}{4\sum_{i=1}^{n}(1-(-1))}\right)$$
$$=\exp\left(-\frac{n\varepsilon^2}{4}\right).$$

Inserting this and using the bound on the cardinality of $\mathcal{H}_{(X,X')^n}$, described by the growth function, yields

$$\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\sum_{h\in\mathcal{H}_{(X,X')^n}}\mathrm{P}_\sigma\left(\sum_{i=1}^{n}\sigma_i\,\left(\mathbb{1}\{h(X_i')\neq Y_i'\}-\mathbb{1}\{h(X_i)\neq Y_i\}\right)\geq\frac{n\,\varepsilon}{2}\right)\right]$$

$$\leq\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right)\right]=\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right).$$

Putting both steps together one obtains as claimed

$$\sup_{h\in\mathcal{H}}\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h)-\overline{\mathcal{R}}_{(X,Y)^n}(h)\geq\varepsilon\right)$$

$$\leq 2\,\mathrm{P}_{(X,Y)^n,(X',Y')^n}\left(\sup_{h\in\mathcal{H}}\left(\overline{\mathcal{R}}_{(X',Y')^n}(h)-\overline{\mathcal{R}}_{(X,Y)^n}(h)\right)\geq\frac{\varepsilon}{2}\right)\leq 2\,\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right).$$

□

The result of Lemma 1.17 can be interpreted as follows. Considering a data distribution $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$, the likelihood of encountering samples that fall short of adequately estimating the true risk of any given hypothesis (with a margin of ε) can be constrained dependent on margin, sample size and growth function of the hypothesis class.
The value of the result now relies on calculating or bounding the growth function. Analytically calculating it is hard, thus a common bound credited to Sauer ([Sau72]) is used. Note that a similar bound is provided at roughly the same time in [VC71]. Sauer formulates the observation that the growth function grows polynomially instead of exponentially as soon as the VC complexity of the hypothesis class is exhausted by the number of samples. At such a ratio of sample size to hypothesis class complexity, a classifier cannot fit all possible data to label correspondences on any data set perfectly anymore and in general, has to balance misclassification. It has to choose the least erroneous hypothesis. It has to adapt to the character of the data and thus, should be less prone to overfitting.

Lemma 1.18 (Sauer's Lemma; [VC71, Theorem 1])**.**
Consider a hypothesis class $\mathcal{H}$ with VC dimension $\Delta_{\mathcal{H}}$ and let n be an arbitrary positive natural number representing the sample size. As long as the number of samples is less or equal to the complexity dimension, i.e. if $n \leq \Delta_{\mathcal{H}}$, the growth function is an exponential

function

$$\mathcal{G}_{\mathcal{H}}(n) = 2^n \leq 2^{\Delta_{\mathcal{H}}}.$$

However, if the number of samples exceeds the complexity, i.e. $n > \Delta_{\mathcal{H}}$ then $\mathcal{G}_{\mathcal{H}}$ can be bound polynomially[4].

$$\mathcal{G}_{\mathcal{H}}(n) \leq \left(\frac{ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}$$

Proof. We present an detailed version of N. Sauer's proofs published in [Sau72]. Note that Sauer's approaches have been extended to modernized and detailed proofs multiple times. Thus, the proof we present is obviously not unique and very similar to for example the formulation in [MRT18, Theorems. 3.17 and 3.18].
Prove both case distinctions separately

- The case "$n \leq \Delta_{\mathcal{H}}$" is trivial. By definition, $\mathcal{H}$ can shatter at least one sample of size $\Delta_{\mathcal{H}}$. Obviously, this means $\mathcal{H}$ can shatter any n sized subset of this sample as well, and thus $\mathcal{G}_{\mathcal{H}}(n) = 2^n$ holds.

- For the case "$n > \Delta_{\mathcal{H}}$" proceed by first showing $\mathcal{G}_{\mathcal{H}}(n) \leq \sum_{i=0}^{\Delta_{\mathcal{H}}} \binom{n}{i}$ for all n and all $\Delta_{\mathcal{H}}$ smaller n. This proof is done inductively. The base case, $n = 1$ and $\Delta_{\mathcal{H}} = 0$, is straightforward. Assume the induction claim is proven for $n-1$ and all $\Delta_{\mathcal{H}} < n-1$. For a set of data points $X = \{x_1, \ldots, x_n\}$ define

  $$\mathcal{H}^{(0)} \coloneqq \left\{h \in \mathcal{H} \mid h(x_n) = 0\right\} \text{ and } \mathcal{H}^{(1)} \coloneqq \left\{h \in \mathcal{H} \mid h(x_n) = 1\right\}.$$

  Apply the induction claim to $\mathcal{H}^{(0)}$:
  If X' in any set shattered by $\mathcal{H}^{(0)}$ then $x_n \notin X'$. However, since we assumed a Hypothesis space to be symmetric, i.e. $h \in \mathcal{H}$ implies $(1-h) \in \mathcal{H}$ (Assumption 1.12), it holds

  $$\left\{h \,\middle|\, h \in \mathcal{H}^{(0)} \text{ or } 1 - h \in \mathcal{H}^{(0)}\right\} \subset \mathcal{H}.$$

  Thus, $X' \cup \{x_n\}$ is shattered by $\mathcal{H}$ and further, $\Delta_{\mathcal{H}^{(0)}} \leq \Delta_{\mathcal{H}} - 1$ must hold. Considering this, the induction hypothesis yields

  $$\left|\mathcal{H}_X^{(0)}\right| = \left|\mathcal{H}_{X\setminus x_n}^{(0)}\right| \leq \sum_{i=0}^{\Delta_{\mathcal{H}}-1} \binom{n-1}{i}.$$

  The same argumentation holds for $\mathcal{H}_X^{(1)}$ and putting them together[5] yields

  $$\left|\mathcal{H}_X\right| = \left|\mathcal{H}_X^{(1)}\right| + \left|\mathcal{H}_X^{(0)}\right| \leq \sum_{i=0}^{\Delta_{\mathcal{H}}} \binom{n}{i}.$$

  Transitioning to $\sup_{X \in \mathcal{X}^n} |\mathcal{H}_X|$ concludes the induction step. To finalize the proof

[4] To avoid confusion, note that e is simply Euler's number, not a variable we forgot to define.

[5] Use $\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$ to obtain $2\sum_{i=0}^{k}\binom{n-1}{i} \leq \sum_{i=0}^{k+1}\binom{n-1}{i} + \sum_{i=0}^{k}\binom{n-1}{i} = \binom{n}{0} + \sum_{i=1}^{k+1}\binom{n-1}{i} + \binom{n-1}{i-1} = \sum_{i=0}^{k+1}\binom{n}{i}$.

it remains to bound $\sum_{i=0}^{\Delta_\mathcal{H}} \binom{n}{i}$. The claimed bound is an immediate consequence of $\binom{n}{i}\frac{1}{n^i} \leq \frac{1}{i!}$ and $\Delta_\mathcal{H} < n$:

$$\sum_{i=0}^{\Delta_\mathcal{H}} \binom{n}{i} \leq \left(\frac{n}{\Delta_\mathcal{H}}\right)^{\Delta_\mathcal{H}} \sum_{i=0}^{\Delta_\mathcal{H}} \binom{n}{i} \left(\frac{\Delta_\mathcal{H}}{n}\right)^{i} \leq \left(\frac{n}{\Delta_\mathcal{H}}\right)^{\Delta_\mathcal{H}} \sum_{i=0}^{\Delta_\mathcal{H}} \frac{1}{i!}\, \Delta_\mathcal{H}^i \leq \left(\frac{ne}{\Delta_\mathcal{H}}\right)^{\Delta_\mathcal{H}}$$

□

As already mentioned, the main work for our goal to investigate generalizability using Vapnik-Chervonenkis methods is done. Using symmetrization one can prove an expression of the form

$$\mathcal{R}(h) \leq \overline{\mathcal{R}}(h) + \varepsilon,$$

evaluating, dependent on ε, whether the empirical misclassification risk provides either an accurate or a pessimistic estimate of the true risk of the hypothesis or not. We allow for the generalization bound to be probabilistic but it should hold for a majority of possible sample realizations. The bound has to hold with certainty $1 - \delta$, i.e. any possible learned h maintains a generalization gap smaller or equal to ε with a maximal error tolerance δ concerning the data sampling to calculate the empirical risk.
The VC generalization gap ε_{VC} is defined in Definition 1.19. After this, we prove that it yields a probabilistic uniform generalization bound. In this bound, ε_{VC} quantifies the usefulness of VC theory for applications. For this reason, it will also be the subject of closer investigation in Section 2 later.
Note that the definition of ε_{VC} will slightly differ from source to source due to different proof details.

Definition 1.19 (Vapnik-Chervonenkis Gap)**.**
The VC generalization gap of a hypothesis from class $\mathcal{H}$, when considering a sample of size n is defined as

$$\varepsilon_{VC}(\Delta_\mathcal{H}, n, \delta) \coloneqq 2\sqrt{\frac{1}{n}\left(\Delta_\mathcal{H} \log \frac{2ne}{\Delta_\mathcal{H}} + \log \frac{2}{\delta}\right)}.$$

The third unknown, δ quantifies an error the probabilistic generalization bound cannot surpass.

Remark 1.20 (Symmetrization Condition on ε)**.**
Symmetrization requires $\varepsilon \geq \sqrt{\frac{2}{n}}$ (cf. Lemma 1.17). The Vapnik-Chervonenkis gap $\varepsilon_{VC}(\Delta_\mathcal{H}, n, \delta)$ fulfills this condition for all feasible $\mathcal{H}$, n and δ.

Proof. Obviously,

$$\varepsilon_{VC}(\Delta_\mathcal{H}, n, \delta) \geq \sqrt{\frac{2}{n}} \iff 4\left(\Delta_\mathcal{H} \log \frac{2ne}{\Delta_\mathcal{H}} + \log \frac{2}{\delta}\right) \geq 2.$$

The right expression is true as $4 \log \frac{2}{\delta} \geq 4 \log 2 \geq 2$ holds for all $\delta \in (0, 1]$. □

Now, we proceed by presenting the *Vapnik-Chervonenkis generalization bound* using the VC gap Definition 1.19. Afterward, Section 2 will discuss whether bounding the generalization error is always possible in a mathematical environment building on symmetrization as a foundation. This will also justify the choice of VC theory to investigate the proposal.

Theorem 1.21 (VC Generalization Bound; [Vap99])**.**
Let $\mathcal{H}$ be a hypothesis class with finite complexity $\Delta_{\mathcal{H}}$ and $(X,Y)^n$ be a finite i.i.d. sample, distributed according to $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$. Let $h \in \mathcal{H}$ be an arbitrary hypothesis and let $\delta \in (0,1]$.

- *If $2n > \Delta_{\mathcal{H}}$, then the generalization bound*

$$\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \leq \overline{\mathcal{R}}_{(X,Y)^n}(h) + \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta) \tag{1.4}$$

  holds with probability $1-\delta$ with respect to the sample.

- *In the overparameterized case $2n \leq \Delta_{\mathcal{H}}$, any uniform bound using symmetrization will be vacuous.*

Proof. Applying the assertion of Lemma 1.17 yields

$$\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right) \leq 2\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right)$$

for any hypothesis in $\mathcal{H}$.

- For $2n > \Delta_{\mathcal{H}}$ this is further bound according to Lemma 1.18 by

$$\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right) \leq 2\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}\exp\left(-\frac{n\varepsilon^2}{4}\right).$$

  Inserting ε_{VC} and simplifying the terms yields

$$\begin{aligned}
&2\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}\exp\left(-\frac{n\varepsilon_{VC}^2}{4}\right)\\
&= 2\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}\exp\left(-n\sqrt{\frac{1}{n}\left(\Delta_{\mathcal{H}}\log\frac{2ne}{\Delta_{\mathcal{H}}} + \log\frac{2}{\delta}\right)}^2\right)\\
&= 2\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}\exp\left(-\Delta_{\mathcal{H}}\log\frac{2ne}{\Delta_{\mathcal{H}}}\right)\exp\left(-\log\frac{2}{\delta}\right)\\
&= 2\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{\Delta_{\mathcal{H}}}\left(\frac{2ne}{\Delta_{\mathcal{H}}}\right)^{-\Delta_{\mathcal{H}}}\left(\frac{\delta}{2}\right) = \delta.
\end{aligned}$$

  This is equivalent to $\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \leq \overline{\mathcal{R}}_{(X,Y)^n}(h) + \varepsilon_{VC}\right) \geq 1-\delta$ and proves our bound.

- In the case $2n \leq \Delta_{\mathcal{H}}$, symmetrization yields a vacuous bound already. This can be seen by application of Sauer's Lemma (Lemma 1.18):

$$\begin{aligned} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right) &\leq 2\mathcal{G}_{\mathcal{H}}(2n)\exp\left(-\frac{n\varepsilon^2}{4}\right) \\ &= 2^{2n+1}\exp\left(-\frac{n\varepsilon^2}{4}\right) \overset{!}{=}: \delta \end{aligned} \tag{1.5}$$

  Note that decreasing ε increases δ and the other way around. Thus the bound is vacuous if for all gaps $\varepsilon < 1$ the error tolerance is $\delta \geq 1$, i.e. an error of 100% is allowed. With either a gap or tolerance larger than 1, the uniform bound $\mathrm{P}\left(\mathcal{R} \geq \overline{\mathcal{R}} + \varepsilon\right) \leq \delta$ yields a weaker statement than the definitions of risk and probability on their own, as both are by definition never larger than 1. We show that if $\varepsilon \leq 1$ then it must hold $\delta \geq 1$. Assume $\varepsilon \leq 1$. First, it holds

$$\log\left(2^{2n+1}\exp\left(-\frac{n}{4}\right)\right) = (2n+1)\log 2 - \frac{n}{4} = n(2\log 2 - \frac{1}{4}) + \log 2 \geq 0.$$

  Therefore,

$$2^{2n+1}\exp\left(-\frac{n\varepsilon^2}{4}\right) \geq 2^{2n+1}\exp\left(-\frac{n}{4}\right) \geq 1.$$

  This means, any bound $\mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon\right) \leq \delta$ based on symmetrization is vacuous if $2n \leq \Delta_{\mathcal{H}}$. □

Remark 1.22 (Do not Test on Training Data)**.**
The VC generalization bound is a probabilistic uniform bound

$$\sup_{h\in\mathcal{H}} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \leq \overline{\mathcal{R}}_{(X,Y)^n}(h) + \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta)\right) \geq 1 - \delta.$$

Hence, we bound the probability that the the empirical risk for any hypothesis h on any realization of the sample $(X,Y)^n$ underestimates the true risk of h significantly, i.e. with gap greater or ε, by δ.
This does not mean, that one can train on a training sample $(x,y)^n$ to find a classifier h minimizing the empirical risk $\overline{\mathcal{R}}_{(x,y)^n}(h)$ and then use this empirical risk to predict $\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h)$. Choosing h dependent on the sample makes the consideration biased towards lower empirical risk and thus, the training sample does not represent a randomly chosen sample or the whole distribution well. No matter how small we choose delta, the training data set is more likely to violate the bound than other samples.
This is a possible motivator for the common practice of testing trained classifiers on test samples which have not been seen during training, in order to evaluate their performance.

Note, that Theorem 1.21 is presented in the form common in literature, claiming that the true error cannot be far worse than the empirical estimate. It is not concerned

with hypotheses that expectedly perform far better on unseen data than on the chosen sample, i.e. we do not require

$$\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \geq \overline{\mathcal{R}}_{(X,Y)^n}(h) - \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta)$$

additionally to Eq. (1.4). However, the presented proofs also hold for a slightly stronger formulation considering the absolute error between true and empirical misclassification risk. All constants can actually be transferred to the absolute error case.

Corollary 1.23.
Let $\mathcal{H}$ be a hypothesis class with finite complexity $\Delta_{\mathcal{H}}$ and $(X,Y)^n$ be a finite i.i.d. sample distributed according to $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$.

- *If $2n \geq \Delta_{\mathcal{H}}$ then the generalization bound*

$$\left|\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h)\right| \leq \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta) \tag{1.6}$$

  holds for all δ with probability $1-\delta$ with respect to the sample.

- *If however $2n < \Delta_{\mathcal{H}}$, any uniform bounds using the previous symmetrization as in Lemma 1.17 will be vacuous.*

Proof. Since it adds no value to this thesis, a detailed proof is omitted here. Repeat the steps of the proofs of symmetrization, Sauer's Lemma, and finally VC-generalization. □

This concludes this thesis' theoretical insights into generalization for now. We proceed in Section 1.3 by extending the investigation of the Machine Learning part of Proposal 0.2 to Domain Adaptation problems. Afterwards, in Section 2 we discuss the practical value of the provided theory in the context of Neural Networks. As announced, this latter discussion includes an in-depth analysis of the Vapnik-Chervonenkis generalization gap ε_{VC}.

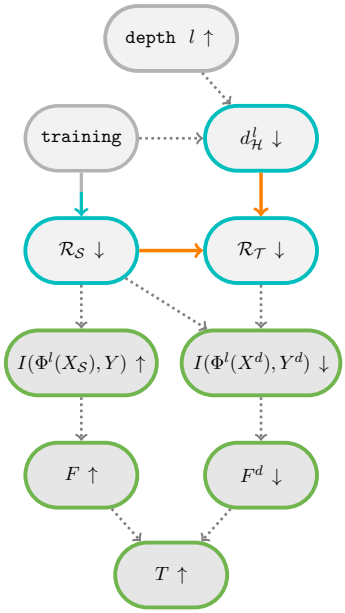
1.3. Domain Adaptation Bounds

We have already investigated a bound for the true risk using empirical estimates after observing samples following the true data distribution. Remember that we already introduced the case of Domain Adaptation in Example 1.9. We want to make statements about the true data risk from knowledge about a similar but different distribution compared to the true data distribution.

This means deriving statements about the impact of a domain shift. To quantify the performance changes after a domain shift we want to prove a bound of the form

$$\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}} \leq \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}} + \varepsilon$$

first. Again, this problem is highlighted in orange on the right. As proposed, this Domain Adaptation bound is dependent on a measure of separability of the two domains by $\mathcal{H}$, the $\mathcal{H}$-divergence $d_{\mathcal{H}}$ in the following defined in Definition 1.24. We omit empirical errors and for now primarily work on random variables. The relation between such empirical errors and the true risk is already thoroughly analyzed in Theorem 1.21.



Obviously, the possible error caused by domain shift, ε needs to cover some measure of similarity of the domains. A simple possibility is to capture the ability of the hypothesis space to label data points by the domain they originate from.
Intuitively speaking, if there are no hypotheses that can discriminate between the domains, then the classification error after domain shift should not change significantly. The definition was introduced by S. Ben-David, J. Blitzer et al.

Definition 1.24 ($\mathcal{H}$-DIVERGENCE; [BDBCP06])**.**
Consider a hypothesis class $\mathcal{H}$ and two domains $\mathcal{S} = (\mathcal{X}, \mathrm{P}_{\mathcal{S}})$ and $\mathcal{T} = (\mathcal{X}, \mathrm{P}_{\mathcal{T}})$. Define the **domain discrimination capability** of $h \in \mathcal{H}$ as the ability to fit different labels to data from different domains:

$$d_h(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) \coloneqq \Big| \mathrm{P}_{\mathcal{S}}\big(h(X) = 1\big) - \mathrm{P}_{\mathcal{T}}\big(h(X) = 1\big) \Big|$$

Generalizing d_h to the whole hypothesis space induces the definition of **$\mathcal{H}$-divergence** between $\mathcal{S}$ and $\mathcal{T}$ as

$$d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) \coloneqq \sup_{h \in \mathcal{H}} d_h(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}).$$

Let $x_{\mathcal{S}}^n$ and $x_{\mathcal{T}}^n$ be n-sized samples of realizations for both domains and collect them in the set X. Then, define the **empirical domain discrimination error** and the **empirical $\mathcal{H}$-divergence**.

$$\overline{d_h}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \coloneqq \frac{1}{n}\left(\sum_{i=1}^{n} \mathbb{1}\{h(x_{\mathcal{S},i}) = 1\} - \sum_{i=1}^{n} \mathbb{1}\{h(x_{\mathcal{T},i}) = 1\}\right)$$

$$\overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \coloneqq \max_{h \in \mathcal{H}_X} \overline{d_h}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \tag{1.7}$$

Using Definition 1.24, it is possible to bound the misclassification error after shifting from source to target domain using three easily graspable expressions:

- The risk of the given hypothesis h on the source domain.
- The shift between the domains measured by $d_{\mathcal{H}}$.
- The error of the optimal hypothesis across both domains, i.e. the ability of $\mathcal{H}$ to perform the classification on both domains at the same time with the same hypothesis h. This is dependent on the data distributions (as in the previous item) but also on the label distribution.

Below we provide a proof similar but slightly more detailed to the ones originally provided in [BDBCP06] and [BCK+08].

Theorem 1.25 (Domain Adaptation Bound; [BCK+08])**.**
Let $\mathcal{H}$ be the considered hypothesis class. Let C^* be the minimal combined misclassification risk across two domains $\mathcal{S}$ and $\mathcal{T}$.

$$C^* \coloneqq \inf_{h \in \mathcal{H}} \left(\mathcal{R}_{P_{\mathcal{T},\mathcal{Y}}}(h) + \mathcal{R}_{P_{\mathcal{S},\mathcal{Y}}}(h)\right)$$

Then, for all $h \in \mathcal{H}$ it holds

$$\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}(h) \leq \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) + 2d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) + C^* \tag{1.8}$$

Proof. Note that the true and predicted labels Y and $h(X)$ are both binary random variables and thus, for an arbitrary distribution P on $\mathcal{X} \times \mathcal{Y}$, it holds

$$\mathcal{R}_{\mathrm{P}}(h) = \mathbb{E}_{\mathrm{P}}\left[\mathbb{1}\{h(X) \neq Y\}\right] = \mathbb{E}_{\mathrm{P}}\left[\,\left|h(X) - Y\right|\,\right].$$

For any $\varepsilon > 0$, there exists a hypothesis $h_\varepsilon \in \mathcal{H}$ such that

$$\mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] + \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] \leq C^* + \varepsilon.$$

Therefore, one can conveniently use the triangle inequality and the additivity of expected values multiple times to obtain

$$
\begin{aligned}
&\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}(h) = \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - Y\right|\,\right] \\
&\leq \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] + \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] \\
&\leq \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] + \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] + \\
&\quad \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] - \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right]\right| \\
&\leq \mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] + \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h_\varepsilon(X) - Y\right|\,\right] + \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - Y\right|\,\right] + \\
&\quad \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] - \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right]\right| \\
&\leq C^* + \varepsilon + \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) + \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] - \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right]\right|.
\end{aligned}
$$

It remains to show that the absolute value on the right-hand side can be bound using $d_{\mathcal{H}}$. For readability, we write "$\iint_{\mathcal{X}\,\mathcal{Y}} \ldots \mathrm{d}y\mathrm{d}x$" instead of "$\int_{\mathcal{X}} \sum_{y\in\mathcal{Y}} \ldots \mathrm{d}x$", even though $\mathcal{Y}$ is finite.

$$
\begin{aligned}
&\left|\mathbb{E}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right] - \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}\left[\,\left|h(X) - h_\varepsilon(X)\right|\,\right]\right| \\
&= \left|\iint_{\mathcal{X}\,\mathcal{Y}} \left|h(x) - h_\varepsilon(x)\right| \Big(\mathrm{P}_{\mathcal{T},\mathcal{Y}}(x,y) - \mathrm{P}_{\mathcal{S},\mathcal{Y}}(x,y)\Big)\mathrm{d}y\mathrm{d}x\right| \\
&\leq \left|\iint_{\mathcal{X}\,\mathcal{Y}} h(x)\Big(\mathrm{P}_{\mathcal{T},\mathcal{Y}}(x,y) - \mathrm{P}_{\mathcal{S},\mathcal{Y}}(x,y)\Big)\mathrm{d}y\mathrm{d}x\right| + \\
&\quad \left|\iint_{\mathcal{X}\,\mathcal{Y}} h_\varepsilon(x)\Big(\mathrm{P}_{\mathcal{T},\mathcal{Y}}(x,y) - \mathrm{P}_{\mathcal{S},\mathcal{Y}}(x,y)\Big)\mathrm{d}y\mathrm{d}x\right| \\
&\leq 2\sup_{h\in\mathcal{H}}\left|\iint_{\mathcal{X}\,\mathcal{Y}} h(x)\Big(\mathrm{P}_{\mathcal{T},\mathcal{Y}}(x,y) - \mathrm{P}_{\mathcal{S},\mathcal{Y}}(x,y)\Big)\mathrm{d}y\mathrm{d}x\right| \\
&= 2\sup_{h\in\mathcal{H}}\Big|\mathrm{P}_{\mathcal{S}}\big(h(X)=1\big) - \mathrm{P}_{\mathcal{T}}\big(h(X)=1\big)\Big| \\
&= 2\, d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}})
\end{aligned}
$$

Putting the previous results together yields $\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}(h) \leq C^* + \varepsilon + \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) + 2d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}})$. Finally, considering ε approaching 0 concludes the proof. □

As the last investigation of this chapter, Corollary 1.27 will quantify the feasibility of the empirical $\mathcal{H}$-divergence as a predictor for $\mathcal{H}$-divergence. For this purpose, two equivalent definitions of empirical $\mathcal{H}$-divergence, also sometimes appearing in literature, can be useful.

Remark 1.26.
Consider samples $x_{\mathcal{S}}^n$ and $x_{\mathcal{T}}^n$ collected in the set X. For the empirical $\mathcal{H}$-divergence $\overline{d_{\mathcal{H}}}$ as in Eq. (1.7) it holds

$$\text{(a)} \quad \overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) = \frac{1}{n} \max_{h \in \mathcal{H}_X} \left| \sum_{i=1}^n h(x_{\mathcal{S},i}) - \sum_{i=1}^n h(x_{\mathcal{T},i}) \right| \geq 0 \tag{1.9}$$

$$\text{(b)} \quad \overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) = 1 - \frac{1}{n} \min_{h \in \mathcal{H}_X} \left(\sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 1\} + \sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{T},i}) = 0\} \right) \tag{1.10}$$

The main definition Eq. (1.7) and Eq. (1.9) aim at estimating the true $\mathcal{H}$-divergence intuitively by replacing expected values with empirical means. Equation (1.10) on the other hand displays $\overline{d_{\mathcal{H}}}$ as a measure of the power of $\mathcal{H}$ as domain classifier hypotheses by counting "misclassifications" on both domains.

Proof.

(a) Eq. (1.9) displays two changes. First, h is binary and thus $h(x)$ equals $\mathbb{1}\{h(x) = 1\}$ for all x. The second novelty is considering only absolute values i.e. claiming that $\overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \geq 0$. This non-negativity claim is an immediate consequence of Assumption 1.12. For each $h \in \mathcal{H}$ insertion of the complementary hypothesis $1 - h \in \mathcal{H}$ yields

$$\overline{d_h}(x_{\mathrm{P}_{\mathcal{S}}}^n, x_{\mathrm{P}_{\mathcal{T}}}^n) = -\overline{d_{1-h}}(x_{\mathrm{P}_{\mathcal{S}}}^n, x_{\mathrm{P}_{\mathcal{T}}}^n).$$

Thus the maximum never attains negative values and is equivalent to considering the absolute value only.

(b) Due to the already mentioned symmetry of the hypothesis class (Assumption 1.12) we can switch maximization with the negative of minimization which results in

$$\begin{aligned}
\overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) &= \frac{-1}{n} \min_{h \in \mathcal{H}_X} \left(\sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 1\} - \sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{T},i}) = 1\} \right) \\
&= \frac{-1}{n} \min_{h \in \mathcal{H}_X} \left(\sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 1\} - \left(n - \sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 0\} \right) \right) \\
&= 1 - \frac{1}{n} \min_{h \in \mathcal{H}_X} \left(\sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 1\} + \sum_{i=1}^n \mathbb{1}\{h(x_{\mathcal{S},i}) = 0\} \right)
\end{aligned}$$

□

Using these two equivalent definitions , we can prove a generalization result for the empirical $\mathcal{H}$-divergence. While similar results can be found for example in [BCK+08],

to the best of our knowledge, no detailed proof is shown in the literature.

Corollary 1.27 ($\mathcal{H}$-Divergence Generalization; [BCK+08])**.**
Let $X_{\mathcal{S}}^n$ and $X_{\mathcal{T}}^n$ be i.i.d. samples distributed according to $\mathrm{P}_{\mathcal{S}}$ and $\mathrm{P}_{\mathcal{T}}$ and let δ be a constant in $[0,1]$. Let $\mathcal{H}$ be the considered hypothesis space and define

$$\varepsilon \coloneqq 2\varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \frac{\delta}{2}) = 4\sqrt{\frac{1}{n}\left(\Delta_{\mathcal{H}} \log \frac{2ne}{\Delta_{\mathcal{H}}} + \log \frac{4}{\delta}\right)}.$$

If $\Delta_{\mathcal{H}} < 2n$ and if there exists a hypothesis h^* achieving the value of the supremum

$$d_{h^*}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) = d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) = \sup_{h \in \mathcal{H}} d_h(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}),$$

then for every $h \in \mathcal{H}$ it holds

$$\mathrm{P}_{X_{\mathcal{S}}^n, X_{\mathcal{T}}^n}\left(d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) \leq \overline{d_{\mathcal{H}}}(X_{\mathcal{S}}^n, X_{\mathcal{T}}^n) + \varepsilon\right) \geq 1 - \delta.$$

Proof. The proof boils down to rewriting the argument for the purpose of enabling the application of previous generalization results. This is mainly an issue of removing the different maxima in the definitions of empirical and true $\mathcal{H}$-divergence. By VC generalization (cf. Corollary 1.23) for any h it holds

$$\mathrm{P}_{X_{\mathcal{S}}^n}\left(\left|\mathbb{E}_{P_{\mathcal{S}}}(h(X)) - \frac{1}{n}\sum_{X_{\mathcal{S}}^n} h(x)\right| \geq \varepsilon_{VC}(n, \Delta_{\mathcal{H}}, \delta)\right) \leq \delta.$$

The same expression is true for the target domain as well. For an arbitrary set of sample realizations $x_{\mathcal{S}}^n$ and $x_{\mathcal{T}}^n$ collected in the set S it holds

$$\begin{aligned}
&d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) - \overline{d_{\mathcal{H}}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \\
&= \left|\mathrm{P}_{\mathcal{S}}\big(h^*(X) = 1\big) - \mathrm{P}_{\mathcal{T}}\big(h^*(X) = 1\big)\right| - \frac{1}{n}\max_{h \in \mathcal{H}_S}\left|\sum_{i=1}^n h(x_{\mathcal{S},i}) - \sum_{i=1}^n h(x_{\mathcal{T},i})\right| \\
&\leq \left|\mathrm{P}_{\mathcal{S}}\big(h^*(X) = 1\big) - \mathrm{P}_{\mathcal{T}}\big(h^*(X) = 1\big)\right| - \frac{1}{n}\left|\sum_{i=1}^n h^*(x_{\mathcal{S},i}) - \sum_{i=1}^n h^*(x_{\mathcal{T},i})\right| \\
&\leq \left|\mathrm{P}_{\mathcal{S}}\big(h^*(X) = 1\big) - \mathrm{P}_{\mathcal{T}}\big(h^*(X) = 1\big) - \frac{1}{n}\Big(\sum_{i=1}^n h^*(x_{\mathcal{S},i}) - \sum_{i=1}^n h^*(x_{\mathcal{T},i})\Big)\right| \\
&\leq \left|\mathrm{P}_{\mathcal{S}}\big(h^*(X) = 1\big) - \frac{1}{n}\sum_{i=1}^n h^*(x_{\mathcal{S},i})\right| + \left|\mathrm{P}_{\mathcal{T}}\big(h^*(X) = 1\big) - \frac{1}{n}\sum_{i=1}^n h^*(x_{\mathcal{T},i})\Big)\right| \\
&= \left|\mathbb{E}_{\mathrm{P}_{\mathcal{S}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^n h^*(x_{\mathcal{S},i})\right| + \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^n h^*(x_{\mathcal{T},i})\right|.
\end{aligned}$$

Therefore, for $\varepsilon = 2\varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \frac{\delta}{2})$ we obtain

$$
\begin{aligned}
&\mathrm{P}_{X_{\mathcal{S}}^n, X_{\mathcal{T}}^n}\left(d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) - \overline{d_{\mathcal{H}}}(X_{\mathcal{S}}^n, X_{\mathcal{T}}^n) \geq \varepsilon\right) \\
&\leq \mathrm{P}_{X_{\mathcal{S}}^n, X_{\mathcal{T}}^n}\left(\left|\mathbb{E}_{\mathrm{P}_{\mathcal{S}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{S},i})\right| + \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{T},i})\right| \geq \varepsilon\right) \\
&\leq \mathrm{P}_{X_{\mathcal{S}}^n, X_{\mathcal{T}}^n}\left(\left|\mathbb{E}_{\mathrm{P}_{\mathcal{S}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{S},i})\right| \geq \frac{\varepsilon}{2} \text{ or } \left|\mathbb{E}_{\mathrm{P}_{\mathcal{T}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{T},i})\right| \geq \frac{\varepsilon}{2}\right) \\
&\leq \mathrm{P}_{X_{\mathcal{S}}^n}\left(\left|\mathbb{E}_{\mathrm{P}_{\mathcal{S}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{S},i})\right| \geq \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \frac{\delta}{2})\right) + \\
&\quad \mathrm{P}_{X_{\mathcal{T}}^n}\left(\left|\mathbb{E}_{\mathrm{P}_{\mathcal{T}}}\left[h^*(X)\right] - \frac{1}{n}\sum_{i=1}^{n} h^*(X_{\mathcal{T},i})\right| \geq \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \frac{\delta}{2})\right) \\
&\leq \delta.
\end{aligned}
$$

□

For completeness, note that fusing our results Theorem 1.21, Theorem 1.25, and Corollary 1.27 yields the following definitive generalization bound.

Corollary 1.28.
Let $(X, Y)_{\mathcal{S}}^n$ and $X_{\mathcal{T}}^n$ be i.i.d. samples distributed according to $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$ and $\mathrm{P}_{\mathcal{T}}$ and let δ be a constant in $[0, 1]$. Let $\mathcal{H}$ be the considered hypothesis space and define

$$C^* \coloneqq \inf_{h \in \mathcal{H}} \left(\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}(h) + \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h)\right).$$

Further assume there exists $h^* \in \mathcal{H}$ such that $d_{\mathcal{H}}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) = d_{h^*}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}})$. Then for all $h \in \mathcal{H}$ it holds

$$\mathcal{R}_{\mathrm{P}_{\mathcal{T},\mathcal{Y}}}(h) \leq \overline{\mathcal{R}}_{(X,Y)_{\mathcal{S}}^n}(h) + 2\,\overline{d_{\mathcal{H}}}\,(X_{\mathcal{S}}^n, X_{\mathcal{T}}^n) + C^* + \varepsilon_{VC}\left(\Delta_{\mathcal{H}}, n, \sqrt{\delta}\right) + 4\,\varepsilon_{VC}\left(\Delta_{\mathcal{H}}, n, \frac{\sqrt{\delta}}{2}\right)$$

with probability at least $1 - \delta$ as long as $2n \leq \Delta_{\mathcal{H}}$.

Proof. Apply Theorem 1.21 and Corollary 1.27 to Theorem 1.25. □

Corollary 1.28 finishes our theory section regarding Domain Adaptation and Machine Learning problems. Two key directions of Proposal 0.2 have been proven in Theorem 1.21 [Proposal 0.2, Claim (a) (i)] and Theorem 1.25 [Proposal 0.2, Claim (a) (iii)]. After investigating this theory, further questions regarding the application of the results may arise.

- For the results we require $2n > \Delta_{\mathcal{H}}$. Is this a strong limitation? How large is $\Delta_{\mathcal{H}}$ for common classifiers, especially for deep Neural Networks?
- We fit the generalization gap $\varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta)$ to the needs of our theory. Inside our framework, it is barely improvable. For which problem setups, i.e. $\Delta_{\mathcal{H}}$, n and δ, is it conveying meaningful generalization bounds? When do the bounds get vacuous?
- Can we derive statements regarding the size of $d_{\mathcal{H}}$ and thereby Proposal 0.2 claim (a) (ii)?

The first two key points will be subject of the upcoming Section 2. The third key point will be discussed empirically in the course of experiments in Section 4.

2. Neural Networks and Disproportionate Uniform Bounds

"[...] a kind of haunting doubt came over me. [...] in theory there is no difference between theory and practice, while in practise there is."
- Benjamin Brewster [Bre82]

With the theory established in Section 1 it is a reasonable next step to investigate whether Vapnik-Chervonenkis generalization is applicable in practice or remains a primarily theoretical approach. As already hinted at, for Neural Network applications the results can be expressionless. This section is dedicated to illustrating the problems directly linked to the properties of Theorem 1.21 and other uniform bounds.

2.1. Investigation of the VC Generalization Gap

Uniform generalization bounds, i.e. upper bounds on the estimated misclassification error of a hypothesis for unseen data after observing any accordingly distributed sample, do not yield meaningful results when the VC complexity of the underlying hypothesis space is very high. This can easily be explained when thinking about the common problem of overfitting in Machine Learning. If the number of training points n is small compared to the VC dimension $\Delta_{\mathcal{H}}$, in the worst case a hypothesis is found during training that easily labels all points of a bad sampled points correctly. This is due to the definition of the VC dimension/complexity (Definition 1.13). There exists at least one sample of size $n \leq \Delta_{\mathcal{H}}$ on which labels can be predicted perfectly by memorization, rather than learning meaningful inference. In the worst case, this does not only hold for one certain but many different samples. A uniform bound has to cover such worst-case examples and thus the generalization gap has to be large. We start by providing a feeling for the size of the VC dimension for different classifiers and sample sizes. Then, we show that complexity is large for Neural Networks and how strong they are regarding measurements other than VC dimension.

Dependence on Sample Size and Complexity

Remember that

$$\varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta) = 2\sqrt{\frac{1}{n}\left(\Delta_{\mathcal{H}} \log \frac{2ne}{\Delta_{\mathcal{H}}} + \log \frac{2}{\delta}\right)}$$

is the VC gap as long as $2n > \Delta_{\mathcal{H}}$. We will consider an error tolerance of $\delta = 0.1$, i.e. up to 10% of data realizations do not need to fulfill VC generalization bound.
In this case, we are left with the impact quantities of the number of samples n and the complexity of the classifier $\Delta_{\mathcal{H}}$.
For a qualitative analysis, we can look at the left and right expressions inside the root separately. Let

$$T_1 + T_2 \coloneqq \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta)^2 = \left(\frac{4}{n}\Delta_{\mathcal{H}} \log \frac{2ne}{\Delta_{\mathcal{H}}}\right) + \left(\frac{4}{n} \log \frac{2}{\delta}\right).$$

Assume $\Delta_{\mathcal{H}}$ is a fixed complexity and define the ratio of samples to complexity as $r \coloneqq \frac{n}{\Delta_{\mathcal{H}}}$.

- The left expression
$$T_1 \coloneqq \frac{4}{r} \log 2er.$$
can be decreased by increasing r, i.e. when considering more samples or less complex classifiers. Some important intermediate values of T_1 are visualized in Table 1. These indicate that in order to obtain bulletproof guarantees for classifiers with high complexity, a very large number of samples is needed to bound the first component of ε_{VC}.

Table 1: Examples for n-to-$\Delta_{\mathcal{H}}$ ratios r achieving T_1 values close to 1, 0.5, 0.25, 0.125, 0.05 and 0.01.

r	18	20	45	100	250	650	4000
T_1	1.02	0.94	0.49	0.25	0.12	0.05	0.01

- The right expression
$$T_2 \coloneqq \frac{4}{n} \log \frac{2}{\delta}$$
can be estimated by
$$\frac{11.98}{n} \leq T_2 \leq \frac{12}{n}$$
after inserting $\delta = 0.1$ This indicates that even independent of the complexity, a decently large sample number is desirable.

Putting T_1 and T_2 together again justifies the intuition, that an empirical mean estimator like our empirical misclassification risk is only precise if the classifier is trained on many samples. Further, the performance of more powerful or complex classifiers is increasingly harder to estimate. A numerical example with a decently large sample size is provided below.

Example 2.1.
Consider the case of $n = 100\,000$ which describes a very large yet for many benchmark problems realistic sample size. Obviously, at this sample size, T_2 is negligible and ε_{VC} is roughly equal to $\sqrt{T_1}$.
The impact of rather small values of $\Delta_{\mathcal{H}}$ is illustrated in Fig. 5. Thus as indicated above, with $\Delta_{\mathcal{H}} = 1000$ or equivalently $r = 100$ the gap is already at 0.5. In this case, the VC bound indicates that however small our training error might be, the true risk cannot be predicted or bound better than just flipping a coin.

To evaluate whether Vapnik-Chervonenkis bounds are practically applicable in modern classifiers or if they are too weak, we should investigate different hypothesis classes with respect to their VC dimension. Applicability is mainly about complexity since any practical application shows eagerness towards using as many training samples as possible anyway. Our main goal will be to gain knowledge about Neural Networks.

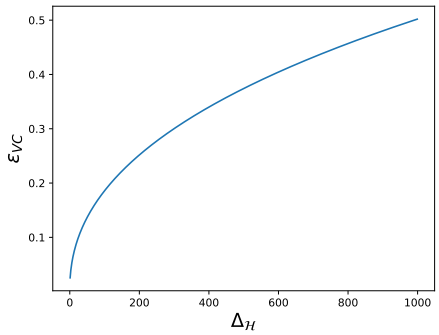


Figure 5: Values of the VC generalization gap for sample size $n = 100\,000$ and error tolerance $\delta = 0.1$.

2.2. Complexity of Different Classifiers

The previous discussion investigated what the ratio of samples to complexity should be to obtain a valid VC bound. Now, we investigate the complexity $\Delta_{\mathcal{H}}$ of different classifiers to get a feeling for its size.

2.2.1. Simple Classifiers

Finite $\mathcal{H}$. Finiteness of the hypothesis class yields a bound on the VC dimension, as shattering a set of size n requires at least 2^n different hypotheses. Thus it holds

$$\Delta_{\mathcal{H}} \leq \lceil \log_2 |\mathcal{H}| \rceil$$

where $\lceil \cdot \rceil$ denotes rounding up to the nearest integer.

Linear Classifiers. Linear (hyperplane) classifiers assume data consist of two classes that can be separated by affine linear hyperplanes $\{x \mid w^t x + b = 0\}$. For these linear classifiers in $\mathbb{R}^m$, it is easy to see that

$$\Delta_{\mathcal{H}} = m + 1.$$

A proof from [Lau14] is presented in the appendix (Appendix B, Theorem B.2).

Binary Tree Classifiers. Random forest classifiers, based on decision tree learning are claimed to be among the best-performing classifiers (cf. [FDCBA14]). Yet decision trees are quite simple to investigate as each node is nothing but linear discrimination with respect to only one variable of the data.
It is straightforward that the VC dimension of a single node performing a binary decision

is smaller than the complexity of linear classifiers. Yıldız proves in [Yil15, Theorem 1.] that for m-dimensional data a single decision node has VC dimension

$$\Delta_{\mathcal{H}} = \lfloor \log_2(m+1) + 1 \rfloor$$

where $\lfloor \cdot \rfloor$ indicates rounding down to the nearest integer. This can be generalized for example to full trees of height l. For this purpose, [Yil15, Corollary 2.] proves

$$\Delta_{\mathcal{H}} = 2^{l-1} \left(\lfloor \log_2(m-l+2) \rfloor + 1 \right).$$

Sine Classifiers. The typical example showing that sometimes the maximal number of points shattered have to be positioned very specifically and at the same time that the VC dimension does not need to be finite can be found in [MRT18, Example 3.16.]. This considers sine graphs as separating subspace. As already mentioned in Remark 1.15 (c) for

$$\mathcal{H} = \{h \mid \exists w \in \mathbb{R}^m \colon\ h(x) = \mathbb{1}_{\mathbb{R}_{\geq 0}}(\sin w^t x)\}$$

it holds

$$\Delta_{\mathcal{H}} = \infty$$

in $\mathbb{R}^m$. This result also implies that in comparison to trees, one does not need more parameters for more complex classifiers, but the very own character of the functions in the process of classification is important. Later in Section 2.3 it is implied that the complexity of Neural Networks can be influenced by choosing more or less simple activation functions.

Proof Idea. The proof is not constructive. The points to be shattered have to be arranged specifically, instead of using more general 0-1-vectors in the previous examples. For any $n \in \mathbb{N}$, the set $X_n = \{(2^{-m}, 0, \ldots, 0)^t \mid m \in \mathbb{N},\ m \leq n\}$ can be shattered by sine functions along the x_1-axis with certain frequencies. Thus, $\Delta_{\mathcal{H}} \geq n$ for all $n \in \mathbb{N}$ which implies, that the VC dimension is infinite. □

Putting these investigations together, one could expect Neural Networks might have very large complexity as they are composites of high dimensional (affine) linear functions and often highly non-linear activation functions.

2.2.2. Deep Classifiers

The core classifier class of this thesis's proposal is Neural Networks. Let us start by defining them, investigating their properties, and especially investigating their Vapnik-Chervonenkis dimension.

Definition 2.2 (Fully Connected Neural Networks; [BGKP22])**.**
A **(Fully Connected) Neural Network** (FCNN) is a function $\Phi_a(\cdot\,;\theta) \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ defined in the following way:

(a) For $L \in \mathbb{N}$, $N \coloneqq (N_0, \dots, N_L) \in \mathbb{N}^{L+1}$ and a map $\sigma \colon \mathbb{R} \to \mathbb{R}$ define a **Neural Network architecture** as

$$a \coloneqq (N, \sigma).$$

(b) The **parametrization** of the network with given architecture is denoted as

$$\theta \coloneqq (W_0, b_0, W_1, b_1, \dots, W_{L-1}, b_{L-1})$$

where $W_i \in \mathbb{R}^{N_{i+1} \times N_i}$ and $b_i \in \mathbb{R}^{N_{i+1}}$. The total number of parameters is denoted as W and the space of all feasible parameterizations is denoted as Θ.

(c) A **realization** of a Neural Network is a composition of affine linear functions from $\mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ for $i \in \{0, \dots, L-1\}$ given by the parametrization, alternated with the component-wise application[6] of the activation function:

$$\Phi_a(x; \theta) \coloneqq W_{L-1}\sigma\left(\dots\left(W_1\sigma\left(W_0 x + b_0\right) + b_1\right)\dots\right) + b_{L-1}$$

(d) Regarding Neural Networks a hypothesis class is the set of all possible realizations given a fixed architecture $\mathcal{H}_a = \{\Phi_a(\,\cdot\,; \theta) \colon \theta \in \Theta\}$.

This means that typically there is not one hypothesis class considering all possible Neural Networks but one for all Neural Networks sharing the same architecture.

Notation 2.3.
Consider the parameters in Definition 2.2. L is the number of layers of the network after the input and is called the network **depth**. N_i denotes the **width** of the i-th layer, and σ is called **activation function**.
Note that the choice of architecture is almost completely up to the user. The depth of the network and the widths $N_1, \dots, N_{L-1}$ of the so-called **hidden layers** can be chosen freely. Only the input and output dimensions, N_0 and N_L, are a priori specified by the problem-specific data and labels. In this thesis, we consider the case of $N_0 = m$ and $N_L = 1$, i.e. fitting to the defined data spaces $\mathcal{X}$ and $\mathcal{Y}$. To achieve classification, the network classifier induces binary labels by thresholding the output of the Neural Network.

Example 2.4 (CNNs are a Special Kind of FCNN)**.**
Some readers familiar with the topic might want to insist, that modern Neural Networks consist of more than just linear layers. One of the most common layer types used in almost all modern Neural Networks with image data as inputs are convolutional layers. In these layers, a convolutional kernel k that is smaller than the input is shifted over patches of the input. The convolution (or often only the dot product) between the patch and the kernel is calculated as one entry of the output.
We will use a small illustrative example to show that this operation is linear. Consider

[6] For $\mathbf{x} = (x_1, \dots, x_n)^t \in \mathbb{R}^n$ component-wise application means $\sigma(\mathbf{x}) \coloneqq (\sigma(x_1), \dots, \sigma(x_n))^t$

a 2×2 convolutional kernel k and a 3×3 input x. In this case, the convolutional layer calculates the following:

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix} \xmapsto[\text{kernel } k]{\text{conv. layer}} \begin{pmatrix} \begin{pmatrix} x_1 & x_2 \\ x_4 & x_5 \end{pmatrix} * k & \begin{pmatrix} x_2 & x_3 \\ x_5 & x_6 \end{pmatrix} * k \\ \begin{pmatrix} x_4 & x_5 \\ x_7 & x_8 \end{pmatrix} * k & \begin{pmatrix} x_5 & x_6 \\ x_8 & x_9 \end{pmatrix} * k \end{pmatrix}$$

$$= \begin{pmatrix} x_1k_4 + x_2k_3 + x_4k_2 + x_5k_1 & x_2k_4 + x_3k_3 + x_5k_2 + x_6k_1 \\ x_4k_4 + x_5k_3 + x_7k_2 + x_8k_1 & x_5k_4 + x_6k_3 + x_8k_2 + x_9k_1 \end{pmatrix}$$

Using the bijections $\mathbb{R}^{3\times3} \cong \mathbb{R}^9$ and $\mathbb{R}^{2\times2} \cong \mathbb{R}^4$, it is easy to see that this convolutional layer performs simply a linear operation.

$$\begin{pmatrix} k_4 & k_3 & 0 & k_2 & k_1 & 0 & 0 & 0 & 0 \\ 0 & k_4 & k_3 & 0 & k_2 & k_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_4 & k_3 & 0 & k_2 & k_1 & 0 \\ 0 & 0 & 0 & 0 & k_4 & k_3 & 0 & k_2 & k_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \end{pmatrix}$$

Thus mathematically speaking, a typical convolutional layer is just a special type of sparse linear layer with matrix input and output and few parameters or weights to train.

In the course of this thesis, we often do not care about the output of the whole network but only intermediate outputs up to certain layers. This is called deep representations of the data:

Definition 2.5 (Representation)**.**
We define the **representation function** of the l-th layer as

$$\Phi_a^l(x;\theta) = \sigma\left(W_{l-1}\sigma\left(\dots\left(W_1\sigma\left(W_0x + b_0\right) + b_1\right)\dots\right) + b_{l-1}\right)$$

for l in $\{1, \dots, L-1\}$.

Investigating the VC dimension of Neural Networks is not straightforward. It is hard to quantify the complexity precisely. Ideas for lower bounds are examined by finding examples using simple constructions. Bartlett et al. (cf. [BHLM17]), based on techniques from [BMM98], construct an elementary Neural Network that shatters a set of cardinality $m_1\, m_2$ consisting of simple 0-1-vectors in $\mathbb{R}^{m_1+m_2}$. Using this, they conclude that the VC dimension dependent on the number of parameters W and depth L an architecture can be bounded from below by $\Delta_{\mathcal{H}} \geq WL \log \frac{W}{L}$.

Lemma 2.6 (Neural Network Shattering; [BHLM17, Theorem 9])**.**
Let m_1, m_2, r be positive natural numbers and set $k := \lceil \frac{m_1}{r} \rceil$. Then one can construct a Neural Network architecture with the following properties.

- *Depth* $= 3 + 5k$
- *Input size* $= m_1 + m_2$,
- *Number of parameters* $= 2 + m_2 + 4m_1 + k\left((11 + r)2^r + 2r + 2\right)$
- *VC-dimension at least* $m_1\, m_2$

Proof Idea. The proof revolves around carrying out so-called "bit-extraction" techniques (see [BMM98]) using the operations of Neural Network layers. The idea is to consider the set

$$X = \left\{ \left(u^t, v^t\right)^t \;\middle|\; u \in \{0,1\}^{m_1},\, \|u\|_1 = 1,\, v \in \{0,1\}^{m_2},\, \|v\|_1 = 1 \right\} \subset \mathbb{R}^{m_1+m_2}$$

of concatenated unit vectors of lengths m_1 and m_2. The cardinality is $|X| = m_1 m_2$. For any labels on X, encoded as a binary matrix A of size $m_1 \times m_2$, Bartlett et al. parameterize a network that is constructed as follows.

- Given $(u, v) = (e_i, e_j)$, the first layer calculates a bit-representation of $A_{i,:}$ corresponding to $1 + m_1$ weights.
- Then in k blocks of 5 layers and $(11 + r)2^r + 2r + 2$ weights each, sequentially extract the elements $A_{i,1}$ to A_{i,m_2}.
- Finally, the network selects the one certain entry $A_{i,j}$ using $1 + 4m_2$ more weights.

The technical details on performing these tasks using Neural Network operations can be looked up in [BHLM17]. □

As mentioned, the authors apply this result to obtain a lower bound of the VC dimension lower in terms of the number of weights instead of input size.

Corollary 2.7 (Main Lower VC Dimension Bound; [BHLM17, Theorem 3])**.** *There exists a constant C such that for any W, L in $\mathbb{N}_{>0}$ with $W > CL > C^2$ there is a network architecture with*

- *depth at most L,*
- *number of parameters at most W and*
- *ReLU[7] activation function*

such that the architecture has a VC dimension of at least $\frac{WL}{C}\log(\frac{W}{L})$.

Proof. The proof boils down to parameterizing the network constructed in Lemma 2.6 dependent on L and W such that the claim is fulfilled. For this purpose, assume that C has a sufficiently large value and choose

$$r \coloneqq \frac{1}{2}\log\frac{W}{L},\; m_1 \coloneqq r\left(\frac{L-7}{5}\right) \text{ and } \; m_2 \coloneqq W - 5m_1 2^r.$$

[7] First introduced by K. Fukushima in [Fuk69]

The remaining proof consists of inserting these into the construction from Lemma 2.6. For the sake of brevity, the exact calculations are shifted to the Appendix to Corollary B.3. □

The results of Lemma 2.6 and Corollary 2.7 demonstrate the power of Neural Networks in terms of VC complexity using simple ideas and constructions. This also draws attention to an overarching problem regarding the architectures of state-of-the-art Neural Networks.

Modern Neural Networks typically feature a substantial surplus of parameters relative to the available training samples.

Given the previous results this suggests, that the VC dimension of state-of-the-art Neural Networks might also be far bigger than typical sample sizes. This means in the worst case, Neural Networks are capable of fitting samples for any application easily by memorization. However, the results of Lemma 2.6 and Corollary 2.7 are not formulated for arbitrary architectures and thus, estimating the VC dimension of an arbitrary Neural Network might not be feasible. For example, the constructions show a dependence of the input size on the other architecture-defining parameters L and W.
Nevertheless, combined with the overwhelming range of examples in literature, attaining perfect classification on a multitude of different data sets, we can assume that the VC dimension of deep Neural Networks is inconceivably huge and, considering the analysis of the VC generalization gap at the beginning of this chapter, we need an impractically enormous number of training samples to achieve a non-vacuous bound ε_{VC} considering Vapnik-Chervonenkis generalization (Theorem 1.21).
However, large VC complexity is not only a problem of bounding Neural Networks generalization using Theorem 1.21 but of any bound of the form

$$\sup_{h\in\mathcal{H}} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}}(h) - \overline{\mathcal{R}}(h) \geq \varepsilon\right) \leq \delta. \tag{2.1}$$

Remember the motivation for VC generalization: With a large VC dimension, the hypothesis class has much capacity to fit labels and therefore also a capacity to make generalization errors. The classifier's behavior is too unpredictable. This is an issue for other uniform bounds as well. Even if a bound is far tighter, it would still need to regard the hypothesis class' large capacity and the unpredictability of generalization.
A simple example of this issue is fitting labels that are not dependent on the data, i.e. label ill-defined classification data. Since we suspect Neural Networks to have very large VC complexity and might be able to shatter or overfit to many samples, this will present a general problem when bounding their generalization performance.

Example 2.8 (Worst Case Example)**.**
If the complexity $\Delta_{\mathcal{H}}$ is very large compared to any reasonable sample size, classification machines might in the worst case be strong enough to overfit to any labeled training sample. This means they can memorize the labels without, colloquially speaking, learning about the distribution.

Consider a labeling independent of data, e.g.

$$\mathrm{P}_{Y|X}(Y = 0|X) = \mathrm{P}_Y(Y = 0) = \frac{1}{2}.$$

Then, the risk for any classifier h is 50%[8]. Assume we learn from the sample $(x, y)^n$ and assume $\Delta_{\mathcal{H}} \gg n$, then we can learn $h^{(0)} = \mathcal{A}\left((x, y)^n\right) \in \mathcal{H}$ for some learning algorithm $\mathcal{A}$ (cf. Definition 1.11) such that the empirical risk of $h^{(0)}$ on $(x, y)^n$ is approximately zero. Thus, for this sample realization, the gap between the empirical and true risk of $h^{(0)}$ is $\frac{1}{2}$.
If we consider a deterministic uniform generalization bound, i.e. $\mathcal{R}_{\mathrm{P}}(h) - \overline{\mathcal{R}}(h) \geq \varepsilon$ has to be satisfied on all possible sample realizations and for all hypotheses in $\mathcal{H}$, then choosing $(x, y)^n$ and $h^{(0)}$ as constructed above yields $\varepsilon \geq \frac{1}{2}$. Any deterministic bound has to be vacuous for such powerful classifiers.
Thus, if very complex classifiers can overfit arbitrary samples, especially those that might not depict the true distribution risk well, then generalization bounds might face some issues. Regarding probabilistic bounds, we have seen that Theorem 1.21 is an expressionless descriptor of generalization if $\Delta_{\mathcal{H}}$ and consequently ε_{VC} are very large.
Different bounds might be tighter but will still have the same problem when the VC dimension is high. They have to take into consideration that very complex classifiers might achieve zero loss on many even ill-defined classification samples. If different gaps are not dependent on Δ_{VC} but on different quantities that explain classification power, all uniform bounds might be vacuous because they need to regard immense overfitting possibilities.
Probabilistic uniform bounds are meant to measure general generalization performance and ignore strange behavior on outlier samples. However, since very complex hypothesis classes have the capacity to perform well on any single sample one could claim that there is no general behavior of very complex classifiers. Since the hypothesis class has so much power, every sample can be a bad predictor for the general distribution.

2.3. More Insights on the Immense Power of Neural Networks

We have seen that the VC complexity of Neural Networks might be very large and thus the potential to overfit is high. However, the VC dimension evaluates hypothesis classes on their ability to shatter one sample. This includes technical constructions like the already discussed sine classifiers. The theory then establishes bounds and statements on the worst-case scenario, that all not just single samples of given size can be shattered. This subsection is dedicated to showing that Neural Networks not only have high VC dimension but also behave closely to this assumed worst-case overfitting abilities. For this purpose, we present several more results from literature and mathematical theory, apart from the size of the VC dimension, suggesting that deep Neural Networks may be too powerful of algorithms to be investigated within the scope of classical results and

[8]For every h the risk is calculated by independence as $\mathrm{P}_{X,Y}(h(X) = 1, Y = 0) + \mathrm{P}_{X,Y}(h(X) = 0, Y = 1) = \frac{1}{2}\left(\mathrm{P}_X(h(X) = 0) + \mathrm{P}_X(h(X) = 1)\right) = \frac{1}{2}$.

uniform bounds.
The most common results regard Neural Networks approximating continuous functions. This deviates slightly from the classification focus of this thesis but still continues to provide evidence that Neural Networks are incomprehensibly powerful structures. Further, as both investigate similar problems within different spaces and with different losses, regression and classification are closely related.

A Brief Thematic Shift to Regression Problems This paragraph investigates so-called universal approximation theorems which show that *wide but not very deep* networks can already estimate any continuous function on compact support without any strict assumptions. Afterward, a result indicating that *deeper* Neural Networks might be able to do the same while being more parameter-efficient, is provided. Detailed proofs would be out of the scope of this thesis, but we shortly present some proof ideas that are unique and interesting throughout this section.
The first result we discuss abuses the high non-linearity of so-called sigmoids to prove such a result.

Theorem 2.9 (Universal Approximation Theorem I; [Cyb89])**.**
Let the activation function $\sigma \in C(\mathbb{R};\mathbb{R})$ be a continuous sigmoid, i.e. a continuous, bounded, measurable function with

$$\lim_{t\to\infty} \sigma(t) = 1 \quad \textit{and} \quad \lim_{t\to-\infty} \sigma(t) = 0.$$

Let the data space be $\mathcal{X} = [0,1]^n$ in $\mathbb{R}^n$ for some $n \in \mathbb{N}_{>0}$. Then, the set of all 2-layer Neural Networks with architectures $\{a \coloneqq (n, N_1, 1, \sigma) \,|\, N_1 \in \mathbb{N}_{>0}\}$ is dense in $C([0,1]^n;\ \mathbb{R})$. This means for all $f \in C([0,1]^n;\mathbb{R})$ and $\varepsilon > 0$ there exists $N_1 \in \mathbb{N}$ and a network $\Phi_a(\,\cdot\,;\theta)$ approximating f such that

$$\sup_{x\in[0,1]^n} \big|f(x) - \Phi_a(x\,;\theta)\big| < \varepsilon.$$

Proof Idea. The proof is done by contradiction. We will skip the details and only summarize the major steps the author, George Cybenko, does. Refer to the original paper for complete proof. Assume that the set

$$\mathcal{F} \coloneqq \{\Phi_a(\,\cdot\,;\theta) \,|\, N_1 \in \mathbb{N}_{>0},\, a = (n, N_1, 1, \sigma),\, \theta \in \Theta\} = \operatorname{span}\{\sigma(\langle w, \cdot\rangle + b) \,|\, w \in \mathbb{R}^n,\, b \in \mathbb{R}\}$$

of 2-layer Neural Networks with arbitrary width values for N_1 is no dense subspace of $C([0,1]^n;\ \mathbb{R})$. The contradiction is found by doing the following three steps.

(a) According to the separation theorem by Hahn-Banach ([Alt12, Theorem 4.16]) there exists a linear functional $F \neq 0$ on $C([0,1]^n;\mathbb{R})$ such that $F|_{\mathcal{F}} \equiv 0$.

(b) The Riesz-Markov-Kakutani representation theorem implies that F can be represented as

$$F\colon C([0,1]^n;\mathbb{R}) \to \mathbb{R},\ h \mapsto \int_{[0,1]^n} h(x)\ \mathrm{d}\mu(x).$$

Notably, this holds for continuous functions of the form $h = \sigma(\langle w, \cdot \rangle + b) \in \mathcal{F}$.

(c) Lastly, one observes, that sigmoid functions are always *discriminatory*. This is defined as the property

$$\forall \mu' : \int_{[0,1]^n} \sigma(w^t\, x + b)\ \mathrm{d}\mu'(x) = 0 \quad \forall\, w \in \mathbb{R}^n, \forall\, b \in \mathbb{R} \implies \mu' = 0.$$

To conclude, (a) and (b) together imply that for the measure μ defining F it holds $\int_{[0,1]^n} \sigma(w^t\, x + b)\ \mathrm{d}\mu(x) = 0$ for all weights w and biases b. Thus, (c) implies $\mu = 0$ which obviously contradicts $F \neq 0$. □

Another result of universal approximation properties is presented below. In contrast to Theorem 2.9 it does not abuse the discriminatory property of sigmoids and thus displays a more general claim.

Theorem 2.10 (Universal Approximation Theorem II; [LLPS93])**.**
Let $\sigma \in C(\mathbb{R}; \mathbb{R})$ be a continuous activation function. Then ***exactly one*** *of the following is true.*

Either ***a)*** *σ is polynomial*

or ***b)*** *for all m and n in $\mathbb{N}$, $K \subset \mathbb{R}^n$ compact, $f \in C(K; \mathbb{R}^m)$ and $\varepsilon > 0$*
there exists $N_1 \in \mathbb{N}$ and a network with architecture $a = (n, N_1, m, \sigma)$
approximating f as

$$\sup_{x \in K} \big\| f(x) - \Phi_a(x, \theta) \big\| < \varepsilon.$$

Proof. The original proof is very sophisticated and comprehensible. Therefore, we suggest working through [LLPS93, Theorem 1] if interested. □

While the previous results mainly regard flatter networks of rather small depths and enlarge the network widths up to infinity for their construction, R. Eldan and O. Shamir investigate "*the power of depth for feedforward Neural Networks*" in a 2015 paper of the same name ([ES15]). They provide a minimal example illustrating that there exist sufficiently complex functions for which less deep networks need exponentially more width and parameters than deeper ones to approximate these functions.

Theorem 2.11 (Depth is More Efficient than Width; [ES15])**.**
Let σ be the ReLU or the 0-1-activation function. Then there exist $c > 0$ and $C > 0$ such that for all $d > C$ there exist a measure μ on $\mathbb{R}^d$ and a 3-layer network $\Phi_a(\,\cdot\,; \theta) \colon \mathbb{R}^d \to \mathbb{R}$ with $a = (d, N_1, N_2, 1, \sigma)$ and parameterization θ such that:

- *$\Phi_a(\,\cdot\,; \theta)$ attains values in $[-2, 2]$ and is supported on $\{x : \|x\| \leq C\sqrt{d}\}$.*
- *$\Phi_a(\,\cdot\,; \theta)$ has limited width*

$$\max\{N_1, N_2\} \leq C d^{\frac{19}{4}}$$

- *For any 2-layer Neural Network* $f \coloneqq \Phi_{a'}(\cdot, \theta') \colon \mathbb{R}^d \to \mathbb{R}$*, with* $a' = (d, N_1', 1, \sigma)$ *and width*
$$N_1' \leq ce^{cd}$$
it holds
$$\mathbb{E}_{x\sim\mu}\left[\big(f(x) - \Phi_a(x;\theta)\big)^2\right] \geq c.$$

This means one can construct three-layer Neural Networks of polynomial width that can only be approximated using exponentially growing width.

Proof Idea. The idea found in [ES15] and broken down to the absolute minimum, is to construct a certain radial function $g \coloneqq g^{(0)} \circ \|\cdot\|$ such that 2-layer networks with a width less than ce^{cd}, fail to approximate g on a subspace far away from the origin. This subspace is large enough, that it contributes at least a constant approximation error. The construction is based on two observations.
First, the measure μ can be chosen in a way that the margin between f and Φ_a or g can be expressed by the L_2 distance of the respective Fourier transforms each convoluted with an indicator function φ of the unit ball:
$$\mathbb{E}_{x\sim\mu}\left[\big(f(x) - g\big)^2\right] = \|\hat{f} * \varphi - \hat{g} * \varphi\|_{L_2}^2$$
Second, the Fourier transform of 2-layer Neural Networks can be shown to only be supported on "a union of [finitely many] tubes of bounded radius passing through the origin" [ES15, Claim 15]. Thus, $\hat{f} * \varphi$ is zero in space far away from the origin, where the tubes are distant from each other. However, a radial function g can be constructed to have "L_2-mass" on the space in between the tubes [ES15, Lemma 10 and 14]. The number of tubes which is dependent on the network width, has to be large to fill our space.
The 3-layer network can split the work of approximating the Euclidean norm and the univariate function $g^{(0)}$ onto different layers, making them comparable trivial tasks [ES15, Section 4.4]. Thus g is constructed to be approximated well by a network Φ_a and badly by any network of depth two. Therefore Φ_a also cannot be approximated by 2-layer networks. To obtain insights into the details and technicalities, consider reading [ES15]. □

Theorem 2.11 presents a small-depth proof of concept but the published proof is already very long and technical. Extending this result and its proof idea to deeper networks might neither be possible nor meaningful. However, Theorem 2.11 is a clear indicator, that especially for complex relationships, very deep networks might outperform flatter ones, and therefore properties like universal approximation might be possible for deeper, potentially more efficiently parameterized networks.

Neural Network Shattering of Arbitrary Sets The discussed results investigate how powerful Neural Networks are, regarding the slightly different task of regression. To finish off our investigation concerning the overfitting potential of Neural Networks, we proceed from these well-known approximation results back to classification. The upcoming results are similar to the discussion of VC dimension but are only applicable to very

specific Neural Network architectures. They will provide evidence that Neural Networks might be as powerful as the construction in Example 2.8, making uniform bounds less useful.
Zhang et al. present an example that substantiates previous claims on the possibly gigantic VC dimension of Neural Networks in [ZBH$^+$17]. This publication has made a significant impact on the research scene. A part of the paper presents a result similar to universal approximation applicable to classification tasks. They evaluate the ability of Neural Networks with limited depth and width to *memorize* a finite data set and its labels by heart. Their construction only uses one node per sample point.

Theorem 2.12 (Finite Sample Expressivity; [ZBH$^+$17, Theorem 1])**.**
For any labeled sample s^n *with data in* $\mathbb{R}^m$ *there exists a Neural Network with ReLU activation* σ*, architecture* $a = (m, n, 1, \sigma)$ *and up to* $2n + m$ *uniquely chosen weights fitting the labels perfectly, i.e. achieving zero empirical risk.*

Proof. The proof, provided by [ZBH$^+$17], uses linear algebra to rewrite fitting labels into a solvable system of linear equations: Consider the Neural Network

$$\Phi_a(x) = w_1\,\sigma(W_0 x + b) = \sum_{j=0}^{n} (w_1)_{1,\,j}\,\max\left\{\left\langle \left(e_j^t W_0\right)^t,\, x\right\rangle + b_j,\, 0\right\}$$

for e_j being the j-th unit vector in $\mathbb{R}^n$. Let $\mathrm{s}^n = (x, y)^n$ denote an n-sized labeled sample where all x_i are different. The idea is to set up a linear system based on the Neural Network evaluation and prove that it is solvable. This yields a parameterization fitting all n labels.
Assume all rows of W_0 are equal, i.e. there exists w_0 such that $w_0 = e_j^t W_0 \in \mathbb{R}^{1\times m}$ holds for all j in $\{1, \dots, n\}$. After possibly renaming our data, we can choose w_0 and b such that

$$-b_1 < w_0 x_1 < -b_2 < \cdots < -b_n < w_0 x_n.$$

With the purpose of fitting all samples, we are left to solve

$$(y_1, \dots, y_n) = w_1 \begin{pmatrix} \max\{w_0 x_1 + b_1, 0\} & \dots & \max\{w_0 x_n + b_1, 0\} \\ \vdots & \ddots & \vdots \\ \max\{w_0 x_1 + b_n, 0\} & \dots & \max\{w_0 x_n + b_n, 0\} \end{pmatrix}$$

$$= w_1 \begin{pmatrix} w_0 x_1 + b_1 & & 0 \\ \vdots & \ddots & \\ w_0 x_1 + b_n & \dots & w_0 x_n + b_n \end{pmatrix}$$

By construction, we have an invertible diagonal matrix on the right-hand side. Therefore we can find a weight vector w_1, such that the 2-layer Neural Network fits the sample perfectly. □

This result by Zhang et al. allows for an immediate statement on VC complexity of some Neural Networks.

Corollary 2.13 (VC dimension of 2-Layer Networks)**.**
The hypothesis class $\mathcal{H}_a$ of Neural Networks with architecture $a = (m, n, 1, \sigma)$ where the activation function σ is ReLU has VC complexity

$$\Delta_{\mathcal{H}_a} \geq n.$$

*What is more these networks can shatter **any** set of size n or smaller.*

This theorem about so-called finite sample expressivity might not be surprising and the proof is almost elemental as well. However, the remaining content of the publication [ZBH+17] illustrates similar empirical effects for far bigger Neural Networks. The authors present the potential power of Neural Networks to fit data as badly processed as literal noise. The contents of the upcoming example can be interpreted as a direct continuation of the motivating example (cf. 2.8). Example 2.14 visualizes conveniently that not one set like in Definition 1.13 but seemingly any set of points or sample can be shattered by Neural Networks.

Example 2.14 (The Enormous Power of Neural Networks; [ZBH+17])**.**
Just as suggested by the thought-provoking paper title "*understanding deep learning requires rethinking generalization*" Zhang et al. provide a hyperbole on the immense potential power of deep architectures by exaggerating the issue of poor generalization. They investigate the behavior of three deep Neural Networks oriented on state-of-the-art classifiers after training on distorted data. In line with the universal approximation theorems, they find that state-of-the-art Neural Networks are able to memorize big data sets of the worst kinds.
They provide several experiments from which we briefly present the following one. The used network is a simplified version of Alexnet ([KSH12b])[9] and the data sets are the benchmark set CIFAR10 and variants thereof. The optimizer is a variant of *Stochastic Gradient Descent*. The set CIFAR10 consists of 10 classes totaling 50 000 training and 10 000 test data points in the form of RGB images of size $32 \times 32 \times 3$. Note that the sample size of 50 000 is already quite large for a real-world application.
The test error on this benchmark is negligibly small, refer to [DBK+21] for several networks with errors below 3% with some being as low as less than 0.5% for "ViT-H/14". The experiment is set up to train the small Alexnet on the following variants of CIFAR10:

(a) original data and labels

(b) original data and randomized labels

(c) fixed but random permutation of the pixels of all data points

[9] The simplified architecture in "implementation terms" is Conv 5×5 → Pool 3×3 → Normalization → Conv 5×5 → Pool 3×3 → Normalization → Dense 384 → Dense 192.

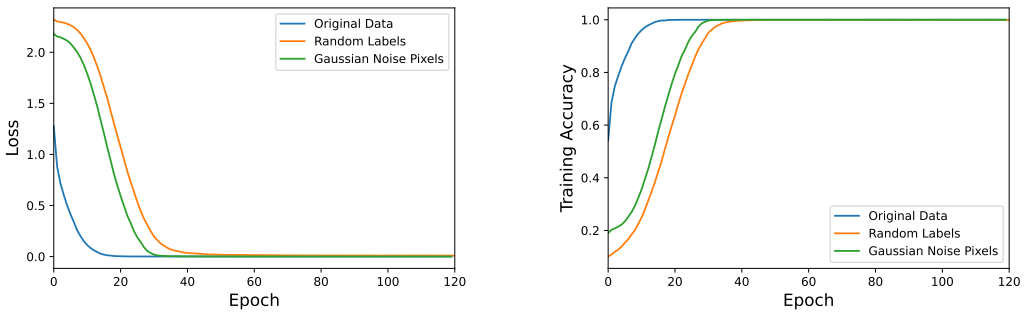


Figure 6: Reproduction of results of [ZBH+17]. The given Neural Network easily achieves zero loss and 100% training accuracy on the experimentation variants (a), (b) and (e). The convergence of noise data or labels is slower but as good as the convergence of the original labeled data.

(d) different random permutations of the pixels of all data points

(e) all pixels are randomly sampled from Gaussian distributions

Intuitively, one would expect a worse training performance, the more we degenerate data from humanly learnable patterns. However, the results of the paper are different. We reproduced the results of (a), (b) and (e) and plotted them in Fig. 6. It is observable, that the **training errors** approach zero for all three cases while the **testing error**, which we use as an unbiased indicator for generalization performance, is 22.3% for the original data, 90.1% for the random labels, and 85.7% for Gaussian pixel data. The second case, i.e. random labels, is especially interesting for our theoretical framework, as fitting random labels perfectly would fit the definition of the VC dimension. Approaching perfect training accuracy here means we almost or completely shatter the set.
Further, the authors interpolate between the first two cases and introduce partially corrupted labels, i.e. each label has a probability of p to replace the true label with a uniformly sampled one. The results thereof are as expected, the testing error increases with more label corruption.
Lastly, Zhang et al. extend these observations to larger and more complex data sets to strengthen the message further. Refer to the cited publication for more information.

Conclusion of the Machine Learning Part of the Proposal We can conclude that Neural Networks, especially state-of-the-art architectures with millions of parameters, most likely have all the power to overfit any real-world data set. Not enough samples can be provided or sometimes even stored in regular workspaces. Thus, uniform bounds such as Vapnik-Chervonenkis generalization will not suffice to explain the proposals' connections like Neural Network generalization and Domain Adaptation.
However, as a way forward, the theory from Section 1 can also be applied solely to the

last layer of a Neural Network, which is nothing but a linear classifier on the data representation $\Phi_a^{L-1}(X;\ \theta)$. In this case, the provided theory is more powerful. Still, one has to investigate how and why which representations are chosen. As one can imagine, this is a problem as big as understanding deep learning itself. Nevertheless, this shift of focus might be a necessary step to explain Neural Network performance.
As already hinted at in the proposal, the information-theoretical part tries to give a perspective or idea where one could start research at and find results based on representations.

2.4. Outlook: Tighter Generalization Bounds for Neural Networks

Before we investigate Sarishvili's Hypothesis 0.1 further, Section 2.4 provides a non-exhaustive overview of modern generalization results that may estimate the generalization error for Neural Networks. Readers only interested in the investigation of the claims of Proposal 0.2 might entirely skip this part and continue with Section 3.
The results presented in this outlook are tighter and stronger than Vapnik-Chervonenkis bounds. We have seen that uniform bounds like VC generalization might be vacuous when faced with very powerful hypotheses classes like Neural Networks due to immense overfitting potential. As compensation for tighter bounds, the upcoming results need to make assumptions closely related to exact problem settings such as data distribution and sample. Therefore, the interpretability of the bounds is significantly reduced. They help to analyze individual applications but not the overarching performance of deep Neural Networks in general.

Remember that we proved a uniform generalization error bound

$$\sup_{h\in\mathcal{H}} \mathrm{P}_{(X,Y)^n}\left(\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \overline{\mathcal{R}}_{(X,Y)^n}(h) \geq \varepsilon_{VC}(\Delta_{\mathcal{H}}, n, \delta)\right) \leq \delta$$

in Section 1 with the purpose of investigating whole hypothesis classes regarding their generalization performance. A special emphasis is on Neural Networks. We showed that if the hypothesis class is too powerful and if we do not make assumptions on the underlying data distribution $\mathrm{P}_{(X,Y)^n}$, such uniform bounds have a tendency to be expressionless. In the literature, several different approaches beyond the VC dimension to quantify generalization gaps are investigated.
Some ideas aim to extend or slightly differ Vapnik-Chervonenkis's theory and provide different uniform bounds. Two well-known examples do not use VC dimension $\Delta_{\mathcal{H}}$ and instead are based on fat-shattering dimension ([BLW96]) or the Rademacher complexity ([SSBD14, Theorem 26.5]). Others include Bayesian statistics in so-called PAC-Bayes learning theory ([McA99]). As mentioned, all of these provide uniform generalization bounds when trying to investigate hypotheses arbitrarily chosen from a function class. These bounds are selectively tighter than VC generalization but suffer from the same problems as discussed throughout this section. They are likely vacuous considering deep Neural Networks.

Just note that special training approaches to achieve predictable generalization by manipulating already trained networks have been developed building up on the PAC-Bayes framework. For details on this approach refer to [DR17].
We figured when using uniform bounds for generalization errors a problem arises if the considered hypothesis space can fit labels independent of the data, i.e. it can work with meaningless ill-defined classification data (cf. Example 2.8). Thus, any result that provides non-vacuous generalization predictions has to include problem-specific assumptions on the considered hypothesis space, the training data, or its distribution. We shortly present two results recently published by K. Kawaguchi, Y. Bengio and L. Kaelbling which can tightly estimate generalization.
The first result adapts to problems we had with VC generalization. The authors circumvent the large value of $\Delta_{\mathcal{H}}$ for Neural Networks and at the same time directly regard the training sample bias we mentioned in Remark 1.22.

Training-Validation Framework The training of Machine Learning algorithms in practice is no topic of this thesis. For the first bound, just note that training is often done using a training-validation method. This means the training data is split into one set on which several different classification models are learned and one set to compare or validate the trained classifiers' performances. Thus, we use an unbiased additional sample unseen during training to validate the models' risks.
Considering only selected, trained classifiers instead of all possible Neural Network hypotheses leads to the following result.

Theorem 2.15 ([KBK22, Proposition 5])**.**
Let $\mathcal{H}_{val}$ be any finite selection of classifiers, not necessarily originating from a shared hypothesis class and let $\mathrm{S}^{n_{val}}$ be an i.i.d. sample distributed by $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$. Define a singe point generalization gap random variable for each classifier and each data point in $\mathrm{S}^{n_{val}}$:

$$\kappa_{h,i} \coloneqq \mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) - \mathcal{L}(h(X_i), Y_i) \quad \text{for all } h \in \mathcal{H}_{val} \text{ and } \mathrm{S}_i = (X_i, Y_i) \in \mathrm{S}^{n_{val}}.$$

Let $\delta \in (0, 1]$. Assume γ and C are positive real numbers such that

$$|\kappa_{h,i}| \leq C \text{ a.s. and } \mathbb{E}[\kappa_{h,i}^2] \leq \gamma^2 \text{ for all } h \in \mathcal{H}_{val} \text{ and } i \in \{1, \dots, n_{val}\}.$$

Then the generalization bound

$$\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \leq \overline{\mathcal{R}}_{S^{n_{val}}}(h) + \frac{2C \log\left(\frac{|\mathcal{H}_{val}|}{\delta}\right)}{3n_{val}} + \sqrt{\frac{2\gamma^2 \log\left(\frac{|\mathcal{H}_{val}|}{\delta}\right)}{n_{val}}} \tag{2.2}$$

holds for all $h \in \mathcal{H}_{val}$ with probability $1 - \delta$ w.r.t. the sample.
Proof. [KBK22, Appendix C6]. □

Note, that calling the sample validation set instead of sample does not make this bound tight. Also, tightness in comparison to VC generalization (Theorem 1.21) does not come from referring to the sample as a validation set $\mathrm{S}^{n_{val}}$. This renaming yields no difference in the result, it only displays a guideline for the application like Remark 1.22. Using the training sample realization to validate the model and estimate the true risk yields biased results. Instead, it is tight because of the finite cardinality of the considered functions $\mathcal{H}_{val}$ and because the relation between the number of classifiers and samples might be small. Only considering finitely many classifiers allows for far less complex classification. The authors provide the following numerical example showing that the bound is tight in practice.

Example 2.16 (The Validation Bound is Tight; [KBK22])**.**
Consider a validation set of $n_{val} = 10\,000$ (cf. MNIST in Section 4 or CIFAR-10 in Example 2.14) and an error tolerance $\delta = 0.1$. Further, assume $\mathcal{H}_{val}$ are the classifiers visited during training after each optimization step. As a proof of concept use artificially large numbers of considered classifiers: We might consider 20 different architectures trained 1000 epochs each where an epoch makes 50000 training steps, i.e. considers each training sample of MNIST or CIFAR once. In $\mathcal{H}_{val}$ we save each of these $100\,000\,000$ intermediate network parameterizations. If we add an order of magnitude for convenience, this example leads to an immensely large set of pre-chosen classifiers we want to validate with $|\mathcal{H}_{val}| = 1\,000\,000\,000$, i.e.

$$\log\left(\frac{|\mathcal{H}_{val}|}{\delta}\right) \approx 23.03.$$

The worst case value for $|\kappa_{h,i}|$ is by definition 1 and for $C = \gamma^2 = 1$ it holds

$$\frac{2C\log\left(\frac{|\mathcal{H}_{val}|}{\delta}\right)}{3n_{val}} \approx 0.00154 \quad \text{and} \quad \sqrt{\frac{2\gamma^2\log\left(\frac{|\mathcal{H}_{val}|}{\delta}\right)}{n_{val}}} \approx 0.0679.$$

In conclusion, it holds $\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(h) \leq \overline{R}_{\mathrm{S}^{n_{val}}}(h) + 7\%$ for all hypotheses in $\mathcal{H}_{val}$. If we investigate the single point generalization gaps $\kappa_{h,i}$ further and decrease γ^2 instead of the worst case $\gamma = 1$, this bound can be estimated far tighter.

This result is a numerical observation applicable to any Machine Learning algorithm and neglects all information about the overarching hypothesis class and its power. Kawaguchi et al. also elaborate on Neural Network generalization directly and consider deep data representations in another theorem.
The authors consider a Neural Network as a directed acyclic graph (DAG) and rewrite the simple calculation of linear layers and activation functions as a sum of all paths through the DAG. The construction of the Neural Network DAG is as follows.

Deep Path Through Neural Networks For simplicity assume we have no biases in the network layers' linear calculations. Given the input x and the parameterization θ

there is one node for each component of the intermediate representations $\Phi_a^l(x;\theta) = \sigma(W_{l-1}\sigma(\dots(W_1\sigma(W_0x)\dots)))$. The edge from the j-th component of layer l to the i-th component of layer $l+1$ has weight $(W_l)_{i,j}$ if $\Phi_a^{l+1}(x;\theta)_i \neq 0$ and 0 else. In the latter case, where $\Phi_a^{l+1}(x;\theta)_i = 0$ any information on this path does not influence the output of the whole network $\Phi_a(x;\theta) = W_{L-1}\sigma(\dots(W_1\sigma(W_0x))\dots)$. In such cases, where the output component of an edge gets saturated by ReLU during Neural Network feed-forward evaluation of x, the edge is deemed "**inactive**".
Using this DAG, one can rewrite the output of the parameterized Neural Network as the sum of paths from input x to output $\Phi_a(x;\theta)$, where each path's value is the product of its edge weights.

Example 2.17 (Deep Paths to Calculate Representations; [KBK22, Section 5.2])**.**
Consider a trained Neural Network with input X and ReLU-activation function as a DAG and consider paths from input to output layer through the graph. Let $\hat{\sigma}_l$ be the 0-1-vector, indicating if the edges ending at layer l are active or not. Then, we can rewrite the calculation of the output $\Phi_a(x\,;\theta)$ as

$$\Phi_a(x;\theta) = \sum_{j_{L-1}=1}^{N_{L-1}} \cdots \sum_{j_0=1}^{N_0} (W_{L-1})_{j_{L-1}} \prod_{l=0}^{L-2} (W_l)_{j_{l+1},j_l} \prod_{l=1}^{L-1} (\hat{\sigma}_l)_{j_l}(x,\theta)\, x_{j_0}.$$

This is just a complicated way to execute the chained linear function and ReLU calculations. Note that the calculation can be extended to higher dimensional outputs when considering multiclass instead of binary classification.
To calculate bounds we need a vector z with an entry z_j for each path $j \in \{1,\dots,N_0\} \times \cdots \times \{1,\dots,N_L\}$ indicating if all edges in the path are active, i.e. if the value of the path contributes to the output.

$$z_j \coloneqq \prod_{l=1}^{L-1} (\hat{\sigma}_l)_{j_l}(x;\theta)\, x_{j_0}$$

Similarly define w as the vector of the values of the paths.
If the input of the network is a random variable X, then the Z and W are the corresponding random variables spring active paths and path weights.

Using these deep paths, Kawaguchi et al. provide the following deterministic generalization bound for the specific trained network one obtains after training.

Theorem 2.18 (Deep Paths; [KBK22, Theorem 7])**.**
Let $(x,y)^n \sim \mathrm{P}_{\mathcal{S},\mathcal{Y}}$ be a labeled i.i.d. sample realization and let (X,Y) be a random variable also distributed according to $\mathrm{P}_{\mathcal{S},\mathcal{Y}}$. Let Z and z_i for $i = \{1,\dots,n\}$ representing deep paths as above, i.e. Z corresponds to X and z_i to (x_i,y_i). Define

$$G \coloneqq \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}[ZZ^t] - \frac{1}{n}\sum_{i=1}^{n} z_i z_i^t \quad \text{and} \quad v \coloneqq \frac{1}{n}\sum_{i=1}^{n} y_i z_i - \mathbb{E}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}[YZ].$$

Let $\{\lambda_j\}_j$ and $\{u_j\}_j$ denote eigenvalues and corresponding orthonormal eigenvectors of G. Let $\theta_{u_j,w}$ and $\theta_{v,w}$ denote the angles between the u_j and w and v and w.

Then it holds

$$\begin{aligned}\mathcal{R}_{\mathrm{P}_{\mathcal{S},\mathcal{Y}}}(\Phi_a(\,\cdot\,;\theta)) - \overline{\mathcal{R}}_{(x,y)^n}(\Phi_a(\,\cdot\,;\theta)) &= \left(2\|v\|\|w\|\cos\theta_{v,w} + \|w\|^2 \sum_j \lambda_j \cos\theta_{u_j,w}\right) \\ &\quad + \left(\mathbb{E}_{\mathrm{P}_{\mathcal{Y}}}[\|Y\|^2] - \frac{1}{n}\sum_{i=1}^{m}\|y_i\|^2\right) \\ &\leq \left(2\|v\|\|w\| + \|w\|^2\lambda_{max}(G)\right) + \left(\mathbb{E}_{\mathrm{P}_{\mathcal{Y}}}[\|Y\|^2] - \frac{1}{n}\sum_{i=1}^{m}\|y_i\|^2\right).\end{aligned}$$

Proof. [KBK22, Appendix C6]. □

The first equation calculates the generalization gap based on the weights. Using the angles between the vectors of deep data paths and the corresponding weights measures a kind of similarity between path activeness and path weight. The authors describe this as a compression in the calculation of the output representation $\Phi_a(x;\theta)$ induced by the weights. Further, they describe the second bracket which is only dependent on the sample and not the training, as a label concentration.
As we suggested in the conclusion of Section 2, Theorem 2.18 links the generalization of deep Neural Networks to the chosen representations. However, this bound has to be calculated for each considered Neural Network parameterization individually. This gives a hard time when making general claims about Neural Network generalization and thus is difficult to interpret.
After seeing those insights into modern generalization bounds, we come back to the task of this thesis, the investigation of general deep classifiers by validating Sarishvili's Hypothesis 0.1.

3. Information Theory

"The key to artificial intelligence has always been the representation."
- Jeff Hawkins

The second part of this thesis will shift the focus from Machine Learning to Information Theory as suggested in Proposal 0.2. We will first introduce the reader to the foundation (Section 3.1) required to investigate a possible interplay between our framework of minimizing risks to an optimization problem considering information-theoretical definitions (Section 3.2 and Section 3.3).

3.1. Information-Theoretic Basics

Information Theory is typically concerned with the predictability of random variables. The main properties discussed in this thesis will be *entropy*, a measure for the uncertainty or degree of randomness a human perceives when sampling a random variable, and *mutual information*, a measure to quantify how much information sampling one random variable will provide about a second random variable.

The Intuition of Entropy There are various possible interpretations for entropy and thereby Information Theory. Most are dependent on interpreting the quantity

$$\log_2 \frac{1}{\mathrm{P}_X(x)} = -\log_2 \mathrm{P}_X(x)$$

and entropy is defined as the mean thereof (cf. Definition 3.1).
$-\log_2 \mathrm{P}_X(x)$ is often referred to as **information** obtained by or the **surprise** of observing the event x. Consider an example illustrating the aspect of **surprise**:
If an unfair coin X^0 has $\mathrm{P}_{X^0}(\text{"heads"}) = 0.9$, flipping it is a quite predictable process. Most of the time heads will show and the surprise of this event is low. Seeing "heads" when flipping a fair coin X^1 feels more surprising or random. Similarly, an unlikely or unpredictable event feels more surprising. The corresponding surprise values are written down in Table 2.

	$-\log_2 \mathrm{P}_{X^0}(x)$	$-\log_2 \mathrm{P}_{X^1}(x)$
$x =$ "heads"	0.152	1
$x =$ "tails"	3.332	1

Table 2: "Humanly perceived surprise" of the outcomes of an unfair coin X^0 and a fair coin X^1 measured by $-\log_2 \mathrm{P}_X(x)$.

Next, we extend this idea to the interpretation of gained **information** when observing x. This is based on the idea that highly probable events provide low information and

rare events are very informational. One could justify this claim by considering physical experiments. In this context, observing unlikely events in an experimental iteration could lead to the observation of unlikely outcomes. Therefore, the understanding of underlying concepts might be improved further if the outcome of the experiment is more improbable.
These two interpretations are pretty comfortable as they rely on simple human perception but at the same time, they might not be fully satisfying explanations.
A third interpretation regards cryptography and the theory of encoding messages with letters from a finite alphabet $\mathcal{X}$. It is desirable to encode messages as efficiently as possible. Naive encoding like assigning a unique bit-vector of length $\log_2 |\mathcal{X}|$ to each letter can be vastly improved if the letter distribution of the average message is known. In this case, the most efficient encoding assigns each letter a bit vector of length dependent on its frequency of occurrence. Such construction is done by Huffman's algorithm, published in [Huf52]. The lengths correspond to our information content $-\log_2 \mathrm{P}_X(x)$. Thus it can also be referred to as the (optimal) **cost of encoding** a letter x with the occurrence probability $\mathrm{P}_X(x)$. This means a good encoding uses knowledge about the letter distribution. Entropy describes the average encoding cost per letter.

Obviously, these three interpretations are substantially different and the first two do not even rely on the exact form of $-\log_2 \mathrm{P}_X(x)$ but also hold for different possible definitions of surprise and information. Nevertheless, this form directly yields useful properties like a chain rule (Proposition 3.3) which will be the foundation for the connection between misclassification risk and Information Theory seen in Proposal 0.2.
With possible interpretations in mind, this subsection will define entropy as the "expected information gain" or "expected surprise" when sampling a random variable and then provide the basic theory necessary to further investigate the proposal in Section 3.2 and Section 3.3.
Note that the content of this sub-chapter is based on [CT06, Chapter 2]. In the upcoming definitions, we include equivalent definitions without proof. The rather straightforward calculations proving these can also be found in the cited book by Cover and Thomas.

Definition 3.1 (Entropy)**.**
Let X and Y be random variables on the sets $\mathcal{X}$ and $\mathcal{Y}$. If X is distributed by P_X, we define the **entropy** H of X

$$H(X) \coloneqq \mathbb{E}_{X\sim \mathrm{P}_X}\left[\log_2 \frac{1}{\mathrm{P}_X(X)}\right]$$

The **conditional entropy** $H(X|Y)$ of X conditioned on observation of Y is defined as

$$H(X|Y) \coloneqq \mathbb{E}_{(X,Y)\sim \mathrm{P}_{X,Y}}\left[\log_2 \frac{1}{\mathrm{P}_{X|Y}(X|Y)}\right]$$

If X and Y are discrete this is sometimes written equivalently as

$$\begin{aligned} H(X|Y) &\coloneqq \sum_{y\in\mathcal{Y}} \mathrm{P}_Y(y)\, H(X|Y=y) \\ &\coloneqq -\sum_{x\in\mathcal{X},\, y\in\mathcal{Y}} \mathrm{P}_{X,Y}(x,y)\log_2 \mathrm{P}_{X|Y}(x\,|\,y). \end{aligned}$$

Extending the notion of entropy we can define measurements to compare random variables and distributions.

Definition 3.2 (Similarity Measures)**.**
Let X, Y and Z be random variables on the sets $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$.

(a) Define the **mutual information** $I(X;Y)$ between X and Y as

$$\begin{aligned} I(X;Y) &\coloneqq \mathbb{E}_{(X,Y)\sim \mathrm{P}_{X,Y}}\left[\log_2 \frac{\mathrm{P}_{X,Y}(X,Y)}{\mathrm{P}_X(X)\,\mathrm{P}_Y(Y)}\right] \\ &= I(Y;X) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

We can once again implement conditionals into this measure if we want to embed knowledge of a third random variable Z. This yields the definition of **conditional mutual information** $I(X;Y\,|Z)$[10] of X and Y given Z as

$$\begin{aligned} I(X;Y\,|Z) &\coloneqq \mathbb{E}_{(X,Y,Z)\sim \mathrm{P}_{X,Y,Z}}\left[\log_2 \frac{\mathrm{P}_{X,Y|Z}(X,Y|Z)}{\mathrm{P}_{X|Z}(X|Z)\,\mathrm{P}_{Y|Z}(Y|Z)}\right] \\ &= H(X|Z) - H(X|Y,Z). \end{aligned}$$

(b) If one wants to compare two distributions P and Q on the shared set $\mathcal{X}$, consider the **Kullback-Leibler Divergence**

$$KL(\mathrm{P}||\mathrm{Q}) \coloneqq \mathbb{E}_{X\sim \mathrm{P}}\left[\log_2 \frac{\mathrm{P}(X)}{\mathrm{Q}(X)}\right]$$

It is convention that $\mathrm{P}(x)\log_2 \frac{\mathrm{P}(x)}{\mathrm{Q}(x)} \coloneqq \infty$ for any x with $\mathrm{P}(x) \neq 0$ and $\mathrm{Q}(x) = 0$. Similarly, $\mathrm{P}(x)\log_2 \frac{\mathrm{P}(x)}{\mathrm{Q}(x)} \coloneqq 0$ for all x with $\mathrm{P}(x) = 0$ and arbitrary $\mathrm{Q}(x)$.

[10] One could use brackets as in $I\big((X;Y)|Z\big)$ to emphasize that the mutual information of X and Y is the expressions main focus and it does *not* simply describe mutual information of "X" and "$Y|Z$". Nevertheless, we will stick to the traditional and more common notation.

To obtain some intuition about the given measures some important properties are provided. Those properties will be useful in the remainder of this chapter.

Proposition 3.3 (Properties of Entropy and Mutual Information)**.**
Let X, Y and Z be random variables on the sets $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$. Further let P *and* Q *be two distributions on $\mathcal{X}$. Then one can prove the following bounds.*

(a) Kullback-Leibler divergence is always non-negative, i.e.

$$KL\left(\mathrm{P}||\mathrm{Q}\right) \geq 0$$

where equality holds if and only if $\mathrm{P}(x) = \mathrm{Q}(x)$ holds almost everywhere or, if X is discrete, for all x.

(b) Depending on $\mathcal{X}$, bounds for entropy can or cannot be derived.

(i) The entropy of discrete random variables with finite support can be bound. If $\mathcal{X}$ is finite, then

$$0 \leq H(X) \leq \log_2 |\mathcal{X}|. \tag{3.1}$$

The lower bound is achieved by deterministic X and the upper bound is achieved if and only if X is uniformly distributed on $\mathcal{X}$.

(ii) For continuous random variables no entropy bound can be found. For any $c > 0$ there exist random variables X and Y such that

$$H(X) < -c \text{ and } H(Y) > c.$$

(c) Mutual information is non-negative, i.e.

$$I(X;Y) \geq 0$$

with equality $I(X;Y) = 0$ if and only if X and Y are independent.

Further important properties are the following.

(d) ***Conditioning is a monotone operation*** *for entropy, i.e.*

$$H(X|Y) \leq H(X).$$

Equality holds if and only if X and Y are independent.

(e) Entropy is a ***concave*** *functional w.r.t. underlying probability distributions. This means, if $X \sim P_X$ and $Y \sim P_Y$, then it holds for all $Z \sim \alpha P_X + (1-\alpha)P_Y$ with $\alpha \in [0,1]$ that*

$$H(Z) \geq \alpha H(X) + (1-\alpha)H(Y)$$

(f) There are ***chain rules*** *for entropy and mutual information. Analogous to the probability distribution chain rule it holds*

$$H(X,Y) = H(Y|X) + H(X)$$

and

$$I(X,Y;Z) = I(Y;Z|X) + I(X;Z) \tag{3.2}$$

and similar for larger numbers of random variables.

Proof. These properties and more can be found in [CT06]. For the interested reader the proofs are condensed, changed to fit our framework, and collected in Appendix B, Proposition B.4. □

Remark 3.4.
Using the chain rule for entropy allows rewriting mutual information as

$$I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y).$$

This, as well as Proposition 3.3 (*c*), shows that mutual information measures something similar to dependence between random variables.

Remark 3.5 (Interaction Information; [Yeu91])**.**
Note, that mutual information is only defined for two random variables, separated by ";". Thus, $I(X,Y;Z)$ in the chain rule Eq. (3.2) is the mutual information of the joint random variable (X,Y) and Z.
This is not to be confused with a generalization of mutual information to more than two random variables, sometimes called interaction information. This is inductively defined through

$$I(X;Y;Z) \coloneqq I(X;Y) - I(X;Y|Z).$$

Definition 3.6 (Markov Chain)**.**
The random variables X, Y and Z form a Markov Chain, denoted as $X \to Y \to Z$, if we can simplify the joint probability

$$\mathrm{P}_{X,Y,Z}(x,y,z) \coloneqq \mathrm{P}_X(x)\mathrm{P}_{Y|X}(y\,|\,x)\mathrm{P}_{Z|X,Y}(z\,|\,x,y)$$

to

$$\mathrm{P}_{X,Y,Z}(x,y,z) = \mathrm{P}_X(x)\mathrm{P}_{Y|X}(y\,|\,x)\mathrm{P}_{Z|Y}(z\,|\,y) \tag{3.3}$$

for all realizations x, y and z.

Lemma 3.7 (Equivalent Definition of Markov Chains)**.**
The Markov chain property is equivalent to Z being conditionally independent of X if the outcome of Y is known.

Proof. It holds $X \to Y \to Z$ if and only if $\mathrm{P}_{Z|Y}(z\,|\,y) = \mathrm{P}_{Z|X,Y}(z\,|\,x,y)$ for all realizations x, y and z. Considering the chain rule for probability distributions this is equivalent to

$$\mathrm{P}_{Z,X|Y}(z,x\,|\,y) = \mathrm{P}_{Z|X,Y}(z\,|\,x,y)\mathrm{P}_{X|Y}(x\,|\,y) = \mathrm{P}_{Z|Y}(z\,|\,y)\mathrm{P}_{X|Y}(x\,|\,y),$$

which yields the independence of the conditional probabilities. □

Remark 3.8 (Neural Networks Form Markov Chains)**.**
If a random variable Z can be calculated using a map on Y, i.e. $Z = f(Y)$ for some function f, then $X \to Y \to Z$ is a Markov chain. Thus several different Markov chains can be found in Neural Network representations. Consider the l-th hidden layer and the random variables for data X and labels Y. Then

$$X \to \Phi_a^l(X;\theta) \to \Phi_a^L(x;\theta) = \Phi_a(x;\theta)$$

forms a Markov chain. Similarly

$$Y \to X \to \Phi_a(x;\theta), Y \to X \to \Phi_a^l(x;\theta) \text{ and } Y \to \Phi_a^l(x;\theta) \to \Phi_a(x;\theta)$$

are Markov chains.

Lemma 3.9 (DPI; [CT06, Theorem 2.8.1.])**.**
For any Markov chain $X \to Y \to Z$ the ***data-processing inequality (DPI)***

$$I(X;Y) \geq I(X;Z)$$

holds.
Proof. Reconsider the chain rule for mutual information

$$I(Y,Z;X) = I(X;Z) + I(X;Y|Z) = I(X;Y) + I(X;Z|Y).$$

By the Markov property, we have $I(X;Z|Y) = 0$ due to Lemma 3.7 and thus

$$I(X;Z) \leq I(X;Z) + I(X;Y|Z) = I(X;Y).$$

□

Remark 3.10 (DPI Implication)**.**
If $X \to Y \to Z$, it holds

$$\begin{aligned} 0 &\leq I(X;Y) - I(X;Z) \\ &= H(X) - H(X|Y) - \Big(H(X) - H\left(X|Z\right)\Big) \\ &= H(X|Z) - H(X|Y). \end{aligned}$$

Thus, DPI yields

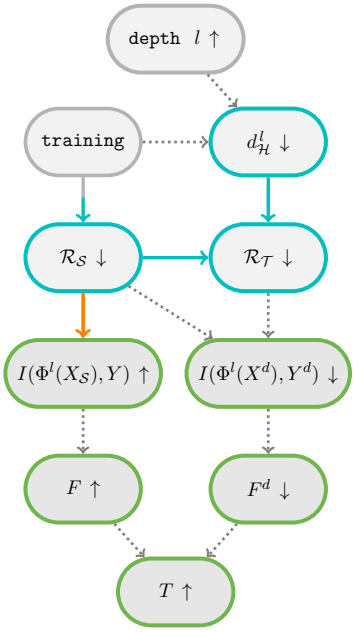
$$X \to Y \to Z \quad \Longrightarrow \quad H(X|Z) \geq H(X|Y).$$

Using the provided collection of definitions and results, we can investigate the interplay between risk and information content in the representation and thereby continue the investigation of Proposal 0.2.

3.2. Interconnection of Risk and Information Theory

3.2.1. Fano's Inequality

Having introduced a mathematical framework to talk about information, we can look into possible connections between misclassification risk and dependence between representations and labels measured by mutual information.
The proposals' connection (b) (i) can be explained using a well-known result proven by Robert Fano in 1961 in his book *Transmission of information: a statistical theory of communications* ([Fan61]). It introduces a random variable to measure misclassification and thereby proves a bound using properties from Proposition 3.3. Afterward, we evaluate whether Fano's inequality can be applied to investigate the quality of representations as proposed in the conclusion of the Machine Learning part.



Theorem 3.11 (Fano's Inequality)**.**
Consider random variables X, Y and $\hat{Y}$ on $\mathcal{X}$ and $\mathcal{Y}$ such that $Y \to X \to \hat{Y}$ form a Markov chain. $\mathcal{Y}$ is finite. Interpret $\hat{Y}$ as an estimator for Y and let $I_\oplus$ be the XOR indicator random variable for Y and $\hat{Y}$, i.e.

$$I_\oplus = \begin{cases} 1, \text{ if } Y \neq \hat{Y} \\ 0, \text{ if } Y = \hat{Y}. \end{cases}$$

Then it holds

$$H(I_\oplus) + \mathrm{P}(Y \neq \hat{Y}) \log_2 |\mathcal{Y}| \geq H(Y|\hat{Y}) \geq H(Y|X). \tag{3.4}$$

Proof. With knowledge about the outcomes of two of the three random variables Y, $\hat{Y}$ and $I_\oplus$, calculation of the outcome of the unknown third is sometimes possible. Thus, $H(I_\oplus|Y,\hat{Y}) = 0$ and $H(Y|\hat{Y}, I_\oplus = 0) = 0$ hold. Using these two "zeros" and the chain rule for entropy (Proposition 3.3, f)) yields

$$\begin{aligned} H(Y|\hat{Y}) &= H(Y|\hat{Y}) + H(I_\oplus|Y,\hat{Y}) = H(I_\oplus, Y|\hat{Y}) = H(I_\oplus|\hat{Y}) + H(Y|\hat{Y}, I_\oplus) \\ &= H(I_\oplus|\hat{Y}) + \mathrm{P}(I_\oplus = 1) H(Y|\hat{Y}, I_\oplus = 1) + \mathrm{P}(I_\oplus = 0) H(Y|\hat{Y}, I_\oplus = 0) \\ &\leq H(I_\oplus) \quad + \mathrm{P}(I_\oplus = 1) H(Y|\hat{Y}, I_\oplus = 1) + 0 \\ &\leq H(I_\oplus) + \mathrm{P}(Y \neq \hat{Y}) \log_2 |\mathcal{Y}|. \end{aligned}$$

The last step uses $H(Y|\hat{Y}, I_\oplus = 1) \leq H(Y) \leq \log_2 |\mathcal{Y}|$. This shows that

$$H(Y|\hat{Y}) \leq H(I_\oplus) + \mathrm{P}(Y \neq \hat{Y}) \log_2 |\mathcal{Y}|$$

holds.
The remaining claim, $H(Y|\hat{Y}) \geq H(Y|X)$, is a direct consequence of the Markov chain property and DPI (Remark 3.10). □

We now want to extend this result to make claims about its tightness and achieve a form usable for the proposed connection we currently investigate. For this purpose, a helpful lemma relating entropy to the underlying probability distribution is introduced. It acts as a reorganized and shortened version of content published by Ho and Verdú in [HV10].

Lemma 3.12 ([HV10])**.**
For any discrete random variable Y with finite support $\mathcal{Y}$, it holds

$$H(Y) \geq 2 - 2 \max_{y \in \mathcal{Y}} \mathrm{P}_Y(y). \tag{3.5}$$

Proof. The idea of the proof is to introduce an intermediate random variable W with the property

$$2 - 2 \max_{y \in \mathcal{Y}} \mathrm{P}_Y(y) \leq H(W) \leq H(Y).$$

One defines W in a way that its entropy can be bound using concavity together with the knowledge about the entropy of uniformly distributed discrete random variables, see Proposition 3.3 (b) (i). Subsequently, proving that the entropy of W is smaller than the entropy of Y yields the claim.
Let $\mathcal{Y} = \{y_1, \ldots, y_m\}$ and assume without loss of generality that the elements are ordered by probabilities, i.e. $i < j$ implies $\mathrm{P}_Y(y_i) \geq \mathrm{P}_Y(y_j)$. In particular, it holds

$$y_1 = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathrm{P}_Y(y).$$

Let $l \coloneqq \left\lfloor \frac{1}{\mathrm{P}_Y(y_1)} \right\rfloor \leq m$ and define the random variable W in terms of its distribution on $\mathcal{Y}$ as

$$\mathrm{P}_W(y_i) \coloneqq \begin{cases} \mathrm{P}_Y(y_1) & \text{, if } i \leq l \\ 1 - l\,\mathrm{P}_Y(y_1) & \text{, if } i = l+1 \\ 0 & \text{, else.} \end{cases} \tag{3.6}$$

Let

$$\alpha \coloneqq l \left(\mathrm{P}_Y(y_1)(l+1) - 1\right)$$

and let U_l and U_{l+1} be the uniformly distributed random variables on $\{y_1, \ldots, y_l\}$ and $\{y_1, \ldots, y_{l+1}\}$ respectively. We can calculate

$$\alpha \mathrm{P}_{U_l}(y_i) = l \left(\mathrm{P}_Y(y_1)(l+1) - 1\right) \mathrm{P}_{U_l}(y_i) = \begin{cases} \mathrm{P}_Y(y_1)(l+1) - 1 & \text{, if } i \leq l \\ 0 & \text{, otherwise.} \end{cases}$$

Similarly,

$$(1-\alpha)\,\mathrm{P}_{U_{l+1}}(y_i) = \big((l+1) - \mathrm{P}_Y(y_1) l\,(l+1)\big)\,\mathrm{P}_{U_{l+1}}(y_i) = \begin{cases} 1 - l\,\mathrm{P}_Y(y_1) & \text{, if } i \le l+1 \\ 0 & \text{, otherwise.} \end{cases}$$

Therefore, it holds

$$\mathrm{P}_W(y_i) = \alpha \mathrm{P}_{U_l}(y_i) + (1-\alpha)\,\mathrm{P}_{U_{l+1}}(y_i) = \begin{cases} \mathrm{P}_Y(y_1) & \text{, if } i \le l \\ 1 - l\,\mathrm{P}_Y(y_1) & \text{, if } i = l+1 \\ 0 & \text{, else} \end{cases} \tag{3.7}$$

for all integers $i = 1, \dots, m$. Thus, concavity of entropy (cf. Proposition 3.3) yields

$$H(W) \ge \alpha H(U_l) + (1-\alpha) H(U_{l+1}) = \alpha \log_2 l + (1-\alpha)\log_2(l+1).$$

Applying the inequality $\log_2 l \ge 2 - \frac{2}{l}$ this can be rewritten as

$$H(W) \ge \alpha\left(2 - \frac{2}{l}\right) + (1-\alpha)\left(2 - \frac{2}{l+1}\right) = 2\left(1 - \alpha\frac{1}{l} - (1-\alpha)\frac{1}{l+1}\right).$$

Inserting $\mathrm{P}_W(y_1) = \alpha\frac{1}{l} + (1-\alpha)\frac{1}{l+1} = \mathrm{P}_Y(y_1)$ concludes the first step of the proof as

$$H(W) \ge 2\left(1 - \mathrm{P}_Y(y_1)\right).$$

To finish the proof, it remains to show that the entropy of W is less or equal to the entropy of Y. For this purpose, a telescoping sum is introduced.

$$\begin{aligned} &H(Y) - H(W) \ge H(Y) - H(W) - KL(\mathrm{P}_W || \mathrm{P}_Y) \\ &= \sum_{i=1}^{m} \mathrm{P}_Y(y_i) \log_2 \frac{1}{\mathrm{P}_Y(y_i)} - \sum_{i=1}^{m} \mathrm{P}_W(y_i) \log_2 \frac{1}{\mathrm{P}_W(y_i)} - \sum_{i=1}^{m} \mathrm{P}_W(y_i) \log_2 \frac{\mathrm{P}_W(y_i)}{\mathrm{P}_Y(y_i)} \\ &= \sum_{i=1}^{m} \big(\mathrm{P}_Y(y_i) - \mathrm{P}_W(y_i)\big) \log_2 \frac{1}{\mathrm{P}_Y(y_i)} \end{aligned}$$

Use that $\log_2 \frac{1}{\mathrm{P}_Y(y_i)} = \sum_{k=i}^{m-1} \left(\log_2 \frac{\mathrm{P}_Y(y_{k+1})}{\mathrm{P}_Y(y_k)}\right) + \log_2 \frac{1}{\mathrm{P}_Y(y_m)}$ for all i:

$$\begin{aligned} \dots &= \sum_{i=1}^{m-1} \left[\big(\mathrm{P}_Y(y_i) - \mathrm{P}_W(y_i)\big) \sum_{k=i}^{m-1} \log_2 \frac{\mathrm{P}_Y(y_{k+1})}{\mathrm{P}_Y(y_k)} \right] + \log_2 \frac{1}{\mathrm{P}_Y(y_m)} \underbrace{\sum_{i=1}^{m} \big(\mathrm{P}_Y(y_i) - \mathrm{P}_W(y_i)\big)}_{=0, \text{ as } \sum_{i=1}^m \mathrm{P}_Y(y_i) = 1 = \sum_{i=1}^m \mathrm{P}_W(y_i)} \\ &= \sum_{i=1}^{m-1} \big(\mathrm{P}_Y(y_i) - \mathrm{P}_W(y_i)\big) \sum_{k=i}^{m-1} \log_2 \frac{\mathrm{P}_Y(y_{k+1})}{\mathrm{P}_Y(y_k)} \\ &= \sum_{k=1}^{m-1} \log_2 \frac{\mathrm{P}_Y(y_k)}{\mathrm{P}_Y(y_{k+1})} \sum_{i=1}^{k} \big(\mathrm{P}_W(y_i) - \mathrm{P}_Y(y_i)\big) \ge 0 \end{aligned}$$

The last expression is non-negative since we assumed that $\mathrm{P}_Y(y_k) \geq \mathrm{P}_Y(y_{k+1})$ and since by construction of W it holds

$$\sum_{i=1}^{k} \mathrm{P}_Y(y_i) \leq \sum_{i=1}^{k} \mathrm{P}_Y(y_1) = \sum_{i=1}^{k} \mathrm{P}_W(y_i)$$

as long as $k \leq l$ and it holds

$$\sum_{i=1}^{k} \mathrm{P}_Y(y_i) \leq 1 = \sum_{i=1}^{k} \mathrm{P}_W(y_i)$$

otherwise. Thus, $H(Y) \geq H(W) \geq 2 + 2\mathrm{P}_Y(y_1)$ concludes the proof. □

It is possible to relax both Fano's inequality (Theorem 3.11) and Lemma 3.12 and collect them in one result connecting information and misclassification risk. This yields the first proposed interconnection.

Corollary 3.13 (Relaxed Fano Inequeality)**.**
Consider random variables X, Y and $\hat{Y}$ on $\mathcal{X}$ and $\mathcal{Y}$ with the latter being a finite set. Let $Y \to X \to \hat{Y}$ form a Markov chain. Then it holds

$$\frac{H(Y) - I(X;Y) - 1}{\log_2 |\mathcal{Y}|} = \frac{H(Y|X) - 1}{\log_2 |\mathcal{Y}|} \leq \mathrm{P}(Y \neq \hat{Y}). \tag{3.8}$$

This inequality's tightness can be analyzed by an similar upper bound on $\mathrm{P}(Y \neq \hat{Y})$. There exists a label estimator $\hat{Y}^*$, precisely the prediction a Bayes classifier[11] would choose, such that

$$\mathrm{P}\left(Y \neq \hat{Y}^*\right) \leq \frac{1}{2}\Big(H\left(Y|X\right)\Big) = \frac{1}{2}\Big(H\left(Y\right) - I\left(X;Y\right)\Big). \tag{3.9}$$

Proof. Having discussed Fano's inequality proving the claim in Eq. (3.8) is straightforward. The left equality of the equation can be achieved by insertion of $I(X;Y) = H(Y) + H(Y|X)$. Further, Fano's Inequality (Theorem 3.11) can be rewritten as

$$H(Y|X) \leq H(I_\oplus) + \mathrm{P}(Y \neq \hat{Y}) \log_2 |\mathcal{Y}|.$$

Since $I_\oplus$ is a binary random variable, it holds $H(I_\oplus) \leq \log_2 2 \leq 1$. Rearranging of the terms yields the second claim in Eq. (3.8).
The second statement, Eq. (3.9), is a direct consequence of the previous discussions in Lemma 3.12.

[11] A Bayes classifier is a theoretical construct labeling data with the optimal labels w.r.t. the generally unknown underlying distribution: $f\colon \mathcal{X} \to \mathcal{Y},\ x \mapsto \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ \mathrm{P}_{Y|X}(y|x)$.

Let

$$f\colon \mathcal{X} \to \mathcal{Y},\, x \mapsto \operatorname*{argmax}_{y\in\mathcal{Y}} \mathrm{P}_{Y|X}(y \,|\, x)$$

be the Bayes classifier and $\hat{Y}^* = f(X)$. It holds

$$1 - \mathrm{P}_{Y|X}\big(f(x) \,\big|\, x\big) = \sum_{y\in\mathcal{Y}\setminus\{f(x)\}} \mathrm{P}_{Y|X}\big(y \,\big|\, x\big) = \sum_{y\in\mathcal{Y}} \mathrm{P}_{Y|X}\big(y \,\big|\, x\big) \mathbb{1}\left\{y \neq f(x)\right\}$$

Hence, calculation yields

$$\begin{aligned}
P(Y \neq \hat{Y}^*) &= \int_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} \mathrm{P}_{X,Y}(x,y) \mathbb{1}\left\{y \neq f(x)\right\} \,\mathrm{d}x \\
&= \int_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}\setminus\{f(x)\}} \mathrm{P}_{X,Y}(x,y) \,\mathrm{d}x \\
&= \int_{x\in\mathcal{X}} \mathrm{P}_X(x) \sum_{y\in\mathcal{Y}\setminus\{f(x)\}} \mathrm{P}_{Y|X}(y \,|\, x) \,\mathrm{d}x \\
&= \int_{x\in\mathcal{X}} \mathrm{P}_X(x) \left(1 - \mathrm{P}_{Y|X}\left(f(x) \,\middle|\, x\right)\right) \mathrm{d}x \\
&\overset{\text{(Lem. 3.12)}}{\leq} \frac{1}{2} \int_{x\in\mathcal{X}} \mathrm{P}_X(x) H(Y \,|\, X = x) \,\mathrm{d}x \\
&= \frac{1}{2} H(Y|X).
\end{aligned}$$

□

Remark 3.14 (Fano For Binary Classification)**.**
The results presented in Corollary 3.13 evaluate the relationship between mutual information and misclassification risk as proposed in Proposal 0.2. For binary classification with uniformly distributed prior distributions P_Y we obtain

$$H(Y) = \log_2 |\mathcal{Y}| = 1.$$

Thus, Fano's inequality yields the same bound on mutual information between data and labels,

$$H(Y) - I(X;Y) - 1 = -I(X;Y) \leq \mathrm{P}(Y \neq \hat{Y}), \tag{3.10}$$

and on mutual information between deep representations at a layer l and the labels,

$$H\left(Y\right) - I\left(\Phi_a^l\left(X;\theta\right);Y\right) - 1 = -I\left(\Phi_a^l\left(X;\theta\right);Y\right) \leq \mathrm{P}(Y \neq \hat{Y}). \tag{3.11}$$

Thus, we cannot improve misclassification bound by Fano's inequality when considering deep representations instead of initial data. What is more, non-negativity of mutual

information and probability implies that both Eq. (3.10) and Eq. (3.11) are true statements for all X, Y and Φ_a anyways, no matter the argument of the mutual information or the probability distribution.

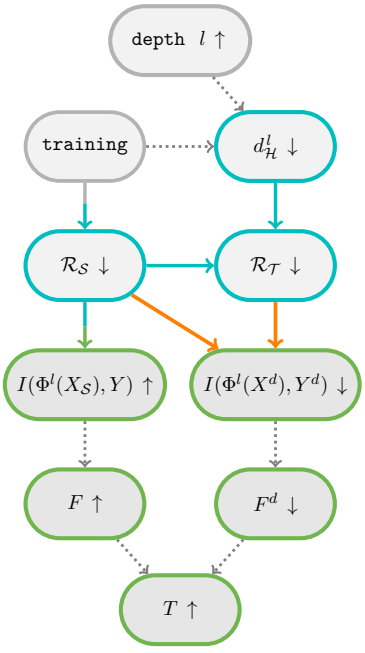
Summarizing, Fano's inequality undoubtedly shows a connection between classification performance and mutual information of data representation and labels. When data or representation yield high values of $H(Y|X)$ and $I(X;Y)$ or $H(Y|\Phi_a^l(X;\theta))$ and $I(\Phi_a^l(X;\theta);Y)$, the misclassification risk cannot be arbitrary small. However, the inequality is vacuous when considering binary classification and it loses tightness in general, when $\mathcal{Y}|$ is small. Further, we can see that the bound on mutual information is not improved when considering representations instead of data. This is because $Y \to X \to \Phi_a^l(X;\theta)$ implies that $I(X;Y) \geq I\left(\Phi_a^l(X;\theta);Y\right)$ it holds

$$H(Y) - I(X;Y) - 1 \leq H(Y) - I\left(\Phi_a^l(X;\theta);Y\right) - 1 \leq \mathrm{P}(Y \neq \hat{Y})$$

by the data-processing inequality (Lemma 3.9). Rather we can say, that if a deep classifier has small risk, the choice of representation cannot be too bad, i.e. decreasing the mutual information in comparison to the initial problem too much. The tightness is more an evaluation of the problem setting and the data, then the Neural Network itself. This

3.2.2. Information Theory and Domain Adaptation

The next step is to discuss Proposal 0.2 (b) (ii). Intuitively the claim projects an aura of rationality and coherence. Suppose a classifier hypothesis performs well on two domains. In this case, these should not differ significantly and thus we might come to the conclusion that we cannot differ the two domains well, i.e. the mutual information between the representation of a data point and the distribution it originates from should be low. The idea seems similar to the domain shift generalization theorem we discussed previously (cf. Theorem 1.25). However, we will shortly illustrate that this connection does not hold in general.



Consider the shared distribution of data from both domains in the random variable X^d, given the domain labels Y^d.

Definition 3.15 (Cross-Domain Data)**.**
Given two domains $\mathcal{S}$ and $\mathcal{T}$ on $\mathcal{X}$ we define **cross-domain classification data** as the pair of random variables (X^d, Y^d) on $\mathcal{X} \times \{0, 1\}$. The distributions are defined by

$$\mathrm{P}_{X^d,Y^d}\left(x^d,\, y^d\right) = \mathrm{P}_{Y^d}\left(y^d\right) \mathrm{P}_{X^d|Y^d}\left(x^d \,\middle|\, y^d\right) \coloneqq \mathrm{P}_{Y^d}\left(y^d\right) \cdot \begin{cases} \mathrm{P}_{\mathcal{S}}\left(x^d\right) \text{, if } y^d = 0 \\ \mathrm{P}_{\mathcal{T}}\left(x^d\right) \text{, if } y^d = 1 \end{cases}$$

and

$$\mathrm{P}_{Y^d}\left(y^d = 0\right) = \mathrm{P}_{Y^d}\left(y^d = 1\right) = \frac{1}{2}.$$

Thus, y^d is simply an indicator for the domain considered, source $\mathcal{S}$ or target $\mathcal{T}$. If we consider labeled classification data on either domain, we include Y as class label as in the previous sections.

Regarding the proposal we will present an example to show that the mutual information between data X^d and its domain of origin Y^d is no predictor for classification risk.

Example 3.16 (Proposal (b) (ii))**.**
As previously, assume the label outcomes of both Y and Y^d are equiprobable, i.e. both are uniformly distributed. Using the independence of Y and Y^d, we can express mutual independence between cross-domain data X^d and domain label Y^d given the class Y in terms of the four different data distributions $\mathrm{P}\left(X^d \,\middle|\, y^d,\, y\right)$.
Note that, for the sake of readability, we shorten

$$\text{“} \int\limits_{x^d \in \mathcal{X}} \sum_{y^d \in \mathcal{Y}^d} \sum_{y \in \mathcal{Y}} \cdots \mathrm{d}x\text{” to “} \iiint\limits_{\mathcal{X},\, \mathcal{Y}^d,\, \mathcal{Y}} \cdots \text{”}.$$

Then rewrite the mutual information in the following way.

$$I(X^d, Y^d|Y) = \mathbb{E}_{\mathrm{P}_{X^d,Y^d,Y}}\left[\log_2 \frac{\mathrm{P}_{X^d,Y^d|Y}(X^d, Y^d|\, Y)}{\mathrm{P}_{X^d|Y}(X^d|\, Y)\, \mathrm{P}_{Y^d|Y}\left(Y^d \,\middle|\, Y\right)}\right] \tag{3.12}$$

$$= \iiint\limits_{\mathcal{X},\mathcal{Y}^d,\mathcal{Y}} \mathrm{P}_{X^d,Y^d,Y}\left(x^d, y^d, y\right) \log_2 \frac{\mathrm{P}_{X^d,Y^d|Y}\left(x, y^d \,\middle|\, y\right)}{\mathrm{P}_{X^d|Y}\left(x^d \,\middle|\, y\right)\ \mathrm{P}_{Y^d|Y}\left(y^d \,\middle|\, y\right)}$$

$$= \iiint\limits_{\mathcal{X},\mathcal{Y}^d,\mathcal{Y}} \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right) \mathrm{P}_{Y^d}\left(y^d\right) \mathrm{P}_Y\left(y\right) \log_2 \frac{\mathrm{P}_{X^d|Y^d|Y}\left(x^d \,\middle|\, y^d, y\right)}{\mathrm{P}_{X^d|Y}\left(x^d \,\middle|\, y\right)}$$

$$= \frac{1}{4} \iiint\limits_{\mathcal{X},\mathcal{Y}^d,\mathcal{Y}} \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right) \left[-\log_2 \frac{\mathrm{P}_{X^d|Y}(x^d|\, y)}{\mathrm{P}_{X^d|Y^d,Y}(x^d|\, y^d, y)}\right]$$

$$= \frac{1}{4} \iiint\limits_{\mathcal{X},\mathcal{Y}^d,\mathcal{Y}} \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right) \left[-\log_2 \left(\frac{1}{2} \frac{\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right) + \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, 1 - y^d, y\right)}{\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right)}\right)\right]$$

The last equality uses that for random variables X and Y with the latter being discrete, it holds $\mathrm{P}(x) = \sum_{y\in\mathcal{Y}} \mathrm{P}(x,y) = \sum_{y\in\mathcal{Y}} \mathrm{P}(x|y)\mathrm{P}(y)$.
The rewriting of $I(X^d, Y^d|Y)$ enables us to take a look at two extreme cases.

- **First**, if for a given class y data from both domains share the **same distribution**, i.e. $\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d = 0, y\right) = \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d = 1, y\right)$ for all x^d and y, the argument of the logarithm is constant and equal to 1 and therefore, the mutual information is 0.

- The **second** border case considers densities that have **approximately disjunct support**, i.e. for each $x^d \in \mathcal{X}$ at least one of the probabilities $\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right)$ or $\mathrm{P}_{X|Y^d,Y}\left(x^d \,\middle|\, 1-y^d, y\right)$ is **not** significantly larger than zero. Thus, the argument of the integral in the last line of Eq. (3.12) is either approximately 0 because $\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right)$ is already approximately equal to 0 or it is approximately equal $\mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right)\left[-\log_2 \frac{1}{2}\right]$. Therefore, it holds

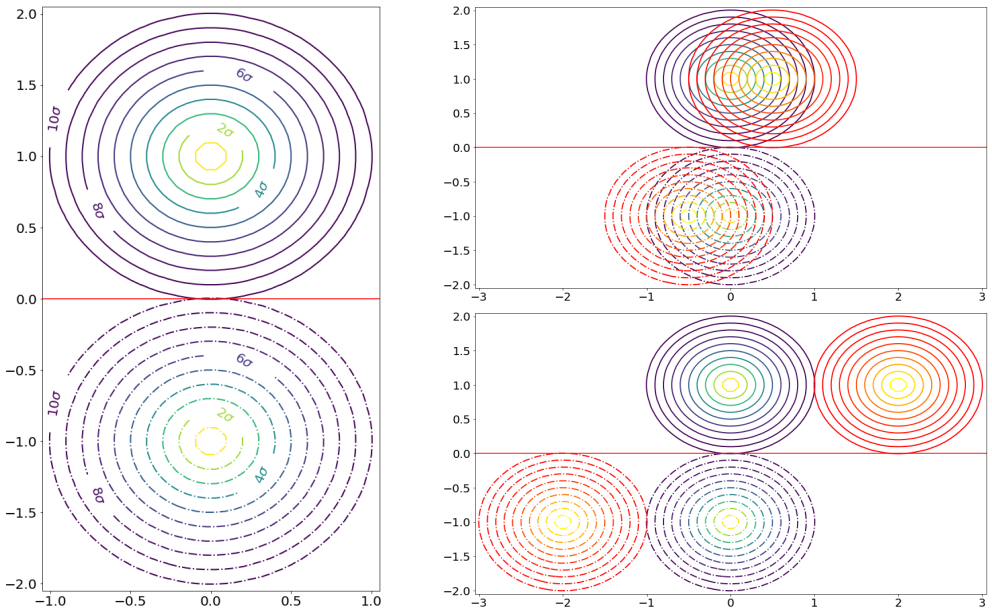
$$\begin{aligned} I(X^d, Y^d \,|\, Y) &\approx \frac{1}{4} \iiint_{\mathcal{X},\mathcal{Y}^d,\mathcal{Y}} \mathrm{P}_{X^d|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right)\left[-\log_2 \frac{1}{2}\right] \\ &= \frac{1}{4} \sum_{y^d\in\mathcal{Y}^d} \sum_{y\in\mathcal{Y}} \int_{x^d\in\mathcal{X}} \mathrm{P}_{X|Y^d,Y}\left(x^d \,\middle|\, y^d, y\right) \mathrm{d}x \\ &= \frac{1}{4} \sum_{y^d\in\mathcal{Y}^d} \sum_{y\in\mathcal{Y}} 1 = 1. \end{aligned}$$

We use an interpolation between these two extreme cases to show that simply observing $I(X^d, Y^d \,|\, Y)$ yields no information about misclassification risk on the domains. For this purpose, we present a simple construction with perfect classification on both domains.

Let the densities of source and target domain for each label be multivariate Gaussian distributions in $\mathbb{R}^2$ where all covariance matrices are equal to a tenth of the identity matrix. This exact value for the covariance's diagonal entries is chosen arbitrarily and solely for visualization purposes.

$$\Sigma \coloneqq \frac{1}{10}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The mean values are set to $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} x_1 \\ 1 \end{pmatrix}$ for the data with label $y = 0$ from source and target domain respectively. The mean values for the two distributions with label $y = 1$ are set to $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} -x_1 \\ -1 \end{pmatrix}$.



(a) Two classes of the source domain. In the first discussed extreme case in Example 3.16 the target domain is equal to the source domain.

(b) Domain shift parallel to the x-axis as in the construction in Example 3.16. **Top:** A smaller shift with a decent overlap of the distributions of the classes. **Bottom:** Distributions of source and target domain do not overlap apart from very improbable realizations.

Figure 7: Visualization of linear classification on normally distributed class data. The circles' centers are the respective class means and each contour line displays a distance of a multiple of σ to the mean. The red line indicates the respective classifying hyperplane. Solid and dashed contour lines discriminate the two classes, and colormap discriminates the domains. With a probability of at least 99% data will be sampled in a 10σ neighborhood of the means. The plots illustrate different levels of domain shifting without changing the classification performance of the classifier.

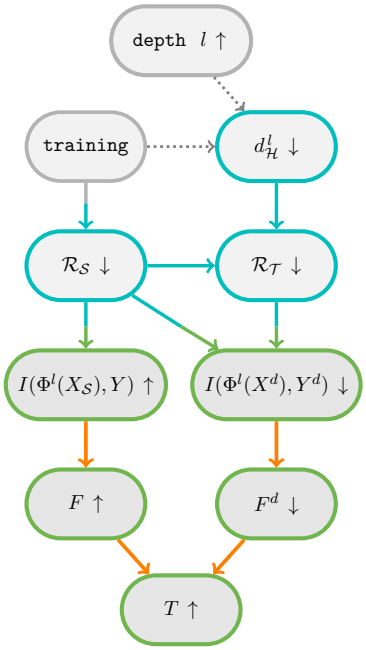
This choice guarantees that the x-axis as linear hyperplane yields almost zero loss classification on both domains and for all x_1: Chebichev's inequality [Appendix A, Theorem A.1] for $\Sigma = \sigma I$ yields

$$\mathrm{P}\left(\|X - \mu\| \geq k\sigma\right) \leq \frac{1}{k^2}.$$

Therefore, the choice $\sigma = 0.1$ implies that X is in a $10\sigma = 1$ neighborhood of μ with a probability of at least 99%. Fig. 7 illustrates the cases investigated above for this exact example, i.e. one with maximal mutual information, one with minimal and one case in between. The misclassification risk is equal for all cases using one and the same linear classifier, while the mutual information varies significantly. This makes it easier to understand that misclassification risk is no valid predictor for mutual information and this part of the proposal can be falsified for general distributions.

3.3. Proposed Information-Theoretical Optimization

The final part of the proposal regards concentrating the two paths we already investigated, source risk minimization and Domain Adaptation, into one information-theoretical measure. The hypothesis intended to link these two problems to the results of Section 3.2 and if this had worked out well, mutual information would have been the tool to unite them again. We will try to work around these problems.
The proposed merging of the two paths is based on the concept of Information Bottleneck and an extension thereof, so-called fidelity (Section 3.3.2). These two quantities have the purpose of rewriting our two Machine Learning problems as one information-theoretical optimization on the representation. We begin by reviewing the concept of Information Bottleneck.



3.3.1. Information Bottleneck for Neural Networks

Proposal 0.2 relates misclassification risk to mutual information. The idea of connecting the good performance of Neural Networks to measuring the flow of information through layers is no novel one. An impactful entry to the literature in this context is categorized under the overarching term **Information Bottleneck** (IB). Introduced by Tishby et al. in [TPB00] and later concisely investigated by Schwartz-Ziv and Tishby in [ST17], IB is a framework to analyze or even train Neural Networks with Information Theory in mind. It is a quantity to investigate Neural Network layer representations, trading off low complexity of the representations and sufficiency for the required classification task. The authors claim the best Neural Network representations preserve all mutual information between data and label while minimizing excess information contained in the input representation X. Let $a = (N_0, \dots, N_L, \sigma)$ be a Neural Network architecture and L be its depth. Then the idea of IB can be formulated as finding the optimal parametrization or representation θ^* in

$$\theta^* = \underset{\theta\colon I(X;\,Y)=I(\Phi_a^{L-1}(X;\theta);\,Y)}{\operatorname{argmin}} I(X;\,\Phi_a^{L-1}(X;\theta)).$$

Due to DPI (Lemma 3.9), we know

$$Y \to X \to \Phi_a^L(X;\theta) \implies I(X;\,Y) \geq I(\Phi_a^{L-1}(X;\theta);\,Y)$$

for any parameterization θ. Thus the condition $I(X;\ Y) = I(\Phi_a^{L-1}(X;\theta);\ Y)$ enforces a representation with maximum information content about the labels.
Finding such representations, i.e. network parameterizations, is a difficult problem and thus the idea is to relax this optimization problem to

$$\theta^* = \operatorname*{argmin}_\theta\ I(X;\ \Phi_a^{L-1}(X;\theta)) - \beta I(\Phi_a^{L-1}(X;\theta);\ Y)$$

for some relaxation parameter $\beta > 0$.

Definition 3.17 (Information Bottleneck; [TPB00])**.**
For a Neural Network Φ_a of depth L, data X and label Y, the target expression

$$\mathcal{L}(\theta) = I(X;\ \Phi_a^{L-1}(X;\theta)) - \beta I(\Phi_a^{L-1}(X;\theta);\ Y)$$

is called the **Information Bottleneck (lagrangian)** of the representation $\Phi_a^{L-1}(X;\theta)$ in layer $L-1$. The Information Bottleneck seeks a balance between good prediction (DPI) and compression, i.e. ignoring redundant and unnecessary information, reducing the internal dimensions of the representation, and focusing on features useful for prediction.

Schwartz-Ziv and Tibshy conclude from a series of experiments, that the success of Neural Networks could be understood by investigating IB. In [ST17], they perform experiments training Neural Networks purely for classification, without optimizing IB directly or by regularization, and observe interesting behavior of the IB lagrangian. We discuss the results in the following.
They illustrate the behavior of $I(\Phi_a^{L-1}(X;\theta);\ Y)$ and $I(X;\ \Phi_a^{L-1}(X;\theta))$ graphically and distinguish two stages during the training of classification on a labeled sample, i.e. fitting θ according to the connection of data and labels. First, training is only concerned with fitting labels, both $I(\Phi_a^{L-1}(X;\theta);Y)$ and $I(X;\ \Phi_a^{L-1}(X;\theta))$ increase and the training misclassification error decreases. When the misclassification error reaches saturation and barely decreases anymore, according to their observation, further training seems to enter a compression phase. $I(\Phi_a^{L-1}(X;\theta);\ Y)$ barely changes anymore, while $I(X;\ \Phi_a^{L-1}(X;\theta))$ decreases significantly. The authors' explanatory guess regards the gradients during the Stochastic Gradient Descent (SGD) optimization steps.
In the first phase, the gradients have rather big means and small standard deviations, speaking about absolute values. This phase is called *drift phase*. The optimization steps are majorly guided by the impact of several misclassifications.
In the second phase, they notice a significant change in the gradients. The means are smaller, while the standard deviations are comparably large. The authors call this the *diffusion phase* since the gradients have high stochasticity and the optimization therefore almost behaves like a random search finding small improvements. For more details on these two-phased training phenomenons refer to [ST17, Section 3.4 and 3.5].
To summarize, the authors claim that during SGD optimization of Neural Network, a representation is found that, measured on IB value, is on the one hand well trained on the task and has a low empirical error while on the other hand is efficient due to compression and diffusion and thus, is not overfit to the training data and generalizes well.

The observations on a two-phased optimization in the information plane *can also be empirically reproduced* for example on a state-of-the-art ResNet architecture. Such reproduction was published by L.N. Darlow and A. Storkey, in [DS19].
Another Information Bottleneck related idea is proposed by A. Achille and S. Soatto in [AS17]. Instead of representation complexity measured in terms of the representation, they propose to consider the mutual information between data and the network weights. In this framework, it can be shown that minimal mutual information between data and weights leads to Neural Networks that are invariant to task-irrelevant perturbations in data, like a domain shift by adding noise.
Information Bottleneck as a concept seems like a very useful approach to investigating Neural Networks. However, in literature, there is no universal agreement that it is a valid general explanation for the success of classification and it seems only to hold empirically.

Remark 3.18 ([SBD+18])**.**
In the research community, reasonable doubt on the Information Bottleneck framework is expressed. Saxe et al. ([SBD+18]) construct cases to show, that the claims we presented above and from the same source, [ST17], do not hold in general. Thus IB on its own cannot be used as sound and fail-safe argumentation for the success of deep Neural Networks.
First of all, the authors critique the claim, that optimization can be observed as a two-phased process. They suspect the cause for entering a compression phase is a double-sided saturated activation function and show, that if activation is not saturated on one side, e.g. with commonly used ReLU, optimization might never enter a compression phase (cf. [SBD+18, Section 2]).
Furthermore, they show that notable compression in information space and overfitting are not exclusive from each other. Thus, generalization is not fully explainable by the observation of two-phased optimization (cf. [SBD+18, Section 3]).
Still, they agree with the original work [ST17] that ignoring information irrelevant to the classification task should aid generalization as the focus only lies on distinct class features. However, they present an artificial example, where a "$I(X;\ \Phi_a^{L-1}(X;\theta))$ to $I(\Phi_a^{L-1}(X;\theta);\ Y)$"-plot provides an insufficiently differentiated view on this effect. They show that the value of $I(X;\ \Phi_a^{L-1}(X;\theta))$ can strictly increase across training epochs while simultaneously considering only task-irrelevant inputs or features collected in $\hat{X}$, $I(\hat{X};\ \Phi_a^{L-1}(\hat{X};\theta))$ traverses the two phases proposed in the Information Bottleneck framework (cf. [SBD+18, Section 5]).
Saxe et al. conclude by acknowledging the usefulness of IB to investigate Neural Networks if developed further and recognize that IB and especially its Lagrangian can be used to improve the process of training Neural Networks, e.g. as regularizer of the objective function.

3.3.2. Fidelity and Target Representation

Having introduced IB and its relevance, we can talk about one of its extensions called *fidelity* which is used in Proposal 0.2. The goal is to rewrite our classification and Domain Adaptation problems as a second, information-theoretic, optimization strung close to Information Bottleneck.
The Information Bottleneck Lagrangian itself can be accepted to be a quantity one wants to minimize to improve a Neural Network's performance. Previous claims state if training in one problem scenario, defined by data, labels and classifier architecture, yields a decrease in Information Bottleneck value, we end up with a better classifier. Nevertheless, we cannot compare different problems to each other since the Information Bottlenecks Lagrangian's values do not relate. Furthermore, the definition of fidelity addresses the last critique of [SBD+18]. Instead of evaluating $I(X;\ \Phi_a^{L-1}(X;\theta))$, they only consider excess information beyond task-relevant features by punishing $I(X;\ \Phi_a^{L-1}(X;\theta)\,|\,Y)$.
This motivates normalizing IB in some way, eventually yielding the definition of fidelity, provided by Noshad et al. in [NCS+21].

Definition 3.19 (Representation Fidelity; [NCS+21])**.**
Let $\beta > 0$. Given a Neural Network Φ_a, the **fidelity** of a representation $\Phi_a^{L-1}(X;\theta)$ of data X with label Y is defined as

$$F(\theta) = \frac{I(\Phi_a^{L-1}(X;\theta);\ Y) - \beta I(X;\ \Phi_a^{L-1}(X;\theta)\,|\,Y)}{I(X;Y)}.$$

Remark 3.20.
Note that the order of terms in the numerator is changed in comparison to Information Bottleneck. While directly extending the idea of IB, Noshad et al. did introduce fidelity as a property to be maximized. This is justified by an elegantly simple optimization bound.

The following Lemma proves that in contrast to IB, fidelity has a clear and achievable optimization bound.

Lemma 3.21 ([NCS+21])**.**
For data X and label Y and a Neural Network representation of X, $\Phi_a(X;\theta)$, it holds

$$F(\theta) \leq 1.$$

Equality holds if and only if

$$I(X;\ \Phi_a^{L-1}(X;\theta)\,|\,Y) = I(X;Y\,|\,\Phi_a^{L-1}(X;\theta)) = 0.$$

Proof. The inequality is a direct consequence of chain rule and conditional independence in Markov chains. The chain rule allows us to rewrite $I(\Phi_a^{L-1}(X;\theta), X;\ Y)$ in two ways.

$$\begin{aligned} I(\Phi_a^{L-1}(X;\theta), X;\ Y) &= I(X;Y) + I(\Phi_a^{L-1}(X;\theta);\ Y\,|\,X) \\ &= I(\Phi_a^{L-1}(X;\theta);\ Y) + I(X;\ Y\,|\,\Phi_a^{L-1}(X;\theta)) \end{aligned}$$

Due to Lemma 3.7, $Y \to X \to \Phi_a^{L-1}(X;\theta)$ implies $I(\Phi_a^{L-1}(X;\theta);\ Y \,|\, X) = 0$.
Thus,

$$\begin{aligned} I(X;\ Y) &= I(\Phi_a^{L-1}(X;\theta);\ Y) + I(X;\ Y \,|\, \Phi_a^{L-1}(X;\theta)) \\ &\geq I(\Phi_a^{L-1}(X;\theta);\ Y) \\ &\geq I(\Phi_a^{L-1}(X;\theta);\ Y) - \beta I(X;\ \Phi_a^{L-1}(X;\theta) \,|\, Y) \end{aligned} \tag{3.13}$$

which is equivalent to

$$\frac{I(\Phi_a^{L-1}(X;\theta);\ Y) - \beta I(X;\ \Phi_a^{L-1}(X;\theta) \,|\, Y)}{I(X;\ Y)} \leq 1.$$

Given the inequalities in Eq. (3.13) the claim on equality is immediate. □

Now we can merge the two paths of the Domain Adaptation discussion, low source risk and good domain shift generalization, in Proposal 0.2 into one optimization to find an, in an information-theoretic manner, optimal representation.

Definition 3.22 (Target Parametrization)**.**
Consider source and target domains $\mathcal{S}$ and $\mathcal{T}$ and a Neural Network architecture a. Let F denote the fidelity of source data $(X,Y) \sim \mathrm{P}_{\mathcal{S},\mathcal{Y}}$. Similarly let F^d denote fidelity of cross-domain data (X^d, Y^d). Then, we call θ_T a **target parametrization** if

$$\theta_T \in \operatorname*{argmax}_\theta \Big(F(\theta) - F^d(\theta) \Big)$$

and we call the corresponding value

$$T \coloneqq F(\theta) - F^d(\theta)$$

the information-theoretical **target value**.

It remains to investigate the connections between mutual information, fidelity, and the target value T provided by Proposal 0.2.
First of all, it is claimed that *mutual information increase implies fidelity increase* and analog for decreasing values. As the following Lemma suggests, this claim does not cover the bigger picture.

Lemma 3.23.
For Data X and Label Y and a Neural Network Φ_a it holds that optimizing $F(\theta)$ is only dependent on the value

$$(1+\beta) I(\Phi_a^{L-1}(X;\theta), Y) - \beta H(\Phi_a^{L-1}(X;\theta)).$$

Proof. The denominator $I(X;Y)$ is constant in $\Phi_a^{L-1}(X;\theta)$. Thus fidelity is proportional to the numerator:

$$F(\theta) \propto I(\Phi_a^{L-1}(X;\theta);\ Y) - \beta I(X;\ \Phi_a^{L-1}(X;\theta) \,|\, Y)$$

Further, we know that

$$I(\Phi_a^{L-1}(X;\theta);Y) = H(\Phi_a^{L-1}(X;\theta)) - H(\Phi_a^{L-1}(X;\theta)\,|\,Y).$$

Additionally, since $\Phi_a^{L-1}(X;\theta)$ is a deterministic function in X, we know

$$\begin{aligned} I(X;\Phi_a^{L-1}(X;\theta)\,|\,Y) &= H(\Phi_a^{L-1}(X;\theta)\,|\,Y) - H(\Phi_a^{L-1}(X;\theta)\,|\,X,Y) \\ &= H(\Phi_a^{L-1}(X;\theta)\,|\,Y). \end{aligned}$$

Putting these realizations together, we obtain

$$\begin{aligned} &(1+\beta)I(\Phi_a^{L-1}(X;\theta);Y) - \beta H(\Phi_a^{L-1}(X;\theta)) \\ &\quad= (1+\beta)\left(H(\Phi_a^{L-1}(X;\theta) - H(\Phi_a^{L-1}(X;\theta)\,|\,Y)\right) - \beta H(\Phi_a^{L-1}(X;\theta)) \\ &\quad= H(\Phi_a^{L-1}(X;\theta)) - H(\Phi_a^{L-1}(X;\theta)\,|\,Y) - \beta H(\Phi_a^{L-1}(X;\theta)\,|\,Y) \\ &\quad= I(\Phi_a^{L-1}(X;\theta);\ Y) - \beta I(X;\ \Phi_a^{L-1}(X;\theta)\,|\,Y). \end{aligned}$$

□

Lemma 3.23 implies that the connections between mutual information and fidelity are not as simple as the proposal graph proposed. Fidelity $F(\theta)$ is not only dependent on $I(\Phi_a^{L-1}(X;\theta),Y) = H(\Phi_a^{L-1}(X;\theta)) - H(\Phi_a^{L-1}(X;\theta)\,|\,Y)$ but also an additional weighted $-H(\Phi_a^{L-1}(X;\theta))$ term. Thus, we cannot confidently claim that fidelity increases if the mutual information $I(\Phi_a^{L-1}(X;\theta),Y)$ does. Rather, it is the case that fidelity increases if the data entropy in general is high and at the same time the difference between data entropy and class entropy is big. Mutual information is only really dependent on the second part. Fidelity additionally rewards very spread out high entropic data when the class is unknown, independent of the relation to class spread as in the mutual information.

Another idea that can be investigated in the context of the proposal is to directly look at the target from Definition 3.22 instead of detouring to the exact properties of fidelity. It could be speculated that by a smart choice of the two different β_1 and β_2 in

$$\begin{aligned} F(\theta) - F^d(\theta) = &\frac{(1+\beta_1)I(\Phi_a^{L-1}(X;\theta),Y) - \beta_1 H(\Phi_a^{L-1}(X;\theta))}{I(X;Y)} \\ &- \frac{(1+\beta_2)I(\Phi_a^{L-1}(X^d;\theta),Y^d) - \beta_2 H(\Phi_a^{L-1}(X^d;\theta))}{I(X^d;Y^d)} \end{aligned}$$

the two overlapping entropy terms might cancel out. This, however, is a subtle flaw that has grown from a small calculation error, when defining the information-theoretical target representation. Since we optimize over the space of possible representations, $H(\Phi_a^{L-1}(X;\theta))$ and $H(\Phi_a^{L-1}(X^d;\theta))$ are changed during the optimization.
A short inspection of both entropy terms conveniently displays this problem. Even for the simple case of $\Phi_a^{L-1}(\cdot;\theta)$ being the identity, they are undeniably different and their

exact relation might not be easily describable. Let $X_{\mathcal{S}}$ and $X_{\mathcal{T}}$ denote domain data and X^d the corresponding cross-domain data. Then,

$$H(X_{\mathcal{S}}) = \int_{x^d \in \mathcal{X}} \mathrm{P}_{\mathcal{S}}(x) \log_2 \left(\frac{1}{\mathrm{P}_{\mathcal{S}}(x)} \right) \mathrm{d}x, \tag{3.14}$$

$$\begin{aligned} H(X^d) &= \int_{x^d \in \mathcal{X}} \sum_{y^d=0}^{1} \mathrm{P}(x^d,\, y^d) \log_2 \left(\frac{1}{\mathrm{P}_{X^d,Y^d}(x^d, Y^d = 1) + \mathrm{P}_{X^d,Y^d}(x^d, Y^d = 0)} \right) \mathrm{d}x \\ &\geq \frac{1}{2} \left(H(X_{\mathcal{S}}) + H(X_{\mathcal{T}}) \right). \end{aligned} \tag{3.15}$$

The inequality holds by the concavity of entropy. Thus, canceling the entropies out with the purpose of making direct statements about our two prominent mutual information values is not immediate. Including the distributions induced by different representations makes the entropy values and their relation even more unpredictable. Nevertheless, this does not mean that the idea of using fidelity is in vain.
Even if the proposed direct connection between $T = F(\theta) - F^d(\theta)$ and the two values $I(\Phi_a^{L-1}(X;\theta), Y)$ and $I(\Phi_a^{L-1}(X^d;\theta), Y^d)$ does not bear fruit, our target representation optimization problem can be viewed as a heuristic which might help Neural Networks generalize better, similarly to Information Bottleneck, if we regularize general training or solely train with respect to fidelity as objective.
From our investigations above, we know that maximizing $F(\theta)$ means searching a representation where $H(\Phi_a^{L-1}(X;\theta))$ is large in comparison to $(1+\beta)H(\Phi_a^{L-1}(X;\theta) \,|\, Y)$. This means the total data itself is spread out and its distribution is unpredictable, while the distribution of data points of each class, i.e. conditioned on knowledge about the labels, is rather predictable. This implies that increasing fidelity induces a representation where the data of different classes is easily separable, i.e. where classification is a simple task. Similarly, minimizing cross-domain fidelity means minimizing

$$H(\Phi_a^{L-1}(X^d;\theta)) - (1+\beta)H(\Phi_a^{L-1}(X^d;\theta) \,|\, Y^d)$$

(cf. Lemma 3.23). Since $H(\Phi_a^{L-1}(X^d;\theta)) - H(\Phi_a^{L-1}(X^d;\theta) \,|\, Y^d) \geq 0$, this implies that minimizing fidelity is about maximizing $\beta H(\Phi_a^{L-1}(X^d;\theta) \,|\, Y^d)$ and keeping $H(\Phi_a^{L-1}(X^d;\theta)) - H(\Phi_a^{L-1}(X^d;\theta) \,|\, Y^d)$ close to zero. A desired representation is as high-entropic as possible and at the same time is not significantly less random if we condition on labels. Thus, the representation behaves as unpredictably as possible. No classifier layer should perform well on such data.

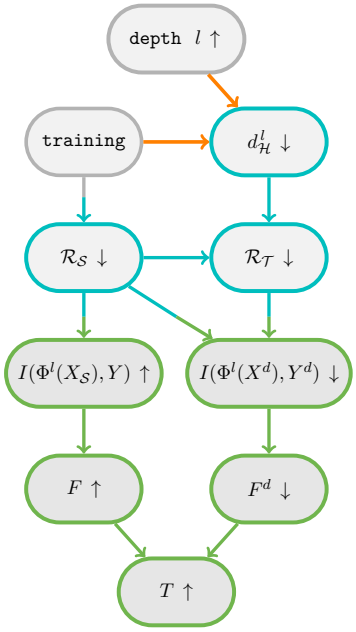
Conclusion of Section 3 The discrepancy between theory and practice displays an old problem showing that the conceptional validity of ideas does not guarantee their applicability in real-world settings. The proposal tried to tie the practical success of Neural Networks to Information Theory. We have shown for one thing, that the proposed ideas are mostly sound and seem to hold in practice. Nevertheless, Remark 3.18 showed possible problems with the IB framework and thereby also on fidelity. Further, regarding

the proposed connection between Machine Learning and Information Theory using an interplay of risks and mutual information, both Remark 3.14 and Example 3.16 presented problems and counterexamples for Proposal 0.2 (b) (i) and (ii). The process of identifying a framework with additional assumptions guaranteeing good prediction of risk using mutual information or similar information-theoretic quantities could require years of dedicated research.

4. Experiments on Deep Representations

"Discovery requires experimentation."
-Werner Reinhardt[12]

As hinted at multiple times it remains to look into the representations and parametrizations of trained Neural Networks. This is a possibility to enable the application of the Machine Learning results discussed in Section 1 and Section 2 to deep classifiers. Additionally, we have not discussed the claim of Proposal 0.2 **(a)** **(ii)** yet. The claim contextualizes a statement of Sarishvili's Hypothesis 0.1 that deep Neural Networks might find representations minimizing the domain shift and thereby already achieve Domain Adaptation invariance through regular training alone. Mathematically this is expressed in a decrease of $\mathcal{H}$-divergence between domain representations in deep layers, called deep $\mathcal{H}$-divergence $d^l_{\mathcal{H}}$.



4.1. Additional Measurements for the Experiments

The claim is concerned with the blind Domain Adaptation capacity of deep Neural Networks and whether they can translate their source domain performance to data from another closely related target domain without additional architecture extending Definition 2.2 or special regularization in the loss. Remember blind Domain Adaptation means that training uses only data points from the source domain (cf. Remark 1.10). In contrast, most state-of-the-art algorithms do unsupervised Domain Adaptation, i.e. additionally to labeled source data, they use unlabeled target domain data during training. To facilitate a comparison of blind and unsupervised Domain Adaptation during the experiments, we present a well-known unsupervised Domain Adaptation algorithm in Section 4.1.1. Thereafter, we present a possible explanation for the Domain Adaptation invariance in form of "manifold disentanglement" in Section 4.1.2. This approach to understanding representations of Neural Networks is then empirically evaluated in the experiments.

[12]Fictional character, Marvel's Agents of S.H.I.E.L.D.

4.1.1. Domain-Adversarial Training of Neural Networks

Remember, our result on Domain Adaptation performance, Theorem 1.25, showed that the risk gap due to domain shift could be estimated by the discriminatory abilities measured by $d_{\mathcal{H}}$. When applying Theorem 1.25 to the linear classification at the last layer of a Neural Network only, Domain Adaptation is concerned with finding representations such that the **deep $\mathcal{H}$-divergence** (cf. Definition 1.24) of the last hidden layer's deep representation in the trained network $\Phi_a(\,\cdot\,;\theta)$

$$d_{\mathcal{H}}^{L-1}(\mathrm{P}_{\mathcal{S}}, \mathrm{P}_{\mathcal{T}}) \coloneqq \sup_{h\in\mathcal{H}} \left| \mathrm{P}_{\mathcal{S}}\big(h(\Phi_a^{L-1}(X;\theta)) = 1\big) - \mathrm{P}_{\mathcal{T}}\big(h(\Phi_a^{L-1}(X;\theta)) = 1\big) \right|$$

or its empirical equivalent on an according sample $(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n)$

$$\overline{d_{\mathcal{H}}^{L-1}}(x_{\mathcal{S}}^n, x_{\mathcal{T}}^n) \coloneqq 1 - \inf_{h\in\mathcal{H}} \frac{1}{n}\left(\sum_{i=1}^{n} \mathbb{1}\{h(\Phi_a^{L-1}(x_{\mathcal{S},i};\theta)) = 1\} + \sum_{i=1}^{n} \mathbb{1}\{h(\Phi_a^{L-1}(x_{\mathcal{T},i};\theta)) = 0\} \right)$$

are small for linear classifiers $\mathcal{H}$. Define deep $\mathcal{H}$-divergence for other layers than $L-1$ equivalently. Note that from the three equivalent definitions of empirical $\mathcal{H}$-divergence (cf. Remark 1.26), we chose this one because its value can be easily calculated by training a domain classifier from $\mathcal{H}$ to distinguish source and target domain representations of points from the sample. Then $\overline{d_{\mathcal{H}}^{l}}$ is calculated by measuring the empirical error of this trained domain classifier on the deep representations of the respective training sample.

Typically, representations are aligned directly in modern Domain Adaptation algorithms, using unsupervised training, i.e. using labeled source and unlabeled target data, and special Domain Adaptation heuristics. These can be special classifier architectures of the network, regularization terms in the loss function or a special form of training.
One famous algorithm tries as a second optimization goal, besides good classification on the source domain, to minimize deep $\mathcal{H}$-divergence during training directly. This algorithm is called *domain-adversarial training of Neural Networks* commonly referred to as DANN and was provided by Ganin et al. in [GUA+16]. When we compare representations and performance of blind and unsupervised Domain Adaptation later, we want to use a DANN architecture as a representative for the latter. Therefore, we introduce it shortly:

Example 4.1 (DANN; [GUA+16])**.**
Domain-adversarial training of Neural Networks is an unsupervised Domain Adaptation approach (cf. Remark 1.10). A training sample is required to consist of labeled source domain data, unlabeled target domain data, and a label to identify the domain:[13]

$$\mathrm{S}^{2n} = \left(\Big(x_{\mathcal{S}},\, y_{\mathcal{S}},\, y^d = 0\Big)^n,\, \Big(x_{\mathcal{T}}, y^d = 1\Big)^n \right)$$

The DANN classifier behaves similarly to a fully connected Neural Network as we introduced them in Definition 2.2, but extends the concept by adding a layer that is

[13] We denote the sample size as $2n$ to suggest a balanced sample. This is an arbitrary choice.

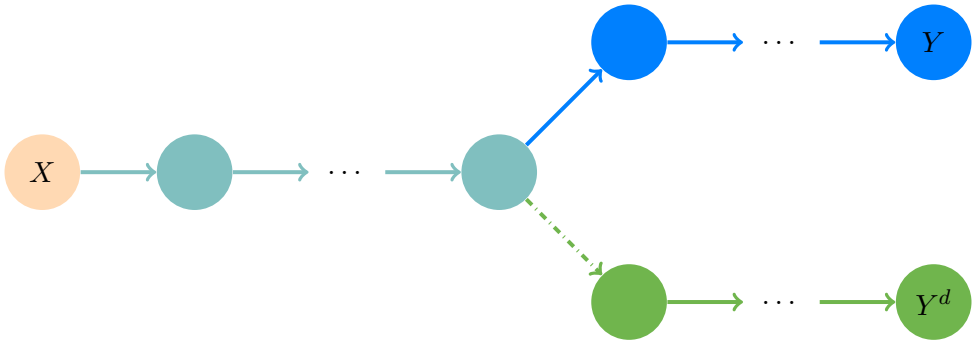


Figure 8: Rough sketch of the DANN architecture. Each node is an intermediate layer representation. The green network is trained to distinguish the domain representations, i.e. predict y^d. The blue network is trained to predict the labels y. At the green dashed arrow, gradients are reversed during backpropagation. The turquoise part is trained to provide representations supporting the label classification performance in the blue and impeding or complicating the domain discrimination by the green sub-network, i.e. decrease the deep $\mathcal{H}$-divergence. Typically, the feature extraction in the turquoise part uses convolutional layers, the two classification sub-networks use standard linear layers.

manipulating gradients during optimization and by allowing for **bifurcations** in the network. Bifurcating in a Neural Network means that at some intermediate layer, the output is fed into two separate subsequent sub-networks. The network before bifurcation tries to provide a data representation that optimally satisfies the needs of both of these sub-networks. These needs are "communicated" through regular backpropagation[14] as common for Neural Network training.
In the case of DANN, one network after the bifurcation is trained to classify the labels y, and the other one is trained to classify the domains, i.e. the domain labels y^d. The special feature of DANN is now a separate layer to manipulate the gradients during the backpropagation such that the feature extraction layers before the bifurcation are trained to provide a representation on which the domain classifier cannot perform well. A sketch of the DANN architecture can be seen in Fig. 8.
The gradient manipulation is done in the following way. To optimize a Neural Network, the gradient of the loss with respect to each layer's weights is calculated using the chain rule. Since the loss only depends on the network's output, this means the gradients w.r.t weights in the earlier layers depend on the following, deeper layers' weights. Thus, backpropagation facilitates dynamic programming to reuse them with the purpose of efficient gradient calculation.
Now in DANN, the **negative gradient** w.r.t the weights of the domain classifier's layers is used for calculation of the gradients regarding the feature extractor part of DANN.

[14] When training a Neural Network, backpropagation is the main step of calculating the gradient of the weights efficiently. It is a dynamic programming approach using repetition in chain rule components of $\partial\Phi(x)/\partial W_l$ for different layers l. The gradients are used for optimization, e.g. Gradient Descent.

Therefore, the earlier layers (before bifurcation) are updated each optimization step towards **worse performance** of the green network. At the same time, since the gradient is reversed directly at the bifurcation, both classification networks are optimized towards predicting y and y^d as well as possible given the feature.
Summarizing, the DANN algorithm is training two classification networks to perform different classification tasks, i.e. fitting the source classes $(x_{\mathcal{S}}, y)^n$ as well as fitting the domain labels $((x_{\mathcal{S}}, y^d = 0)^n, (x_{\mathcal{T}}, y^d = 1)^n)$. At the same time, it is training a feature selection network providing a shared input representation for the two classifiers. The features are trained, such that the source data label classifier can perform well and the domain discriminator fails to distinguish source and target domain. The latter is done by reversing parts of the gradient calculations to punish easily discriminable representations with high $d^l_{\mathcal{H}_a}$-values. Here l denotes the representation of the bifurcation and $\mathcal{H}_a$ denotes the hypothesis class of the domain discrimination network after bifurcation.
Ganin et al. show good performance for several Domain Adaptation benchmark problems. For details refer to [GUA+16, Section 5].

Note, that using DANN is not necessarily aligned with our idea to investigate intermediate representations to enable working with simpler classifiers and obtain tighter generalization bounds. It minimizes $d^l_{\mathcal{H}_a}$ at the bifurcation, not at the end of the network right before the classification. In deep domain discrimination, $\mathcal{H}_a$ are not just linear classifier hypotheses. Thus, VC generalization bounds will probably be vacuous. Nevertheless, the core idea is similar. If the classification on the source domain is good and source and target cannot be distinguished, then performance on the target should be good as well.

4.1.2. Data Manifold Entanglement

Remember, Sarishvili's Hypothesis 0.1 states that additional architecture is only needed when the domain shift is large. The claim Proposal 0.2 (a) (ii) as possible reasoning does not rest on sound research yet. One explanatory approach found in the literature is based on the **manifold hypothesis** for real-world data together with the idea of **entangled class manifolds** being disentangled, flattened or loosened by trained Neural Networks along the layers. The claim is that in these "*disentangled representations*" the domain shift is far smaller than the difference between the classes and thus a classifier should automatically classify target data similarly to source data.
This idea of a manifold disentanglement perspective of Neural Networks and the underlying assumption of the manifold hypothesis are briefly explained below.

Remark 4.2 (Manifold Hypothesis; [FMN13])**.**
In essence, the manifold hypothesis suggests that despite the complexity and high dimensionality of most real-world data, there exists a simpler, intrinsic structure that can be effectively captured in a space with reduced dimension. These simpler structures are referred to as manifolds.

As we will see later, the data clouds of different classes typically are not ordered geometrically. Hence, if the manifold hypothesis is true, the classes live on different manifolds that mix, intersect, and are not separable by simple linear classification or by critically looking at them (as an example, consider Fig. 16 later). Thus, networks creating representations with unfolded, disentangled, and separated class manifolds would unfold the metaphorical "Gordian Knot" of fussy, noisy real-world classification data. Suppose the representations of the different classes are always as separable as possible, i.e. classes can be separated with a large margin between points from different classes. In that case, generalization and small domain shifts should be overcome easily and there is no need for additional architecture like DANN to "cut the knot" by force.

Brahma et al. ([BWS15]) claim that such a disentangled representation can be observed in regularly trained deep fully connected Neural Networks. They further claim that this disentangling leads to the good generalization observed in the many successful Neural Network applications. We claim that if this is true, it might aid Domain Adaptation because overcoming small domain shifts and generalizing to data from a domain close to the source or generalizing to unseen data from the source domain itself are very similar tasks. In both cases, the classifier has to generalize to data slightly differently shaped to the one it trained on.

Example 4.3 (Manifold Disentangling By Deep Networks; [BWS15])**.**
Brahma et al. discuss **manifold disentangling** as an approach to explain the good generalization empirically observed in deep learning. They claim that optimized or trained deep Neural Networks induce data representations with flattened and separated class manifolds. In this framework, layers move, modify and shift the class manifolds as a whole instead of memorizing single points. Overfitting would be unlikely. Thus, any separation of classes should generalize to unseen data points, as these supposedly live on one of these class manifolds.
The authors propose an empirical measurement to observe the disentangling behavior and then use it to empirically validate the hypothesis that deep representations are disentangled for several examples. This empirical measurement is called **manifold entanglement** $\mathcal{E}$. Note that the name is kind of misleading, as $\mathcal{E}$ is constructed to measure the flatness of one class or sample, not the entanglement of the whole data or domain. It compares the pairwise distances between points to the distance when traveling along the surface of the "class manifold". The surface can be considered flat if these two distances are close in value for most points.
To implement this idea and calculate manifold entanglement $\mathcal{E}$, we need to estimate the path lengths along the manifolds. Therefore, we define so-called **geodesic distance** between two points given a class sample x^n of large sample size n. When using the sample points as discrete estimate of the manifold surface, define a path from $x_i =: x_{\pi^k_{(i,j)}[0]}$ to $x_j =: x_{\pi^k_{(i,j)}[k]}$ through k points of the sample as

$$P\left(\pi^k_{(i,j)}\right) := \left(x_{\pi^k_{(i,j)}[0]}, x_{\pi^k_{(i,j)}[1]}, \ldots, x_{\pi^k_{(i,j)}[k]}\right).$$

The indices $\pi^k_{(i,j)}$ denote a list of elements from $\{1, 2, \ldots . n\}$ with cardinality $k+1$ to indicate in which order the path travels through which sample points.

Define the length of $P\left(\pi^k_{(i,j)}\right)$ as

$$\left|P\left(\pi^k_{(i,j)}\right)\right| \coloneqq \sum_{l=1}^{k} \left\| x_{\pi^k_{(i,j)}[l]} - x_{\pi^k_{(i,j)}[l-1]} \right\|.$$

Then the approximate **geodesic distance** is defined as

$$G_M(i,j) \coloneqq \min_{\pi^k_{(i,j)}} \left|P(\pi^k_{(i,j)})\right|.$$

In conclusion, the proposed measure for flatness and entanglement of a sample x^n, the **manifold entanglement** is defined as

$$\mathcal{E}(x^n) \coloneqq \frac{2}{n(n-1)} \sum_{i>j} \frac{\left|G_M(i,j) - \left\|x_i - x_j\right\|\right|}{\left\|x_i - x_j\right\|}. \tag{4.1}$$

The interpretation is immediate. For flat data, the geodesic distance is close or equal to the Euclidean distance, and the value of $\mathcal{E}$ is close to zero. For highly curved data surfaces (and if k is sufficiently large), the geodesic distance for most points should be far larger than the length of the straight path. In this case, the entanglement value is high.
The authors investigate several Neural Networks for disentangling behavior and conclude that flattening takes place and thereby the Neural Network creates representations that are easily classifiable. The flattened classes intersect less and do not tie into each other as much anymore.

Remark 4.4 (Qualitative Assessment of Manifold Entanglement)**.**
We already mentioned that this measure does not quantify the entanglement of the data, but only the flatness of the separate classes. The authors themselves suggest further metrics that "should" quantify the mean overlap of classes.
For our experiments, we chose to use $\mathcal{E}$ together with a supplementary low-dimensional visualization of the data classes. We combine flatness with simple visualization to quantify entanglement more thoroughly than by only observing $\mathcal{E}$.
Further, it has yet to be reasoned or explained that entanglement is a predictor for good performance at all. We acknowledge that compressed and separated class representations simplify classification. Nevertheless, we expect that only observing one value for flatness might neither correlate with good performance nor with nicely disentangled representations. This doubt on the usefulness of $\mathcal{E}$ is also strengthened when thinking about its possible implementation or calculation.
First of all, if the network strongly compresses data along the layers, representations might be far from injective. Thus, there is a risk of dividing by zero. If we do not include points too close to each other the calculated value gets less precise and informative. Therefore, comparison between different representations using $\mathcal{E}$ could be meaningless.
Further, the measurement does not regard the dimension of the points x_i at all. While

experimenting with different architectures, it was noticeable to us that $\mathcal{E}$ tends to be smaller if the representation is lower dimensional. It is unclear to us, whether this empirical dependence on dimension is an indicator that networks are enforced to find more compressed and flat representation, especially in layers with smaller output dimensions, or whether this displays a possible flaw in the definition of $\mathcal{E}$. Additional investigation of this observation might yield more insights but it will be no topic in this work.
For this thesis, we limit ourselves to investigating possible relations between $\mathcal{E}$-values along the network layers and the relation to generalization and Domain Adaptation performance. Since the manifold entanglement perspective has not attracted too much attention in literature yet, this is a step to quantify $\mathcal{E}$ as either a possible measurement or even explanation approach for Neural Network generalization and Domain Adaptation.

4.2. Experiment Goals

Having the untouched parts of Sarishvili's Hypothesis 0.1, its explanation approach Proposal 0.2 and the freshly introduced concepts of DANN and manifold entanglement $\mathcal{E}$ in mind, we can formulate goals and methodology for our experimentation. Consider a Domain Adaptation problem given source and target data.

- **Goal:** By experimentation we want to improve our knowledge about deep representations in trained Neural Networks and the Domain Adaptation capabilities of fully connected Neural Networks. We want to compare representations in Domain Adaptation architectures such as DANN and representations in regular, source-only trained, deep fully connected Neural Networks. Further, we want to look at the influence of depth on possible results.
- **Methodology:** When investigating Neural Networks we have multiple possibilities to evaluate the empirical validity of Sarishvili's Hypothesis 0.1 and Proposal 0.2. First of all, we can compare the **empirical risks** on both domains to show the Domain Adaptation performance itself (Theorem 1.25). Then, we can compare representations with respect to their (deep) **empirical $\mathcal{H}$-divergence** for $\mathcal{H}$ being linear classifiers or small domain discrimination networks. Third, we can look at the flatness of the representation class manifolds by calculating the **entanglement $\mathcal{E}$** (Example 4.3). Last but not least we can take a look at the class data clouds themselves using **dimensionality reduction visualization** methods such as t-SNE (cf. [vdMH08]).

We are forgoing the implementation of information-theoretic measures from Section 3 as the theoretical foundations are deemed too uncertain and unreliable, while at the same time, the calculation complexity is high. Further, we already discussed empirical arguments for and against optimizing the information theoretical target representation when presenting Information Bottleneck in Section 3.3.1.

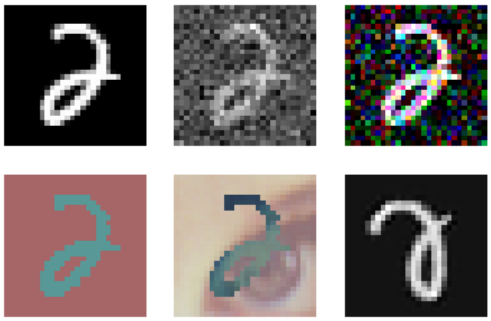


Figure 9: Examples for the different domains MNIST, MNIST-n [GS], MNIST-n [RGB], MNIST-c [RGB], MNIST-l [RGB], MNIST-r[GS]. The parameters are chosen as "Signal-Noise-Ratio" $= 3$ for MNIST-n (cf. Definition 4.6 for SNR) and rotation angle $= \frac{\pi}{4}$ for MNIST-r.

4.3. Experiments on Handwritten Digits

The experiments of this section consider Domain Adaptation problems on variants of MNIST. This is a state-of-the-art dataset with 60 000 training and 10 000 testing points. Each data point is a grayscale image of a handwritten digit between 0 and 9, stored as 28×28 matrix with values in $[0, 1]$. Some variants of MNIST later will consist of RGB images. For comparability, we convert 28×28 images to 3-channel images of size $28 \times 28 \times 3$, by repeating the image three times.
Classical MNIST constitutes the most basic domain. Further domains are constructed as the following variants.

- **MNIST:** 3-channel repetition of the values of the classical 28×28 images of handwritten digits.
- **MNIST-r:** rotated MNIST. Variants are different rotation angles.
- **MNIST-n:** MNIST with added Gaussian noise. Variants of MNIST-n include different magnitudes of noise and the two cases [GS] and [RGB] i.e. adding noise before the 3-channel repetition on the grayscale or thereafter on the RGB image. After adding noise, data points are normalized to $[0, 1]$-values.
- **MNIST-c:** colored MNIST. The digits and the background are each colored in a constant color. The color choice is random for each data point, but a significant contrast between the digit and the background is ensured.
- **MNIST-l:** MNIST where the 3 channel image is superposed with a randomly selected and colored patch of the famous "Lenna-Picture"[15]. Again contrast between

[15] http://www.lenna.org/. This picture is often used for any kind of image processing as it has a mix of detailed and shallow regions. It has clear structures posing different challenges in different parts of the image.

the coloring of the digit and the background is ensured. Further, the images are normalized to $[0, 1]$-values.

Examples for each domain are shown in Fig. 9. The labels are chosen as the displayed digits 0 to 9.

Remark 4.5.
For the upcoming experiments, we lift our assumption of only considering binary classification (Assumption 1.12). This has multiple reasons:

- The empirical knowledge we gain by doing the experiments this way is not significantly different. Binary classification is a tool to make establishing theory much clearer and easier to interpret for the reader.
- Considering the manifold hypothesis, collecting multiple different digits in one class (e.g. one class includes the digits with labels 0 to 4 and the other class the digits with labels 5 to 9) does not make sense. Each new class would consist of 5 class manifolds which would be processed together by training. This would probably make it harder to investigate the surface utilizing visualization and the manifold entanglement measure.
- Restricting the experiments to only two of the ten different classes reduces the number of samples and in general is a biased choice. Some pairs of digits are easier to distinguish than others. Further, we did experiments restricted to multiple pairs of digits and can affirm that the results are similar but less expressive as fewer samples are used.

4.3.1. The Influence of Depth

Proposal 0.2 (a) (ii) claims that deeper representations are increasingly invariant to domain shifts. Therefore, the first experiment tries to compare (blind) Domain Adaptation performances for different Neural Network depths. The result will also help to find a good network architecture in further experiments. We expect to observe a compression or flattening of class manifolds in deeper networks. In this subsection, we will consider two different Domain Adaptation problems to investigate the effects of different network depths. The first one investigates the performance loss after increasingly large domain shifts (MNIST to MNIST-n with different levels of noise). This will show the magnitude of the effects as the domains are easily comparable.
Thereafter, the second and main experiment of this subsection investigating depth considers more complex domains (MNIST-c to MNIST-l) and visualizes possible representation alignment in detail.

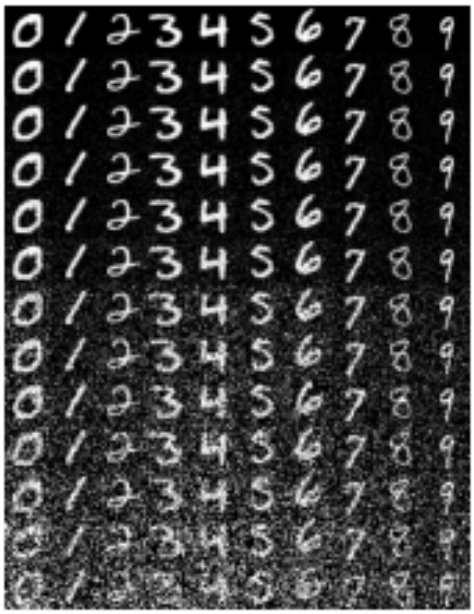


Figure 10: Examples for the images in increasingly noisy MNIST with SNR values: $1:0$, $60:10$, $50:10$, $40:10$, $30:10$, $20:10$, $10:10$, $9:10$, $8:10$, $7:10$, $6:10$, $5:10$, $4:10$.

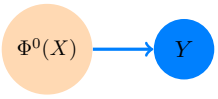
Exp. 1: Gradually Increasing Noise

To gain the maximal amount of information by our initial experiment, we choose an example where the domain shift can be gradually increased. The source domain is MNIST and the different target domains are all grayscale MNIST-n domains with different signal-to-noise ratios (SNR). These SNR values are defined using empirical estimates for the pixel values of each image. For the sake of simplicity, we only introduce SNR for quadratic 1 channel images. This can be generalized to other formats.

Definition 4.6 (Signal-To-Noise Ratio (SNR))**.**
Consider one image $x \in [0,1]^{m \times m}$ and use an estimator $std(x) \in [0,1]$ for the **standard deviation** of the entries or pixels of x. Using this estimate, we can sample a noisy version x^{noise} of x with a **signal-to-noise ratio** $\mathrm{SNR} \in \mathbb{R}$ as a realization of

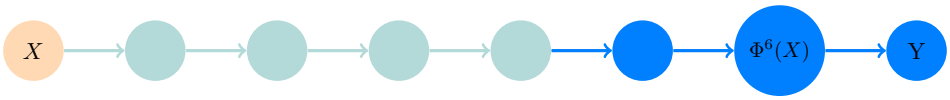
$$x^{noise} = \left(x_{i,j} + \varepsilon_{i,j}\right)_{i,j} \text{ with pixel-wise noise } \varepsilon_{i,j} \sim \mathcal{N}\left(0, \left(\frac{std(x)}{\mathrm{SNR}}\right)^2\right).$$

Note, that SNR can also be set to infinity to signal the case of no added noise. For convenience, we will denote SNR as actual ratios "S:N" of standard deviations of signal and noise instead of reals or infinity. This notation has the advantage that it includes the case 1:0 where no noise is added nicely.
With the possibility of defining domains using SNR, we can describe the first experiment. The **domains** we consider are illustrated in Fig. 10. We go from no humanly perceivable noise (SNR = 60:10) to so much noise that human classification by looking at the images is already hard (SNR = 4:10).
This experiment cosiders **fully connected Neural Network architectures** only. To facilitate comparison later, the architecture are aligned with the classification path of the DANN architecture (Example 4.1). Precisely this means we consider up to four (for network depth 5) convolutional layers for feature extraction. For networks of depth at least 6, we append standard linear layers for classification. All hidden layers have ReLU activation functions. The last layer of the networks is always a linear layer outputting the class prediction, i.e. for MNIST a stochastic vector with 10 entries, one for each class.



(a) Depth 1 consists of only the linear classification layer. The deepest representation is X after the input layer.

(b) Network of depth 3. Two convolutional layers followed by the linear classification layer.



(c) Network of depth 7. The maximum of four convolutional layers is followed by two linear and the classification layer.

Figure 11: Examples of the architecture setup in our first experiment. Arrows represent layer evaluation, i.e. first convolution and in deeper layers linear evaluation. The output or classification layer is always a linear layer with our 10 dimensional label prediction as output. Considering deeper networks, the hidden layers are the first up to four convolutional layers (light blue), then arbitrarily many linear layers (dark blue). $\Phi^{L-1}(X)$ denotes the deep representation we consider for example when looking at $\mathcal{H}$-divergence. This is emphasized by the node size. The exact parameterization is given in Appendix C.

We trained the different architectures on the MNIST domain using TensorFlow ([Bra15]) and the optimization algorithm ADAM ([KB14]).
The networks are trained source-only, that is only on MNIST training samples. We look at the classification performance of the different architectures tested on the dif-

ferent MNIST-n target domains and the ability to discriminate between the respective domain representations. The domain discrimination is tested by training two classifiers on the domain representations, predicting y^d after classification training is finished. One domain classifier is a small Neural Network, one is a linear threshold classifier. The numerical results, sorted by depth and target domain SNR, can be found in Fig. 12 and summarized as follows. On the **source domain**, we observe very good classification results. Solely in very deep and very shallow networks the performance slightly drops. While the performance drop in 1-layer networks, i.e. linear classifiers, can be described by insufficient classification power, the performance decrease in deep ones is highly likely to be caused by local minima in optimization or overfitting. This is apparent as choosing the parameterization of the network with depth 10 and simply appending identity function layers would increase the accuracy. This can be done as the last few layers all output representations of the same dimension (again, for the exact setup refer to Appendix C).
In our experiments, different optimization setups with more optimization steps or different weight initialization did not improve accuracy for very deep architectures compared to Fig. 12. On more powerful computational hardware and larger storage, using many augmented training samples might increase performance.
Considering the **Domain Adaptation** task, i.e. accuracy on (unseen) target domain data, we note the following. For small domain shifts up to SNR 20 : 10 the classification accuracy on the target domain is high. Further, the connection between depth, empirical $\mathcal{H}$-divergence, and performance after domain shift can be seen when considering the deeper networks.
Considering Fig. 12 (b) and (c), we notice that the smaller networks do not produce indistinguishable representations. This can be explained by them only consisting of convolutional layers. Convolution operations compute local outputs only (cf. Example 2.4). It is likely that they only focus on the task of detecting the digit's shape and do not randomly decide to reduce the influence of possible noise e.g. in the corner of the images, when doing source-only training. Convolutional layers are by nature not useful to decrease the impact of the given domain shift enough, without additional heuristics and unsupervised Domain Adaptation training.
In contrast to the convolutional layers, the accuracies on deep linear layer representations show that the domain shifts have a decreasing impact on performance with increasing network depth. Linear and also neural classification of deep representations by their domain y^d is not significantly better than a coin flip as long as the domain shift is small. Small here are SNR-values larger than 20 : 10. For bigger domain shifts, the distinguishability also slightly decreases with increasing depth, noticeable after depth 7.
This first experiment already serves as a solid introduction to validating Sarishvili's Hypothesis' claim of domain alignments in deep representations and yields insightful results. Especially for depths 8 to 10, the deep representations overcome small domain shifts and achieve almost perfect accuracy on both domains. For deeper networks, the domain shifts also do not impact classification performance significantly, i.e. the accuracy on the target domain is comparable to the accuracy on the source. However, we

(a) Classification accuracy on the test dataset of the respective domain. Columns indicate the depths of the tested networks and rows the domains (ordered by its SNR) from which the test data originates.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1:0	0.93	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.90	0.81	0.89
60:10	0.41	0.69	0.92	0.95	0.98	0.97	0.90	0.97	0.97	0.98	0.89	0.78	0.87
50:10	0.37	0.55	0.88	0.93	0.97	0.95	0.85	0.96	0.96	0.98	0.89	0.77	0.86
40:10	0.34	0.35	0.81	0.89	0.96	0.93	0.78	0.93	0.94	0.97	0.88	0.75	0.84
30:10	0.30	0.16	0.69	0.82	0.93	0.88	0.68	0.86	0.90	0.95	0.85	0.72	0.79
20:10	0.23	0.10	0.54	0.67	0.84	0.76	0.53	0.72	0.80	0.88	0.77	0.63	0.66
10:10	0.17	0.10	0.36	0.41	0.55	0.30	0.31	0.34	0.53	0.45	0.56	0.35	0.43
9:10	0.15	0.10	0.34	0.38	0.51	0.24	0.28	0.31	0.48	0.39	0.50	0.30	0.41
8:10	0.15	0.10	0.30	0.35	0.46	0.19	0.25	0.28	0.43	0.33	0.41	0.25	0.38
7:10	0.14	0.10	0.28	0.32	0.41	0.16	0.22	0.25	0.37	0.28	0.33	0.19	0.34
6:10	0.13	0.10	0.27	0.28	0.35	0.13	0.19	0.22	0.32	0.23	0.26	0.16	0.29
5:10	0.13	0.10	0.23	0.25	0.29	0.12	0.17	0.19	0.28	0.20	0.21	0.14	0.22
4:10	0.12	0.10	0.20	0.21	0.24	0.12	0.15	0.16	0.24	0.17	0.18	0.13	0.16

(b) $\overline{d_{\mathcal{H}}}^{L-1}$-values for linear hypotheses $\mathcal{H}$: Discrimination accuracy of deep representations of MNIST and MNIST-n with the given SNR using a linear classifier.

	1	2	3	4	5	6	7	8	9	10	11	12	13
60:10	1.00	1.00	1.00	1.00	1.00	0.67	0.52	0.49	0.49	0.50	0.50	0.51	0.49
50:10	1.00	1.00	1.00	1.00	1.00	0.73	0.53	0.48	0.49	0.49	0.50	0.50	0.49
40:10	1.00	1.00	1.00	1.00	1.00	0.80	0.59	0.48	0.49	0.50	0.49	0.49	0.50
30:10	1.00	1.00	1.00	1.00	1.00	0.85	0.66	0.50	0.49	0.50	0.50	0.49	0.50
20:10	1.00	1.00	1.00	1.00	1.00	0.96	0.81	0.57	0.55	0.52	0.54	0.51	0.52
10:10	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.76	0.71	0.65	0.72	0.58	0.63
9:10	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.80	0.76	0.69	0.74	0.59	0.65
8:10	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.83	0.78	0.72	0.75	0.60	0.70
7:10	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.87	0.80	0.74	0.77	0.64	0.70
6:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91	0.84	0.78	0.81	0.66	0.84
5:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93	0.86	0.80	0.81	0.59	0.78
4:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.88	0.81	0.73	0.76	0.92

(c) $\overline{d_{\mathcal{H}_a}}^{L-1}$-values for deep Neural Network hypotheses $\mathcal{H}_a$: Discrimination accuracy of deep representations of MNIST and MNIST-n with the given SNR using a Neural Network.

	1	2	3	4	5	6	7	8	9	10	11	12	13
60:10	1.00	1.00	1.00	1.00	1.00	0.78	0.55	0.47	0.50	0.50	0.49	0.48	0.49
50:10	1.00	1.00	1.00	1.00	1.00	0.82	0.61	0.48	0.50	0.49	0.50	0.50	0.50
40:10	1.00	1.00	1.00	1.00	1.00	0.87	0.68	0.50	0.48	0.50	0.48	0.50	0.51
30:10	1.00	1.00	1.00	1.00	1.00	0.93	0.82	0.56	0.51	0.48	0.51	0.50	0.53
20:10	1.00	1.00	1.00	1.00	1.00	0.98	0.92	0.64	0.56	0.54	0.58	0.52	0.59
10:10	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.89	0.75	0.68	0.74	0.64	0.75
9:10	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.91	0.77	0.70	0.76	0.65	0.78
8:10	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.94	0.80	0.68	0.74	0.71	0.80
7:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.81	0.75	0.79	0.75	0.76
6:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.87	0.78	0.84	0.78	0.87
5:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.90	0.80	0.87	0.80	0.92
4:10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.92	0.83	0.89	0.81	0.94

Figure 12: Performance of deep Neural Networks of depths $L = 1$ to 13 in the unsupervised Domain Adaptation from source MNIST to target MNIST-n. The rows indicate the different MNIST-n target domains with SNR-values between $60:10$ and $4:10$. All networks are only trained on MNIST, i.e. the domain with SNR $1:0$. The cell values are accuracies of the tasks of predicting the digit labels y in (a) (i.e. calculating 1 minus the empirical risk $\overline{\mathcal{R}}$) or classifying the last layer representation by its domain y^d in (b) and (c) (i.e. calculating empirical deep (representation) $\mathcal{H}$-divergence $\overline{d_{\mathcal{H}}}^{L-1}$). Different colormaps are used to distinguish these two kinds of accuracies. In both, red suggests bad performance. It emphasizes instances with low classification accuracy or good distinguishability of the domain representations.

could not achieve zero loss performance for these very deep networks, even throughout several different training approaches.

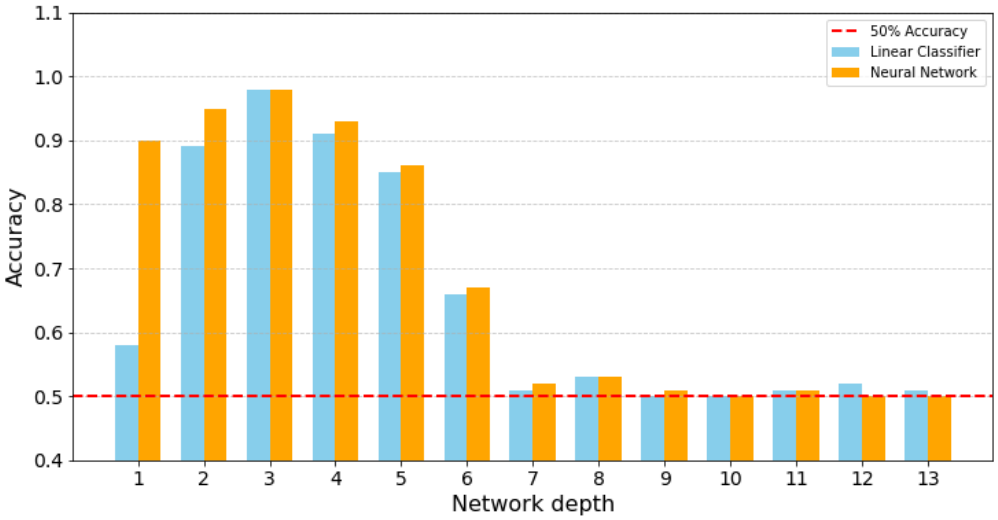
Figure 13: Examples for the classes of the domains MNIST-c (top) and MNIST-l (bottom).

Experiment 2: Colorful Domains To further evaluate the soundness of Sarishvili's Hypothesis 0.1 concerning depth, we look into a different, more complex Domain Adaptation problem. The domains are MNIST-c as the source and MNIST-l as the target. Examples for all classes/digits in both domains are provided in Fig. 13. Both domains are randomly colored, the difference is a superposition with Lenna-patches in MNIST-l. The domain difference can no longer be quantified based on the background being black or noisy, i.e. values being 0 or not. The initial domains are neither linearly classifiable nor linearly separable anymore because most pixels are randomly colored. This means the geometrical structure is mostly based on randomness, not class or domain features. A classifier has to detect shapes and contrasts to classify data. We will see in the results that this requires more than one layer.
The first results are shown in Fig. 14. Again, the first metric we observed is **classification accuracy** on the source and target domain. The results are similar to the previous experiment. Accuracies on both domains are almost perfect for network depths 2 to 11 before experiencing a slight but notable decline for deeper networks. As data is geometrically most dependent on the random color, linear classification of the input data, i.e. the network in the first table column, again does not yield good results. 13% is only slightly better than classifying with no training at all.
Further, **domain discrimination** results are also similar to the case of adding noise as domain shift. A small Neural Network can distinguish the two domains both in the initial data and the representations after a convolutional layer. The deeper representations output by the linear layers are not discriminable anymore. A linear domain classifier also fails to distinguish representations that are at least six layers deep. Performance should be translated from source to target domain. As already shown in the accuracies in Fig. 14 (a), the performance is almost the same before and after the domain shift.

In addition to these results, we look at **deep representations** themselves. We look into class **entanglement** through $\mathcal{E}$ (Example 4.3) and **visualize representations** using

	1	2	3	4	5	6	7	8	9	10	11	12	13
MNIST-c	0.13	0.97	0.98	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.88	0.86
MNIST-l	0.12	0.94	0.97	0.98	0.97	0.98	0.97	0.95	0.95	0.96	0.96	0.86	0.83

(a) Accuracy of label prediction on source domain MNIST-c and target domain MNIST-l with networks of depths 1 to 13.

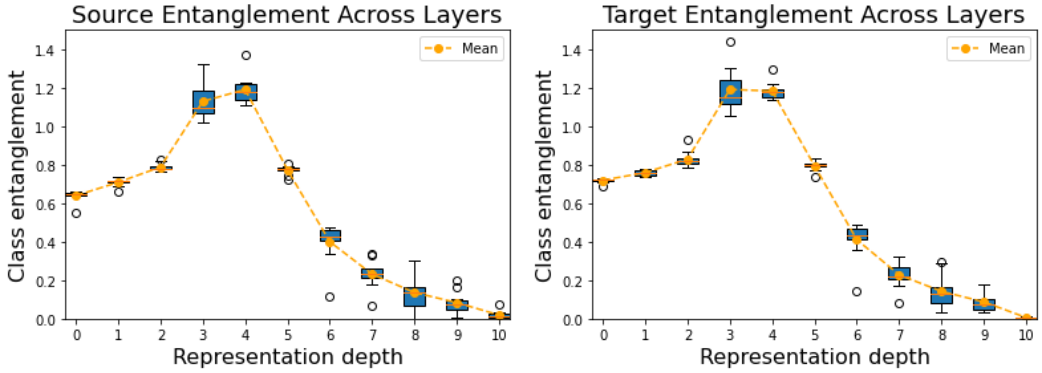


(b) Domain discrimination accuracy on last layer representations by linear and network classification of y^d. The baseline 0.5 indicates complete indistinguishability.

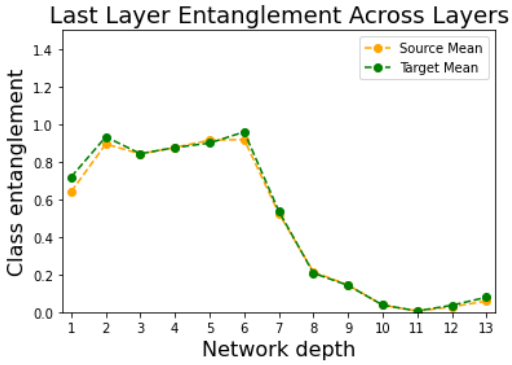
Figure 14: Accuracies of the experiment regarding the domain shift from MNIST-c to MNIST-l and network depths 1 to 13. Networks of different depths are again trained on MNIST-c only.

the dimensionality reduction method t-SNE ([vdMH08])[16].
The values of $\mathcal{E}$ across the layers are shown in Fig. 15 and selected representations are visualized in Fig. 16 below. Thereby, we investigate how data is processed throughout a network. As a *representative architecture*, we choose a network of depth 11. This is the deepest of our networks with good classification accuracy and indistinguishable domain representations. Entanglement values and visualization might provide insight into the data processing of deep Neural Networks. Combining the insights these two metrics provide, we can conclude that the networks in this example have the data clustered by classes/digits already after the convolutional layer, early in the network. Judging by

[16]Note, that t-SNE is only a tool to compare representations. It is a non-linear, and non-deterministic dimensionality reduction method which aims to keep local structures similar. It cannot be used as an absolute measurement for performance or even entanglement, as it changes the relative and absolute positions of all points in some way.



(a) Entanglement $\mathcal{E}$ through the layers of one trained network. The values are calculated for all classes individually and visualized together using box plots (Appendix D.1). The first four layers are convolutional. The succeeding linear layers output representations with reduced mean entanglement across all class manifolds.



(b) Mean entanglement for class representations of the semi-last layers of Neural Networks with different depth. The mean entanglement decreases in deep linear layers. Class manifolds are flattened.

Figure 15: Entanglement $\mathcal{E}$ for the MNIST-c to MNIST-l Domain Adaptation problem. In (a) we look at the changes in entanglement throughout the layers of the Neural Network with a depth of 11. Then for convenience, in (b) we also show the entanglement of the respective deep representations for all the Neural Network architectures of different depths.

the t-SNE visualizations, the classes are separated further in deeper representations and they are compressed to make points of the same class as similar as possible. Further, the entanglement metric decreases significantly in deeper networks implying that data manifolds are flattened across linear layers.
In conclusion, the outcomes of these two experiments seem to partly align with the underlying theoretical claim in Proposal 0.2. Increasing depth by stacking linear layers (and according to our analysis only linear layers, no convolutions), seems to increase the Domain Adaptation capabilities of source-only trained Neural Networks up to a certain depth. We observed that small domain shifts can be overcome, deep representations of

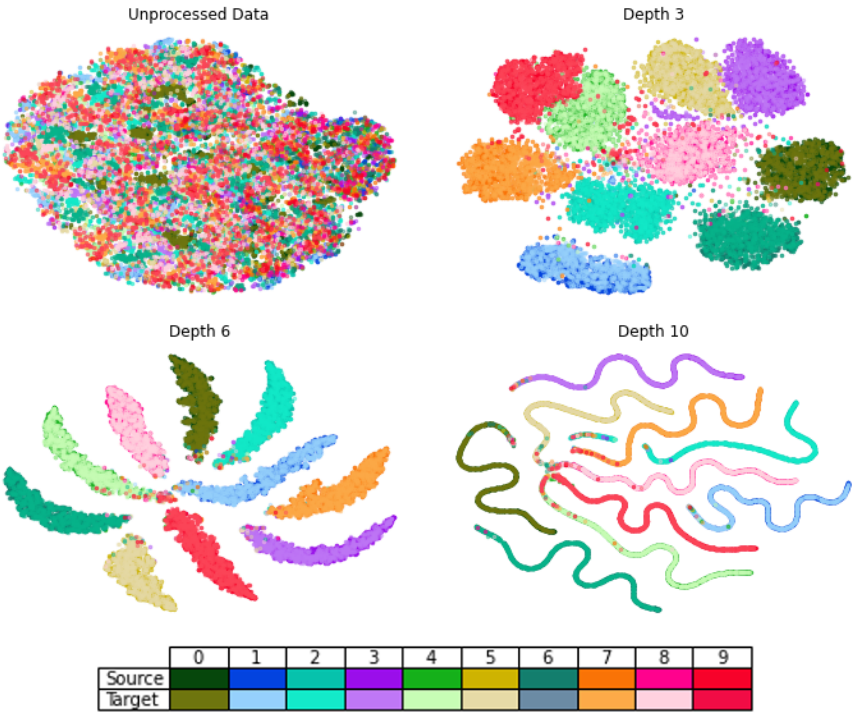


Figure 16: t-SNE visualization of different representations produced by layers in the network with depth 11 in experiment 2. The network is trained source-only on MNIST-c and the corresponding target domain is MNIST-l. Data is notably separated by classes and compressed. Distances between classes seem to increase greatly in relation to the distances between data of the same class which seems to decrease.

the two domains can be aligned and data clouds seem to be compressed and flattened. This validates our claims empirically. Deep Neural Networks could be sufficient to tackle certain Domain Adaptation problems and manifold flattening or disentangling could be a possible metric predicting good performance.

In the following subsection, we are conducting further experiments on more MNIST domains to enhance the empirical evidence base. Instead of the influence of depth, the next focus will be the claim from Sarishvili's Hypothesis 0.1 that special architecture might be unnecessary to perform certain Domain Adaptations.

4.3.2. Necessity of Additional Architecture

With the insights on the effect of depth from the previous experiments, we will proceed to compare the Domain Adaptation performance of standard source-only training with the DANN architecture (Example 4.1). We have seen that small domain shifts, like

small noise or perturbations, do not influence performance and we need no additional architecture. This experiment now tries to show the limits of this property.

Notation 4.7 (The Networks)**.**
Technically speaking, both the previously considered deep networks and the DANN architecture are both forms of Neural Networks as the calculations of the individual layers are based on the same idea. The difference is simply that DANN does not fit into *our* construction in Definition 2.2. To tell the networks apart, we will call the networks fitting into this definition, i.e. with no additional architecture again fully connected Neural Networks (FCNN). Any Neural Network with bifurcation as in Example 4.1 will be called DANN.

The considered **domains** in this experiment are all six introduced MNIST variants already illustrated in Fig. 9: MNIST, MNIST-n [GS] and [RGB], MNIST-c, MNIST-l and MNIST-r. For MNIST-n the SNR is chosen as $30:10$, for MNIST-r the rotation angle is $\frac{\pi}{4}$. In contrast to the previous experiment, we do not only shift from a simple to a more complex but in some way similar domain (MNIST to MNIST-n and MNIST-c to MNIST-l) but instead consider all possible source-target domain combinations.
We train one fully connected Neural Network on each domain as "source-only training" and additionally train a DANN architecture on all possible domain combinations doing unsupervised Domain Adaptation.
This means there are many networks to be trained. Due to limited computational resources and storage, the network architectures are chosen with a smaller number of parameters. Nevertheless, they are chosen with the results of the previous experiment in mind reproducing their results. We consider fewer convolutional layers than before, as we have already concluded they do not really help when investigating the claim. It will be interesting to see, that with DANN training, the convolutional layers can reduce the domain shift. Further, we do not consider the maximal well-performing network depth from the experiments before. Instead, we fall back to networks with fewer linear layers. The FCNNs consist of two convolutional layers followed by four linear hidden layers and the linear output layer. A sketch can be seen in Fig. 17 and the precise setup is again given in Appendix C. The DANN architectures share the same architecture with a bifurcation and a succeeding domain classifier after the second convolutional layer. The domain classifier consists of one gradient reversal and two dense layers, including the output layer.
The numerical **results** of the experiments are shown in Figs. 18 and 19. The tables show the empirical misclassification error of the networks on the target domain and the empirical $\mathcal{H}$-divergence $\overline{d_{\mathcal{H}_a}^{L-1}}$ between the last layer representations of source and target domain, calculated by a small discrimination network. Notably, DANN overcomes multiple of the domain shifts, which the fully connected network cannot reduce. Especially from MNIST and its noisy variants to the colored variants, the classification performance on the target is improved by 70% to 80% when choosing DANN over FCNN. Further, the domain discrimination values clearly show that DANN is very good at creating indistinguishable representations and FCNN can only hold up when domain shifts are small.

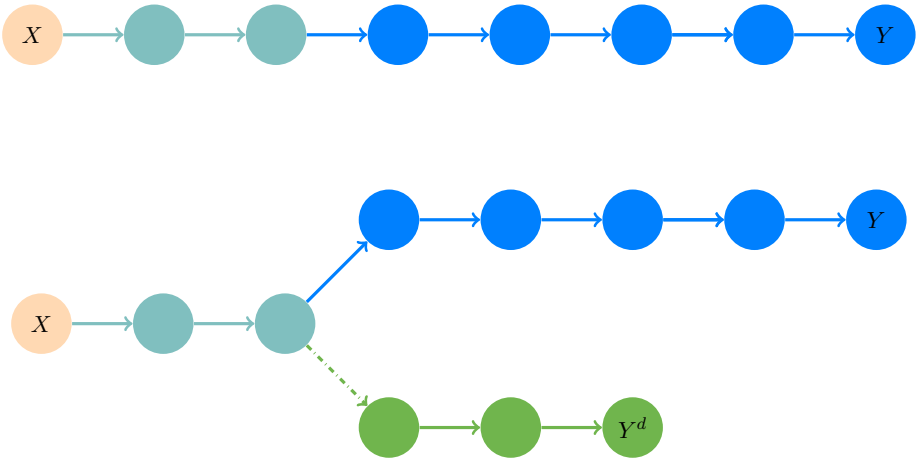


Figure 17: Depth 7 fully connected Neural Network and DANN architectures used in the third experiment. The turquoise and blue layers are building the classification network. The DANN-specific extra architecture is shown in green. The dashed arrow again denotes the gradient reversal to enforce domain indistinguishability.

Classification Accuracy

	MNIST	MNIST-n GS	MNIST-n RGB	MNIST-c	MNIST-i	MNIST-r
MNIST	0.99	0.99	0.99	0.18	0.17	0.65
MNIST-n GS	0.99	0.99	0.99	0.19	0.19	0.57
MNIST-n RGB	0.99	0.99	0.99	0.20	0.18	0.64
MNIST-c	0.99	0.96	0.95	0.98	0.97	0.52
MNIST-i	0.98	0.96	0.89	0.98	0.98	0.45
MNIST-r	0.49	0.50	0.52	0.15	0.17	0.99

Domain Discrimination Accuracy

	MNIST	MNIST-n GS	MNIST-n RGB	MNIST-c	MNIST-i	MNIST-r
MNIST	0.50	0.67	0.50	0.98	0.98	0.84
MNIST-n GS	0.62	0.50	0.50	0.98	0.98	0.87
MNIST-n RGB	0.50	0.53	0.50	0.98	0.99	0.83
MNIST-c	0.82	0.61	0.50	0.50	0.56	0.83
MNIST-i	0.65	0.66	0.84	0.58	0.50	0.85
MNIST-r	0.89	0.93	0.91	0.99	0.99	0.50

Figure 18: Fully connected Neural Networks trained **source-only** for blind Domain Adaptation. The cells values are empirical classification errors and domain discrimination accuracies on deep representations of layer $L - 1 = 6$. The latter are calculated using a small discrimination network. Rows indicate the source domain and columns the target domain.

Classification Accuracy

	MNIST	MNIST-n GS	MNIST-n RGB	MNIST-c	MNIST-I	MNIST-r
MNIST	0.99	0.99	0.99	0.97	0.93	0.65
MNIST-n GS	0.99	0.99	0.99	0.96	0.89	0.59
MNIST-n RGB	0.99	0.99	0.99	0.97	0.91	0.55
MNIST-c	0.98	0.98	0.98	0.98	0.97	0.48
MNIST-I	0.98	0.98	0.95	0.98	0.98	0.52
MNIST-r	0.82	0.76	0.79	0.18	0.17	0.97

Domain Discrimination Accuracy

	MNIST	MNIST-n GS	MNIST-n RGB	MNIST-c	MNIST-I	MNIST-r
MNIST	0.50	0.50	0.50	0.50	0.62	0.72
MNIST-n GS	0.50	0.50	0.50	0.54	0.64	0.73
MNIST-n RGB	0.50	0.50	0.50	0.57	0.60	0.70
MNIST-c	0.50	0.50	0.50	0.50	0.50	0.68
MNIST-I	0.53	0.50	0.54	0.50	0.50	0.72
MNIST-r	0.63	0.66	0.64	0.78	0.75	0.50

Figure 19: DANN trained **unsupervisedly** on the labeled source and unlabeled target data for Domain Adaptation. The cell values are empirical classification errors and domain discrimination accuracies on deep representations of layer $L - 1 = 6$ in the classification path after bifurcation. The latter accuracies are calculated using a small discrimination network. Rows indicate the source domain and columns the target domain.

This is the case for the pairs MNIST and MNIST-n or MNIST-c and MNIST-l.
In the results, it can also be seen that Neural Networks with no additional architecture perform better when going from a complex source domain to a less complex target domain than the other way around, e.g. MNIST to MNIST-c in comparison to the other way around MNIST-c to MNIST. Whether this is generally the case or only for our choices of domains is unclear.
In cases where either the source or target domain is MNIST-r, even DANN fails to adapt. Note, however, that this is not because such Domain Adaptation problems are impossible. Domain Adaptation problems adapting from MNIST to MNIST-r with different angles ***can*** be done well. For such an example, refer to so-called *Kullback-Leibler Guided Domain Adaptation*, presented in [NTG+21].
As in the previous experiment, we can investigate the representations with respect to their **entanglement**. Showing all combinations would be repetitive and therefore, motivated by the accuracies of all the different domain pairs discussed above, we restrict to the following three Domain Adaptation problems:

- MNIST → MNIST-c: one of the cases with the highest performance gap between blind and unsupervised Domain Adaptation. The fully connected Neural Network has an accuracy of 18% on MNIST-c, while DANN achieves 97% accuracy.

- MNIST-c → MNIST: Exactly the other way around, the FCNN matches DANN in classification performance. Both perform well on the target. Representation discriminability differs between the FCNN and the DANN.
- MNIST-r → MNIST-c: In stark contrast to the previous case, both Domain Adaptation performances are miserable and the domain representations are distinguishable easily.

These three problems should cover a diverse view on connections between Domain Adaptation performance, domain discrimination and representation entanglement if there are any.
Fig. 20 shows the investigation of mean $\mathcal{E}$ values on these three Domain Adaptation problems. For each problem, we look at the source and target domain for the fully connected Neural Network and DANN separately. All networks decrease the mean entanglement in all domains. The degree of entanglement in the deepest representation seems to be empirically linked to performance:
For deepest hidden layer representations in FCNNs that show good Domain Adaptation performance, we can see that the value of $\mathcal{E}$ on the target domain classes seems to have values around 0.25, comparable to the value on the source domains (MNIST to MNIST-c with DANN and both networks in MNIST-c to MNIST). The networks that perform worse on the target also tend to have higher entanglement, converging to a value of approximately 0.5. The DANN architectures always achieve the entanglement value of 0.25 on source and target domains, no matter the classification performance after the domain shift. This is evident in the third example with source domain MNIST-r.
The initial, unprocessed data of MNIST-c appears to be quite flat already, while MNIST and MNIST-r show a much higher entanglement, based on $\mathcal{E}$. The t-SNE visualizations support the observations we made using manifold entanglement. Fig. 21 highlights the difference between MNIST to MNIST-c and MNIST-c to MNIST. We first show the initial, unprocessed data, before comparing representations from blind domain adaptation and FCNN to DANN representations. The comparison is done at the deepest layer and right after the convolutional layers, i.e. at the bifurcation. It is evident, that blind Domain Adaptation cannot handle the domain shift from MNIST to MNIST-c which can be seen as the MNIST-c data cloud is never really separated by class. We saw previously, however, that it is slightly disentangled in the deeper layers in comparison to the convolutional layer representations. In the case of MNIST-c to MNIST, this is different. While still easily discriminable in this visualization, both domains are ordered by their label after the convolutional layers of the FCNN. In the deeper representations, the classes of the two domains overlap pairwise and are not easily separable anymore. DANN representations in both of these cases are aligned nicer and faster, as expected when using a representation aligning metric.
The third experiment shows the apparent problem when using MNIST-r as either the source or target domain. Neither the FCNN nor the DANN classifier can separate the target domain classes in a meaningful way. DANN is still able to decrease deep domain discriminability, but the result is just a messy carpet of MNIST-c data points somehow

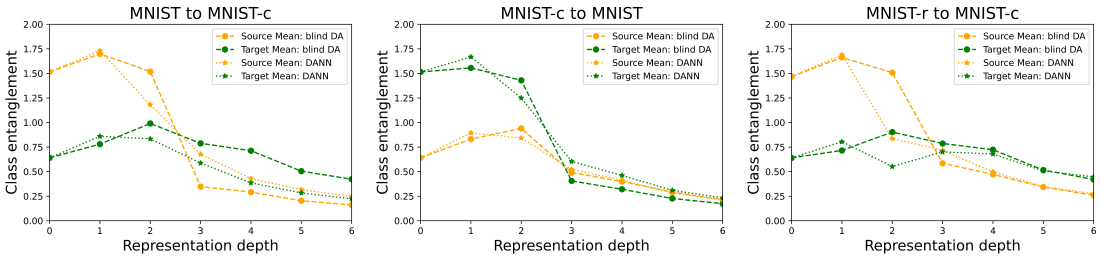
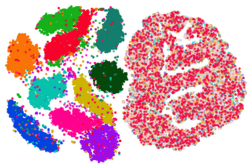


Figure 20: Mean entanglement $\mathcal{E}$ for FCNN and DANN on three Domain Adaptation tasks. After the convolutional layers, entanglement decreases. In the case MNIST-c to MNIST which is the only case of these three with good Domain Adaptation using FCNN, we observe the lowest deep representation entanglement of FCNNs performing blind Domain Adaptation.

aligned with the nicely ordered MNIST-r class clusters. Our DANN does not achieve a representation relating colored digits to rotated digits which would yield good classification on both domains.
This third experiment shows, that even when using DANN and observing low entanglement $\mathcal{E}$ of approximately 0.25 (Fig. 20) and $\mathcal{H}$-divergence of representations at approximately 0.75 (Fig. 19), the accuracy after domain shift can be simply bad. The reason was apparent when observing t-SNE visualizations of the different classes of both domains. Thus we can conclude that in well-performing networks, all the measurements we use as predictors for Domain Adaptation performance seem to suggest this good performance. However, we cannot simply use low entanglement or domain discriminability as a predictor for good performance on the target domain. Further, we have seen that many domain shifts, especially between grayscale and RGB domains, can lead to bad performance on the target when blindly learning on the source domain only. Architectures like DANN are very useful in this case.

4.3.3. Results

The different results on MNIST domains provided insights into the behavior of deep representations with respect to their geometric ordering and compression and with respect to Domain Adaptation invariance. We have seen that deeper fully connected Neural Networks, trained source-only, can find representations which ignore small domain shifts. Stacking linear layers appears to compress and separate classes instead of overfitting to data. Hence, small domain shifts seem to not lead to significantly differing deep representations. Thus, classification on both domains shows similar performance. Larger domain shifts cannot be overcome like this and need additional Domain Adaptation heuristics and at least some targeted unsupervised Domain Adaptation training,



(a) Unprocessed data of MNIST (left) and MNIST-c (right).

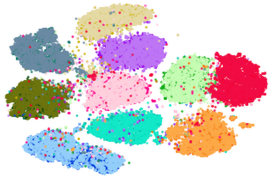
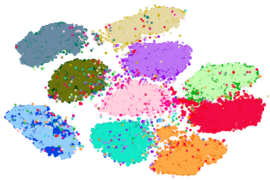
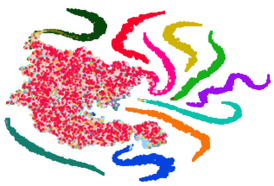
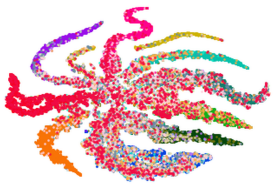
Problem	Depth	FCNN	DANN
MNIST → MNIST-c	2		
	6		
MNIST-c → MNIST	2		
	6		
MNIST-r → MNIST-c	6		

Figure 21: t-SNE visualizations of deep FCNN and DANN representations. The source domain's deep representations are always showing separated and compressed class data clouds. The target domain is either aligned with the source, separated from it or overlapping but not ordered by class. In the networks that showed better performance in Fig. 19, the two domains also seem to align far better. The color code is equivalent to Fig. 16.

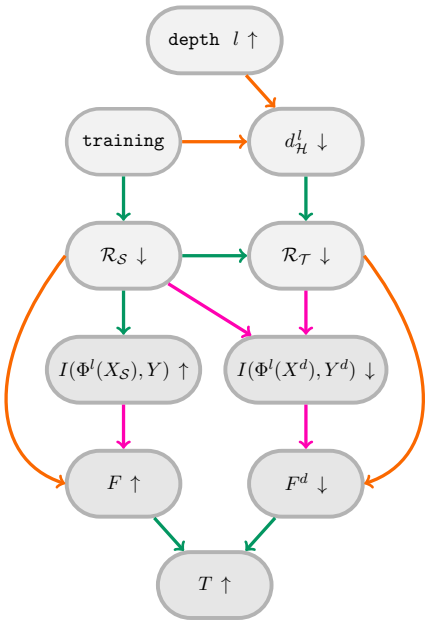
like DANN. We have also seen that t-SNE visualization and the entanglement metric $\mathcal{E}$ can enable discussion about representations and help explain different performances. Nevertheless, neither could be used as a foundation to prove mathematical results. Observation of both can be used as a helpful tool to explain, but not to predict good or bad results.

5. Conclusion and Outlook

"By far, the greatest danger of Artificial Intelligence is that people conclude too early that they understand it."
—Eliezer Yudkowsky [Yud08]

The goal of this thesis was to investigate Sarishvili's Hypothesis 0.1 and the claim that standard training of deep Neural Networks on only one (source) domain already yields some domain adaptation invariance. An idea is that deep representations have a limited capability to ignore features only caused by the domain shift and focus on class-defining features that are similar in both domains. A possibility to describe this Domain Adaptation invariance is found in Information Bottleneck-like approaches.

With the purpose of mathematically investigating the validity of this claim, Proposal 0.2 laid out a step-by-step roadmap to break down the explanation approach into small digestible steps concerning claims, ideas and terminology from classical Machine Learning theory, Information Theory and possible connections of these two fields.
On the right, we reintroduce the infographic of the proposal and color-coded connections that were either **refuted**, **proven** and further investigated regarding applicability to deep Neural Networks, or **empirically investigated** and neither proven nor disproven. To summarize the results we go over them again step by step:



Regarding Machine Learning statements we used the Vapnik-Chervonenkis framework to prove uniform generalization bounds (Theorem 1.21, Corollary 1.23) and a Domain Adaptation bound (Theorem 1.25). We then investigated the validity when applying them to deep Neural Networks and came to the conclusion that empirical measures might be too weak of a predictor for misclassification risk $\mathcal{R}$ and domain distinguishability $d_{\mathcal{H}}$ (Section 2). In terms of theory this is because the VC dimension of Neural Network hypothesis classes is very large and thus realistic sample sizes for applications do not suffice to avoid overfitting in the worst case (Section 2.1). Then, we showed that this worst-case bound for Neural Network generalization given by VC dimension Δ_H is realistic to be tight because they seem to be able to shatter large samples of even noise, can approximate continuous functions on compact support arbitrarily well and need additional problem-specific assumptions to predict the generalization performance

tightly (Section 2.2, Section 2.3). This capability of overfitting makes empirical measures unreliable if the sample size is not comparably large, i.e. for typical practical Neural Network applications.
Furthermore, we proved a connection between misclassification risk and mutual information of data representation and labeling using Fano's inequality (Theorem 3.11, Corollary 3.13). However, we have also shown that Fano's inequality does not vary when using deep data representations instead of initial data (Remark 3.14). This was because the statement $Y \to X \to \Phi_a(X;\theta) \implies H(Y) - I(X;Y) - 1 \leq \mathrm{P}\left(Y \neq \Phi_a(X;\theta)\right)$ does not provide tighter bounds on mutual information when switching X in the markov chain with an intermediate representation $\Phi_a^l(X;\theta)$. Thus, Fano's inequality is not suitable to explain the good performance of deep classifiers. Rather it says, that individual representation that do not preserve information well, cannot attain zero loss anymore. It can evaluate single networks but not whole architectures.
Beyond that we used and motivated fidelity (Definition 3.19) and discussed its value to quantify classification performance on deep representations by discussing the consensus of the literature on the Information Bottleneck framework. The result is that opinions diverge and the IB perspective empirically works on some problems but has issues to generalize to explaining Neural Networks success for general architecture (Section 3.3.1). This direct IB (or equivalently fidelity) perspective is appended in the infographic above by orange skip connections.
Investigating possible information-theoretical explanations for Domain Adaptation invariances did not yield affirming results. A connection of good classification performance on two domains and their indistinguishability measured by low value of $I(\Phi(X^d;\theta), Y^d)$ could be disproven by the construction of a counterexample (Example 3.16).
Further, we showed that a direct connection between fidelity and mutual information can be speculated, but not proven since the value of fidelity also depends on a general representation entropy term changing with variable value. A direct connection needs this additional term to stay constant when changing representations by changing the network parameterization (Lemma 3.23). In conclusion, the information-theoretic part proofs unreliable as a watertight predictor for misclassification risk and Domain Adaptation performance.
Lastly, we empirically investigated deep representations of different data set variants of MNIST in different Domain Adaptation experiments in Section 4. We investigated the influence of depth and showed that the concatenation of multiple linear layers seems to induce representations that overcome small domain shifts and therefore perform well in such Domain Adaptation problems. We then compared regular source-only training to unsupervised Domain Adaptation training by the DANN algorithm (Example 4.1) and showed that the special architecture of DANN enables superior Domain Adaptation performance both in cases with small domain shifts that can be overcome by source only trained deep classifiers and in cases where the simple fully connected Neural Network fails.
During these experiments, we also used visualization and a recently suggested approach found in the literature called manifold entanglement (Example 4.3) to aid interpreta-

tion. We showed that the latter might be used as an aiding measurement to investigate Domain Adaptation problems but might not be useful to evaluate, predict and compare classification performance qualitatively on its own.

Reviewing the literature and providing the different Machine Learning related proofs in a common framework, we showed that classical theory does not suffice to investigate the class of Neural Network classifiers. This emphasizes the possible need for a major shift in how we conceptualize and apply Neural Networks in order to provide theoretical bounds or performance guarantees for application. In the current stage it feels like using Neural Networks is a trial-and-error approach for each new problem and if a classifier fails, it gets enlarged and theoretically even more powerful and prone to overfitting. This approach might meet its limits sooner or later and therefore it is necessary to investigate far more explanatory approaches and novel ideas to extend our knowledge on the background workings of the empirically best-performing classifier class we have nowadays, (deep) Neural Networks. The hypothesis investigated in this thesis might not hold entirely from a mathematical point of view, exceeding the status of an empirically validated idea, but it makes some entrance points to novel research visible.
First, it might be necessary to investigate more and tighter generalization bounds like the few presented in Section 2.4 to simply get a variety of tools to look at deep Neural Networks from different perspectives. Building up on such generalization bounds, there might be guarantees found do distinguish cases, where the value of empirical errors can be used to estimate the true error and cases where it is no good predictor, for example applications where the generalization performance is up to chance. This might lead to more comprehensive theory and possibly new approaches to the training of Neural Networks.
On that note, it is also suggested to further look into IB, fidelity and similar approaches to possibly develop them into sound frameworks with provable relations to classification performance. The information-theoretical viewpoint might have the necessary distance to classical Machine Learning results making it possible to increase understanding of the workings of trained Neural Networks significantly without tripping on the problems classical theory and, as we have shown, uniform generalization bounds show.
The third suggestion is to investigate representations of well and also of poorly performing networks and try to build novel theory thereon. Whether this investigation will be of information-theoretical nature or done via an entanglement perspective or something completely different, is obviously unclear. Nevertheless, since observing Neural Networks simply as a Machine Learning classifier seems to understate its complexity, investigating representations might be a useful viewpoint to find new approaches and possibly prove new results.

References

[ALS22] Md Manjurul Ahsan, Shahana Akter Luna, and Zahed Siddique, *Machine-learning-based disease diagnosis: A comprehensive review*, Healthcare **10** (2022), no. 3.

[Alt12] H.W. Alt, *Lineare funktionalanalysis: Eine anwendungsorientierte einführung*, Masterclass, Springer Berlin Heidelberg, 2012.

[Ama67] Shunichi Amari, *A theory of adaptive pattern classifiers*, IEEE Transactions on Electronic Computers **EC-16** (1967), no. 3, 299–307.

[AS17] Alessandro Achille and Stefano Soatto, *On the emergence of invariance and disentangling in deep representations*.

[BC18] Jacques Bughin and Michael Chui, *Notes from the ai frontier: Modeling the impact of ai on the world economy*.

[BCE+23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang, *Sparks of artificial general intelligence: Early experiments with gpt-4*, 2023.

[BCK+08] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman, *Learning bounds for domain adaptation*, Advances in Neural Information Processing Systems 21 (Cambridge, MA), MIT Press, 2008.

[BDBCP06] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira, *Analysis of representations for domain adaptation.*, vol. 19, 01 2006, pp. 137–144.

[BGKP22] Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen, *The modern mathematics of deep learning*, Mathematical Aspects of Deep Learning, Cambridge University Press, dec 2022, pp. 1–111.

[BHLM17] Peter L. Bartlett, Nick Harvey, Chris Liaw, and Abbas Mehrabian, *Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks*, 2017.

[BLW96] Peter L. Bartlett, Philip M. Long, and Robert C. Williamson, *Fat-shattering and the learnability of real-valued functions*, Journal of Computer and System Sciences **52** (1996), no. 3, 434–452.

[BMM98] Peter Bartlett, Vitaly Maiorov, and Ron Meir, *Almost linear vc dimension bounds for piecewise polynomial networks*, Advances in Neural Information Processing Systems (M. Kearns, S. Solla, and D. Cohn, eds.), vol. 11, MIT Press, 1998.

[Bra15] Google Brain, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015, Software available from tensorflow.org.

[Bre82] Benjamin Brewster, *Theory and practice*, The Yale Literary Magazine **47** (1882), 202.

[BWS15] Pratik Brahma, Dapeng Wu, and Yiyuan She, *Why deep learning works: A manifold disentanglement perspective*, IEEE Transactions on Neural Networks and Learning Systems **27** (2015), 1–12.

[CT06] Thomas M. Cover and Joy A. Thomas, *Elements of information theory (wiley series in telecommunications and signal processing)*, Wiley-Interscience, USA, 2006.

[Cyb89] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals, and Systems (MCSS) **2** (1989), no. 4, 303–314.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, International Conference on Learning Representations, 2021.

[dCZWG20] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J. Gordon, *Domain adaptation with conditional distribution matching and generalized label shift*, CoRR **abs/2003.04475** (2020).

[DGL96] Luc Devroye, László Györfi, and Gábor Lugosi, *A probabilistic theory of pattern recognition*, Stochastic Modelling and Applied Probability, vol. 31, Springer, 1996.

[DR17] Gintare Karolina Dziugaite and Daniel M. Roy, *Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data*, CoRR **abs/1703.11008** (2017).

[DS19] Luke Nicholas Darlow and Amos Storkey, *What information does a resnet compress?*, 2019.

[ES15] Ronen Eldan and Ohad Shamir, *The power of depth for feedforward neural networks*, CoRR **abs/1512.03965** (2015).

[Fan61] R. Fano, *Transmission of information: A statistical theory of communications*, The MIT Press, Cambridge, MA, 1961.

[FDCBA14] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim, *Do we need hundreds of classifiers to solve real world classification problems?*, J. Mach. Learn. Res. **15** (2014), no. 1, 3133–3181.

[Fer82] K. Ferentinos, *On tchebycheff's type inequalities*, 1982.

[FMN13] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan, *Testing the manifold hypothesis*, 2013.

[Fuk69] Kunihiko Fukushima, *Visual feature extraction by a multilayered network of analog threshold elements*, IEEE Transactions on Systems Science and Cybernetics **5** (1969), no. 4, 322–333.

[GUA+16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky, *Domain-adversarial training of neural networks*, 2016.

[Hoe63] Wassily Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American statistical association **58** (1963), no. 301, 13–30.

[Huf52] David A Huffman, *A method for the construction of minimum-redundancy codes*, Proceedings of the IRE **40** (1952), no. 9, 1098–1101.

[HV10] Siu-Wai Ho and Sergio Verdú, *On the interplay between conditional entropy and error probability*, IEEE Transactions on Information Theory **56** (2010), no. 12, 5930–5942.

[IL67] A G Ivakhnenko and V G Lapa, *Cybernetics and forecasting techniques*, Modern analytic and computational methods in science and mathematics, North-Holland, New York, NY, 1967, Trans. from the Russian, Kiev, Naukova Dumka, 1965.

[KB14] Diederik P. Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, 2014.

[KBK22] K. Kawaguchi, Y. Bengio, and L. Kaelbling, *Generalization in deep learning*, Mathematical Aspects of Deep Learning, Cambridge University Press, dec 2022, pp. 112–148.

[KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems (F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[KSH12b] ———, *Imagenet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems (F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[Lau14] Fabien Lauer, *Vc-dimension of hyperplanes*, https://mlweb.loria.fr/book/en/VCdimhyperplane.html, 2014, Accessed: 2023-11-09.

[LJQ20] Wenling Li, Bo Jin, and Yu Quan, *Review of research on text sentiment analysis based on deep learning*, OALib **07** (2020), 1–8.

[LLPS93] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.*, Neural Networks **6** (1993), no. 6, 861–867.

[LRM+12] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng, *Building high-level features using large scale unsupervised learning*, 2012.

[McA99] David A. McAllester, *Pac-bayesian model averaging*, Proceedings of the Twelfth Annual Conference on Computational Learning Theory (New York, NY, USA), COLT '99, Association for Computing Machinery, 1999, p. 164–170.

[MCA+23] Samaneh Madanian, Talen Chen, Olayinka Adeleye, John Michael Templeton, Christian Poellabauer, Dave Parry, and Sandra L. Schneider, *Speech emotion recognition using machine learning — a systematic review*, Intelligent Systems with Applications **20** (2023), 200266.

[MH20] Wafa Mellouk and Wahida Handouzi, *Facial emotion recognition using deep learning: review and insights*, Procedia Computer Science **175** (2020), 689–694, The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology.

[MMR09] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh, *Domain adaptation: Learning bounds and algorithms*, 2009.

[MP43] Warren Mcculloch and Walter Pitts, *A logical calculus of ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics **5** (1943), 127–147.

[MP99] Vitaly Maiorov and Allan Pinkus, *Lower bounds for approximation by mlp neural networks*, Neurocomputing **25** (1999), 81–91.

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of machine learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2018.

[MS23] Gopinath M. and Sibi Chakkaravarthy Sethuraman, *A comprehensive survey on deep learning based malware detection techniques*, Computer Science Review **47** (2023), 100529.

[NCS+21] Morteza Noshad, Jerome Choi, Yuming Sun, Alfred Hero, and Ivo Dinov, *A data value metric for quantifying information content and utility*, Journal of Big Data **8** (2021).

[NTG+21] A. Tuan Nguyen, Toan Tran, Yarin Gal, Philip H. S. Torr, and Atilim Günes Baydin, *KL guided domain adaptation*, CoRR **abs/2106.07780** (2021).

[PLZ23] Svetlana Pavlitska, Nico Lambing, and J. Marius Zöllner, *Adversarial attacks on traffic sign recognition: A survey*, 2023.

[Pop35] Tiberiu Popoviciu, *Sur les équations algébriques ayant toutes leurs racines réelles*.

[Ros58] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, Psychological Review **65** (1958), no. 6, 386–408.

[RV17] Anand Rao and Gerard Verweij, *Sizing the prize what's the real value of ai for your business and how can you capitalise?*, https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf, 2017, Accessed: 2023-09-30.

[RW17] Waseem Rawat and Zenghui Wang, *Deep convolutional neural networks for image classification: A comprehensive review*, Neural Computation **29** (2017), no. 9, 2352–2449.

[Sag97] Carl Sagan, *The demon-haunted world: science as a candle in the dark.*, 1997.

[Sau72] N Sauer, *On the density of families of sets*, Journal of Combinatorial Theory, Series A **13** (1972), no. 1, 145–147.

[SBD+18] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox, *On the information bottleneck theory of deep learning*, International Conference on Learning Representations, 2018.

[Sha48] C. E. Shannon, *A mathematical theory of communication*, The Bell System Technical Journal **27** (1948), no. 3, 379–423.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning - from theory to algorithms.*, Cambridge University Press, 2014.

[ST17] Ravid Shwartz-Ziv and Naftali Tishby, *Opening the black box of deep neural networks via information*, CoRR **abs/1703.00810** (2017).

[TKOSK23] Tala Talaei Khoei, Hadjar Ould Slimane, and Naima Kaabouch, *Deep learning: systematic review, models, challenges, and research directions*, Neural Computing and Applications **35** (2023).

[TPB00] Naftali Tishby, Fernando C. Pereira, and William Bialek, *The information bottleneck method*, 2000.

[Tuk77] John W. Tukey, *Exploratory data analysis*, Addison-Wesley, 1977.

[Vap99] Vladimir N. Vapnik, *The nature of statistical learning theory*, second ed., Springer, November 1999.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability & Its Applications **16** (1971), no. 2, 264–280.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton, *Visualizing data using t-SNE*, Journal of Machine Learning Research **9** (2008), 2579–2605.

[WD21] Mei Wang and Weihong Deng, *Deep face recognition: A survey*, Neurocomputing **429** (2021), 215–244.

[WJ74] Paul Werbos and Paul John, *Beyond regression : new tools for prediction and analysis in the behavioral sciences /*.

[Yeu91] R.W. Yeung, *A new outlook on shannon's information measures*, IEEE Transactions on Information Theory **37** (1991), no. 3, 466–474.

[Yil15] Olcay Yildiz, *Vc-dimension of univariate decision trees*, IEEE transactions on neural networks and learning systems **26** (2015).

[Yud08] Eliezer Yudkowsky, *Artificial Intelligence as a positive and negative factor in global risk*, Global Catastrophic Risks, Oxford University Press, 2008.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, *Understanding deep learning requires rethinking generalization*, 2017.

[ZZXW19] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu, *Object detection with deep learning: A review*, IEEE Transactions on Neural Networks and Learning Systems **30** (2019), no. 11, 3212–3232.

[ZZY+22] Mengli Zhang, Gang Zhou, Wanting Yu, Ningbo Huang, and Wenfen Liu, *A comprehensive survey of abstractive text summarization based on deep learning*, Computational Intelligence and Neuroscience **2022** (2022), 1–21.

A. Appendix A - Inequalities

In this section, we shortly present well-known inequalities used throughout the proofs of this thesis.

Theorem A.1 (Chebychev's Inequality; [Fer82, Sec. 4])**.**
Let $m \in \mathbb{N}$ and let $X = (X_1, \ldots, X_m)$ be a random variable in $\mathbb{R}^m$. Let μ, σ corresponding m-dimensional expected value and variance, such that $\mu_i = \mathbb{E}[X_i]$ and $\sigma_i = Var(X_i)$. Then

$$P(\|X - \mu\| \geq k\|\sigma\|) \leq \frac{1}{k^2}.$$

Theorem A.2 (Hoeffding's Inequality; [Hoe63, Theorem 2])**.**
Let $X_1, \ldots, X_n$ be independent bounded random variables with $a_i \leq X_i \leq b_i$ a.s. Let $S_n = X_1 + \cdots + X_n$ and $\varepsilon > 0$. Then, it holds

$$\mathrm{P}\left(\left|S_n - \mathrm{E}\left[S_n\right]\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Theorem A.3 (Popoviciu's Inequality on Variances; [Pop35])**.**
Let X be a bounded random variable with $a \leq X \leq b$. Then,

$$Var(X) \leq \frac{1}{4}(b-a)^{\cdot}2$$

Theorem A.4 (Jensen's Inequality)**.**
For any random Variable X on a subspace $\mathcal{X} \subset \mathbb{R}$ and convex functions $f\colon \mathbb{R} \to \mathbb{R}$ it holds that

$$\mathbb{E}\left[f(X)\right] \geq f\left(\mathbb{E}\left[X\right]\right).$$

Furthermore, if f is strictly convex, then equality holds if and only if X is constant.

B. Appendix B - Additional Proofs

B.1. Generalization Bound

Example B.1 (Symmetrization by Random Signs)**.**
Let σ be a uniformly distributed random variable on $\{-1,+1\}^n$ and let $(X,Y)^n$ and $(X',Y')^n$ be labeled i.i.d. samples drawn according to some distribution $P_{\mathcal{X},\mathcal{Y}}$. Then

$$\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{h(X_i')\neq Y_i'\}-\mathbb{1}\{h(X_i)\neq Y_i\}\geq\frac{\varepsilon}{2}\right\}\right]$$
$$=\frac{1}{2^n}\sum_{\sigma\in\{-1,1\}^n}\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\Big(\mathbb{1}\{h(X_i')\neq Y_i'\}-\mathbb{1}\{h(X_i)\neq Y_i\}\Big)\geq\frac{\varepsilon}{2}\right\}\right]$$

Proof. Let $\sigma\in\{-1,+1\}^n$ first be arbitrary and fixed. Let $\mathrm{I}^{(1)}$ and $\mathrm{I}^{(2)}$ be disjoint index sets such that $\mathrm{I}^{(1)}\cup\mathrm{I}^{(2)}=1,\ldots,n$ and $\sigma_i=1,$ if $i\in\mathrm{I}^{(1)}$ and $\sigma_i=-1,$ if $i\in\mathrm{I}^{(2)}$. Then simply renaming the sample points corresponding to $\mathrm{I}^{(2)}$ yields

$$\mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{h(X_i')\neq Y_i'\}-\mathbb{1}\{h(X_i)\neq Y_i\}\geq\frac{\varepsilon}{2}\right\}\right]$$
$$=\int_{(x,y)^n,(x',y')^n\in(\mathcal{X}\times\mathcal{Y})^{2n}}\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{h(x_i')\neq y_i'\}-\mathbb{1}\{h(x_i)\neq y_i\}\geq\frac{\varepsilon}{2}\right\}$$
$$\cdot\underbrace{\prod_{i=1}^{n}\Big(\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)\Big)}_{=\text{ joint probability due to independence}}\mathrm{d}(x,y)^n\mathrm{d}(x',y')^n$$
$$=\int_{(x,y)^n,(x',y')^n\in(\mathcal{X}\times\mathcal{Y})^{2n}}\mathbb{1}\left\{\sup_{h\in\mathcal{H}}\frac{1}{n}\Big(\sum_{i\in\mathrm{I}^{(2)}}\mathbb{1}\{h(x_i)\neq y_i\}-\mathbb{1}\{h(x_i')\neq y_i'\}\right.$$
$$\left.+\sum_{i\in\mathrm{I}^{(1)}}\mathbb{1}\{h(x_i')\neq y_i'\}-\mathbb{1}\{h(x_i)\neq y_i\}\Big)\geq\frac{\varepsilon}{2}\right\}$$
$$\cdot\prod_{i\in\mathrm{I}^{(2)}}\Big(\mathrm{P}\left((X_i,Y_i)=(x_i',y_i'),(X_i',Y_i')=(x_i,y_i)\right)\Big)$$
$$\cdot\prod_{i\in\mathrm{I}^{(1)}}\Big(\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)\Big)\mathrm{d}(x,y)^n\mathrm{d}(x',y')^n.$$

Since all of the sampled data points are independently and identically distributed the joint distribution of (x_i,y_i) and (x_i',y_i') is symmetrical for all i, i.e.

$$\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)=\mathrm{P}\left((X_i,Y_i)=(x_i',y_i'),(X_i',Y_i')=(x_i,y_i)\right)$$

and therefore we can change them back and obtain an expected value including the signs.

$$
\begin{aligned}
\dots = \int_{(x,y)^n,(x',y')^n \in (\mathcal{X}\times\mathcal{Y})^{2n}} & \mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n}\Big(\sum_{i\in \mathrm{I}^{(2)}} -1\Big(\mathbb{1}\{h(x_i')\neq y_i'\} - \mathbb{1}\{h(x_i)\neq y_i\}\Big) \\
& + \sum_{i\in \mathrm{I}^{(1)}} 1\Big(\mathbb{1}\{h(x_i')\neq y_i'\} - \mathbb{1}\{h(x_i)\neq y_i\}\Big)\Big) \geq \frac{\varepsilon}{2}\Big\} \\
& \cdot \prod_{i\in \mathrm{I}^{(2)}} \Big(\mathrm{P}\left((X_i,Y_i)=(x_i',y_i'),(X_i',Y_i')=(x_i,y_i)\right)\Big) \\
& \cdot \prod_{i\in \mathrm{I}^{(1)}} \Big(\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)\Big)\, \mathrm{d}(x,y)^n \mathrm{d}(x',y')^n \\
= \int_{(x,y)^n,(x',y')^n \in (\mathcal{X}\times\mathcal{Y})^{2n}} & \mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n}\Big(\sum_{i=1}^{n} \sigma_i\Big(\mathbb{1}\{h(x_i')\neq y_i'\} - \mathbb{1}\{h(x_i)\neq y_i\}\Big) \geq \frac{\varepsilon}{2}\Big\} \\
& \cdot \prod_{i\in \mathrm{I}^{(2)}} \Big(\mathrm{P}\left((X_i,Y_i)=(x_i',y_i'),(X_i',Y_i')=(x_i,y_i)\right)\Big) \\
& \cdot \prod_{i\in \mathrm{I}^{(1)}} \Big(\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)\Big)\, \mathrm{d}(x,y)^n \mathrm{d}(x',y')^n \\
= \int_{(x,y)^n,(x',y')^n \in (\mathcal{X}\times\mathcal{Y})^{2n}} & \mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n}\Big(\sum_{i=1}^{n} \sigma_i\Big(\mathbb{1}\{h(x_i')\neq y_i'\} - \mathbb{1}\{h(x_i)\neq y_i\}\Big) \geq \frac{\varepsilon}{2}\Big\} \\
& \cdot \prod_{i=1}^{n} \Big(\mathrm{P}\left((X_i,Y_i)=(x_i,y_i),(X_i',Y_i')=(x_i',y_i')\right)\Big)\, \mathrm{d}(x,y)^n \mathrm{d}(x',y')^n \\
= \mathbb{E}_{(X,Y)^n,(X',Y')^n} & \left[\mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\Big\}\right]
\end{aligned}
$$

Since there are 2^n sign combinations and none changes the expected value as already discussed, the main proof is straightforward.

$$
\begin{aligned}
& \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\} \geq \frac{\varepsilon}{2}\Big\}\right] \\
& = \frac{1}{2^n} \sum_{\sigma\in\{-1,1\}^n} \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\} \geq \frac{\varepsilon}{2}\Big\}\right] \\
& = \frac{1}{2^n} \sum_{\sigma\in\{-1,1\}^n} \mathbb{E}_{(X,Y)^n,(X',Y')^n}\left[\mathbb{1}\Big\{ \sup_{h\in\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i\Big(\mathbb{1}\{h(X_i')\neq Y_i'\} - \mathbb{1}\{h(X_i)\neq Y_i\}\Big) \geq \frac{\varepsilon}{2}\Big\}\right]
\end{aligned}
$$

□

B.2. VC-dimension

Theorem B.2 (VC DIMENSION OF LINEAR CLASSIFIERS)**.**
Linear (hyperplane) classifiers consider two classes separated by affine linear hyperplanes $\{x \mid w^t x + b = 0\}$*. For these linear classifiers in* $\mathbb{R}^m$*, it is easy to see that*

$$\Delta_{\mathcal{H}} = m + 1.$$

Proof. The upcoming proof was published in [Lau14].

- "$\Delta_{\mathcal{H}} \geq m + 1$": Consider the set of all unit vectors and the 0-vector $X = \{\mathbf{0}, e_1, \ldots, e_m\}$. Then, given binary labels $y_0, \ldots, y_m$, we can define the following simple linear threshold classifier:

$$h(x) = \mathbb{1}_{\mathbb{R}_{\geq 0}} \left(\left\langle 2 \begin{pmatrix} y_1 - \frac{1}{2} \\ \vdots \\ y_m - \frac{1}{2} \end{pmatrix}, x \right\rangle + y_0 - \frac{1}{2} \right)$$

- "$\Delta_{\mathcal{H}} < m + 2$": Let $X = \{x_1, \ldots, x_{m+2}\} \subset \mathbb{R}^m$ and assume for any labeling $\mathbf{y}^{(j)} \in \{0,1\}^{m+2}$ there exist a weight w_j and bias b_j such that

$$y_i^{(j)} = \mathbb{1}_{\mathbb{R}_{\geq 0}} \left(w_j^t \, x_i + b_j \right)$$

Define $M \coloneqq \begin{pmatrix} x_1 & \ldots & x_{m+2} \\ 1 & \ldots & 1 \end{pmatrix}$ and $W \coloneqq \begin{pmatrix} w_1 & \ldots & w_{2^{m+2}} \\ b_1 & \ldots & b_{2^{m+2}} \end{pmatrix}$. Then it holds

$$\mathbb{1}_{\mathbb{R}_{\geq 0}} \left(M^t W \right) = \left(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(2^{m+2})} \right).$$

We show a contradiction by investigating the rank of $M^t W$.
For all $\alpha \in \mathbb{R}^{m+2}$ there exists a row of $(M^t W)_{:,j(\alpha)}$ with the same signs as α. Therefore

$$\alpha^t M^t W = \mathbf{0} \implies \alpha^t (M^t W)_{:,j(\alpha)} = 0 \implies \alpha = \mathbf{0}.$$

Thus the rows are linearly independent and $rank(M^t W) = m + 2$. However, at the same time

$$rank(M^t W) \leq rank(M) \leq m + 1$$

due to dimensionality. ↯
As the set X was chosen arbitrarily this proves that no $m+2$ points in $\mathbb{R}^m$ can be shattered by affine linear threshold classifiers. □

Corollary B.3 (Main Lower VC-Dimension Bound; [BHLM17, Theorem 3])**.** *There exists a constant C such that for any W, L with $W > CL > C^2$ there is a network architecture with*

- *depth at most L,*
- *number of parameters at most W and*
- *ReLU activation function*

such that the architecture has VC-dimension at least $\frac{WL}{C}\log(\frac{W}{L})$.

Proof. The proof boils down to applying Lemma 2.6 with fitting choices of the parameters m_1, m_2 and r dependent on L and W such that the network constructed by Bartlett et al. fulfills our claim.
Assume C has a value high enough such that L and $\frac{W}{L}$ are sufficiently large for the upcoming arguments. Choose

$$\begin{aligned} r &\coloneqq \frac{1}{2}\log\frac{W}{L}, \\ m_1 &\coloneqq r\left(\frac{L-7}{5}\right) \text{ and} \\ m_2 &\coloneqq W - 5m_1 2^r. \end{aligned}$$

Note that m_2 is positive for large L and $\frac{W}{L}$ as $5m_1 2^r$ grows with rate $\sqrt{WL}\log\left(\frac{W}{L}\right)$ and thus slower than $\sqrt{WL}\sqrt{\frac{W}{L}} = W$.
Prove the claims separately by applying Lemma 2.6.

- First, consider the *depth*. Lemma 2.6 constructs a network with depth $3 + 5\lceil\frac{m_1}{r}\rceil$. Insertion of m_1 yields depth less or equal to as long as L is at least 3. L

$$3 + 5\left\lceil\frac{L-7}{5}\right\rceil \leq 3 + 5\frac{L-3}{5} = L$$

- Further the number of *weights* in the previous construction is less or equal to W.

$$\hat{W} \coloneqq 2 + m_2 + 4m_1 + k((11+r)2^r + 2r + 2) \leq W$$

  For the purpose of proving this, insert m_1 and m_2 and reorder the terms of $W - \hat{W}$ first.

Afterward one can prove the non-negativity of $W - \hat{W}$.

$$
\begin{aligned}
& W - \hat{W} \\
& = -2 + 5r\left(\frac{L-7}{5}\right)2^r - 4r\left(\frac{L-7}{5}\right) - \left\lceil\frac{L-7}{5}\right\rceil\left((11+r)2^r + 2r + 2\right) \\
& = 2^r\left(r\,(L-7) - \left\lceil\frac{L-7}{5}\right\rceil(11+r)\right) - r\left(\frac{4}{5}\,(L-7) + 2\left\lceil\frac{L-7}{5}\right\rceil\right) - 2 - 2\left\lceil\frac{L-7}{5}\right\rceil \\
& \geq 2^r\left(r\,(L-7) - \left(\frac{L-3}{5}\right)(11+r)\right) - r\left(\frac{4}{5}\,(L-7) + 2\left(\frac{L-3}{5}\right)\right) - 2 - 2\frac{L-3}{5}
\end{aligned}
$$

Restricting to the most dominant terms, when L and $\frac{W}{L}$ are large, leaves us with

$$\simeq 2^r\left(rL - \frac{L}{5}r\right) = \left(\frac{W}{L}\right)^{1/2}\frac{4L}{5}\log\frac{W}{L} = \frac{4}{5}\sqrt{WL}\log\frac{W}{L}$$

In this case, for sufficiently large sized L and $\frac{W}{L}$, we obtain $W \geq \hat{W}$.

- It remains to prove the lower bound on the *VC dimension*, i.e. $m_1\, m_2 \geq \frac{WL}{C}\log\left(\frac{W}{L}\right)$. We have for some C

$$m_1\, m_2 = r\left(\frac{L-7}{5}\right)\left(W - r(L-7)2^r\right) \geq WL\log\frac{W}{L} - \frac{C-1}{C}WL\log\frac{W}{L}$$

once again using the dominance of the term $WL\log\frac{W}{L}$ for large parameters. Thus we obtain the lower bound as claimed.

□

B.3. Information Theory

Proposition B.4 (Properties of Entropy and Mutual Information)**.**
Let X, Y and Z be random variables on the sets $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$. Further let P *and* Q *be two distributions on $\mathcal{X}$. Then one can prove the following bounds.*

(a) Kullback-Leibler divergence is always non-negative, i.e.

$$KL\,(\mathrm{P}||\mathrm{Q}) \geq 0$$

where we have equality if and only if $\mathrm{P}(x) = \mathrm{Q}(x)$ holds for all x.

(b) Depending on $\mathcal{X}$, we can derive bounds for entropy.

(i) Entropy of discrete random variables with finite support can be bound. If $\mathcal{X}$ finite, then

$$0 \leq H(X) \leq \log_2|\mathcal{X}| \tag{B.1}$$

and equality $H(X) = \log_2|\mathcal{X}|$ holds if and only if X is uniformly distributed on $\mathcal{X}$.

(ii) For continuous random variables this does not hold. For any $c > 0$ there exist random variables X and Y such that $H(X) < -c$ and $H(Y) > c$.

(c) Mutual information is also non-negative, i.e.

$$I(X;Y) \geq 0$$

with equality $I(X;Y) = 0$ if and only if X and Y are independent.

Further important properties are the following.

(d) ***Conditioning is a monotone operation*** *for information theoretical measures, i.e.*

$$H(X|Y) \leq H(X).$$

Equality holds iff X and Y are independent.

(e) Entropy is a ***concave*** *functional w.r.t. to underlying probability distributions. This means, if $X \sim P_X$ and $Y \sim P_Y$, then it holds for all $Z \sim \alpha P_X + (1-\alpha)P_Y$ with $\alpha \in [0,1]$ that*

$$H(Z) \geq \alpha H(X) + (1-\alpha)H(Y)$$

(f) There are ***chain rules*** *for entropy and mutual information. Analogous to the probability distribution chain rule it holds*

$$H(X,Y) = H(Y|X) + H(X)$$

and

$$I(X,Y;Z) = I(Y;Z|X) + I(X;Z)$$

and similar for larger numbers of random variables.

For these and even more properties of information-theoretic measures, refer to [CT06].
Proof.

(a) The positivity of the Kullback-Leibler divergence is a direct consequence of Jensen's inequality [Appendix A, Theorem A.4]. The Logarithm function is strictly concave in the positive real numbers. Thus

$$\begin{aligned} KL(P||Q) &= \mathbb{E}_{X\sim P}\left[\log_2 \frac{P(X)}{Q(X)}\right] = \mathbb{E}_{X\sim P}\left[-\log_2 \frac{Q(X)}{P(X)}\right] \\ &\geq -\log_2 \mathbb{E}_{X\sim P}\left[\frac{Q(X)}{P(X)}\right] \\ &= -\log_2 \int_{\mathcal{X}} P(x)\frac{Q(x)}{P(x)}\,\mathrm{d}x \\ &= -\log_2 1 = 0 \end{aligned}$$

Due to strict concavity of the logarithm, this calculation holds with equality if and only if $\frac{Q(x)}{P(x)}$ is constant in x. Since both are probability measures and thus normalized, it must hold $\frac{Q(x)}{P(x)} = 1$ for all x.

(b) (i) Let X be a discrete random variable on $\mathcal{X} = \{x_1, \ldots, x_n\}$. Then, first of all $\frac{1}{\mathrm{P}(x_i)} \geq 1$ for all i in $\{1, \ldots, n\}$ and thus,

$$H(X) = \sum_{i=1}^{n} \mathrm{P}(x_i) \log \frac{1}{\mathrm{P}(x_i)} \geq 0.$$

For the upper bound consider the uniform distribution $U_{\mathcal{X}}(x_i) = \frac{1}{n}$ for all i in $\{1, \ldots, n\}$. Then

$$\begin{aligned} KL(\mathrm{P}||U_{\mathcal{X}}) &= \sum_{i=1}^{n} \mathrm{P}(x_i) \log_2 \frac{\mathrm{P}(x_i)}{U_{\mathcal{X}}(x_i)} \\ &= \sum_{i=1}^{n} \mathrm{P}(x_i) \log_2 \mathrm{P}(x_i) + \sum_{i=1}^{n} \mathrm{P}(x_i) \log_2 \frac{1}{U_{\mathcal{X}}(x_i)} \\ &= -H(X) + \log_2 n. \end{aligned}$$

Statement (a) now implies

$$H(X) \leq \log_2 n = \log_2 |\mathcal{X}|$$

with equality if and only if $\mathrm{P}(x_i) = U_{\mathcal{X}}(x_i)$ for all i in $\{1, \ldots, n\}$.

(ii) W.l.o.g. $\mathcal{X} \subset \mathbb{R}$ For $t > 0$ and $X \sim U(0, t)$ observe

$$H(X) = \int_0^t \frac{1}{t} \log t \, \mathrm{d}x = \log t.$$

Thus, the image space of entropy covers the whole space $\mathbb{R}$.
The same idea can be extended to higher dimensional random variables.

(c) Both claims are a direct consequence of (a) when rewriting mutual information as

$$I(X;Y) = KL(\mathrm{P}_{X,Y}||\mathrm{P}_X\mathrm{P}_Y).$$

(d) Due to $I(X|Y) = H(X) - H(X|Y)$, Statement (c) directly implies both claims.

(e) Concavity of Entropy can be easily checked using the concavity of the logarithm.

(f) Calculation yields

$$\begin{aligned} H(X,Y) &= -\iint_{\mathcal{X}\,\mathcal{Y}} \mathrm{P}_{X,Y}(x,y) \log_2 \mathrm{P}_{X,Y}(x,y) \, \mathrm{d}y \, \mathrm{d}x \\ &= -\iint_{\mathcal{X}\,\mathcal{Y}} \mathrm{P}_{X,Y}(x,y) \log_2 \mathrm{P}_X(x) \, \mathrm{d}y \, \mathrm{d}x - \iint_{\mathcal{X}\,\mathcal{Y}} \mathrm{P}_{X,Y}(x,y) \log_2 \mathrm{P}_{Y|X}(y|x) \, \mathrm{d}y \, \mathrm{d}x \\ &= -\int_{\mathcal{X}} \mathrm{P}_X(x) \log_2 \mathrm{P}_X(x) \, \mathrm{d}x + H(Y|X) \\ &= H(X) + H(X|Y). \end{aligned}$$

Similarly $H(X,Y|Z) = H(X|Z) + H(Y|X,Z)$. Therefore,

$$\begin{aligned} I(X,Y;Z) &= H(X,Y) - H(X,Y|Z) \\ &= H(X) + H(Y|X) - H(X|Z) - H(Y|X,Z) \\ &= I(X;Z) + I(Y;Z|X). \end{aligned}$$

□

C. Appendix C - Experimental Setup

C.1. The Networks

Below we describe the architectures of the used Networks in Section 4. Note that training was done for 50-200 Epochs depending on the task with varying training batch sizes. We did not use early stopping or training until convergence as it might diminish results. For example, remember, in IB it was claimed that the second compression phase of training was important for generalization. Such phases might be skipped when early stopping.

Investigation of Depth The cooking recipe to set up a Neural Network as in Section 4.3.1 is to append the following hidden layers between input and output in the exact order.
The first hidden layer is a convolutional layer with 64 convolution-kernels of size 5×5. Then up to three convolutional layers with 32 convolution-kernels of size 3×3 respectively are appended. After this only linear layers follow. Their output sizes are 2048, 512, and then 128 after this every layer has output size 64, i.e. the data dimension is not compressed anymore until we reach the output layer.
The output layer always outputs dimension 10. Between all layers, we include dropout (30%) and batch normalization regularization.

Investigation of DANN In the setup for the experiment in Section 4.3.2 we use the following layers. A convolutional layer with 32 convolution-kernels of size 3×3, followed by a convolutional layer with 64 convolution-kernels of size 5×5. Then four linear layers with output size 100 and the classification layer with output dimension 10.
The additional domain discrimination architecture for DANN is appended on the output of the convolutional layer and consists of a gradient reversal layer, i.e. forward evaluation is the identity function but backpropagation switches the gradient sign, followed by a linear layer with output dimension 100, and then the domain prediction layer with output dimension 2. Between all layers, we include dropout(30%) and batch normalization regularization.

D. Visualization

Two tools for the visualizations are not self-explanatory and might require some introduction.

D.1. Box Plots

We orientate on the definition by John Tukey ([Tuk77, Sec. 4.3.3]) Box plots are a tool to easily visualize a set of approximately normally distributed, one-dimensional points and their spread. The construction to visualize a finite non-empty set $X \subset \mathbb{R}$ is done as follows:

- Calculate the mean value $Q_2 \in X$(below emphasized in red).
- We calculate $Q_1 \in X$ and $Q_3 \in X$ as quantile points of X, such that both in $(-\infty, Q_1]$ and $[Q_3, \infty)$ are 25% of data points respectively. These are the upper and lower boundary of the box.
- Similarly we calculate the largest smallest points in the interval $[Q_1 - 1.5(Q_3 - Q_1), Q_3 + 1.5(Q_3 - Q_1)]$ (in the plot below emphasized on the x-axis and by the pink interval going through the data points) and draw the so-called "whiskers" extending the box.
- Points not inside the whiskers or the box are outliers and typically plotted as discrete points.

An example might look like: