Causal Regularization in Deep Learning Using the Average Causal Effect

Kathrin Khadra

Master's thesis

Causal Regularization in Deep Learning Using the Average Causal Effect

Kathrin Khadra

19. November 2021

Chair of Data Processing
Technische Universität München

Kathrin Khadra. *Causal Regularization in Deep Learning Using the Average Causal Effect.* Master's thesis, Technische Universität München, Munich, Germany, 2021.

Supervised by Prof. Dr.-Ing. Klaus Diepold and M.Sc. Matthias Kissel; submitted on 19. November 2021 to the Department of Electrical and Computer Engineering of the Technische Universität München.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

Abstract

Causal Interpretability aims to make decisions of algorithms interpretable by investigating what would have happened under different circumstances. These varying circumstances can be manipulations on the algorithm to assess its causality. In this thesis, I include the causal interpretability mechanism called the Average Causal Effect (ACE) into the training of a neural net. To assess the causality of the model, the ACE uses so-called interventions to manipulate the neural net. The goal is to determine more causal weights and biases during the model training. Using this approach, I evaluate whether including a causal interpretability mechanism as a regularization increases the overall causality of the model. Moreover, I investigate whether this improvement in causality also impacts the generalization ability of the neural net. The developed approach is compared to a standard neural net as well as L1 and L2 regularized neural nets. Furthermore, I conduct these experiments with well-balanced datasets, datasets with a prior probability shift, and datasets with a covariate shift. For all datasets, the results show that the presented causal regularization approach is able to improve the overall causality of the neural net. However, the distribution of the shifted training data highly affects the generalization ability. With an increasing variance of the distribution, the developed approach shows significantly lower test Mean Squared Errors than for training data with less variance. This is because the interventions applied by the ACE depend on the variance of the training data distribution.

Contents

1 Introduction

Interpretable Machine Learning has been a huge field of interest to the research community for several years. Overall, having insights into the way Machine Learning algorithms make choices, can have a number of benefits and help in many ways. Particularly, training algorithms could benefit from interpretable learning. Right now, the current state of training algorithms is based largely on trial and error as well as past experience. This can make them incomprehensible. Furthermore, the resulting model suffers from the risk of making unexpected decisions. This means, with interpretable learning trustworthy models can be generated. Thus, for use cases where decisions can affect people's survival, like healthcare or autonomous driving, interpretable processes can be a game-changer. Furthermore, by enhancing the understanding of the training process, researchers, engineers, and data scientists can make sure that the generated models follow fairness, security and human understanding. This is important due to the European Union passing the "Rights to Explanation" regulation as well as the recent call for diversity and inclusion in AI [14].

In general, interpretable learning falls under the area of explainable artificial intelligence (XAI) [12]. XAI aims to make the models generated by artificial intelligence more trustworthy and understandable [12]. Gilpin et al. state that while XAI might be able to describe the reasoning of a system, interpretable learning makes AI more trustworthy without describing that reasoning [12]. Furthermore, Pearl claims that to achieve explainability one needs a causal model of the environment, as explainability cannot be achieved only by association [34]. Generally, one can explain causality with two variables X and Y. The variable X is a direct cause of the variable Y if a manipulation on X exists, so that X and Y are associated [47]. All other variables are fixed at some value for this scenario [47]. Taking the definition for interpretability and causality one arrives at the subcategory causal interpretability. Causal interpretability aims to answer the question of why a decision was made by a system instead of another [29]. This means, it makes decisions interpretable by gaining insights on what decision would have been made under alternative circumstances [29]. One can achieve that in two ways. First, one can make already trained models interpretable [29]. Secondly, one can create inherently interpretable models by making them interpretable during the training [29].

Within this thesis, I develop an approach that falls into the second category. Thus, I aim to create a causal neural net through a change in the model training. To my knowledge, most work in this field aims to analyze already trained models. Thus, they fall into the first category. As mentioned before, in the standard case models are trained for correlations or association. Thus, they could find patterns that are not representing the real world. This makes it difficult to trust the created machine learning models. Furthermore, by also including patterns that are not causal, standard machine learning models might be worse at generalizing than a causal machine learning model. This leads to two research questions for this thesis:

1. Is a causal interpretability mechanism, that is applied to the training of a neural net, able to enhance its causality?
2. Is a neural net with an enhanced causality able to generalize better?

Within this thesis, I aim to answer these questions by incorporating the Average Causal Effect (ACE) into the model training [7]. I achieve that by including the ACE into a regularization term added to the loss function of the training. The ACE analyzes the inner state of the model. Thus, it examines neuron activation to determine causal model parameters. To my knowledge, most of the work done in the field of causal interpretability does not analyze the inner state. By doing this, the regularization mechanism within the training should maximize the overall causality of the machine learning model. Furthermore, the investigated model is a model solving a regression problem, as most interpretability research work investigates classification problems. The developed algorithm is tested on a well-balanced dataset, a dataset with prior probability shift, and a dataset with covariate shift. Lastly, the results are compared to the results of a model with L1 and L2 regularization in its model training. This is due to the fact that Janzing found the indication that L1, as well as L2 regularizations, can enhance the model causality [18].

In the following Chapter 2, I discuss related work and the still existing gap in this research field. Afterward, in Chapter 3, I provide some definitions in the area of deep learning as well as causal modeling and learning. Furthermore, I explain the developed method in Chapter 4 and discuss the results in Chapter 5.

2 Literature Review

In the following section, I explore related work connected to this thesis. The properties of each paper can be found in Table 2.1. There, the methods are sorted into four categories: *Model Training, Causality, White Box*, and *Regression*. The category *Model Training* represents that the authors apply the method during model training. This means the method takes influence on the training process and does not analyze an already trained model. Furthermore, the category *Causality* stands for methods that are implemented based on a causality mechanism. So, methods that make use of mathematical theories that depend on the principle of causality. Approaches that fall into the category *White Box* are approaches that open the black box of the model that they analyze. Thus, an approach that analyzes the inner state of the model they investigate. The last category *Regression* applies for approaches that are able to analyze models solving regression problems. Using the mentioned categories, one can sort the papers into five sections. First, methods applied within the model training that are based on causality. Those approaches apply to the categories *Model Training* and *Causality*. Then, methods which alter model training but are not based on a causality mechanism. Hence, approaches falling into the category *Model Training*. The third section belongs to the category *Causality* and deals with approaches that are based on causality but analyze already trained models. After that I introduce authors that analyze the inner state of a trained model using approaches not based on causality. Thus, they apply to category *White Box*. Lastly, I discuss papers which analyze trained models regarding explainability or interpretability. In this thesis, I develop an approach which fullfills all the mentioned categories: *Model Training, Causality, White Box*, and *Regression*. Hence, my approach falls into the first section. Therefore, I first introduce approaches which developed methods based on causality mechanisms and applied them in model training. Then, I work my way through the other sections.

2.1 Causal Mechanisms Built in Model Training

During this thesis, I apply a method based on causality in the model training. So, I analyze the papers [3, 18, 22, 27], which fall into the category *Model Training* and *Causality*.

Bahadori et al., in [3], developed a regularization method for neural nets to make the overall model more causal. The regularization method uses a causality score within the cost function of the training. This improves the causality of the overall model. The authors interpreted the mentioned causality score as a probability, which shows how likely it is that a feature X_i does not cause the output Y_i. As a next step, Bahadori et al. included the probability into the cost function of the training to control the influence of the weights w_i of X_i. This means, that if X_i is less likely to cause Y_i, the weight of X_i will have less influence in the cost function. Subsequently, this makes features with a larger causal effect more influential. Therefore, it improves the overall causality of the model. Using causal clinical data, the authors were able to show that their algorithm interprets the right features as causal. Furthermore, the results show that one can achieve an improvement of 20% in the overall causality score of a neural net using this method. Looking at Table 2.1, one can see that the approach belongs to category *Model Training, Causality, White Box*, and *Regression*. I also include a causality mechanism into a regularization term to improve the overall causality. Thus, this approach is the most similar method to my approach. However, I maximize the overall causality directly during training. In contrast, the authors here increase the influence of causal weights and features.

In the next two papers, [18, 27], the developed approaches apply to category *Model Training, Causality*, and *Regression*. In [27], Kyono, Zhang and van der Schaar introduced the algorithm Causal Structured Learning (CASTLE). CASTLE is a causal regularization method for models. In detail, the method uses a causal directed acyclical graph (DAG) to reconstruct only the features which act as an optimal predictor. This means that the feature has a causal relationship to the target variable, which indicates that the feature is causal. In order to achieve that, the DAG acts as an adjacency matrix within the input layer of the neural net. CASTLE achieves better out-of-sample predictions compared to other common benchmark regularizers. During the experiments, the authors used multiple synthetic and real life datasets. Similarly, in [18], Janzing found the indication that standard neural net regularization improves the causality of models. Table 2.1 shows that this work is more an analysis rather than a newly developed approach. However, one can still classify it as an approach used within model training. In this work, the author used the standard L1 and L2 regularization to reduce over fitting. Subsequently, he was able to decrease the confounding effect. Furthermore, Janzing determined the size of the regularization parameter of the L1 and L2 regularization. For that, he estimated the strength of confounding beforehand. By reducing the confounding effect of the model, the model is able to generalize from observational distributions to interventional distributions. Through that Janzing was able to show an indication that the L1 and L2 regularization improves the causality of a model. During this thesis, I also develope an approach for causal regaulrization. However, I look at the inner state of the model and take the inner state into account during my analysis.

In [22], the goal is to make the computation of the steering angle of an autonomous car explainable. As displayed in Table 2.1, the authors applied their method inside the training process and based the method on causality. To be more precise Kim and Canny used visual attention to highlight image regions that influence the model's output. Then, the authors applied the principle of causal attention. That means, they covered up the identified regions in the picture in order to explain which elements have a causal effect on the model. The results show that the proposed framework does not degrade the model performance. Furthermore, the model demonstrates causal behavior. In detail, the algorithm concentrates on features used by humans while driving. I draw from the mentioned approaches regarding the application during training and the used causality mechanism (see Table 2.1). However, I also analyze at the internal state of the model. In contrast. the authors here only look at the input and output relationship. Furthermore, in [22] classification models where analyzed. I apply my method to regression problems.

In sum, the above mentioned approaches belong to the category *Model Training* and *Causality*. Some, as [3], also fall into category *White Box* and *Regression*. In this thesis, the approach has all the mentioned attributes. However, I directly maximize the overall causality of the model. Thus, I include the causality measurement, as a regularization term, in the training's cost function. In contrast, the other papers concentrate on regulating the influence of causal features or non-causal features. Or rather, they focus on choosing the right features or simply making the model explainable.

2.2 Explainable and Interpretable Model Training

As I mentioned before, one goal for this thesis is to develop an approach which falls into the category *Model Training*. In the following papers [1, 20, 46], the authors included their methods into the model training. This way, they aim to make the model interpretable or explainable. In contrast to the section before, they do not apply to category *Causality*.

In [46], the authors used Cook's distance on large scale datasets. Cook's distance is a classical statistical technique to estimate the influence of data points [46]. Using that, they evaluated the influence of training samples on regression models. This means, they can identify samples with an extraordinary strong influence on the training of the model. In order to use Cook's distance on a large and high-dimensional dataset, Wojnowicz et al. introduced the method influence sketching to identify influential training samples. Influence sketching applies random projections within the influence computations. The authors tested their algorithm on a malware detection dataset. The dataset included over 2 million executable files with almost $100,000$ features. They showed that the impactful samples tend to be mislabeled. When testing with the mislabeled data, the results show that deleting the impactful samples brings the accuracy down to 90.24% from 99.47%. Deleting the same number of random samples only brings the accuracy down to 99.45%. This additionally proofs how impactful the samples are. Their approach is also applicable to other datasets. The authors in [1] used the attention mechanism sequential attention. Here, Arik and Pfister applied a deep tabular learning architecture, called TabNet, in order to select the most salient features. The results show that TabNet outperforms other similar methods on multiple synthetic and real-world tabular datasets. Furthermore, the method achieves interpretable feature attributions. Both methods connect to my work, as they belong to the category *Model Training* and *Regression*. Yet, the authors used their methods for sample selection and feature selection.

Kim, Shah and Doshi-Velez addressed feature selection and extraction in [20]. The Mind the Gap Model (MGM) embeddeds interpretability criteria into the model design. This way, they could tweak interpretability parameters and generate identifiable dimensions. Then, the authors used that for data exploration and hypothesis generation. The method obtains identifiable features on a range of diverse real-life datasets. My approach is similar, because the authors influenced the model training directly.

In sum, all three papers use their approaches for feature or sample selection. Additionally, they take influence on the model training and some of them are applicable to regression problems. However, my approach concentrates on influencing the model training to directly improve the overall causality.

2.3 Causality Mechanisms Used on Trained Models

Next, I analyze approaches belonging to category *Causality*. Thus, those methods analyze already trained models. This connects to my work, as I also develope an approach based on causality.

The following papers [6, 16] fall into the category *Causality* and *White Box*. First, Cawley investigated a regression problem to select causal features in [6]. Here, the author used inference of the Markov Blanket, of direct causes and of direct effects. For the non-causal case, Cawley derived the non-causal feature selection from logistic regression. For that, he used Laplace prior based Bayesian regularisation. Cawley then evaluated linear classifiers regarding their difference between the training and test data. For this, he utilized the causal relationship of input and response variables as a measurement. The causal feature selection method shows no significant enhancement regarding the predictive accuracy of the classifiers over a non-causal feature selection and/or using all features provided. In sum, the causality mechanism applied has no positive effect on the accuracy of the model. Furthermore, Harradon, Druce, and Ruttenberg, in [16], built a causal machine learning model utilizing salient concepts within CNNs. The authors used a human-understandable representation of network activations to extract the salient concepts. For that, they trained autoencoders to extract the human-understandable representations of network activations. Using the extracted salient concepts, they developed a Bayesian causal model to make the classification interpretable. Lastly, the authors tested the algorithm for image classification. This way, they could identify and visualize features with significant causal influence. The results show that they could provide a novel approach to building causal models. Next, I look at papers applying to category *Causality* and *Regression*.

In [7, 35, 40, 49], the authors analyzed trained models using a method based on causality. First I look at the approach in [40]. Schwab and Karlen derived a causal explanation model (CXPlain). CXPlain estimates the impact of machine learning inputs on the respective outputs. Here, the authors calculated the uncertainty of the model's feature importance. For that, they used Bootstrap ensembling. This way, a model with high-dimensional data can be made explainable by estimating the importance of each feature. The results show that CXPlain is more accurate and faster than comparable algorithms in the field of estimating feature importance. Within the next approach [7], Chattopadhyay et al. used the average causal effect (ACE) to compute the influence features have on the output. To compute the causal effect, the authors viewed a neural net as a Structural Causal Model. The described algorithm can efficiently compute the ACE even for large dimensional data. Furthermore, Chattopadhyay et al. tested the method for a standard classification task with a real-life dataset and an LSTM structure with a synthetic dataset. For both cases, causal effects could be calculated making the causality of the neural nets more transparent. In [35], Peter, Bühlmann, and Meinshause described a causal model as a model which remains invariant under interventions. The interventions consist of different changes in the model environment. This means they applied different interventional real-life datasets to the models. The authors then classified a model as causal, if it showed invariance in their predictive accuracy. The result is a confidence interval for the causality of the model. The results show that, especially for Gaussian structural equation models, the method can identify the set of causal models. Furthermore, in [49], the authors used counterfactual explanation to built fair decision-making systems. They developed three measurements: the counterfactual direct effects (Ctf-DE), the counterfactual indirect effects (Ctf-IE), and the counterfactual spurious effects (Ctf-SE). These measurements show the level of fairness of the decisions made by the respective models. All of the quantifications use counterfactuals for their assessment. The Ctf-DE measures the natural direct effect between the input X and target Y. This means it assesses the effect of the direct causal path between $X \rightarrow Y$. Furthermore, the Ctf-IE quantifies the natural indirect effect between $X \rightarrow Y$. In detail, it measures the underlying causal effect between $X \rightarrow Y$ caused by another variable in the system. Lastly, the Ctf-SE assesses the spurious effects between X and Y. This means, it calculates effects caused by backdoor paths of X and Y. Backdoor paths are paths only pointing at and not coming out of X or Y. In sum, the authors were able to provide a measurable approach to build a fair decision-making system. The papers above unite the fact that they use causality mechanisms to analyze trained models. Furthermore, one can apply them to regression problems. The next paper falls into the category *Causality* but the analyzed model solves a classification problem.

In order to measure the causal effect within a neural net, the authors in [15] used the Causal Concept Effect (CaCE) as a basis of their method. The CaCE provides the opportunity to evade mistakes caused by confounding. As one cannot simulate a $do(\cdot)$ operator effortlessly, simulating the CaCE has its challenges. That's why, Goyal Shalit and Kim use Variational Auto Encoder (VAE) to develope the VAE-CaCE. The results show that the VAE-CaCE can measure the CacE effect for various number of datasets. Looking at Table 2.1, the main difference is that this approach is only applicable to classification problems.

In sum, the approaches in this section are based on causality and analyze already trained models. Additionally, most of them are applicable for regression problems. In my thesis, I use the average causal effect (ACE), mentioned in [7]. The main difference is that I apply this mechanism inside the model training creating a causal model from the start. In the next section, I cover approaches that analyze the inner state of the model with regards to interpretability and explainability.

2.4 Interpretability and Explainability of the Trained Model's Inner State

So far, I have looked at papers that fall into one of the following or both categories *Causality* and *Model Training*. Within this section, I analyze work that deals with non-causality mechanisms outside of the training process. They belong to the category *White Box*, as they analyze the inner state of the model. In my thesis, I investigate the inner state of the model as well.

In [4, 45], the authors analyze an already trained model regarding interpretability or explainability. As one can see in Table 2.1, they also inspect the inner state of the model. Additionally, both approaches are suitable for regression problems. One way to make machine learning models explainable is to extract rules or interpretable models from them. In [4], the authors extracted Decision Trees from neural nets, to make the neural nets explainable. They generated the Decision Trees using DecText. DecText utilizes the algorithm C4.5 which applies the concept of information entropy. Furthermore, DecText is also able to work with continuous features. The previous work only concentrates on MofN type Decision Trees. Their splits follow a m-of-n rule, which means for a split m of n Boolean conditions need to be satisfied. As MofN Decision Trees are only suitable for MofN problems, the authors provide an alternative method for regular high dimensional real-world problems. The authors of [45] also measured feature importance through counterfactuals. Sundararajan, Taly, and Yan generated the counterfactuals by scaling down the original inputs. Then, they analyzed the counterfactual's gradient, called interior. Using the gradient, they measured the importance of the respective feature. The authors then applied the algorithm to an LSTM language model and a GoogleNet for image object recognition. Furthermore, they chose to test it on a ligand-based virtual screening network with categorical features. With this method, they were able to calculate interior gradients as efficiently as standard gradients. Furthermore, the algorithm applies to various neural nets. The results conclude that the feature importance improves the prediction score. For both papers, they analyzed an already trained model by looking at the inner state. Furthermore, they were able to apply their methods to regression problems. In the following, the approaches also fall into the category *White Box*. However, they are only applicable for classification tasks.

In the following papers [21, 42], the authors analyzed the inner state of an already trained model using non-causality mechanisms. In [42], Shirkumar, Greenside, and Kundaje took the activations of each neuron into account to interpret the model. Called DeepLIFT, the algorithm backpropagates the influence of each neuron to every feature. This way, DeepLIFT generates a contribution score of each neuron. For that, the algorithm compares the neuron's current activation to its reference activation. Through that, DeepLIFT identifies features with the largest contribution to the output. The authors applied the approach to neural nets trained with MNIST and genomic data. The results show, that compared to other methods, DeepLIFT reveals novel dependencies within the model. Moreover, in [21], Kim et al. used Concept Activation Vectors (CAVs) to explain the internal state of a model. The authors developed a testing framework with CAVs (TCAV), which uses directional derivatives. TCAV quantifies the importance of the model's features for the result of the model. To test TCAV, Kim et al. evaluated image classification models. The results show that CAVs can give insights regarding the model's prediction via the internal state. Furthermore, CAVs apply from standard image classifications to models solving specialized medical problems. As mentioned before, both papers analyze the inner state of the model with methods not based on causality. Additionally, they can perform on classification tasks only.

In sum, the work above analyzes the inner state of a trained model regarding interpretability or explainability, so fall into the category *White Box*. In this case, the approaches in [4, 45] can handle regression problems. On the contrary, the algorithms presented in [21, 42] are only applicable to classification tasks. In this thesis, the approach also belongs to the category *White Box* and *Regression*. The next section covers other relevant explainability or interpretability methods for machine learning.

2.5 Interpretability and Explainability of Trained Models

Up to this point, I looked at papers that fall into one or a number of the following categories: *Model Training, Causality* and *White Box*. Lastly, I look at approaches that provide other interesting methods that aim to make models more explainable or interpretable without falling into the categories *Model Training, Causality* and *White Box*. Those papers are relevant as I also intend to make a model more interpretable.

First, I analyze work that falls into the category *Regression*. Riberio, Singh, and Guestrin developed the framework LIME in [38]. LIME trains an interpretable model locally around the prediction. The authors showed that interpretability is useful for multiple models in trust-related tasks. In detail this means in text and image domains as well as both expert and non-expert users. In [41], Sharma, Hendersion and Ghosh built CERTIFAI to test any machine learning model regrading robustness, transparency, fairness and interpretability. The paper introduces a genetic algorithm to generate counterfactuals. The results show that the genetic algorithm brings flexibility to the model, by generating custom counterfactuals. Furthermore, the authors developed CERScore to quantify the robustness of the tested machine learning model. CERScore uses the generated counterfactuals. The algorithm is able to perform equivalently to similar methods. For another approach dealing with counterfactual explanation, the authors of [25] used an influence function. The influence function computes the neural net input's effect on the output. In this case, Koh and Liang tested linear regression and a CNN using image data. With that, they determined the effect of the input on the output. Using the proposed method, the authors were able to understand the model behavior and debug models. Furthermore, they could detect dataset errors and create visually indistinguishable training-set attacks. In sum, as one can see in Table 2.1, the approaches above are able to provide a method that analyzes an already trained regression model regarding explainability or interpretability.

Lastly, I analyze papers that are only able to analyze methods solving classification tasks. The authors in [8] introduced Quantitative Input Influence (QII) using interventions. QII quantifies the influence the inputs of a model have on the output. The measurement can make decisions, made by a model on individuals or groups, transparent. Furthermore, the method measures the joint influence of inputs, while it can account for correlated inputs. Additionally, the authors quantified the marginal influence of individual inputs as well. For that, they used principled aggregation measures, like Shapley values. A Shapely value can show the average expected marginal contribution of one neuron. Overall, QII can generate explanations of correlated inputs and can be approximated efficiently. In [28], the authors made interpretability of predictions, with high-dimensional vectorized features, possible. Here, Le et al. use explanation by intervention to elucidate the labels generated by the prediction. The created framework, GRACE, generated 60% more accurate post-explanation decisions than the competing baseline method LIME. Both of the methods can be applied to models solving classification tasks.

In this section, I explored approaches that make already trained models interpretable or explainable. The papers are relevant, as they aim to make machine learning models explainable or interpretable.

To conclude, looking at Table 2.1, most authors applied their approaches outside of the training process. This means, they analyzed already trained models. I suspect that intervening with the training process to make the model causal can be a more efficient approach. Hence, I aim create a causal model directly. Furthermore, methods, that fall into the category *Model Training*, mainly engineer the feature selection not the model itself (see Table 2.1). Additionally, the authors mostly treated the model as a black box without analyzing the inner state. In this thesis, I directly influence the overall causality of the model by analyzing the inner state. Thus, I create causal weights during the training process which increase the overall causality of the model. Furthermore, the approaches belonging to the category *White Box* often do not fall into the category *Causality*. The authors used other interpretability or explainability approaches to analyze the inner state. However, I analyze the inner state of the model with a method based on causality. The paper which is most similar to my approach is [3]. Yet, the main difference is that Bahadori et al. regulated the influence of weights using a causality score. The causality score, as mentioned before, is a soft-score that rates the causality of each weight. In contrast, I directly stir the weights into a causal direction by enhancing the overall causality of the model. All in all, the unique contribution here is to directly create causal weights during the training process by looking at the inner state of the model. Next, I dive deeper into the background of this thesis. As I apply my method to neural nets, I first give a brief overview of the definition of neural nets.

Papers	Objective of Method	Model Training	Causality	White Box	Regression
Causal Mechanisms Built in Model Training					
[3]	Model Causality Measured	●	●	●	●
[27]	Feature Selection	●	●	○	●
[18]	Model Causality Measured	◑	●	○	●
[22]	Feature Selection	●	●	○	○
Explainable and Interpretable Model Training					
[46]	Training Sample Selection	●	○	○	●
[1]	Feature Selection	●	○	○	●
[20]	Feature Selection	●	○	○	○
Causality Mechanisms used on Trained Models					
[6]	Feature Selection	○	●	●	●
[16]	Feature Selection	○	●	●	○
[7]	Feature Selection	○	●	○	●
[35]	Model Causality Measured	○	●	○	●
[49]	Model Explainability	○	●	○	●
[15]	Training Sample Selection	○	●	○	○
Interpretability and Explainability of the Trained Model’s Inner State					
[4]	Model Explainability	○	○	●	●
[45]	Feature Selection	○	○	●	●
[42]	Model Explainability	○	○	●	○
[21]	Feature Selection	○	○	●	○
Interpretability and Explainability of Trained Models					
[38]	Model Explainability	○	○	○	●
[41]	Model Explainability	○	○	○	●
[25]	Training Sample Selection	○	○	○	●
[8]	Feature Selection	○	○	○	○
[28]	Model Explainability	○	○	○	○

Table 2.1: Approaches are mainly used outside of model training and often do not pair causality with an analysis of the inner state

3 Background

In this section, I explain the methods this thesis is based on. First, I introduce models and concepts of deep learning used throughout thesis. Second, I present the concept of causal modelling and learning as well as causal inference. This serves as a foundation of the methods proposed in this thesis. In the following section, I discuss the concepts of deep learning.

3.1 Deep Learning

One main model of deep learning are deep feedforward neural nets. In the following, I introduce the definition of deep feedfoward neural nets used in this thesis. Furthermore, I review the main loss function used throughout this thesis, the Mean Squared Error. Lastly, I introduce the mechanism of neural net regularization and the concept of datasetshifts.

3.1.1 Deep Feedforward Neural Nets

In this thesis, I define a neural net like Nielsen in [30]. Thus, a neural net is a model that makes predictions using data. There are two kinds of predictions: classification and regression. Classification takes the inputs and sorts them into a category. If one uses pictures as an input, for example, the neural net would determine the objects within that picture. Regression, on the other hand, uses the inputs to predict a numerical value. So for example, taking the location of a house, the size of a house, and the year they built the house, the neural net can predict the house price. Within this thesis, I concentrate on regression problems.

As displayed in Figure 3.1, a neural net consists of layers. The layers, in turn, contain neurons. The neurons form a weighted sum of their inputs and apply a, so-called, activation function to that weighted sum. In Figure 3.1, the neurons are displayed as circles. Furthermore, the neurons are connected between the layers. Generally, each neural net has an input layer and an output layer. Between them are one to several hidden layers. When performing computation with the neural net, the input layer brings the initial data samples into the neural net. After that, the hidden layers perform the computations and the output layer presents the result. The result in turn represents a prediction or classsification. Throughout this thesis, I only used feedforward neural nets. They are the easiest form of neural nets, as they only pass information from one layer to its following layer. Hence, there are no cycles or recurrent connections in the network. For simplicity when referring to neural nets, I mean only the feedforward type.

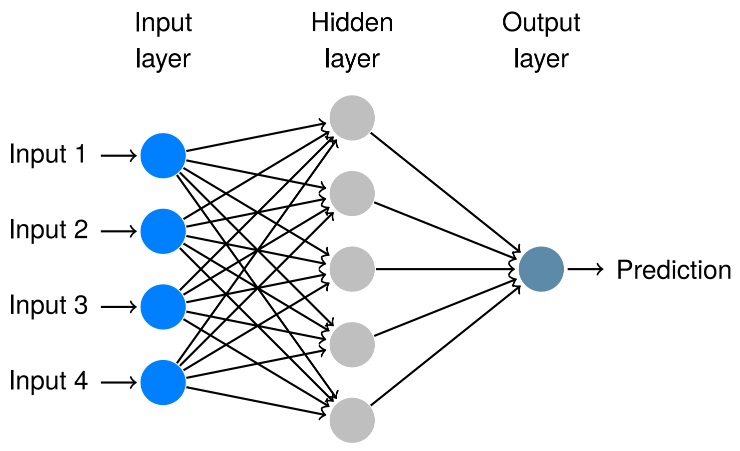

Figure 3.1: A feedforward neural net has several layers which consist of neurons

To train the neural net, one uses the, so-called, training data. During the training, the training algorithm adjusts the neural net to represent the input-output relationship of the training data as good as possible. For that, the training algorithm tunes the parameters of the neural net. The mentioned parameters are weights and biases. The training algorithm assigns these weights and biases to the neurons in each layer [30]. In more detail, given a set of inputs $\mathbf{x} = x_1, x_2, ..., x_n$, the parameters $\mathbf{w} = w_1, w_2, ..., w_n$ are the respective weights to the inputs x_i. Furthermore, each neuron has a bias b. Using the weights, inputs, and the bias, the activation $\mathbf{z}$ of each neuron becomes

$$\mathbf{z} = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i}^{n} w_i x_i + b \tag{3.1}$$

with n being the number of inputs $\mathbf{x} = x_1, x_2, ..., x_n$.

As each neuron of a hidden layer has multiple inputs, the input x_i is passed to the neuron with its respective weight w_i. Then, then the bias b of the neuron is added. Afterward, the model passes the result of this equation to the next layer.

As one can see, with Equation (3.1) the neuron is linear. In order, to introduce non-linearity, one can use activation functions [39]. In this case, I use the Sigmoid function. If one takes the inputs $\mathbf{x} = x_1, x_2, ..., x_n$, the weights $\mathbf{w} = w_1, w_2, ..., w_n$ and the bias b of each neuron, the computation of each neuron becomes

$$f(\mathbf{w}, \mathbf{x}, b) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}} \tag{3.2}$$

Thus, Equation (3.2) makes sure that the output, the so-called activation, always stays between 0 and 1. The Sigmoid has the advantage that it is continuously differentiable [5]. This means, the neurons are differentiable. This is important to calculate the ACE described in Section 4.3.

To be able to predict values a training algorithm uses data to train the model. In this thesis, I use supervised learning. Supervised learning uses labels to train the model [39]. The labels are the true labels of the dataset. First the training data is inserted into the model and the model computes a prediction. Then, using a loss function the training error is calculated. The training error is the difference between the current prediction and the training label. Afterward, the backpropagation algorithm updates the model parameters to minimize the training error. For that, the partial gradient of the loss function with respect to the model parameters is calculated for the last layer. The last layer is analyzed first because the parameters of the last layer do not influence the other model parameters. That calculated gradient, then, shows how to update the weights and biases to minimize the training error of the model. Afterward, the gradient for the next layer is calculated progressing backwards until the algorithm arrives at the beginning of the network. This way, the weights and biases are updated from the last layer to the beginning of the network to minimize the training error according to the gradient of the loss function.

3.1.2 Mean Squared Error as a Loss Function

I use the Mean Squared Error (MSE) as the loss function in this thesis. As written above, the loss function, and in this case the MSE, computes the training error. For that, the model takes the inputs $\mathbf{x} = x_1, x_2, ..., x_n$ and predicts $\mathbf{y} = y_1, y_2, ..., y_n$. As it is supervised learning, the dataset has labels $\hat{\mathbf{y}} = \hat{y}_1, \hat{y}_2, ..., \hat{y}_n$. The labels are the true labels of the dataset. Using the predictions $\mathbf{y}$ and labels $\hat{\mathbf{y}}$ the MSE becomes

$$L = \frac{1}{n} \sum_{j=1}^{n} \left(y_j - \hat{y}_j\right)^2 \tag{3.3}$$

with n being the number of training samples the neural net has.

According to Equation (3.3), the loss function computes the error between the current prediction $\mathbf{y}$ and optimal prediction $\hat{\mathbf{y}}$. It then squares that error. Afterward, the loss computes the mean of all of the existing errors. Then, the training algorithm adjusts the weights $\mathbf{w}$ and biases $\mathbf{b}$ to minimize the MSE. The MSE is an appropriate loss function if one wants to penalize outliers. That is because the MSE squares the error. This makes errors above 1 more impactful. Furthermore, it is one of the most commonly used loss function, as it is very useful for normally distributed data [39].

3.1.3 Neural Net Regularization

One common problem in training neural nets is that they can overfit. This means, that the model performs very well on the training data but not as good on the test data. Thus, the model is not able to generalize. One way to improve this is regularization. Goodfellow et al. defines regularization as any type of adjustment made to a learning algorithm that aims to decrease the test but not the training error [13]. Thus, regularization can be any adjustment to a learning algorithm that reduces overfitting. There are multiple novel ways how to regularize a neural net. However, in the following, I mention the most relevant regularization techniques for this thesis. Those are techniques addressing the neurons and their activations directly. This means, they either adjust the neuron's parameters or deactivate neurons altogether. The most common way to do so are dropout, L1 regularization, and L2 regularization [13].

Dropout Regularization

The dropout method randomly deactivates neurons within the hidden layers. With that, the method lowers the model complexity as the network becomes smaller. Within a network, neurons can balance out the mistakes of other neurons [43]. This leads to complex co-adaptation between neurons. As this co-adapatations do not translate to the test data, overfitting happens as a result [43]. Through randomly dropping nodes, the dropout method avoids this co-adaptation. Therefore overfitting is reduced.

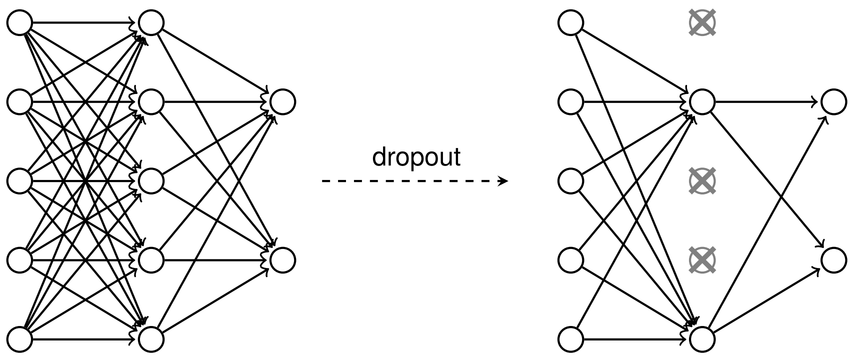

Figure 3.2: The dropout regularization randomly drops neurons through the training process reducing co-adaptation of neurons

L1 and L2 Regularization

The L1 and L2 regularization both reduce overfitting by keeping the values of the weights small [13]. They do so by adding a component to the loss function penalizing larger weights [13].

For the L2 regularization, L_{T_2} is the new loss function that computes the training error of the neural net. Furthermore, L_D is the original loss function like the MSE. With λ as a parameter to tune the regularization, with the weights $\mathbf{w} = w_1, w_2, ..., w_m$ of the neural net, and with L_{L2} as the L2 term, the loss function becomes

$$L_{T_2} = L_D + \lambda L_{L2} = \frac{1}{n}\sum_{j=1}^{n} L_D\left(y_j, \hat{y}_j\right) + \lambda\|\mathbf{w}\|_2^2 = \frac{1}{n}\sum_{j=1}^{n} L_D\left(y_j, \hat{y}_j\right) + \lambda\sum_{l=1}^{m} w_l^2 \quad (3.4)$$

Here, n is the number of training samples and m the total number of weights in the neural net. As one can see in Equation (3.4), $L_D\left(y_j, \hat{y}_j\right)$ is the original loss function, using the predictions $\mathbf{y}$ and labels $\hat{\mathbf{y}}$ to determine the training error. The original loss function can be the MSE, for example. The L2 term L_{L2} is the euclidean length of the weights $\mathbf{w}$. For the regularization, the euclidean length is squared. It should be noted that the euclidean length is the 2-norm. This means, the L_{L2} results in the sum of squared weights.

The L1 regularization is very similar to the L2 regularization. L_{T_1} is the new loss function combining the original loss function L_D and the L1 component L_{L1} [13]. As before, L_{T_1} determines the training error of the neural net [13]. Using λ as a tuning parameter for the regularization the equation becomes

$$L_{T_1} = L_D + \lambda L_{L1} = \frac{1}{n}\sum_{j=1}^{n} L_D\left(y_j, \hat{y}_j\right) + \lambda\|\mathbf{w}\|_1 = \frac{1}{n}\sum_{j=1}^{n} L_D\left(y_j, \hat{y}_j\right) + \lambda\sum_{l=1}^{m}|w_l| \quad (3.5)$$

with n being the number of labels and the weights $\mathbf{w} = w_1, w_2, ..., w_m$ with the total number of weights m. Looking at Equation (3.5), one can see that the original loss function $L_D\left(y_j, \hat{y}_j\right)$ uses the predictions $\mathbf{y}$ and labels $\hat{\mathbf{y}}$ to compute the training error. However, the L1 term uses the 1-norm of the weights $\mathbf{w}$. This results in adding up all absolute values of the weights.

For both L1 and L2 regularization, the loss function keeps the absolute value of the weights small. Throughout the training process, the training algorithm tries to minimize the loss function. Thus, it also tries to minimize the regularization components within the loss. As the regularization component includes the weights, the absolute value of the weights is driven down.

Looking at Equation (3.1), one can see that the smaller the values of the weights $\mathbf{w}$, the smaller the activation of the neuron $\mathbf{z}$. If the algorithm reduces the activation by keeping the weights small, the overall function fitted to the data can be less complex. This might results in less overfitting [13].

Although both regularization techniques are quite similar, they have key differences. In Table 3.1, I summarize the differences [13]. The L1 regularization handles outlier better than its L2 counterpart [13]. The reason for that is the squared component in the L2 regularization. It makes large error terms even more impactful. Furthermore, L1 is able to reduce features [13]. Thus, provide us with a sparse model. This is because L1 is able to reduce coefficience to 0. In constrast, L2 is only able to bring coefficience to a small value. That means, by using L1, features could be entirely removed from the model. Additionally, L1 only has one optimal solution and L2 can provide more than one optimal solution [13].

	L1	L2
Robustness to outliers	+	-
Sparsity	+	-

Table 3.1: L1 regularization is more robust to outliers and can provide sparsity [13]

In sum, the dropout regularization can reduce model complexity by randomly dropping neurons. The L1 and L2 regularization might decrease the complexity of the functions within a neural net by keeping the size of the weights small. For that L1 uses the absolute norm and L2 the squared euclidean norm. Ultimately, all of the methods modify the neural net to decrease the test error while not changing the training error. This means, they can reduce overfitting.

3.1.4 Datasetshift in Deep Learning

One reason, that a neural net is unable to generalize, can be the concept of datasetshift. In general, datasetshift means that the distribution $P_{tra}\left(x, y\right)$ of the training data $\left(x, y\right)$ is different to the distribution $P_{tst}\left(x', y'\right)$ of the test data $\left(x', y'\right)$ [37]. This is because the training algorithm uses the distribution of the training data to tune the model parameters. Thus, the parameters are tuned for that training distribution. If the test data then has a different distribution the parameters do not fit any longer and therefore the prediction quality declines. There are different types of datasetshift. The most common are covariate shift, prior probability shift, and concept shift [26, 37].

Covariate Shift

In the case of covariate shift the distribution of the input x changes [10, 37]. However, the relationship between input x and the corresponding output y stays the same [10, 37]. This means, relationship of the distribution of the training input $P_{tra}(x)$ and the distribution of the test input $P_{tst}(x')$ is $P_{tra}(x) \neq P_{tst}(x')$. But, the relationship of input and output of the training data $P_{tra}\left(y \middle| x\right)$ and test data $P_{tst}\left(y' \middle| x'\right)$ has no shift.

Prior Probability Shift

Prior probability shift is similar but here the distribution of the output y changes [10, 37]. Yet again, the relationship of input and output does not change [10, 37]. In detail, the distribution of the test output $P_{tst}\left(y'\right)$ and the distribution of the training output $P_{tra}\left(y\right)$ are giving the equation $P_{tst}\left(y'\right) \neq P_{tra}\left(y\right)$. Again, the relationship between input and output is $P_{tra}\left(x \middle| y\right) = P_{tst}\left(x' \middle| y'\right)$.

Concept Shift

In the case of concept shift, the relationship of input and output is different between the training and test data [26, 37]. However, there are two cases. First the distribution of the inputs can be the same [26, 37]. That means, $P_{tra}(x)$ is the same as $P_{tst}(x')$. Then, the distributions of the relationship of input and output becomes $P_{tra}\left(y \middle| x\right) \neq P_{tst}\left(y' \middle| x'\right)$ [26, 37]. The second case is that the distribution of the training outputs $P_{tra}\left(y\right)$ is the same as the distribution of the test outputs $P_{tst}\left(y'\right)$ [26, 37]. Similar here, the distributions of the relationship between input and output becomes $P_{tra}\left(x \middle| y\right) \neq P_{tst}\left(x' \middle| y'\right)$ [26, 37].

In sum, datasetshift is caused by a difference between the distributions of training and test data. The most common shift is either between the inputs, outputs or the relationship of input and output. This causes a neural net to be worse at generalizing from the training data.

3.2 Causal Modelling and Learning

In order to apply causal inference to a system, it is necessary to model the system as a causal model. Thus, I also apply causal modelling and learning in this work. In Chapter 1, I introduce the definition of causality that I use here. Essentially, a causal model is a probabilistic model with additional information [36]. This is especially important when creating the causal models. Creating the models requires a learning process just like with probabilistic models. For causal models, one calls the learning process causal learning [36]. Causal learning does not just draw from observational data. It can additionally include data under interventions or uses distribution changes for example [36]. These mechanisms provide additional information. In turn, this additional information ensures that the causal model represents causal relationships and not correlations.

One definition for causal relationships is Reichenbach's common cause principle [32]. It states that, if there is a correlation between two variables X and Y, then they either have a common cause Z, X causes Y, or Y causes X (see Figure 3.3) [32].

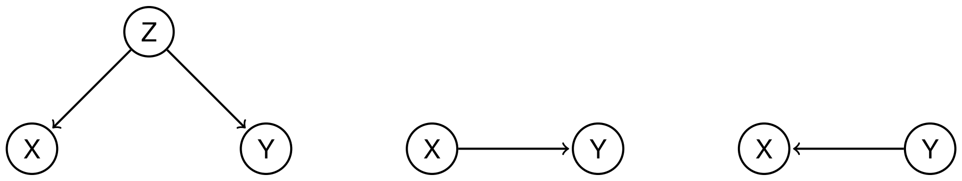

Figure 3.3: Reichenbach's common cause principle defines the causal relationship between X and Y

Thus, a causal model represents only causal relationships according to Reichenbach's common cause principle. It is able to do so because additional information, like data under interventions, is provided throughout the learning process.

3.2.1 Structural Causal Models

For a causal model, I use structural causal models (SCMs) during this thesis. Like other causal models it defines cause-effect relationships between random variables [32]. By definition an SCM is a 4-tupel (X, U, f, P_u) [36]:

- Endogenous variables X: The observable deterministic variables within the system.
- Exogenous variables U: The stochastic noise in the system.
- Set of functions f: The functions $x_i = f_i(Par, u_i)$ determining the causal mechanism within the SCM with $\forall x_i \in X_i, u_i \in U_i$, and Par being a subset of $X - x_i$.
- Probability distribution P_u over U: The probability distribution determining the noise U.

The SCMs that I consider in this thesis, are no feedback causal models. That means, they do not have any feedback loops. Furthermore, not every variable in the SCM has noise. Using this definition of an SCM one can construct a graphical model: a causal Bayesian network.

Causal Bayesian Network

For an SCM $M(X, U, f, P_u)$ one can construct a causal Bayesian network $G = (V, E)$ [32]. Figure 3.4 shows an example of a causal Bayesian network. Here, the variables V are the vertices corresponding to the endogenous variables X [32]. In Figure 3.4, the variables X are A, B, and C. The variables E are the edges of the graph symbolizing the function f [32]. Thus,the connections in Figure 3.4. So, given a function $x_i = f_i(Par, u_i)$ with $\forall x_j \in Par$ means that x_j causes x_i. Thus,this translates into a direct edge from $v_j(x_j)$ to $v_i(x_i)$ within the graph. This also means that v_j is a, so-called, parent and v_i a, so-called, child [36]. The local Markov Model property states that the parent vertices are the only influence for the distribution of a child vertex within a Bayesian network [19]. Furthermore, these parent and child vertices can create a, so-called, path. A path is the sequence of unique vertices $v_i \longrightarrow v_{i+1}$ with edges between them. In sum, an SCM consists of vertices and edges which represent endogenous variables X and their functions f [36]. A path, consisting of vertices, symbolizes their causal relationship.

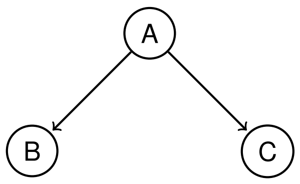

Figure 3.4: An SCM with the vertices A, B, and C as exogenous variables and the edges symbolizing the causal function f

Conditional Independence

The variables within the SCM can be conditionally independent given random variables Z [36]. This means, if two variables X and Y are conditionally independent, they do not have a causal relationship [9]. Hence, they do not influence each other in a causal way. In order to be conditionally independent, they need to be d-separated in the causal Bayesian network $G = (V, E)$ [31].

For two variables X and Y to be d-separated, their vertices v_x and v_y need to be d-separated in $G = (V, E)$ [31]. This means, all paths connecting v_a and v_b are blocked by a set of random variables Z [11]. The variable Z can be observed or unobserved. Observed means that evidence regarding the variable Z is available. Here, all paths means all paths directed in any way from X to Y [31]. If paths are blocked they are considered not active and no active paths constitute conditional independence [11].

Paths between v_x and v_y are considered blocked by Z for the following cases:

- Causal chain: There is an observed $z \in Z$ on the path that is connected by an incoming and outgoing edge: $\rightarrow z \rightarrow$
- Common cause: There is an observed $z \in Z$ on the path that is connected by two outgoing edges: $\leftarrow z \rightarrow$
- Common effect: There is an unobserved $z \notin Z$ on the path that is connected by two incoming edges: $\rightarrow z \leftarrow$. Furthermore, z cannot be connected to a descendant of Z.

For the causal chain and the common cause $z \in Z$ is observed. In both cases the independence of X and Y can be proven given the conditional probability $P(y|x, z)$ of y under x and z. For a observed $z \in Z$ with the joint probability $P(x, z, y)$ of x, z, and y the conditional probability becomes

$$P(y|x, z) = \frac{P(x, z, y)}{P(x, z)} = \frac{P(x)P(z|x)P(y|z)}{P(x)P(z|x)} = P(y|z) \tag{3.6}$$

As the joint probabilities can be written as a product of the conditional probabilities, $P(y|x, z)$ becomes $P(y|z)$. This equation holds for the case of a causal chain and the common cause. Here the evidence of a random variable $z \in Z$ eliminates the causal influence of x. For the cause of common effect, the random variable $z \notin Z$ needs to be unobserved to block the path. Thus, if the path between X, and Y is blocked given Z, X and Y are conditionally independent.

Interventions

A way to find causal effects in a given SCM is through interventions. Interventions are different to observations [36]. With observations, one would observe a variable y having the value 4 and then conclude the impact. On the contrary, interventions change the variables to analyze their impact [36]. By intervening on a value, the system receives another distribution that differs from the observational distribution [36]. The goal is then to observe whether parts of the system change after the intervention [36]. In the case that any intervention can lead to an arbitrary change of system, the two distributions become unrelated [36]. To execute these interventions, one uses the $do(\cdot)$ operator.

The $do(\cdot)$ operator is a way to perform interventions on a given SCM [33]. For hard interventions the intervened variable is set to a certain value. So for example, if one intervenes on x, one sets $do(x = 4)$ [36]. Given that x has a causal effect on y, the corresponding SCM has a changed distribution $P\left(y \middle| do(x = 4)\right)$ for y [36]. For soft interventions on the other hand, one changes the function of x instead of setting a fixed value [36]. This means, if $x = g_x\left(y\right) + N_x$, with N_x being a random noise and $g_x\left(y\right)$ being the causal dependency between x and y, one can change only the noise. So, $x = g_x\left(y\right) + N_x$ becomes $x = g_x\left(y\right) + \tilde{N}_x$ [36]. Thus, with the $do\left(\cdot\right)$ operator, one can perform hard as well as soft interventions.

It should be noted that the conditional expectation $\mathbb{E}\left[y \middle| x = 1\right]$ is not the same as the interventional expectation

$$\mathbb{E}\left[y \middle| do(x = 1)\right] \tag{3.7}$$

In this equation, the expectation of the random variable y is taken over its interventional distribution $P\left(y \middle| do(x = 1)\right)$ and not the conditional distribution [33].

In sum, one can use interventions to asses the causal relationship of variables by changing the distribution induced into the model. For that, soft and hard interventions using a $do\left(\cdot\right)$ operator are suitable. Furthermore, soft interventions change the function of the variables while hard interventions set the variable to a fixed value.

Overall, structural causal models are a way to model causal relationships between data. The structural causal model can then be translated into a graphical model which is a Bayesian Network. To classify the relationship between the variables, one can use conditional independence and interventions. Using SCMs, interventions, and the $do\left(\cdot\right)$ operator, the causal effect of variables within a system can be determined.

3.2.2 Causal Inference

Another main concept in this thesis is causal inference. Essentially, causal inference is the concept of determining the causal effects on an outcome of a system [31]. This means, given

- Covariates X as background variables: e.g. features of a neural net
- Treatments T as actions: e.g. manipulation, intervention or additional feature of a neural net
- Outcome Y as a result: e.g. predicition of a neural net
- Confounders Z as a causal effect on treatment and outcome determined outside of the system: e.g. input samples of a neural net

causal inference determines whether the treatment has a causal effect on the system [48]. In essence, causal inference learns the effect a treatment has on a system. The causal effect measured with causal inference can then be used as a causal attribution. A causal attribution is defined as an explanation for causal behavior [48]. The causal attribution in this thesis is the Average Causal Effect defined later in this section. According to [7], there is a uniform way to analyze causal attributions with the following axioms.

The Axioms for Causal Attributions

According to Chattopadhyay et al. in [7], five axioms were developed, in order to assess the quality of causal attributions. The five axioms are conservativeness, sensitivity, implementation invariance, symmetry preservation, and input invariance [7]. The quality and functionality of a causal attribution like the Average Causal Effect can be determined by analyzing it with these five features.

The first axiom, conservativeness, states that the relevance of any neuron consists of the sum of incoming messages [2]. The relevance in this case is how influential a neuron is on the prediction [2]. Thus, the relevance score R_{ij} of neuron j in layer i is defined as

$$R_{i_j} = \sum_{k=0}^{K} R_{(i,i+1)_{(j \leftarrow k)}} \tag{3.8}$$

where $\forall k \in K$ is the indice of an incoming messages and $R_{(i,i+1)_{(j \leftarrow k)}}$ is an incoming message from the neuron k in layer $i+1$ to neuron j in layer i. All of the summed up incoming messages $R_{(i,i+1)_{(j \leftarrow k)}}$ for $\forall k \in K$ create the relevance score of neuron j in layer i. In the case of causal attribution, the first axiom, conservativeness, would be interpreted as

$$\sum_{i} atr_i = f(inp) - f(baseline) \tag{3.9}$$

with atr being the causal effect of the input, which can be seen as a relevance score. Furthermore, $f(inp)$ is the neural net function with the input inp and $f(baseline)$ is the neural net function with a baseline $baseline$ of the prediction [7]. The baseline should be a neutral predicition that is set along the decision boundaries of the neural net [7]. Equation (3.9) shows how much the prediction changes against a neutral prediction for all inputs. These changes should then summed up to be the causal effect.

The sensitivity assesses the sensitivity in regards of the features [44]. Given a system with a predefined baseline and inputs, the number of features for any input is changed in comparison to the baseline [44]. Then, the predictions of the inputs and baseline are observed [44]. If the predicition of the inputs and the baseline differ, the differing feature should have a non-zero attribution [44].

The axiom, implementation invariance, analyzes the models itself. Given two functionally equivalent models, the measured attribution should have the same value for both models [44]. Functionally equivalent means that the outputs of the models are equal for all inputs although the models might have a different implementation [44].

A causal measurement should also be symmetry preserving. This means, two variables (x, y) are symmetric with respect to a function F if and only if $F(x, y) = F(y, x)$ for all values of x and y [44]. For attribution methods this means, if all inputs have identical values for symmetric variables and all baselines have identical values for symmetric variables, then the symmetric variables receive identical attributions [44].

Lastly, input variance assess whether a model is invariant to a constant shift in input [7]. Thus, does the output of the model y change for a linear shift of the model input x. Given a model $\forall i, f_1\left(x_1^i\right)$ with the input $x_1^i \in X_1$ and a model with the identical weights and biases $\forall i, f_2\left(x_2^i\right)$ with the input $x_2^i \in X_2$, a linear shift is performed for the input x_1^i making $x_2^i = x_1^i + m_2$ with the linear constant m_2 [23]. The neuron activation z_2 of model f_2 becomes

$$z_2 = \mathbf{w}^T x_2 + b_2 = \mathbf{w}^T \left(x_1^i + m_2\right) + b_1 - \mathbf{w}^T m_2 = \mathbf{w}^T x_1 + b_1 = z_1 \tag{3.10}$$

with $x_2^i = x_1^i + m_2$ and $b_2 = b_1 - \mathbf{w}^T m_2$ the neuron activation of f_2 is equal to the neuron activation of f_1 [23]. Thus, the output of f_1 and f_2 are equal making $f_2\left(x_2^i\right) = f_2\left(x_1^i + m_2\right) = y^i = f_1\left(x_1^i\right)$ [23].

The Average Causal Effect

The Average Causal Effect (ACE) is a mechanism, based on causal inference, to determine the causal effect of a binary random variable x on another random variable y within an SCM [36]. As defined in [36], given an SCM with a binary x and y the ACE is

$$ACE = \mathbb{E}\left[y \middle| do(x = 1)\right] - \mathbb{E}\left[y \middle| do(x = 0)\right] \tag{3.11}$$

This means, the ACE takes the concept of interventions to compute the interventional expectation for y. The interventional expectation is computed with an intervention of $do(x = 1)$ and with an intervention of $do(x = 0)$. Then, one subtracts them from one another, which shows us the causal influence x has on y. For example, if x is a variable to show whether a patient got a treatment (1 patient is treated, 0 patient is not treated) and y would be the outcome of that treatment, $\mathbb{E}\left[y \middle| do(x = 1)\right]$ would be the average outcome a number of treated patients had. Furthermore, $\mathbb{E}\left[y \middle| do(x = 0)\right]$ would be the average outcome a number of non-treated patients had. It should be noted that the treatment itself is an intervention. By subtracting the two expectations, one can see if the treatment had an effect relatively to non-treated patients. Thus, the ACE shows the causal effect of a binary random variable x on another random variable y [36]. In this thesis, the definition of the ACE is expanded to non binary variables.

Using the ACE on a causal Bayesian network means intervening with the graph itself [36]. A causal Bayesian network $G = (V, E)$ creates a joint distribution over its vertices $P_V = \prod_{v_i \in V} P\left(v_i \middle| parents(v_i)\right)$ [36]. This means, if one intervenes with a random variable x_i, then x_i depends on the intervention and not the original causal model [36]. Thus, this process changes the distribution P_V. By changing the distribution, the structure of the causal Bayesian network G changes as well. Hence, intervening with x_i means removing incoming edges to the vertices V_{x_i} of x_i of the graph G [31]. The changed distribution of G is defined as $P_{(V|do(V_{x_i}))} = \prod_{v_i \in V - V_{x_i}} P\left(v_i \middle| parents(v_i)\right)$, with $do(V_{x_i})$ being the intervention [36]. Furthermore, $V - V_{x_i}$ shows that the interventional distribution ignores the intervened random variable x_i. Additonally, an intervened causal Bayesian network automatically implies an intervened SCM. An intervened SCM $M^i(X, U, f^i, P_u)$ is identical to an SCM $M(X, U, f, P_u)$ with the intervention $do(x = \hat{x})$ [36]. Here, one replaces the causal mechanism f_x of x with the constant function $\hat{x}$ [36]. Thus, for f all the instances of x are replaced with the arguments of $\hat{x}$ creating f^i [36]. Hence, using the ACE on causal Bayesian networks is the equivalent of changing its structure and joint distribution.

Overall, the ACE uses interventions to evaluate the causal effect of a binary random variable x. It does that by calculating the difference between both intervention scenarios and therefore determining the relative causal effect of x. Furthermore, using the ACE means intervening with the causal Bayesian network and the SCM of the system. For the causal Bayesian network, the intervention changes the structure of the graph and subsequently the graph's joint distribution.

In sum, causal modelling means creating a structure to show the causal relationship between various random variables like in Figure 3.3. For that SCMs can be used and transferred into a graphical model the causal Bayesian networks. With SCMs and causal Bayesian networks one can determine conditional independence. Furthermore, interventions can asses the causal effect of a random variable x. Furthermore, using interventions the ACE evaluates the average causal effect of a random variable x within a model. Thus, utilizing SCMs, causal Bayesian networks, and interventions together with the ACE the causal effect of random variables can be determined.

To conclude, during this chapter, I present the background of this thesis. This included the structure and mathematical foundations of deep feedforward networks and the Mean Squared Error as a neural net loss function. Furthermore, I explain neural net regularization. More specifically, I introduce dropout regularization and L1/L2 regularization as they directly influence the neurons in the network. Additionally, I present a way to model causal relationships through structural causal models and causal Bayesian networks. Lastly, I introduce interventions and the ACE to measure causal relationships between random variables in a structural causal model. As I establish all main definitions, the state of the art in this area, and the background of this thesis, I start with the approach of this thesis in the next chapter.

4 Methodology

The goal of the approach in this thesis is to answer the question whether creating causal parameters during training enhances the generalization capabilities of the neural net. Thus, if I include a causal regularization mechanism in the neural net training, is the neural net able to generalize even if data is subject to dataset shift. Within the next chapter, I introduce the method developed during this thesis. First, I show the overall system and general idea of the method. Afterwards, I explain how to use neural nets as SCMs. Then, I discuss the Average Causal Effect component of the algorithm. Lastly, I discuss the composed loss function and optimization technique for the parameters.

4.1 The Training Algorithm with the Negative Causal Effect (NCE)

First, I introduce the overall methodology of the method. Thus, I explain the different algorithm parts and how they interact with each other. In Figure 4.1, the overall procedure of the neural net training is described. One can see that the neural net training consists of 3 parts. First the training algorithm forwards propagates all training samples. Then, it calculates the ACEs. Afterward, the trainings algorithm calculates the model parameters using the loss function with the negative causal effect (explained in Section 4.4). Through that, the algorithm maximizes the causal effect of the neural net throughout the model training. Within the ACE calculation, the neural net is sliced (explored in Section 4.2) and the ACEs are calculated for each layer (explained in Section 4.3).

As one can see in Figure 4.1, the loss function consists of two parts. The goal is to maximize the causal effect of each neuron. For that the training algorithm calculates the ACEs of each neuron for each training step. With the ACEs the negative causal effect NCE of the neural net is calculated. Then, the algorithm uses the original loss function, the MSE, and the negative causal effect NCE of the neural net to calculate the model parameters. Thus, I incorporate the ACEs into the regularization term. This aims to make the overall causal effect of the net grow.

The method calculates the negative causal effect of the whole net by calculating the ACEs of each neuron within the neural net. For this, the training algorithm takes the training data and the model in which the training data was forward propagated. Then, the algorithm slices the neural net like in Figure 4.2. First, it calculates the ACEs of each neuron in the first layer. Then, the first layer is cut away treating the remaining layers as a new neural net. The outputs of the first layer become the new inputs. With that, I calculate the ACEs of each neuron in the second layer. For the ACE calculation, the interventional values, the output of the removed neurons before the activation, and the remaining layers of the neural net are used. This is done for every layer within the model. It should be noted that it makes no difference in which direction the neural net is sliced. Afterwards, the algorithm takes the ACEs of each neuron and creates the causal effect of each neuron, explained in Section 4.4. Then, the algorithm takes all causal effects of all neurons and calculates the expectation. Afterward, the NCE is created by changing the sign of that expectation. This way when being incorporated into the loss function the ACE is maximized through the negative causal effect being minimized. I provide more details on that in Section 4.3.

Overall, the ACE is included as a regularization mechanism within model training. For that, the method calculates and averages the causal effect of every neuron in the model. After that, it includes the MSE and the negative causal effect in the loss function. Here, the MSE is responsible for penalizing the difference between target output and prediction of the model. The negative causal effect, on the other hand, has the task to enhance the causality of the neural net. Then, the training algorithm chooses the optimal parameters to minimize the MSE and the negative causal effect. The next section provides more detail on how I use the neural net as an SCM in this method.

4.2 Feedforward Neural Nets as Structural Causal Models (SCM)

Within Figure 4.1, one can see that the neural net is included into the calculation of ACE. In order to do that one needs to interpret the neural net as an SCM. This is because the ACE is based on the fundamentals of causal inference and interventions. Causal inference and interventions need an SCM to be applicable. Thus, I interpret the neural net as an SCM.

As other authors like [7, 24] have already stated, feedforward neural net can be interpreted as an SCM. Generally, a neural net can be seen as a directed acyclic graph [7, 24]. One can write each neuron as a function of the previous layer [7]. So, for the layer $\forall l_i \in L$ the neuron's activation $\forall l_{i,j} \in l_i$ can be calculated using $l_{i,j} = f_{i,j}(l_{i-1})$ with $f_{i,j}$ being the function of $l_{i,j}$ and l_{i-1} being the previous layer. For the input layer l_1, one can view the input samples as independent noise variables $u_j \in U$ with the probability distribution P_u [7]. Thus, the neurons $\forall l_{1,j} \in l_1$, in the input layer l_1, can ve calculated by $l_{1,j} = f_{1,j}(u_j)$ with $f_{1,j}$ being the function of l_1. So, translating that

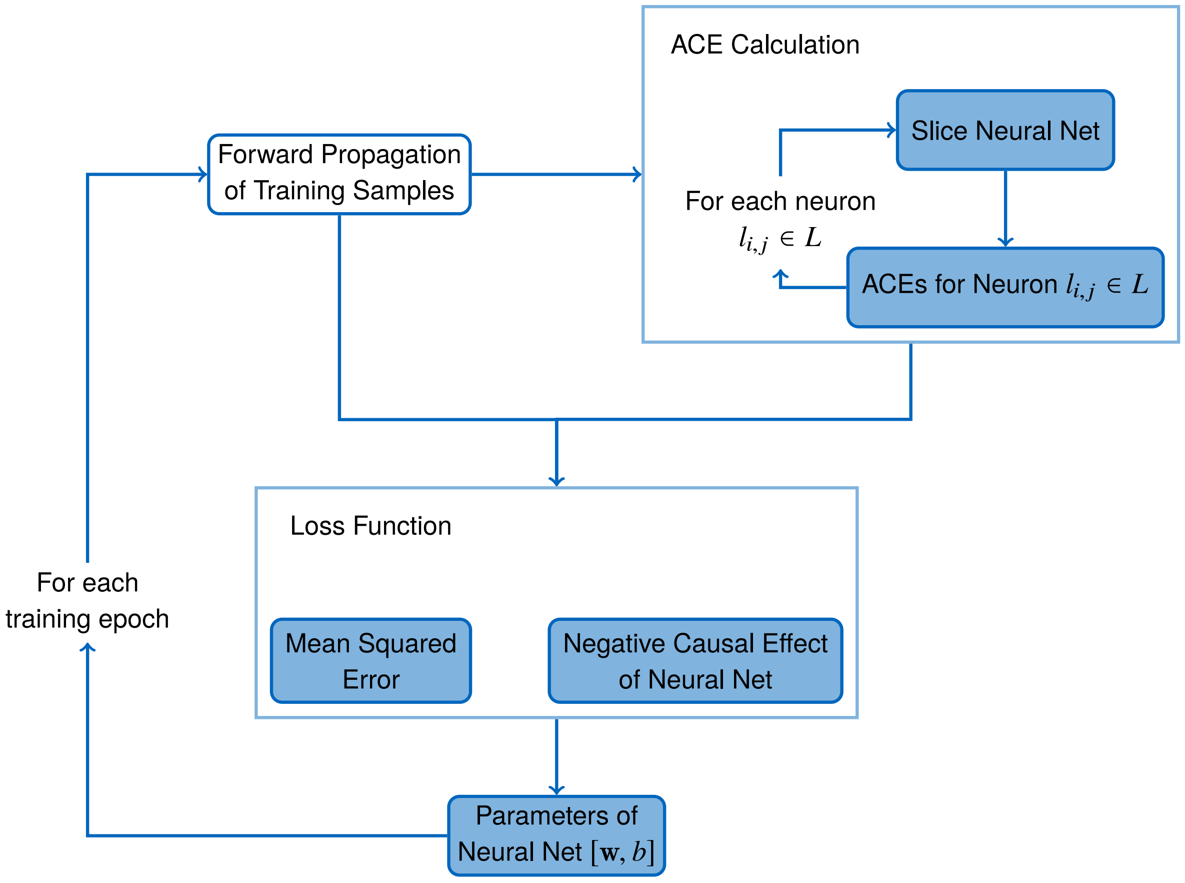
ACE Calculation
Slice Neural Net
For each neuron $l_{i,j} \in L$
ACEs for Neuron $l_{i,j} \in L$
Forward Propagation of Training Samples
Loss Function
Mean Squared Error
Negative Causal Effect of Neural Net
Parameters of Neural Net $[\mathbf{w}, b]$
For each training epoch

Figure 4.1: The method incorporates the usage of SCMs and the calculation of the ACE in the model training

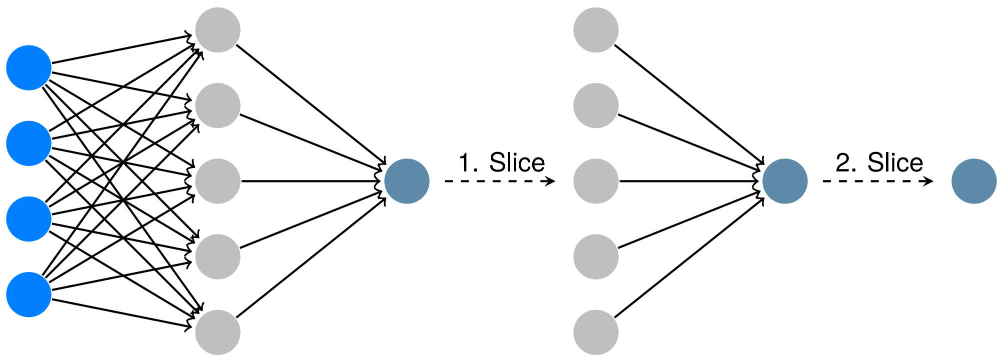

Figure 4.2: Slicing the neural net creates multiple smaller neural nets

to an SCM means that the layers l_i become the endogenous variables $x_i \in X$. The relationship f_i between the layers become the causal function of X. Thus, one can view a neural net as an SCM with the 4-tuple $M\left([l_1, l_2, ..., l_n], U, [f_1, f_2, ..., f_n], P_u\right)$ [7]. Here, l_1 is the input layer and l_n is the output layer. Furthermore, for every layer l_i there is a set of causal functions f_i to determine each neuron within the layer l_i. Additionally, the input variables U are the set of endogenous variables. It is important to note that my goal is to only identify causal relationships within the SCM and not any direction of the causality.

According to [7], one can reduce the SCM $M\left([l_1, l_2, ..., l_n], U, [f_1, f_2, ..., f_n], P_u\right)$ to the SCM $M'\left([l_1, l_n], U, f', P_u\right)$. This means, the input and output layer are directly connected in the acylic graph and I neglect the edges between the hidden nodes. Figure 4.3 shows an example of neglecting the edges of hidden nodes in the acyclic graph. One can do so because the causal function between the output layer l_n and the input layer l_1 depends on all causal functions within the graph. This means, for a neuron $l_{n,i} \in l_n$ the causal function $f_{n,i}(l_{n-1})$ is

$$f_{n,i}(l_{n-1}) = f_{n,i}(f_{n-1,1}(l_{n-2}), f_{n-1,2}(l_{n-2}), f_{n-1,3}(l_{n-2}), ..., f_{n-1,N}(l_{n-2})) \tag{4.1}$$

The short description for that is $l_{n,i} = f'_{n,i}(l_{n-2})$. This shows that the layer l_n can also be written as a function of the layer l_{n-2}. One can backpropagate this function to the first layer transforming $l_{n,i} = f_{n,i}(l_{n-1})$ into $l_{n,i} = f'_{n,i}(l_1)$ for $\forall l_{n,i} \in l_n$. In sum, the causal functions of the output layer is dependend on all the causal function in the graph. This means, one can reduce the SCM to a acyclic graph including only the input and output layer.

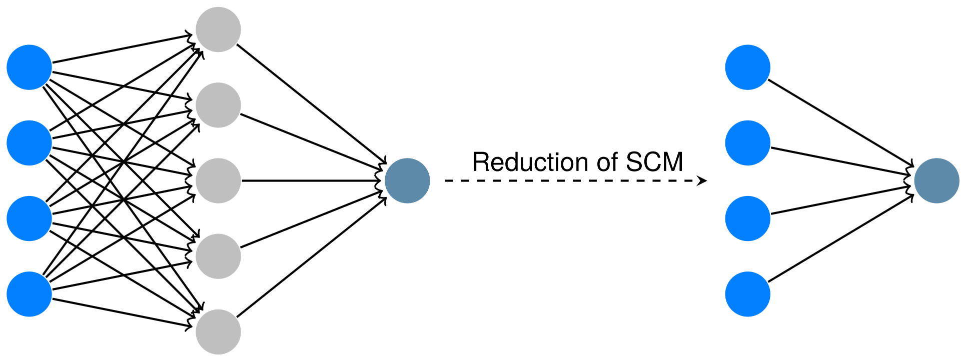

Figure 4.3: The input and output layer can be directly connected in the acyclic graph

As mentioned before, in order to get all the ACEs within the neural net, I slice the model as in Figure 4.2. For each iteration, I remove one layer beginning with l_1. This means, I remove the layer l_1 to l_k and the remaining layers l_{n-k} to l_n become the new neural net. Thus, after each slice, I treat the remaining neural net like a new model. The removed layers provide the new model with the exogenous variable U. In detail, the output of the neurons within the removed layer become the input samples and therefore the exogenous variable U. As I treat the remaining model as a new model, each remaining neural net becomes its own SCM $M\left([l_{n-k}, l_n], U, f', P_u\right)$ for k being the number of removed layers. This creates multiple SCMs of one model. Thus, each hidden layer acts as an input layer once. Without slicing the neural net the causal function of a hidden node would always depend on the original input layer l_1. This is because the neural net can be reduced down to an SCM only including the input and output layer. With slicing the SCM and treating the output of the removed layers as the exogenous variable U, the hidden layers become input layers. This means, I can create the causal functions f' for each hidden node. In sum, with every iteration, I remove a layer that provides new input samples. Furthermore, the remaining layers create a new SCM.

In order to use the ACE, as described in Section 4.3, the neurons need to be d-separated [36]. If they are not the neurons will influence each other. Hence, the interventions performed on the neurons are not separated from each other. This means, if one manipulates neuron $l_{1,i}$ the other neurons $[l_{1,1}, l_{1,2}, ..., l_{1,i-1}, l_{1,i+1}, ...l_{1,n}]$ are also influenced. A consequence would be that the intervention on neuron $l_{1,i}$ would alter the activation of neuron $l_{1,i-1}$, for example. With a changed activation, the then calculated ACEs of neuron $l_{1,i-1}$ would not have their true values, as they depend on the activation of the neuron. This would mean that the ACE of $l_{1,i-1}$ would not have its true value. Thus, I make the same assumption as in [7]. For a given neural net with a reduced SCM $M\left([l_{n-k}, l_n], U, f', P_u\right)$ and a causal Bayesian network $G = (V, E)$ the neurons are d-separated. This makes them conditional independent. I can make this assumption because the features have common exogenous parent vertices V [7]. As mentioned before, if one views a neural net as an SCM the input samples become the exogenous variables [7]. Thus, the input features all have the input samples as exogenous parent vertices in the causal Bayesian network. Looking at the definition of d-separation in Section 3.2.1, one can see that an observed random variable Z can block the path between two variables. This holds if Z is an observed random variable and the two variables on the path have a the common cause Z or are on a causal chain with Z. Thus, as one can see in Figure 4.4, if Z is an input sample, and two neurons would be connected, they would lie on a causal chain with Z or have the common cause Z. The third case, common effect, would mean that the neurons would be the effect of the input samples. Thus, this case can be neglected as the input samples are propagated into the neurons. Here, the input samples are exogenous variables which are observed random variables. Because of the fact that the input samples are observed, the path between the neurons can be considered blocked. As the input samples are observed variables, the features of a neural net can be considered d-separated.

As I produce multiple SCMs out of one network, I apply the assumption to each of the

SCMs. For this, I use the output of the removed layer as the new inputs. After removing a layer, the features of the remaining SCM $M\left(\left[l_{n-k}, l_n\right], U, f', P_u\right)$ have observed exogenous variables in the form of inputs as well. Hence, the new neurons in layer l_{n-k} also have a blocked path between them. In short, the features of a neural net are d-separated due to the inputs blocking the path as exogenous variables. Furthermore, the sliced neural nets do use inputs as exogenous variables as well which means I view the input neurons as d-separated as well. With this, I apply the ACE to all the neurons within the neural net.

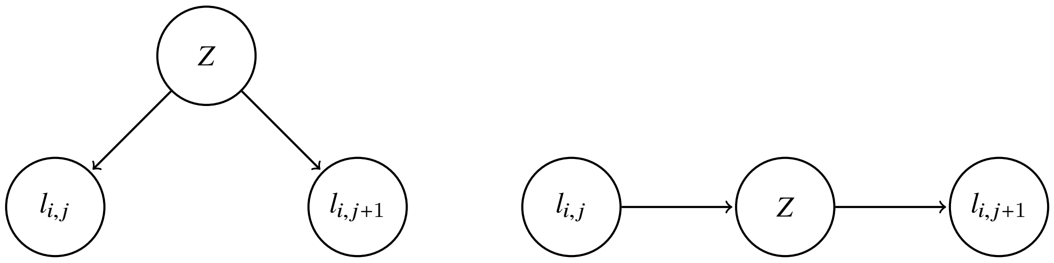

Figure 4.4: The SCM of the input neurons and the input samples acting as the random variable Z

To conclude, each neural net can be viewed as an SCM with the neurons being the endogenous variables X, the functions of the neurons being the causal functions, and the input samples being the exogenous variables U. Due to the causal functions depending on one another one can cut down the acyclic graph to the input and output layer of the system. Additionally, I slice the neurons creating multiple SCMs out of one neural net. This way I can isolate the causal functions for each hidden node. Furthermore, I assume that the features of each created SCM are d-separated due to the input variables acting as exogenous variables. As I can isolate the causal function of each hidden node with an SCM, I calculate the ACE as state in the following section.

4.3 Applying the Average Causal Effect (ACE) to Neural Nets with Continuous Values

Generally, one can apply the ACE to binary values as shown in Section 4.3. As the data used here is continuous, one needs to alter the computation of the ACE, so it can handle continuous data. Chattopadhyay et al., in [7], defined the ACE for the continuous case as

$$ACE^{y}_{do(x_i=\alpha)} = \mathbb{E}\left[y \middle| do\left(x_i = \alpha\right)\right] - baseline_{x_i} \tag{4.2}$$

with y being the output, x_i the intervened input, α the intervention on x, and the $baseline_{x_i}$ being a neutral prediction. The ACE calculates the causal effect of x_i on the output y [36]. Here, the ACE uses the interventional expectation $\mathbb{E}\left[y \middle| do\left(x_i = \alpha\right)\right]$, with the neutral prediction as a reference point [7]. The original definition of the ACE uses a binary variable x. Bringing the medical example back to mind, x represents a treatment. So, for $x = 1$ the doctor gave a treatment to the patient and for $x = 0$ the patient remains untreated. Subtracting both of the cases shows if the treatment had a causal effect, as the untreated patient acts as a ground truth. Thus, the effect of a treatment is calculated relative to a non-treated patient. As other effects not related to the treatment also occur for non-treated patients, these effects are taken out of account. For the continuous case, the baseline represents the ground truth [7]. That's why it should be a neutral prediction. After one calculates the interventional value of a neuron, it is subtracted from that neutral prediction. This way, other side effects occurring in the net are taken out of account. With this relative value, one can identify the real impact of the neuron. In sum, the continuous ACE calculates the causal effect of the intervened neuron x_i has on the output y using a neutral prediction as a reference point and the interventional expectation.

As mentioned before, the baseline should be a neutral prediction. In detail, a neutral prediction would be a prediction along the decision boundary of the model [7]. Furthermore, the baseline could be domain-specific [7]. So, in the case of image data, it could be a fully black picture. However, a fixed baseline has disadvantages as the ACE is dependent on the inputs [7]. Thus, features, which largely deviate from the other features, can generate a high ACE without actually having a large causal effect. Thus, I also make the baseline dependent on the feature x_i. As in [7] the baseline becomes

$$baseline_{x_i} = \mathbb{E}_{x_i}\left[\mathbb{E}\left[y \middle| do\left(x_i = \alpha\right)\right]\right] \tag{4.3}$$

which means the interventional expectation $\mathbb{E}\left[y \middle| do\left(x_i = \alpha\right)\right]$ is averaged over all interventional values of x_i. By averaging the causal effect over all interventions, the side effect occurring within the net during the interventions are taken out of accounts. This way, the effect of the individual interventions can be measured by the ACE. Thus, if the baseline is dependent on x_i, one can avoid errors in the ACE for large differences between the features.

One way to check whether one has a functioning baseline is to verify if it enables the ACE to measure 0 causal effect [7]. If x_i would have no causal effects on y, the interventional expectation would not change [7]. This means, all of the interventional expectations keep the value γ, for example. The mean of all of the interventional expectations then becomes γ as well. Thus, the baseline becomes γ. With all interventional values being γ and the baseline being γ as well the ACE becomes 0.Thus, the baseline introduced in Equation (4.3) enables the ACE to measure 0 causal effect.

The ACE is suitable for this thesis for multiple reasons. First, the ACE provides a metric to evaluate every single neuron and their activations regarding causality. As I aim to tune the weights and biases of the neurons to improve their causality, I need to assess the causality of each neuron separately. Furthermore, as explained in Section 4.4, I use the ACE as a regularization term within the loss function. One way to do this is by adding the ACE as a metric to the loss function. Furthermore, the ACE fulfills the axioms in Section 3.2.2 except for axiom one [7]. Axiom one is relevant if one wants to linearly approximate the neural net [7]. As explained in Section 3.2.2 for axiom one, the subtraction of an input from a arbitrary baseline should be the overall causal attribution of a linear system. However, as the ACE measures the causal strength of the individual neurons, axiom one does not need to be fulfilled [7]. Looking at Equation (4.2), one can see that the axiom sensitivity holds. The ACE gives a non-zero score to values that change the prediction measured against the baseline. Furthermore, the implementation invariance and the symmetry preservation are fulfilled, as the ACE directly calculates the interventional expectation. This means, the ACE does not depend on the implementation of the model [7]. So, as long as the function of the model stays equivalent the implementation does not matter [7]. Lastly, as derived in Section 3.2.2, a non-linear model is invariant to a linear shift in inputs. Thus, the last axiom, input invariance, holds as the ACE depends on the neural net function f. This means, the ACE with a constant shift in inputs stays equal to the ACE without a shift.

To conclude, the one can define the continuous ACE as the the interventional expectation subtracted from a neutral prediction. This way the relative effect of the features can be determined. A robust baseline should be stable for large changes in features. Thus, the baseline I chose here is individual for each feature. Furthermore, the ACE fulfills four of the five axioms for causal attribution. In order to calculate the ACE, one needs to intervene on the input features of the SCM.

4.3.1 Choosing the Interventional Values

As explained in Section 3.2.1, intervening on a variable means setting the variable to a certain value. Due to the lack of prior information, like in [7], I assume that the $do(\cdot)$ operator is equally likely to intervene x_i over the whole domain of x_i. Thus, the intervention for any value between the lowest value low^i of x_i and the highest value $high^i$ of x_i is equally likely [7]. Because of that, the intervention of x_i becomes the uniform distribution $U\left(low^i, high^i\right)$.

Furthermore, like in [7], I only intervene on one feature at a time. So, given the neural net with the causal function $y = f'_y(x_1, x_2, ..., x_k)$ with y being the output and $x_1, x_2, ..., x_k$ being the features, the intervened causal function becomes

$$y = f'_{y|do(x_i=\alpha)}(x_1, x_2, ..., x_{i-1}, \alpha, x_{i+1}, ..., x_k) \tag{4.4}$$

with α being the intervention at the feature x_i [7]. For simplicity, I shorten $f'_{y|do(x_i=\alpha)}$ to f'_y in the following.

4.3.2 Calculating the Interventional Expectation

In the previous chapter, I discussed the definition of the ACE for the continuous case. That mentioned definition uses the interventional expectation, which I introduce in this chapter. The general definition of the conditional expectation is $\mathbb{E}\left[Y = y \middle| X = x\right] = \sum_y yP\left(Y = y \middle| X = x\right)$. For the continuous case the sum can be written as an integral and the conditional expectation becomes $\mathbb{E}\left[Y = y \middle| X = x\right] = \int_y yp\left(Y = y \middle| X = x\right) dy$. With this definition, like in [7], the interventional expectation is

$$\mathbb{E}\left[y \middle| do\left(x_i = \alpha\right)\right] = \int_y yp\left(y \middle| do\left(x_i = \alpha\right)\right) dy \tag{4.5}$$

As I intervene only one feature x_i per iteration, the integral is calculated by going through all other features x_j keeping $x_i = \alpha$. For a large dataset that can be time-consuming.

Due to that, I approximate the ACE with a Taylor expansion like in [7]. The value to approximate around is the interventional mean

$$\mu_j = \mathbb{E}\left[x_j \middle| do\left(x_i = \alpha\right)\right] \tag{4.6}$$

of the features x_j with $\forall x_j \in l_i$ [7]. The interventional mean is the observational mean of the feature x_j with an intervention performed in another feature x_i. The Taylor expansion can be centered around that because, as described in Section 4.2, I assume that the features are d-separated. This means, if the algorithm intervenes one feature in the neural net, the probability distributions of the other features do not change. Hence, if the algorithm intervenes on the feature x_i, for the other features $x_j \neq x_i$ the interventional probability distribution is $P\left(x_j \middle| do(x_i = \alpha)\right) = P(x_j)$. Thus, the covariance and the mean of the remaining features $x_j \neq x_i$ are equal to the covariance and mean of the input data. Those are the observational covariance and mean. Solely the mean of the intervened feature

$$\mathbb{E}\left[x_i \middle| do\left(x_i = \alpha\right)\right] = \mathbb{E}\left[\alpha\right] = \alpha \tag{4.7}$$

changed, as I set x_i to the fixed value of α. Furthermore, as the features are d-separated and the algorithm intervenes on x_i, the feature x_i has no joint variability with the other features [7]. This means, the covariance $Cov\left(x_i, x_j \middle| do\left(x_i = \alpha\right)\right)$ with $\forall x_j \in l_k$ becomes 0. So, within this approximation, the intervened feature is set to the interventional value, and the other features are set to the observational mean and covariance. With this, the ACE can be measured without the values of other features interfering with the intervention of x_i.

In order to approximate the interventional expectation, Chattopadhyay et al. approximated the intervened causal function of the neural net [7]. In Section 4.3.1, the causal function of the intervened neural net is f_y'. Furthermore, the general definition of a Taylor expansion for a function $f(x)$, and the approximation point $x = a$, is

$$T_{f(x;a)} = \sum_{n=0}^{\inf} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{4.8}$$

with n being the order of derivatives. Thus, the second Taylor expansion for a function $f(x)$, and the approximation point $x = a$, is $f(x) = f(a) + \nabla^T f(a)(x-a) + \frac{1}{2}(x-a)^T \nabla^2 f(a)(x-a)$. Like in [7], using that the second Taylor expansion around the interventional means $\mu = \left[\mu_1, \mu_2, ..., \mu_k\right]^T$ of the input layer l_i, the causal function of the layer l_i becomes

$$f_y'(l_i) = f_y'(\mu) + \nabla^T f_y'(\mu)\left(l_i - \mu\right) + \frac{1}{2}\left(l_i - \mu\right)^T \nabla^2 f_y'(\mu)\left(l_i - \mu\right) \tag{4.9}$$

In order to get the interventional expectation, one applies the interventional expectation on both sides of the equation. As the expectation is linear, the summands of Equation (4.9) can be divided into seperate individual expectations. This means, I can take the expectation of the first and second order term separately. For the first order term $f_y'(\mu)$ the expectation becomes $\mathbb{E}\left[f_y'(\mu)\middle| do\left(x_i = \alpha\right)\right] = f_y'(\mu)$. For the second order derivative the expectation is derived as

$$\begin{gathered}\mathbb{E}\left[\nabla^T f_y'(\mu)\left(l_i - \mu\right)\middle| do\left(x_i = \alpha\right)\right] = \\ \nabla^T f_y'(\mu)\left(\mathbb{E}\left[l_i \middle| do\left(x_i = \alpha\right)\right] - \mathbb{E}\left[\mu \middle| do\left(x_i = \alpha\right)\right]\right) = 0\end{gathered} \tag{4.10}$$

as $\nabla^T f_y'(\mu)$ is a fixed value and the expectation is linear, they can be put in front of the expectation. The expectation $\mathbb{E}\left[\left(l_i - \mu\right)\middle| do\left(x_i = \alpha\right)\right]$ can also be divided into two separate expectations, because the expectation is a linear operator. Moreover, by taking the interventional expectation of the input layer l_i, one gets the interventional mean

μ. This is because the interventional means μ are the means of the input features in layer l_i. Furthermore, the interventional expectation of μ is μ itself. Thus, the first-order derivative becomes 0. This results in the interventional expectation becoming

$$\mathbb{E}\left[f_y'(l_i)\middle|do\left(x_i=\alpha\right)\right]=f_y'+\mathbb{E}\left[\frac{1}{2}\left(l_i-\mu\right)^T\nabla^2 f_y'(\mu)\left(l_i-\mu\right)\middle|do\left(x_i=\alpha\right)\right] \quad (4.11)$$

Next, to extracted the covariance, I apply the trace operator on the second-order term [7]. The trace $Tr(A)$ of a 1×1 matrix A is simply the 1×1 matrix A itself. So, it is possible to use the trace, on the interventional expectation, because one can view the expectation within the trace as a 1×1 matrix. Thus, the interventional expectation from Equation (4.11) becomes

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{2}\left(l_i-\mu\right)^T\nabla^2 f_y'(\mu)\left(l_i-\mu\right)\middle|do\left(x_i=\alpha\right)\right]=\\
&\frac{1}{2}\nabla^2 f_y'(\mu)Tr\left(\mathbb{E}\left[\left(l_i-\mu\right)^T\left(l_i-\mu\right)\middle|do\left(x_i=\alpha\right)\right]\right)=\\
&\frac{1}{2}\nabla^2 f_y'(\mu)\mathbb{E}\left[Tr\left(\left(l_i-\mu\right)^T\left(l_i-\mu\right)\middle|do\left(x_i=\alpha\right)\right)\right]=\\
&\frac{1}{2}\nabla^2 f_y'(\mu)\mathbb{E}\left[Tr\left(\left(l_i-\mu\right)\left(l_i-\mu\right)^T\middle|do\left(x_i=\alpha\right)\right)\right]=\\
&\frac{1}{2}Tr\left(\nabla^2 f_y'(\mu)\mathbb{E}\left[\left(l_i-\mu\right)\left(l_i-\mu\right)^T\middle|do\left(x_i=\alpha\right)\right]\right)
\end{aligned} \quad (4.12)$$

Using the linearity of the trace operator and linearity of the expectation, $\frac{1}{2}$ as well as $\nabla^2 f_y'(\mu)$ can be placed outside of the trace operator. The linearity of the trace operator allows us to push the expectation outside. Furthermore, the trace operator $Tr(AB)$ can also be written as $Tr(BA)$. Generally, matrix multiplications are not commutative. However, if A is of dimension $m\times n$ and B has the dimension $n\times m$ the property holds. This is possible, as the trace is the sum of all diagonal elements of a matrix, thus changing the order of the matrices does not change the result of the trace. For this equation, viewing $\left(l_i-\mu\right)^T$ as A and $\left(l_i-\mu\right)$ as B, the order can be changed. The property holds, as $\left(l_i-\mu\right)^T$ is the transpose of $\left(l_i-\mu\right)$. Then again due to the linearity of the trace operator, the expectation can be pushed inside again. Furthermore, the covariance is defined as $Cov(X)=\mathbb{E}\left[\left(X-\mathbb{E}[X]\right)\left(X-\mathbb{E}[X]\right)\right]$. Due to that, the interventional expectation $\mathbb{E}\left[\left(l_i-\mu\right)\left(l_i-\mu\right)^T\middle|do\left(x_i=\alpha\right)\right]$, in Equation (4.12), becomes the interventional covariance.

Overall, the interventional expectation is approximated with the second Taylor expansion. Due to using the trace and the linearity of the expectation one can reduce the function to

$$\mathbb{E}\left[y\middle|do\left(x_i=\alpha\right)\right]=f_y'(\mu)+\frac{1}{2}Tr\left(\nabla^2 f_y'(\mu)Cov\left(l_i\middle|do\left(x_i=\alpha\right)\right)\right) \tag{4.13}$$

as $f_y'(l_i)$ is the output y of the neural net [7].

Using the approximation and the interventional values from Section 4.3.1, the interventional expectation is computed according to Algorithm 1 [7]. Here, the inputs are the outcome y, the intervened neuron x_i, the interventional values $U\left(low^i, high^i\right)$, number of interventions m, the observational means μ, the observational covariance matrix $Cov(X|do\left(x_i=\alpha\right))$, and the causal function of neural net f_y'. It should be noted that the causal function of the neural net f_y' is the function of the neural net f_y. Then, as the covariances between the intervened input and all other inputs are 0, the algorithm sets the corresponding values within the observational covariance matrix to 0 [7]. After that, the algorithm intervenes the input neuron x_i for values between the lowest value low^i of x_i and the highest value $high^i$ of x_i. This is done over a uniform distribution and for a number of interventions n.

This way, the variable x_i is set to the fixed value of α throughout the intervention. With the observational means and the interventional values the input $\mu_\alpha = \left[\mu_1,\mu_2,...,\alpha,...\mu_{n-1},\mu_n\right]$ is created for each iteration. Then, for each μ_α, the algorithm computes the interventional expectation according to Equation (4.13).

Algorithm 1 Calculate the Interventional Expectation $\mathbb{E}\left[y\middle|do\left(x_i=\alpha\right)\right]$ for the intervened neuron x_i [7]

```
Input: outcome $y$, intervened neuron $x_i$, interventional values $\alpha = [\alpha_1,\alpha_2,...,\alpha_m]$, observational means $\mu = \left[\mu_1,\mu_2,...\mu_n\right]$, observational covariance matrix $Cov(X|do\left(x_i=\alpha\right))$, causal function of neural net $f_y$
Output: Interventional Expectation $\mathbb{E}\left[y\middle|do\left(x_i=\alpha\right)\right]$ for each interventional value
Require: $Cov(x_i,x_j|do\left(x_i=\alpha\right))=0, \forall x_j \in X$
  for $\alpha = [\alpha_1,\alpha_2,...,\alpha_m]$ do
    $\mu_\alpha = \left[\mu_1,\mu_2,...,\alpha,...\mu_{n-1},\mu_n\right]$
    $\mathbb{E}\left[y\middle|do\left(x_i=\alpha\right)\right] = f_y'(\mu_\alpha)+\frac{1}{2}Tr\left(\nabla^2 f_y'(\mu_\alpha)\cdot Cov\left(X\middle|do\left(x_i=\alpha\right)\right)\right)$
  end for
```

In sum, the continuous ACE is the interventional expectation subtracted from a baseline. That baseline is the average of the interventional expectation. Using that reference point, the ACE becomes a relative value to the neutral prediction. Furthermore, the interventions done on the input neurons are along the uniform distribution within the domain of the intervened values. Additionally, the interventional expectation is approximated with the second Taylor series due to large computation costs. This way, the interventional expectation can be calculated using the observational mean, observational covariance with the interventional values of the intervened neuron x_i.

4.4 Introducing the ACE as a Neural Net Regularization

The main goal of this thesis is to introduce a regularization technique for neural net training, which should make the neural net more causal. I do this by using the negative causal effect NCE, as a regularization term. I construct the regularization term similarly to the L1 and L2 regularization explained in Section 3.1.3. The new loss function L_T consists of the original loss function L_D and the regularization term L_W with the regularization parameter λ. Using the NCE as the regularization term the loss function becomes

$$L_T = L_D + \lambda L_W = L_D + \lambda NCE_N \tag{4.14}$$

Generally, the training algorithm minimizes the loss function. Thus, it drives the loss function against zero. This means, the NCE is minimized in this case as well.

For a better understanding, I dive deeper into the calculation of the NCE. Calculating the NCE requires the causal effect $CE_{x_i}^y$ of each neuron within the net. As described in Section 4.2, I slice the the SCM of the neural net creating multiple SCMs with a hidden layer as an input layer. Then, I calculate the causal effect using the interventional expectation for each neuron of the input layers. That way, I can compute the $CE_{x_i}^y$ for every neuron in the neural net. Using the $CE_{x_i}^y$ of every neuron, I estimate the overall causal effect in the neural net as

$$NCE_N = -\mathbb{E}_{x_i}\left[CE_{x_i}^y\right], \forall l_{i,j} \in l_i \tag{4.15}$$

Here, I take the causal effect $CE_{x_i}^y$ of each intervened neuron x_i in every layer and calculate their mean. As the negative causal effect is minimized in the loss function, the $CE_{x_i}^y$ would be minimized as well without a negative sign. That's why I introduce the minus in Equation (4.15). This way by minimizing the NCE the causal effect $CE_{x_i}^y$ of each neuron is maximized.

For the calculation of the causal effect $CE_{x_i}^y$ of a single neuron, I use the ACEs $ACE_{x_i}^y$ of that intervened neuron for every interventional value. For the following, it is important to understand that the individual ACEs $ACE_{x_i}^y$ of the interventional values can be either negative or positive in this case [36]. A positive ACE means the neuron affects the output in a positive causal way and a negative ACE means the neuron affects the output in a negative causal way [36]. Let's bring the medical example back to mind, where the intervention is a treatment and the output is the outcome for the treated patients. If the ACE for a performed treatment would be positive, the treatment would heal the patient. On the contrary, if the ACE would be negative for a performed treatment, the treatment would worsen the condition of the patient. Thus, a negative ACE, as well as a positive ACE, show a causal relationship between the treatment and the outcome. In this case, the treatments are the neuron activations and the outcome is the prediction. In order to build a net with causal activations, the positive and negative causal relationships should be fostered in order to predict the correct outcome. Thus, the $CE^{y}_{x_i}$ becomes

$$CE^{y}_{x_i} = \left|\text{median}(ACE^{y}_{x_i})\right|, \forall \alpha \in x_i \tag{4.16}$$

Here, I take the median of $ACE^{y}_{x_i}$ for all interventional values $\alpha \in x_i$ applied to the intervened neuron x_i. In this case, I use the median and not the mean because, through testing, I discovered that the distribution of the ACEs of one neuron can be a skewed distribution. With this, the mean would be dominated by the outliers. For this thesis, I assume that the outliers are less representative of the causal relationship. Hence, I use the median to calculate the ACE of a neuron. This median is either negative or positive which shows if the input has a predominantly positive or negative causal effect on the outcome. I assume that this shows a first indication of the final causal relationship. As both, negative and positive causal relationships should be fostered the absolute value is taken. Thus, for activations with a negative median of $ACE^{y}_{x_i}$, the median of $ACE^{y}_{x_i}$ is minimized and for activations with a positive median of $ACE^{y}_{x_i}$, the median of $ACE^{y}_{x_i}$ is maximized.

As discussed before, I aim to tune the weights $\mathbf{w}$ and biases b of the neurons by minimizing NCE. This is possible because the ACEs $ACE^{y}_{x_i}$ for each interventional value are dependent on the weights and biases of each neuron. $ACE^{y}_{x_i}$ is defined as

$$ACE^{y}_{x_i} = \mathbb{E}\left[f'_y \middle| do\left(x_i = \alpha\right)\right] - \mathbb{E}_{x_i}\left[\mathbb{E}\left[f'_y \middle| do\left(x_i = \alpha\right)\right]\right] \tag{4.17}$$

with f'_y being the function of the neural net [7]. As, f'_y depends on the weights $\mathbf{w}$ and biases b, with the regularization the training algorithm determines weights and biases minimizing the negative causal effect NCE of the neural net. By doing this, the algorithm maximizes the causal effect $CE^{y}_{x_i}$ of each neuron.

Thus, I include the negative causal effect NCE of the neural net as a regularization in the loss function. This way the training algorithm determines $\mathbf{w}$ and biases b to maximize the causal effect $CE^{y}_{x_i}$ for each neuron according to amount. This is necessary because a positive, as well as a negative causal effect, should be accounted for, in order to achieve a correct prediction.

To conclude, the neural net is viewed as an SCM with a causal Bayesian network. With this model, the causal effect can be calculated for all input neurons using the interventional expectation. In order to calculate the causal effect for all neurons, the neural net is sliced into multiple SCMs. This way each hidden neuron can act as an input layer and the causal effect can be calculated. Furthermore, the interventional expectation is approximated with a Taylor expansion up to the second order. Lastly, the overall negative causal effect of the neural net is used as a regularization term within the loss function. Through that, the training algorithm maximizes the causal effect of each neuron during model training determining causal weights and biases. Next, the experiments and validation of the proposed method are discussed.

5 Experiments and Results

Within this chapter, I introduce the experiments performed with the developed approach. The goal of this chapter is to show the limitations and capabilities of the developed approach. First, I describe the set-up of the experiment. This entails the used datasets, the applied training approaches, and the investigated model. Furthermore, I discuss the results generated by this experimental set-up. Specifically, I evaluate the test loss and training loss together with the causal effect of the neural net measured. In the following, I discuss the experiment set-up.

5.1 Preparation and Conduct of Experiments

For this section, I summarize the data used on the developed approach. This includes one well-balanced training set, three datasets with a covariate shift and three datasets with a prior probability shift. Furthermore, the training methods are described. Besides the approach developed in this thesis, I include an L1 as well as L2 regularization in the model training. Lastly, I describe the model used for the mentioned methods and data.

5.1.1 The Boston Housing Dataset

As mentioned before, there are three different types of datasets used in this thesis. The datasets are a well-balanced dataset, three with covariate shift, and three with a prior probability shift. For all datasets, the boston housing dataset from [17] is used. The dataset can be used to predict house prices according to different properties of the neighborhood, air pollution, structure of the building, and accessibility [17].

I use 14 variables of the boston housing dataset, which I describe in Table 5.1. From these 14 variables, the neighborhood, the air pollution, the structural and the accessibility variable are the features. Subsequently, the median value of property occupied by owners in $\$10,000$, with the acronym MEDV, becomes the target variable. Furthermore, the dataset has 506 samples overall. For the training, I use 50 samples and I test the model with the remaining samples. For the approach developed here, the 50 training samples need one week for the training. Additionally, as backpropagation is used, all of the training gradients need to be saved. For the 50 training samples the gradients use the 15GB computers, which are available for this thesis, to capacity. Because of the RAM capacity and the time the experiments take the training samples size is reduced. Furthermore, Boston Housing is considered to be a problem located on the easier end when it comes to machine learning and the generalization of the neural net is in the focus for this work. Thus, having a larger test set can be seen as beneficial.

Name of Variable	Description of Variable
Neighborhood Variable	
CRIM	The crime rate per capita by city
ZN	The proportion of residential land zoned for properties over 25,000 sq.ft.
INDUS	The proportion of business areas which are not retail per city
CHAS	The Charles River variable (1 if on banks of the river, 0 if not)
TAX	The rate of full-value property-tax calculated per $10,000
PTRATIO	The rate of pupil-teacher per city
B	With Bk being the proportion of people of color by city - $1000(Bk - 0.63)^2$
LSTAT	The rate of people with lower status by percentage
Air Pollution Variable	
NOX	The nitric oxides concentration in parts per 10 million
Structural Variable	
RM	The average number of rooms per property
AGE	The ratio of homes occupied by owners which where built before 1940
Accessibility Variable	
DIS	The distances to five Boston employment centres weighted
RAD	The index of accessibility to access roads
Target Variable	
MEDV	The median value of property occupied by owners in $1000's

Table 5.1: The variables of the boston housing dataset [17]

In Figure 5.1, one can see the importance of each feature for the model. I assess the importance of each feature by leaving out each feature once in the model training. Then I compare the test MSE afterward. If the feature is important the prediction without this feature will be less accurate. This means, the test MSE is going to be higher. If the feature is less important the accuracy of the prediction will suffer less. Thus, the MSE is going to be lower. After randomly selecting the training and test split and randomly intializing the neural net, I compute the test MSE three times for each feature and then calculate the mean. The results are displayed in Figure 5.1. Here, one can see that the shape of the curve first is concave and then goes into a convex form. As the test MSEs over the value of 0.029 fall into the convex part of the graph, they are

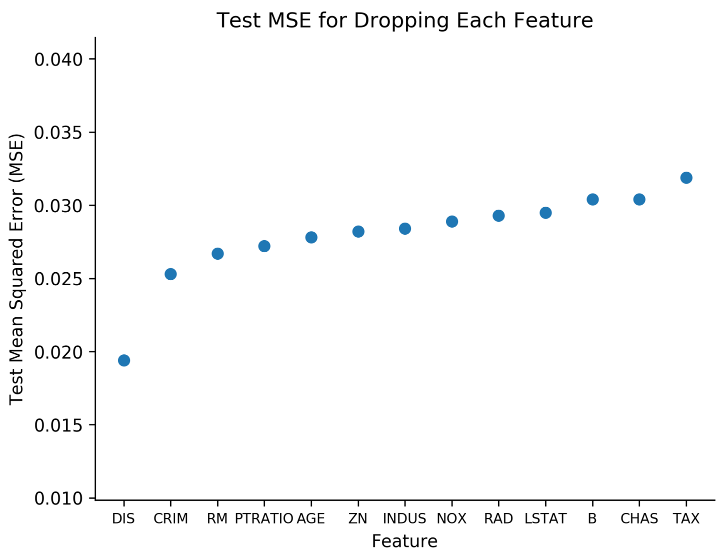

Figure 5.1: The feature TAX, CHAS and B seem to have the strongest influence on the result of the neural net

considered the most important features. Thus, the features TAX, CHAS, and B are the most important while the feature DIS is the least important.

As mentioned before, I use a well-balanced training set. However, I also create datasets with a dataset shift. For the case of covariate shift, it means that the distribution of test and training data differ from one another. For that, a wide, a narrow, and a skewed split are computed. According to Table 5.1, the wide split has a shift in the feature TAX, B, LSTAT, RAD, INDUS, NOX, and RM. For the narrow and skewed split, the shifts are in TAX, B, RAD, CRIM, INDUS, NOX, and RM. As one can see that the wide split has one more important feature with a shift compared to the narrow and skewed split.

Variable with shift more important	Variable with shift less important
The Wide Split	
TAX,B,LSTAT,RAD	INDUS,NOX,RM
The Narrow Split	
TAX,B,RAD	CRIM,INDUS,NOX,RM
The Skewed Split	
TAX,B,RAD	CRIM,INDUS,NOX,RM

Table 5.2: The three splits and the respective variables with a shift divided by importance

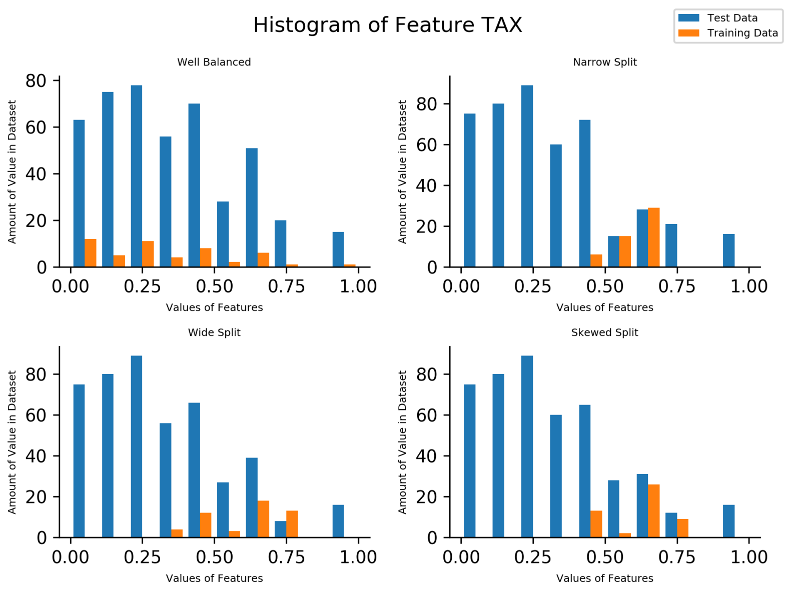

Figure 5.2: The distribution of the TAX feature for different splits

Although, the wide, narrow, and skewed split have shifts in the same features the quality of the prediction will still be influenced differently. A reason for that is how the shifted training data is distributed. In Figure 5.2, one can see a well-balanced test and training dataset as well as the test and training data of the three splits for the feature TAX. All three have a dataset shift for this feature. However, the distributions of the training data for the wide and skewed split are of more uneven shape than the training data for the narrow split. Furthermore, the training data for the wide split for example is distributed a bit wider than for the other splits for example. For feature B, in Figure 5.3, the training data of the narrow and the skewed split are less widely distributed than for the wide split. Generally, the distribution for the training data is of more uneven shape for the wide and skewed split. Furthermore, the wide split has one more shift in an important feature.

Besides the covariate shift, I also generate training data with a prior probability shift. For this, I again generate 3 splits. Looking at Figure 5.4, which is the distribution of the target variable, one can see the three sections for the dataset split. First, I only train the data with the lowest house prices. This means, I take training data only out of the blue section in Figure 5.4. Then I create a training set that only includes mid-range house prices. Thus, I take the samples out of the red area. Lastly, the training samples are taken out of the green section. Thus, they only include high house prices.

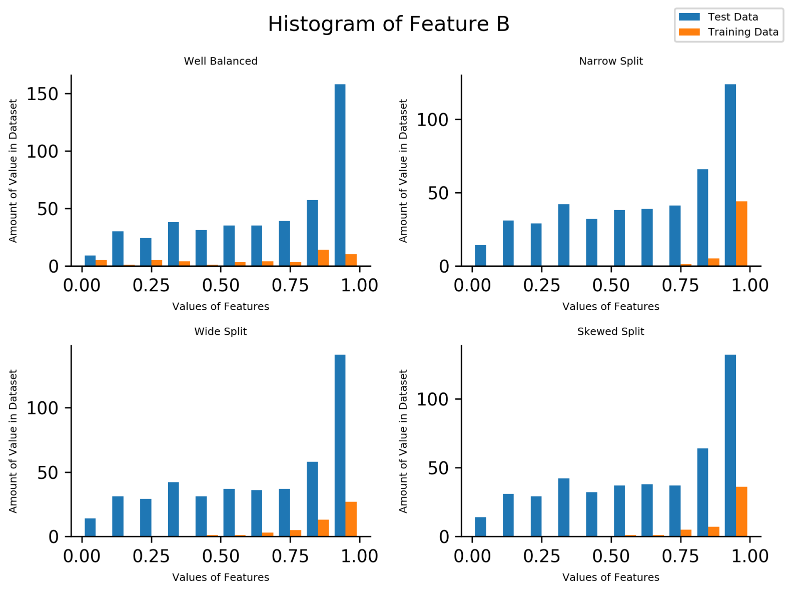

Figure 5.3: The B feature distributed for a well-balanced, wide, narrow, and skewed split

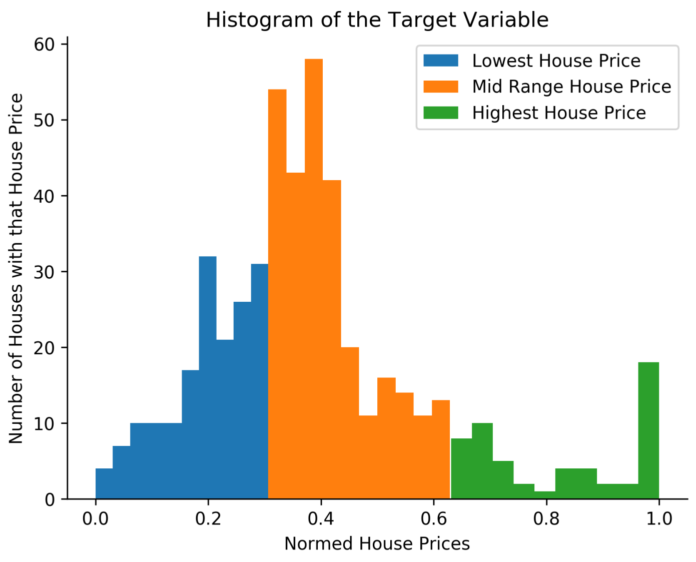

Figure 5.4: The different sections of the target variable

5.1.2 Experiment Set-Up

In this thesis, I train a neural net with the given dataset. The neural net has 13 input neurons and 2 hidden layers, which both have 50 neurons. The last hidden layer results in a single output neuron. For each layer, the activation function used is a sigmoid, as I assume smooth activation functions for the calculation of my method. For the training of the neural net, I apply backpropagation and use the Mean Squared error as a loss function. Furthermore, the Adam optimizer determines the step size for the training process. With the remaining samples, the neural net is tested.

As one can see in Figure 5.5, I take each dataset and train four models with it. First, I train a standard neural net three times. The resulting value from that training and testing will act as a baseline. By running each experiment three times, the results are more representative and reproducible. As the experiments take a week to finish for one dataset, I consider three times to be appropriate in the time frame given for this thesis. Then, I use L1 as well as L2 regularization in the model training. The neural net is trained three times with L1 regularization and three times with L2 regularization. Here, the goal is again to make the results more representative and reproducible. Furthermore, I apply the L1 and L2 regularizations as a baseline as well, as Janzing found the indication that L1 as well as L2 regularization can improve the causality of a neural net [18]. Thus, for L1 and L2 regularization, the only lowest test MSE is displayed here to benchmark my method against the best performing approaches. The last case is that I train a neural net with the approach presented in this thesis. This is done for regularization factors between $e - 05$ and 100. For each dataset and each model, the CE values, the test MSE, as well as the training MSE are examined. For all the cases above the CE is calculated with 10 interventions per feature. Here, again as the experiments take a week and take up a sizable amount of RAM, 10 interventions are chosen. Here the CE value of the whole net is the NCE with a changed sign.

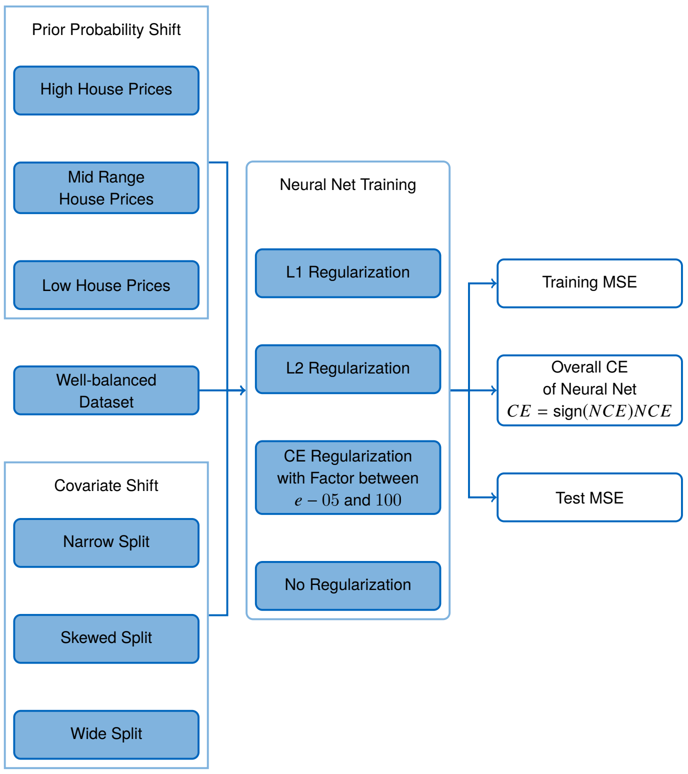

Figure 5.5: The experiment set-up generates the training MSE, the overall CE of the neural net and the test MSE for seven different training sets

5.2 Results and Discussion

I run the described experiments three times for each dataset. For each case, I measure the causal effect of the neural net, the standard deviation of the causal effect within the net, the training MSE, as well as the test MSE. In the following section, I analyze the values measured.

5.2.1 Development of the Causal Effect (CE)

As I mentioned before, the causal effect (CE) of the neural net should be increased through the method developed in this thesis. In Section 4.4, I introduce the CE as a regularization mechanism. The CE is a measurement for the causality of the whole neural net. Thus, I discuss the values of the causal effect measured in the following.

Figure 5.6 shows the CE development over the training epochs for each training method. I train each dataset three times. Then I calculate the standard deviation as well as the mean of the CE. For the mean and standard deviation, I use the values generated by all datasets. Then, I display the CE in Figure 5.6. As one can see, the CE regularization with the factor 100 produces the largest CE value. The other cases generate CE values, which are very close together. However, the CE regularization with factor 10 and the L2 regularization generate a slightly higher mean value compared to the other training methods. Generally, the distribution of the generated CE values has a high standard deviation (shown by the shadows around the lines). As a next step, it is worth looking at the CE value, produced by the different kinds of datasets, separately.

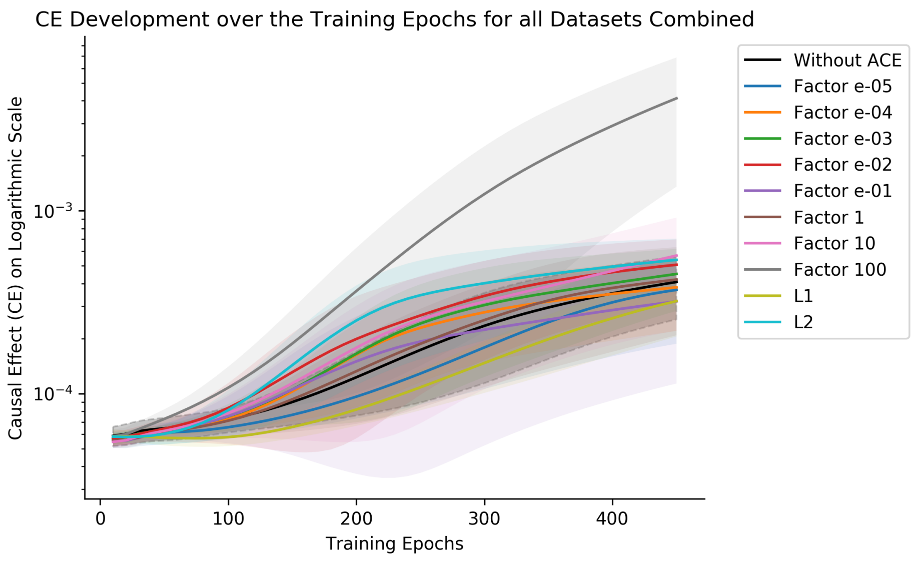

Figure 5.6: The causal effect for all datasets is the highest for the CE regularization with factor 100, factor 10, and the L2 regularization

As one can see in Figure 5.7, for the datasets with an introduced prior probability shift the CE regularization with the factor 100 as well as 10 also provides the largest CE value. Furthermore, the L2 and L1 regularization do not seem to enhance the CE value of the neural net significantly more than not applying a regularization at all. For a dataset with a covariate shift, displayed in Figure 5.8, the CE regularization with the factor 100 is again the largest value. Furthermore, the L2 regularization is the second-highest value. However, the third highest value is the CE regularization with the factor $e-02$. Generally, the mean of the CE value does not get smaller with a smaller factor. Specifically, the factors 10 and 1 lie under factors like $e-02$ and $e-03$. Lastly, the well-balanced dataset, in Figure 5.9 shows, that the CE regularization with factor 100 and factor 10 generates the largest CE value. Similarly, to the plot of all datasets combined, the other factors besides the factors 100 and 10 lie very close to each other.

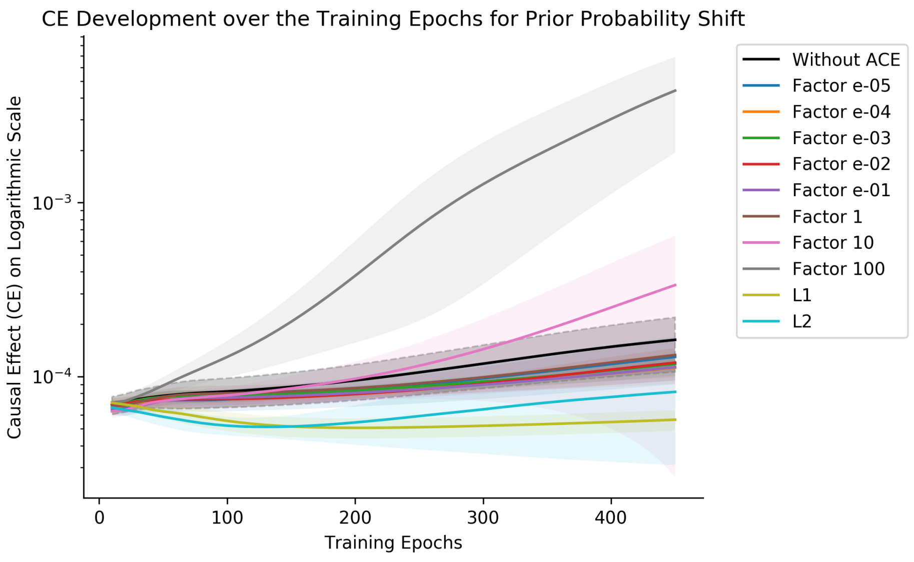

Figure 5.7: For datasets with a prior probability shift the CE regularization with the factor 100 and 10 achieve a noticeable higher CE value

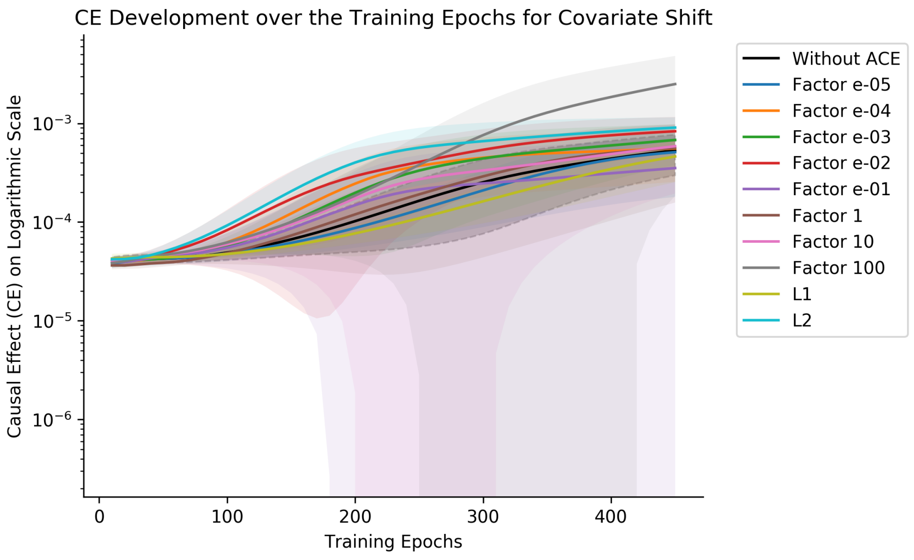

Figure 5.8: The CE value for datasets with a covariate shift is the highest for CE regularization with factor 100, $e-02$ and the L2 regularization

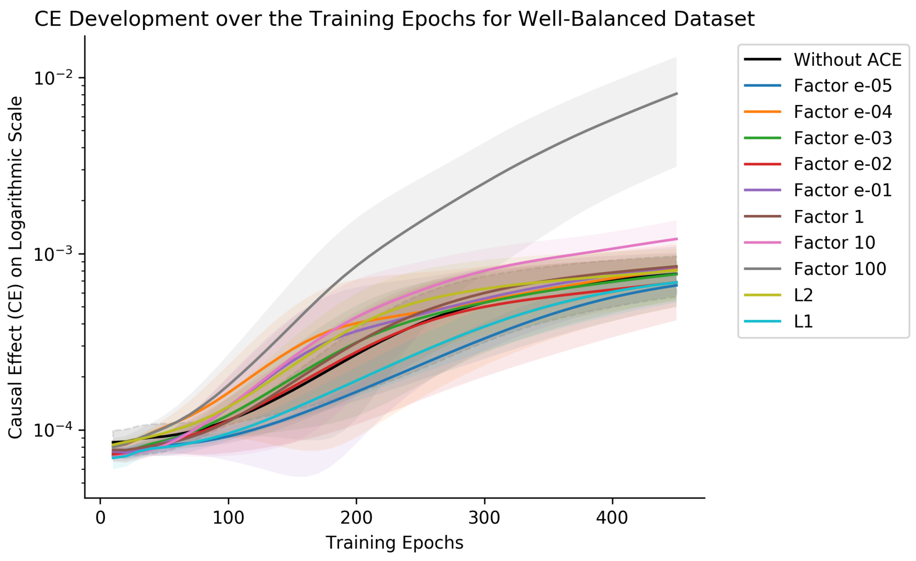

Figure 5.9: For well-balanced datasets the factors 10 and 100 with the CE regularization achieve the highest CE value

Generally, the CE regularization with the value 100 seems to generate the largest CE value of the neural net. For the dataset with a prior probability shift as well as the well-balanced dataset, the CE values of other CE regularizations, the L1 regularization, the L2 regularization as well as the standard neural net training lie near each other. This is because the value of the CE is small compared to the Training MSE. The CE's highest value lies in the area of 10^{-3}. In comparison, the MSE's smallest value lies in the area of 10^{-2}. This means, if the CE value is multiplied with factors between 10 and $e-05$, its impact compared to the MSE can be seen as small. Thus, the CE value will not grow as large for factors between 10 and $e-05$ compared to factor 100.

However, for the covariate shift one can see that with increasing factor for the CE regularization the CE value does not necessarily increase in its mean. As I mentioned before, the CE value has a high standard deviation. One reason could be the randomly initialized weights and biases. In the first epoch of the training, the weights and biases are randomly initialized. This determines the neuron activations within the neural net. As the interventions are dependent on the neuron activations, they are also dependent on the weights and biases. Thus, if the weights and biases are initialized in a way that is less causal, it makes it harder for the CE to increase. This can result in a large standard deviation of the CE value. Thus, even with a higher factor for the CE regularization, the CE value can be smaller if the weights and biases are randomly initialized in a less causal way. The high standard deviation could also stem from the absolute value used in the CE calculation. As this is also related to the MSE, I discuss that in Section 5.2.5.

Another way to see if a causality measurement is working is to check whether it is able to measure zero causal effect. At the beginning of the training, the causality should be zero or at least very close to zero. This is the case because the neural net as not identified any causal relationships yet. It can happen that part of the seed set at the beginning of the training is causal. That's why it can also just be close to zero. As one can see in all four figures, at the start of the training, when all the weights and biases are set randomly and no learning has been executed, the CE is zero or very close to zero. In the following, I compare the test MSE to the CE values discussed in this section.

5.2.2 Causal Effect (CE) Relative to the Test Mean Squared Error

In this thesis, I pose the question of whether the generalization ability of the neural net increases with a larger CE in the neural net. I test the generalization ability of the neural net by using the datasets with a dataset shift. Then, I calculate the test MSE for these datasets and the test MSE that is generated by a well-balanced dataset. Afterward, I take the test MSEs and compare them in relation to the calculated CE value. Here similarly to the figures before, I take the mean of the test MSE and the CE for all training methods trained with all datasets.

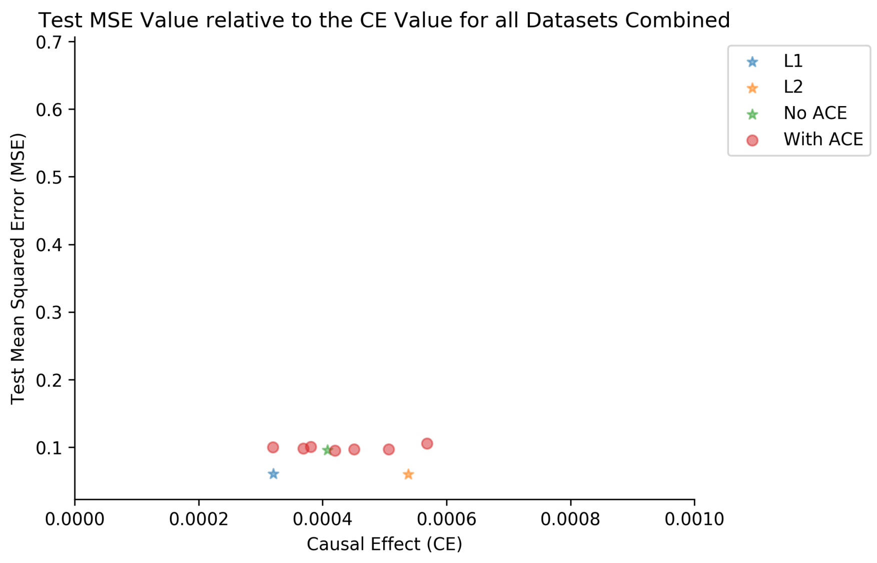

Figure 5.10: The test MSE for all datasets combined do not show a correlation between a improved test MSE and a higher CE value

In Figures 5.10 all datasets are combined. Here, the test MSE values do not seem to decrease with larger CE. However, looking at the datasets separately, one can see two different trends in Figure 5.11, Figure 5.12 and Figure 5.13. For the case of prior probability shift in Figure 5.11, there is still no improvement of the test MSE with increasing CE. But, for the case of covariate shift in Figure 5.12 and the well-balanced dataset in Figure 5.11, the test MSE slightly improves for a growing CE. Furthermore, one can observe that a very large CE results in a large test MSE. This indicates that the test MSE is only improved by a large CE until the CE reaches a certain value. When the CE becomes a too large value the test MSE will not continue to improve. Furthermore, for the covariate shift and the well-balanced dataset, the L2 regularization shows a larger CE with an improved test MSE compared to the standard neural net training. I discuss the test MSE in more detail in the next section.

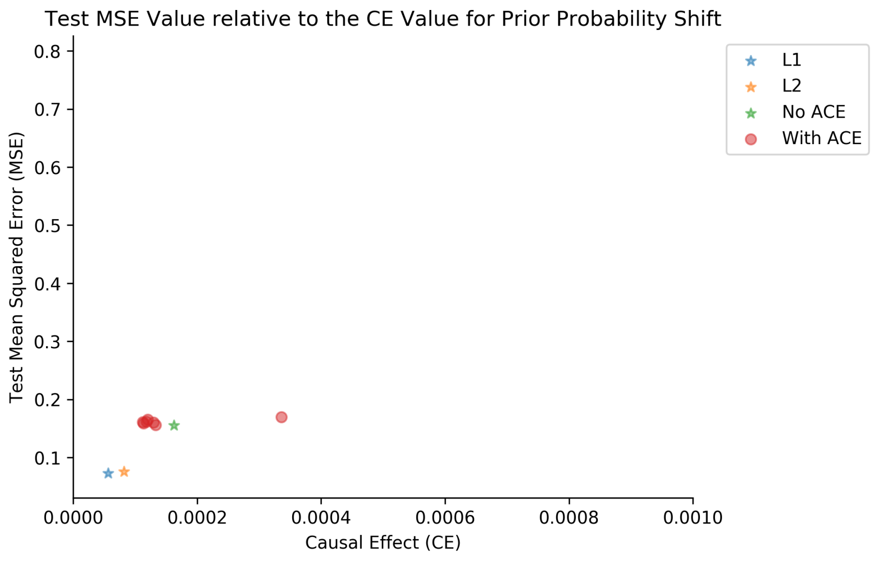

Figure 5.11: For datasets with a prior probability shift there is no correlation between an improved test MSE and a higher CE value

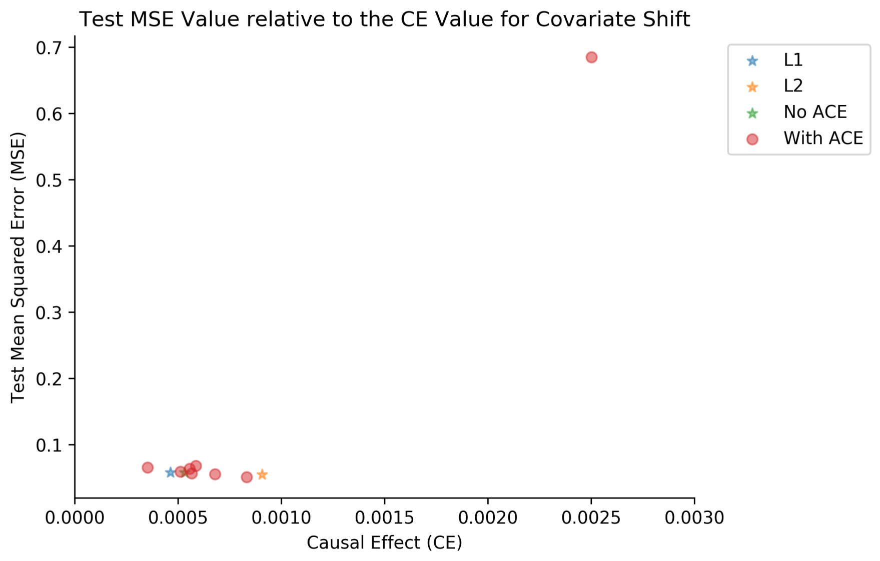

Figure 5.12: For datasets with a covariate shift there is an indication that an improved test MSE correlates with a higher ACE

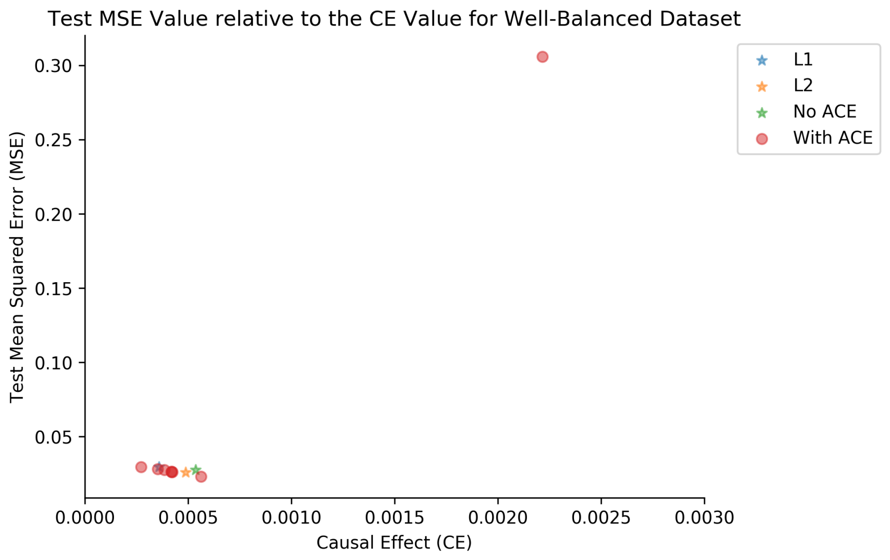

Figure 5.13: An improved test MSE for well-balanced datasets correlates with a higher CE value

5.2.3 The Relative Test Mean Squared Error

After analyzing the MSE in relation to the CE, I take a closer look at the MSE of each factor of the CE regularization. For this, I calculate the relative MSE for each dataset and each training method as

$$MSE_{relative} = \frac{MSE_{test} - MSE_{NoReg}}{MSE_{NoReg}} \tag{5.1}$$

where $MSE_{relative}$ is the relative MSE. For MSE_{test}, I take the mean of the test MSEs of each dataset. This is done for each training method involving a regularization. Furthermore, the variable MSE_{NoReg} is the mean of the test MSE of each dataset with no regularization in the model training. As the case with no regularization acts as a baseline, with Equation (5.1) one can calculate the relative improvement or deterioration of the test MSE MSE_{test} compared to the baseline. This way, I can compare the different MSEs with their different absolute values more accurately.

Similar to the MSE, the standard deviation will also be calculated relative to the baseline deviation. This means, the standard deviation for each training method becomes

$$STD_{relative} = \frac{STD_{test} - STD_{NoReg}}{STD_{NoReg}} \tag{5.2}$$

where $STD_{relative}$ is the relative standard deviation, STD_{test} is the standard deviation of the test MSEs of each dataset and each training method involving a regularization. Furthermore, STD_{NoReg} is the standard deviation of the test MSE of each dataset with no regularization in the model training. With this calculation, the deviation can be compared relative to the baseline and with each other.

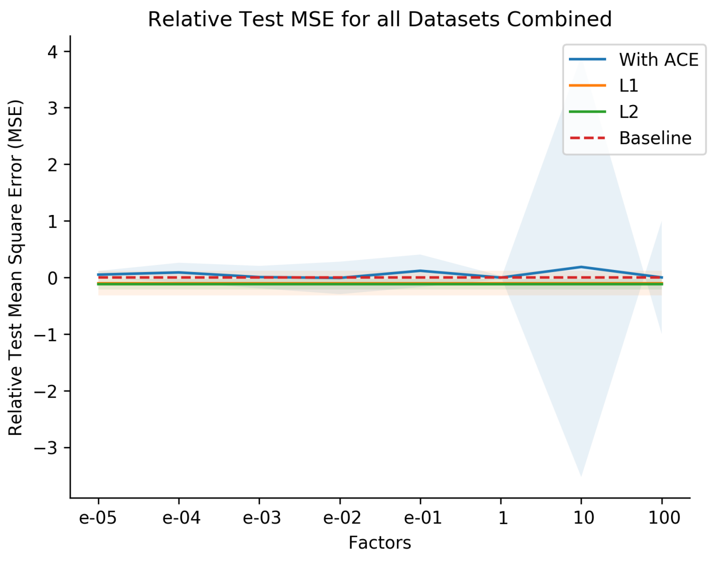

Figure 5.14: The relative test MSE of all datasets combined shows no improvement for the CE regularization

For Figure 5.14, I combine all datasets and calculate the test MSE for all training methods. As I calculate the relative MSE to the baseline, the baseline lies at0 and is indicated by the red dotted line. As one can see, the two regularization methods L1 and L2 have a slightly improved test MSE compared to the baseline. The CE regularization performs similarly to the baseline. There is a large standard deviation especially for the factor 10. To gain more insights, I again split up the datasets into three figures. One can see that, for the dataset with prior probability shift in Figure 5.15, the relative MSE of the CE, the L1, and the L2 regularization perform similarly to the baseline. For the datasets with covariate shift in Figure 5.16, one can see that the CE regularization with factor $e - 03$ and $e - 02$ results in an improved MSE compared to the baseline. In Figure 5.17, one can see that the CE regularization for the well-balanced dataset with the factor 10 generates an improved MSE. With the factors $e - 04$, $e - 01$, and 1 the mean also appears to be better than the baseline MSE. However, they all fall into the standard deviation (shadow with the dashed outline) of the baseline. This means, the improvement is not very significant. For the prior probability shift dataset, the standard deviation seems to be very high for factor 10. Furthermore, for the covariate shift dataset, the standard deviation seems to be high for all CE factors as well as the L1 regularization.

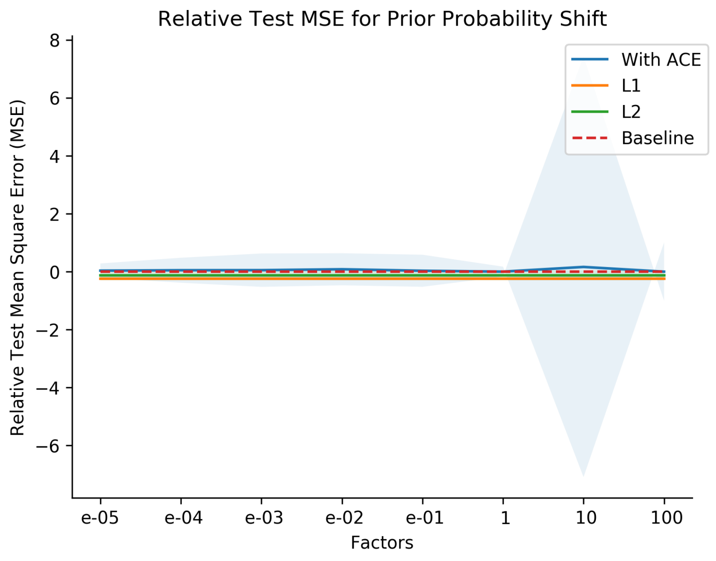

Figure 5.15: For the dataset with prior probability shift there is no improvement for the relative Test MSE of the CE regularization

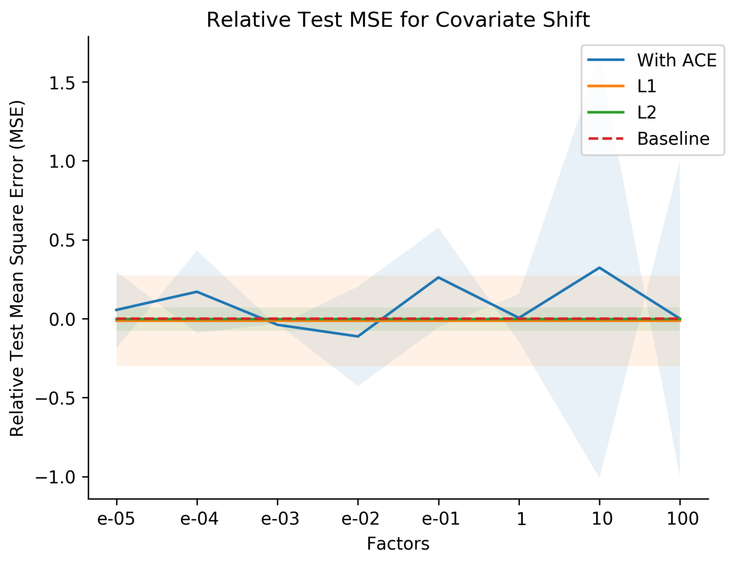

Figure 5.16: The factor $e - 03$ and $e - 02$ of the CE regularization can achieve a improvement of the relative test MSE for a datasets with a covariate shift

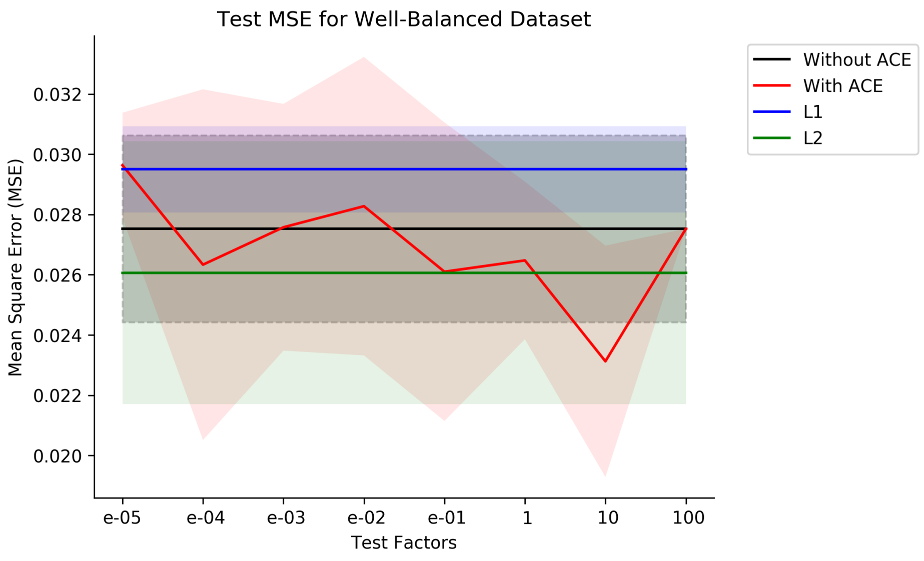

Figure 5.17: For well-balanced datasets the CE regularization with factor 10 can achieve a improved test MSE which is even lower than the test MSE of the L2 regularization

Taking a closer look at the three splits of the covariate shift, it shows that for the wide split (Figure 5.18) the CE regularization with the factors $e-03$, $e-02$, and 1 show an improved test MSE compared to the baseline and its standard deviation. Generally, the mean of the CE regularization lies below the mean of the training method without the regularization. However, except for factors $e-03$, $e-02$, and 1, they still lie within the standard deviation of the baseline. On the other hand, the L2 regularization shows an even larger improvement of the test MSE. For the skewed split, in Figure 5.19, the mean of the test MSE for the CE regularization also has a dip at $e-02$. However, all the values are within the standard deviation of the baseline. This means, I cannot conclude an improvement of the test MSE. However, the same can be concluded for the L2 as well as L1 regularization. Lastly in Figure 5.20, the CE regularization for narrow split results in no improvement at all for the MSE. Furthermore, the figure shows a very large deviation for the factor 10. However, the L1 and L2 regularization also perform worse compared to the baseline. In sum, the wide split that has a dataset shift in a larger number of important features has an improvement for the test MSE. Furthermore, the wide split also has a larger number of skewed datasets and a larger number of wider distributed datasets in the training set compared to the narrow split for example. The narrow and skewed split have the same number of important features with a dataset shift. However, the skewed split also has a larger number of skewed datasets and a larger number of wider distributed datasets in the training set compared to the narrow split.

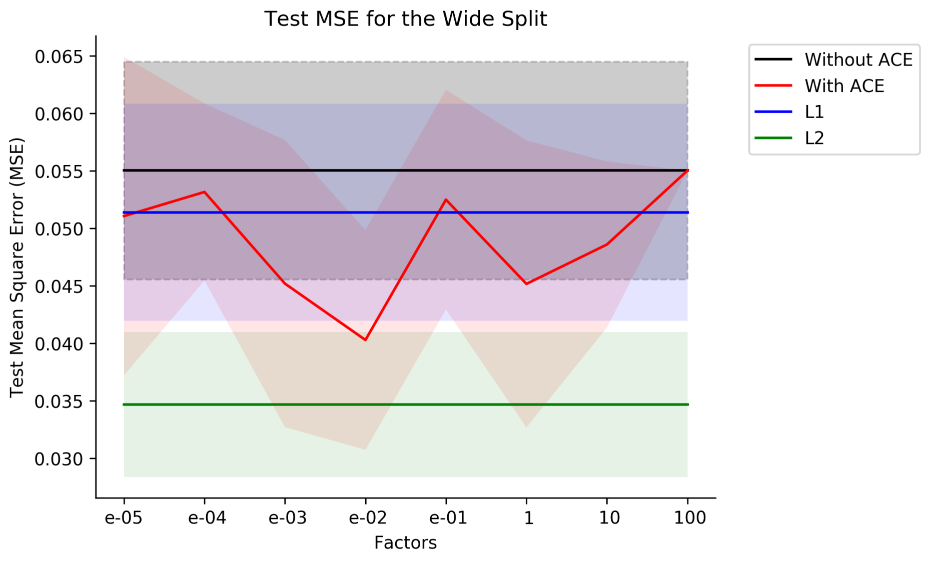

Figure 5.18: The covariate shift with wider distributions is able to improve the test MSE of the CE regularization with factor $e-03$, $e-02$, and 1

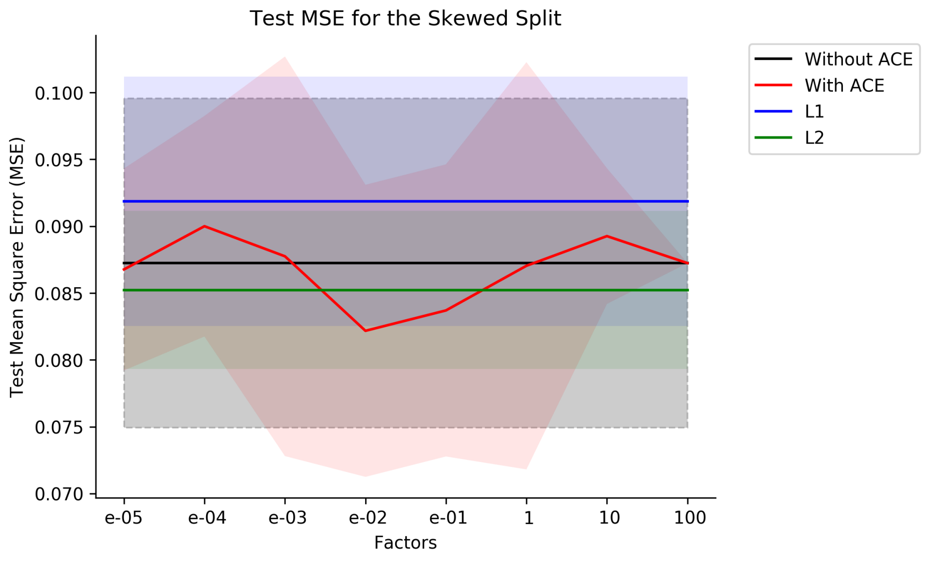

Figure 5.19: The test MSE of the CE regularization for the skewed split falls into the standard deviation of the baseline

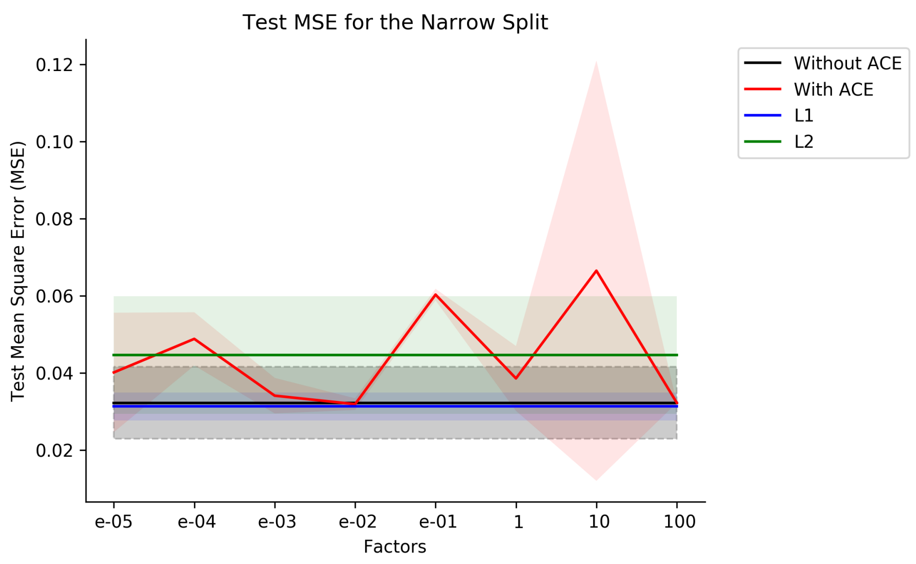

Figure 5.20: For the narrow split the test MSE of the CE regularization falls into the standard deviation of the baseline or becomes worse

Generally, looking at the CE value compared to the test MSE and the relative MSE value. One can conclude that the approach presented in this thesis is not able to generate an improved MSE for the prior probability shift case. Looking at the CE value for this case, one can also conclude that a higher CE value does not result in a higher MSE. A high CE is not the case for the L1 or L2 regularization either. This raises the question of whether the ACE is even suitable for that kind of dataset shift. One reason could be that the ACE intervenes on the neuron activations as well as inputs. Because of that the output variable, so the variable where I apply the dataset shift, is a fixed variable in this process. This can be explained with a simple example of causality. Imagine that the input variables of an algorithm are whether rain is falling and whether the wind is very strong. The output variable would be if people are wearing jackets. The ACE could determine if there is a causal effect between these two input variables and the output variable. Thus, the ACE intervenes on the input variables and activations to see whether the output is changing. Now, I introduce a prior probability shift. Thus, taking out parts of the output. This could mean that I remove all data points from the training set that include the variable that people are wearing a jacket. Thus, the training set now only includes data samples where people are not wearing a jacket. If the ACE intervenes now, the only output present is that people are not wearing jackets. Thus, the output is not changing which means the algorithm cannot form any causal relationships. Translating this to the model in this thesis, one can say that by applying a prior probability shift the causality is only calculated for this specific range in outputs. Hence, the causal connections for other outputs are not detected. This means, if one applies other house prices out of the range of the training data, the causal relationships formed during the training do not improve the calculation. For the covariate shift on the other hand an improvement of the test MSE could be detected for some splits. According to the section before, one could also see that with a growing CE, the test MSE could be improved up to a certain CE value. As I mentioned before, the CE calculates the causal effect of the inputs on the outputs. Thus, this time I apply the dataset shift to the intervened variables. Thinking back to the rain, wind, and jacket example. Now, with a covariate shift, I take out parts of the input variables. Thus, I remove the data points that include rain for example. However, as I intervene on the input variables the output is still able to change. Thus, causal relationships can still be formed without the removed data points. However, what can happen is that some causal relationships go missing, as the interventions depend on the input samples. Thus, if I remove crucial input samples the interventions will not be performed across the whole input space. That will result in fewer causal relationships formed. As the wide and skewed split have a training dataset where the shifted features are wider distributed than for the narrow split, they are able to generate more impactful interventions. With that, the training algorithm results in a better test MSE than the narrow split compared to the baseline. Additionally, as I have multiple inputs with a dataset shift, the impact on all causal relationships is limited. The reason for that is that the other inputs are still there to form causal relationships. For the prior probability shift case, the severity of the shift has a larger impact on how the CE regularization performs. Besides that, only having one input amplifies the effect a sever prior probability shift has on the performance. As the ACE uses that single output to measure causality, a dataset shift in that output sabotages all causal relationships within the net. With the rain, wind, and jacket explanation, it also makes sense that the well-balanced dataset shows an improvement with the CE regularization. For this case, the result of the CE regularization performs better even compared to the L2 regularization for factor 10. As there is no dataset shift here, the interventions can cover the whole realm of input and output data. This means, a maximum of causal relationships can be formed. In sum, the nature of the ACE could cause that the CE regularization is unable to deal with large prior probability shifts for models with one output. Furthermore, as the interventions of the CE regularization depend on the input samples of the training, the covariate shift can also pose problems for this approach. In the following section, I analyze the training MSE and set it into context with the explanation above.

5.2.4 Training Error Developement

In this section, I analyze the ACE and test MSE, I discuss the training MSE and its progress over the training epochs. Here, I calculate the mean of all training MSEs of all datasets for each training method. This is then observed for each training epoch.

Figure 5.21 displays the training MSE for all datasets combined. Here, the training MSE decreases until a certain value for all factors of the CE regularization. However, for the factor 100, the training MSE starts increasing again. This is because, in the beginning, the CE value is very small. Thus, the MSE overpowers the CE within the loss function. As mentioned, when the MSE is decreased enough, the CE can weigh in within the loss function and can drive up the MSE value. It is also logical that this effect appears for the factor 100, as it gives the CE more influence in the loss function. Furthermore, especially for factor 100, the training MSE shows a large standard deviation. All the other factors as well as no regularization produce a similarly small training MSE. On the other hand, the training MSE of the L1, as well as L2 regularization, are not decreasing as in the other cases. Looking at the datasets separately, one can see that the prior probability shift cases, in Figure 5.22, show a training MSE for the L1 and L2 regularization which is not decreasing accordingly. However, as shown in the previous section the test MSE is still improved compared to the baseline. This indicates that the training MSE of the value produced by the L1 as well as L2 regularization is desirable. The cases of covariate shift and the well-balanced dataset, in Figure 5.23 and Figure 5.24, do not show any special behavior in their training MSE.

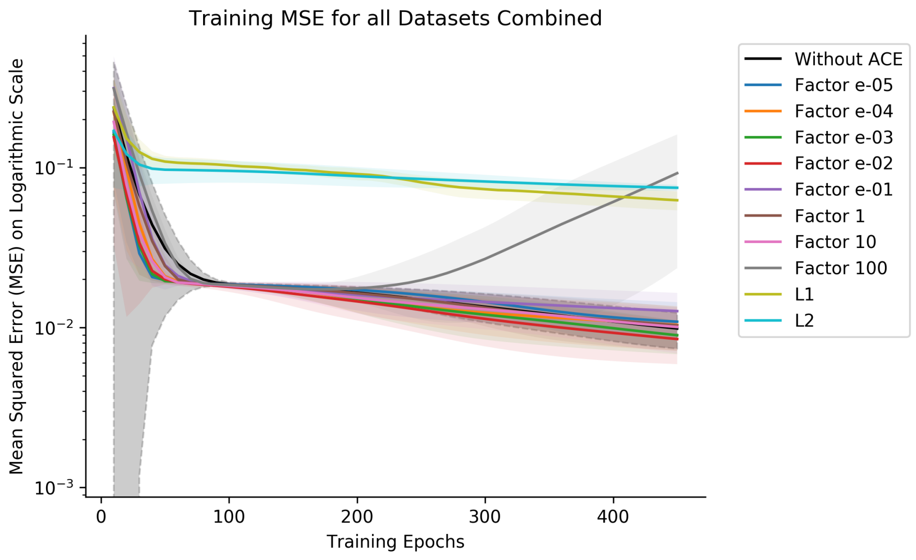

Figure 5.21: The training MSE for all datasets combined shows that the CE regularization can overpower the MSE for factor 100

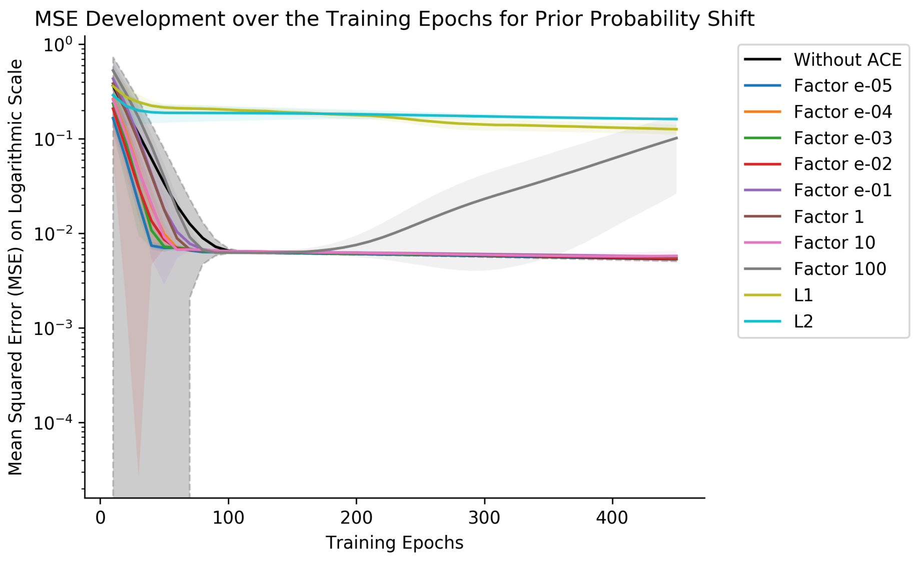

Figure 5.22: The L1 and L2 regularization for datasets with a prior probability shift produce a much higher training MSE

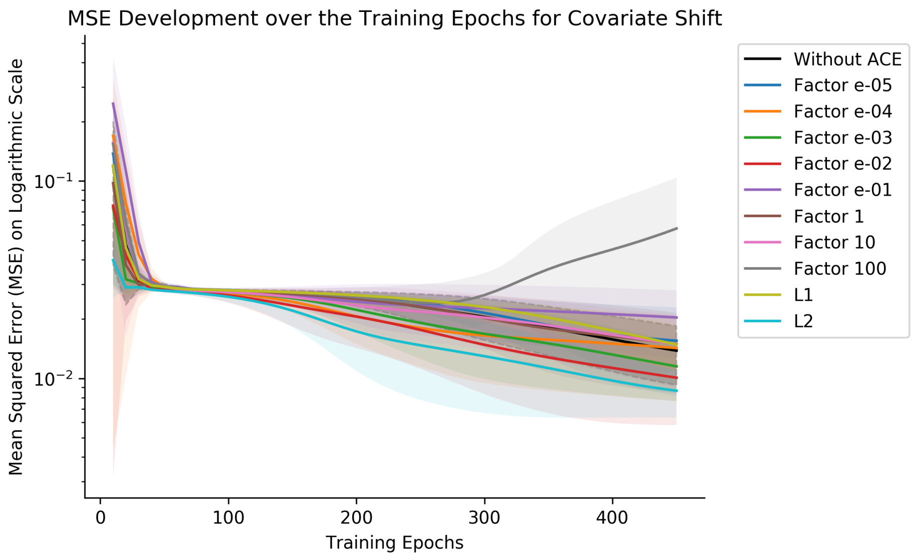

Figure 5.23: For datasets with a covariate shift all the regularization techniques produce a decreasing training MSE except for the CE regularization with factor 100

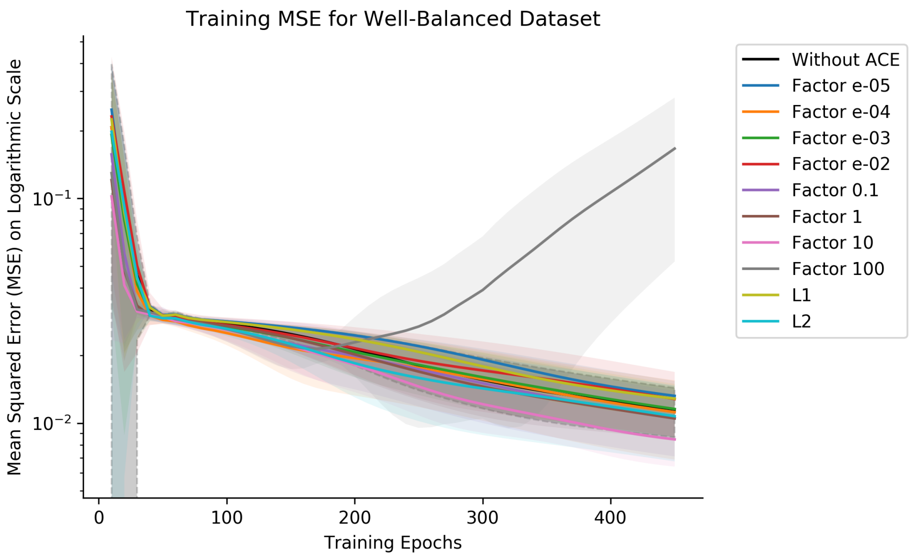

Figure 5.24: For a well-balanced dataset the CE regularization with factor 100 can overpower the training MSE

5.2.5 Summary and Discussion of Results

As explained before, the approach, developed in this thesis with the chosen parameters, is not stable for prior probability and covariate shifts. This is because the interventions are made over the realm of the training input samples. For the prior probability case, this means that the training algorithm does not form causal relationships for certain outputs. Furthermore, for the covariate shift case, some interventions might be missing resulting in the training algorithm missing some causal connections. This is also indicated by the fact that, compared to the other datasets, the well-balanced dataset could be improved the best according to its test MSE. Another indication is that for the measured CE value one cannot see a correlation between increased CE value and improved test MSE for the prior probability shift case. Another important point is the large standard deviation in both the training MSE and the CE value. As mentioned before, interventions rely on neuron activations. The neuron activations, on the other hand, rely on the seed set at the beginning of the training. So depending on what kind of seed is set, the CE can increase faster if the seed is set in a way that favors causality. Or the CE increases slower if the seed is set less favorably. With the speed of the CE increasing also the training MSE is impacted. If the CE is increasing fast, the CE is able to weigh in heavier in the loss function. If the CE is increasing slower, its impact will be less. This means, the training MSE can weigh in more or less depending on the seed resulting in a large standard deviation. Another reason could be that I incorporate the absolute value of the median of the ACEs into the CE. Thus, at the beginning of each training epoch, the regularization determines whether the ACE is positive or negative for each neuron. This value is then fed into the loss function as an absolute value and the training algorithm updates the weights accordingly. There is a great chance that this first calculation of the ACE is not always accurate. This means, that some neurons might be forced into a negative ACE by the regularization although they should be positive. Especially at the beginning of the training when the ACE values are close to zero, this is very likely to happen. Thus, it cannot be assured that the regularization term determines the right sign for the ACE for each neuron. Hence, it can happen that for one training the regularization is right for all neurons, and for another training, it picks a lot of wrong ACEs. Depending on that the training, as well as test MSE, can be influenced positively or negatively. This could result in a large standard deviation for the CE value and the training MSE.

6 Conclusion

In this work, I aim to answer the questions whether a causal interpretability mechanism, incorporated into the model training, will enhance the causality of a neural net and whether a neural net with a higher causal interpretability mechanism has a better generalization ability. In order to answer these questions, I incorporate the Negative Causal Effect (NCE) into the model training as a regualrization term. The NCE is minimized during the training together with the loss function. Through that, the weights and biases should become more causal throughout the training. In order to construct the NCE, the Average Causal Effect (ACE) is used. The ACE is calculated for every neuron in the neural net by slicing the neural net and intervening on every neuron. The interventions are uniformly distributed over the realm of the training data. In order to assess the generalization ability, I perform experiments using a well balanced dataset, three datastets with a covariate shift and three datasets with a prior probability shift. For all datasets, I run a standard training, a training with L1 as well as L2 regularization, and multiple trainings using the CE regularization with regularization factors from $e-05$ to 100. For all CE regularizations, especially for regularization factors above 10 the CE value of the neural net could be enhanced. Janzing found, in [18], that L1 as well as L2 regularization can enhance the causality of the model. The experiments also demonstarted an enhancement of the causality of the neural net for the L1 as well as L2 regularization. Furthermore, I found that the way the dataset shift is distributed has an impact. As the interventions are connected to the training data a sever shift means that the interventions can only cover a small realm of the dataset. With the interventions not covering all input values, not all causal connections can be formed. Moreover, I suspect that the initialized weights and biases influence the growth of the CE value. For the prior probability shift, the output variable cannot change as much with the intervention. So, not all causal connections could be formed. Additionally, the absolute value within the CE regularization can determine the wrong sign for the ACE. Thus, the ACE of a neuron can be optimized for the wrong sign. This also influences the test MSE and the growth of the CE value. For future work, besides using this approach on a larger training samples sizes as well as on other models and datasets, the selection of the right interventions could be explored further. Lastly, the determining of the sign of the ACE for each neuron could be optimized as well.

Bibliography

[1] S. Ö. Arik and T. Pfister. “TabNet: Attentive Interpretable Tabular Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021).

[2] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10 (2015).

[3] M. T. Bahadori, K. Chalupka, E. Choi, R. Chen, W. Stewart, and J. Sun. “Causal Regularization”. In: *Computing Research Repository* abs/1702.02604 (2017).

[4] O. Boz. “Extracting Decision Trees from Trained Neural Networks”. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2002, pp. 456–461.

[5] O. Campesato. *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning & Information, 2020.

[6] G. C. Cawley. “Causal & Non-Causal Feature Selection for Ridge Regression”. In: *Workshop on the Causation and Prediction Challenge at WCCI 2008*. PMLR, 2008, pp. 107–128.

[7] A. Chattopadhyay, P. Manupriya, A. Sarkar, and V. N. Balasubramanian. “Neural Network Attributions: A Causal Perspective”. In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 981–990.

[8] A. Datta, S. Sen, and Y. Zick. “Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems”. In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2016, pp. 598–617.

[9] A. P. Dawid. “Conditional Independence in Statistical Theory”. In: *Journal of the Royal Statistical Society* 41 (1979).

[10] J. Dockès, G. Varoquaux, and J.-B. Poline. “Preventing dataset shift from breaking machine-learning biomarkers”. In: *GigaScience* 10 (2021).

[11] D. Geiger, T. Verma, and J. Pearl. “Identifying independence in bayesian networks”. In: *Networks* 20 (1990).

[12] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. “Explaining Explanations: An Overview of Interpretability of Machine Learning”. In: *5th IEEE International Conference on Data Science and Advanced Analytics DSAA*. IEEE Computer Society, 2018, pp. 80–89.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[14] B. Goodman and S. Flaxman. “EU regulations on algorithmic decision-making and a "right to explanation"”. In: *AI Magazine* 38 (2016).

[15] Y. Goyal, U. Shalit, and B. Kim. “Explaining Classifiers with Causal Concept Effect (CaCE)”. In: *Computing Research Repository* abs/1907.07165 (2019).

[16] M. Harradon, J. Druce, and B. E. Ruttenberg. “Causal Learning and Explanation of Deep Neural Networks via Autoencoded Activations”. In: *Computing Research Repository* abs/1802.00541 (2018).

[17] D. Harrison and D. L. Rubinfeld. “Hedonic Housing Prices and the Demand for Clean Air”. In: *Journal of Environmental Economics and Management* 5 (1978).

[18] D. Janzing. “Causal Regularization”. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, pp. 1401–1412.

[19] H. Kiiveri, T. P. Speed, and J. B. Carlin. “Recursive causal models”. In: *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics* 36 (1984).

[20] B. Kim, J. A. Shah, and F. Doshi-Velez. “Mind the Gap: A Generative Approach to Interpretable Feature Selection and Extraction”. In: *Advances in Neural Information Processing Systems*. Curran Associates, 2015, pp. 2260–2268.

[21] B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. B. Viégas, and R. Sayres. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)”. In: *ICML*. PMLR, 2018, pp. 2673–2682.

[22] J. Kim and J. F. Canny. “Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2961–2969.

[23] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. “The (Un)reliability of Saliency Methods”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham: Springer International Publishing, 2019, pp. 267–280.

[24] M. Kocaoglu, C. Snyder, A. G. Dimakis, and S. Vishwanath. “CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training”. In: *International Conference on Learning Representations*. ICLR, 2018.

[25] P. W. Koh and P. Liang. “Understanding Black-box Predictions via Influence Functions”. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894.

[26] M. Kull and P. A. Flach. “Patterns of dataset shift”. In: *First International Workshop on Learning over Multiple Contexts (LMCE)*. ECML-PKDD, 2014.

[27] T. Kyono, Y. Zhang, and M. van der Schaar. “CASTLE: Regularization via Auxiliary Causal Graph Discovery”. In: *Advances in Neural Information Processing Systems*. Curran Associates, 2020, pp. 1501–1512.

[28] T. Le, S. Wang, and D. Lee. “GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model’s Prediction”. In: *2020 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 26 (2020).

[29] R. Moraffah, M. Karami, R. Guo, A. Raglin, and H. Liu. “Causal Interpretability for Machine Learning - Problems, Methods and Evaluation”. In: *SIGKDD Explorations Newsletter* 22 (2020).

[30] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[31] J. Pearl. “Causal inference in statistics: An overview”. In: *Statistics Surveys* 3 (2009).

[32] J. Pearl. *Causality*. Cambridge University Press, 2009.

[33] J. Pearl. “The Do-Calculus Revisited”. In: *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012, pp. 3–11.

[34] J. Pearl. “The seven tools of causal inference, with reflections on machine learning”. In: *Communications of the ACM* 62 (2019).

[35] J. Peters, P. Bühlmann, and N. Meinshausen. “Causal inference by using invariant prediction: identification and confidence intervals”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78 (2016).

[36] J. Peters, D. Janzing, and B. Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[37] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[38] M. Ribeiro, S. Singh, and C. Guestrin. ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2016, pp. 97–101.

[39] J. Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015).

[40] P. Schwab and W. Karlen. “CXPlain: Causal Explanations for Model Interpretation under Uncertainty”. In: *Advances in Neural Information Processing Systems*. Curran Associates, 2019, pp. 10220–10230.

[41] S. Sharma, J. Henderson, and J. Ghosh. “Certifai: A Common Framework to Provide Explanations and Analyse the Fairness and Robustness of Black-Box Models”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. SIGAI, 2020, pp. 166–172.

[42] A. Shrikumar, P. Greenside, and A. Kundaje. “Learning Important Features Through Propagating Activation Differences”. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, pp. 3145–3153.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014).

[44] M. Sundararajan, A. Taly, and Q. Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. PLMR, 2017, pp. 3319–3328.

[45] M. Sundararajan, A. Taly, and Q. Yan. “Gradients of Counterfactuals”. In: *Computing Research Repository* abs/1611.02639 (2016).

[46] M. T. Wojnowicz, B. Cruz, X. Zhao, B. Wallace, M. Wolff, J. Luan, and C. Crable. ““Influence sketching”: Finding influential samples in large-scale regressions”. In: *2016 IEEE International Conference on Big Data* (2016).

[47] J. Woodward. *Making Things Happen: A Theory of Causal Explanation*. Oxford University Press, 2003.

[48] G. Xu, T. D. Duong, Q. Li, S. Liu, and X. Wang. “Causality Learning: A New Perspective for Interpretable Machine Learning”. In: *IEEE Intelligent Informatics Bulletin* 20 (2020).

[49] J. Zhang and E. Bareinboim. “Fairness in Decision-Making - The Causal Explanation Formula”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 2037–2045.