On the Study of Data processed in Autoencoders

Maria Walch

December 2021

Contents

1 Motivation **5**

2 Neuronal Networks and Autoencoders **9**
2.1 Neuronal Networks . . . 9
2.1.1 Constituents of a Neuronal Network . . . 10
2.1.2 Calculations within a Neuronal Network . . . 11
2.1.3 Hyperparameters . . . 14
2.1.4 Possible Pitfalls . . . 14
2.2 Autoencoders . . . 15
2.3 Categorical Perspectives on Neuronal Nets . . . 16
2.3.1 Spivaks Compositional Perspective . . . 16
2.3.2 Related Work . . . 18
2.3.2.1 Connection to the **Lens** Category . . . 18
2.3.2.2 Consequences . . . 19
2.3.3 Categorical Probability . . . 19
2.4 Autoencoders and Data Structure . . . 22
2.4.1 The Input Data Space . . . 22
2.4.2 Structure within the (Transformed) Data . . . 23
2.4.3 The First Level - I. . . . 24
2.4.3.1 Usage of Persistent Homology on Abstraction Level I . . . 24
2.4.4 The Second Level - II. . . . 25
2.4.4.1 Usage of Persistent Homology on Abstraction Level II . . . 26
2.4.5 The Third Level - III. . . . 26
2.4.5.1 Usage of Persistent Homology on Abstraction Level III . . . 26
2.5 Complexification . . . 28
2.5.1 Simplicial Complexes . . . 28
2.5.1.1 Abstract Simplicial Complexes . . . 28
2.5.1.2 Geometric Simplicial Complexes . . . 29
2.5.2 Covers . . . 29
2.5.2.1 Clustering Functor . . . 30
2.5.3 Fuzzy Simplicial Complexes . . . 30
2.5.4 The Spivak-McInnes Adjoint Pair . . . 30
2.5.5 Kleinbergs Impossibility Theorem and Hierarchical Clustering . . . 31
2.5.6 Maximal and Single Linkage . . . 31
2.5.7 Complexification/Triangulation Algorithms . . . 32
2.5.7.1 Data within the Layers of a Neuronal Network . . . 32
2.5.7.2 Nerve . . . 33
2.5.7.3 Vietoris Rips Complex . . . 34
2.5.7.4 Čech Complex . . . 34
2.5.7.5 Weak and Strong Delauny Complexes . . . 35
2.5.7.6 Witness Complexes . . . 36
2.5.7.7 α-Complexes . . . 36

2.5.7.8 Tangential Delaunay Complexes . . . 37
2.5.8 The Codomain Category of $\mathcal{F}_{AE,\text{struct}}$ . . . 37
2.5.8.1 Topological Invariants . . . 38
2.5.8.2 Homotopy . . . 38
2.5.8.3 Simplicial Sets . . . 39

3 Data Topoi **43**

4 Homotopical and Homological Investigations **45**

5 A Categorical Notion of Stochastic Optimization **47**

6 Stochastic Optimization within Autoencoders **49**

A Neuronal Networks **51**
A.1 Quivers . . . 51
A.1.1 Thin Quiver Representation . . . 51
A.1.2 Stratified Graph and Layers . . . 51
A.2 Hyperparameters . . . 51
A.2.1 Activation Functions . . . 51
A.2.2 Dropout . . . 52
A.2.3 Optimization Function . . . 52
A.2.4 Learning Rate . . . 54
A.2.5 Batch Size . . . 54

B About Kan Complexes and Grothendiecks Homotopy Hypothesis **55**
B.1 Model Categories . . . 56
B.1.1 Top . . . 57
B.1.2 sSet . . . 57
B.2 Kan Complexes . . . 58
B.3 ∞-Categories . . . 59
B.4 $(\infty,1)$-Categories and Fundamental Groupoids . . . 60
B.5 Getting $(\infty,1)$-Categories from Model Categories . . . 60
B.6 Grothendiecks Homotopy Hypothesis . . . 61

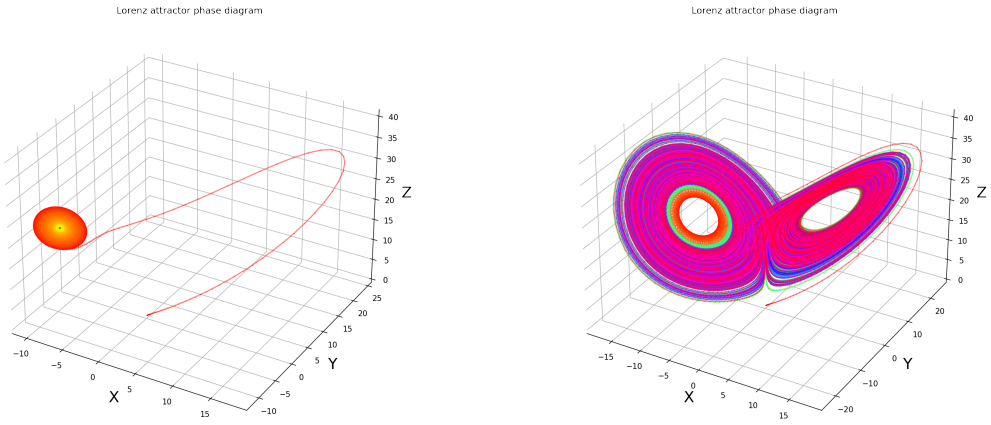
1

Motivation

> ***Model:*** *a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs*
>
> – Merriam-Webster Dictionary

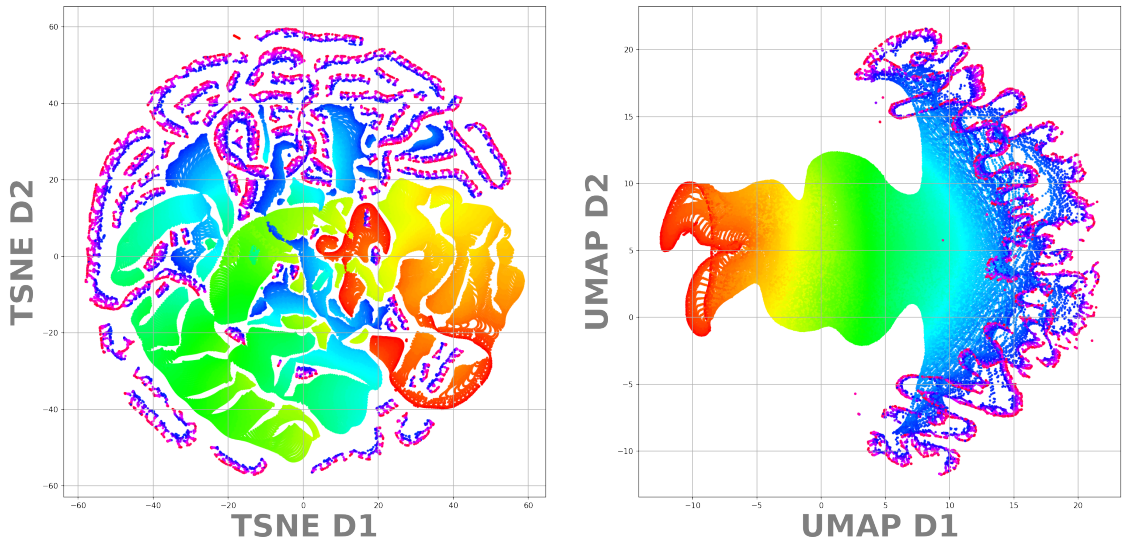
Physics has always been about model building; since Karl Popper, a model is accepted as long as there is no better one to describe the *observations*. The scientific modus operandi of which Karl Poppers "Logik der Forschung" - *the logic of research* laid the foundations was called by him "die deduktive Methodik der Nachprüfung" - *the deductive methodology of falsification*. This demands highest standards not only from the model itself but also from its syllogisms and the 'language' in which these are formulated. Scientific claims need not only to be testable but falsifiable. Once these criteria are accomplished, they make scientific reasoning more pure and exact and they guarantee that the research is as close as it can be to a true knowledge gain - "dem Erkenntnisgewinn". Over the last century, the process of building models from observations has changed fundamentally. While it was difficult to construct precise measuring devices and collect and store data before the development of powerful computers, the problems are nowadays mirrored. Data is available in vast amounts not only for the physical ground work experiments, of which the CERN might be called the most significant one; but also for more basic technical processes of daily life. This comprises medicine, finances, automotive engineering, material sciences, the social sphere and will also cause many legal changes within the basic structure of our society.
Regarding the technical aspect, the vast amount of data poses sensitive questions to the researcher who wants to build a model upon it: "Which parts of the data are relevant?" "How can these parts be interpreted?" "Which changes to the data do alter their interpretability?"
Presently, many of these questions are worked on with **neuronal networks**, which are - regarding the flux of development funds - definitely one of the hottest topics of the 21st century. In our work, we study one particular kind of neuronal networks - **autoencoders**. Their descriptive architecture consists of a *bottleneck* through which the input is forced resulting in a lower dimensional *representation*, that consecutively is used as a basis to reconstruct the original full size input as close as possible. In figure 1.1 two Lorenz systems with numerically close ρ values are depicted. The corresponding two dimensional embeddings are shown in figure 1.2 and figure 1.3. These should give the reader an intuition about the difficulty to (visually) interpret this representation and the consequent need to find abstract tools to validate it, although it is evident that the latent representation comprises *some* information about the initial data. As for the present example, one can identify the quasi-periodic structure also as the reed structure within the UMAP embedding in figure 1.3b. But how about the collapsing case in figure 1.1a? For this toy example we exactly know the dynamics of the input data space, the Lorenz system is a well studied one. However, ideally (with respect to the conceptualisation of the autoencoder), we would know little or nothing about it.

Autoencoders are used in the expectation, that 'minimal information is lost through the bottleneck', that they, like a zip-file, *represent* the *most relevant* parts of the input that are needed



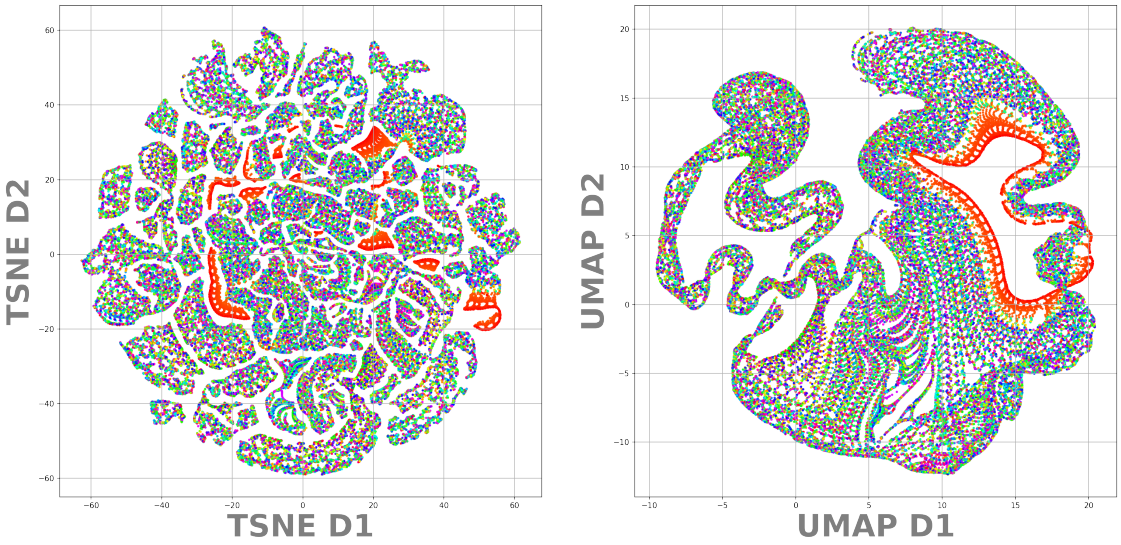
(a) $\rho = 24$ (b) $\rho = 24.1$

Figure 1.1: Three dimensional plot of the Lorenz system for two numerically close ρ values. Figure created by Amala Paulson



(a) TSNE Embedding (b) UMAP Embedding

Figure 1.2: Two-dimensional embeddings of an 8 dimensional latent space of a vanilla autoencoder whose input data was the Lorenz system with $\rho = 24$. Figure created by Amala Paulson



(a) TSNE Embedding (b) UMAP Embedding

Figure 1.3: Two-dimensional embeddings of an 8 dimensional latent space of a vanilla autoencoder whose input data was the Lorenz system with $\rho = 24.1$. Figure created by Amala Paulson

to reconstruct it [67]. Based on this expectation, their (ongrowing) applications are nowadays widespread. To give a glimpse on current applications, they range from semivisible JET detection [72], automated anomaly detection for new physics searches at the LHC [73], the investigation of neutron scatter events in condensed matter physics [84], breast cancer subtype classification [83], automated exoplanet transit detection [81] to anomaly detection in smart farming concepts [66]. From the above variety of data that autoencoders are applied on and the respective bundle of structures that are hoped to be identified within it, the reader might have got the intuition that autoencoders are used as **automated model builders**. In respect of Karl Popper and the constructivist epistemology we ask ourselves the question "*Is this justified?*" In deep belief of its intrinsic beauty and methodical elegance we approach this question within the field of abstract mathematics. In the following, we present our problem setting and the methods we are using to work on it in an inductive deductionistic way. I am formalizing the visual analysis of dynamic systems performed with Amala and introducing neural persistence into the analysis to overcome Michael Joswigs concern with the wrongly represented high dimensional knots within the 2D projection.

2

Neuronal Networks and Autoencoders

2.1 Neuronal Networks

Autoencoders are a subspecies of *neuronal* or equivalently: *neural networks*. Most abstractly, neuronal networks are directed acyclic graphs (dags). The structure of neuronal networks interpreted as abstract graphs can be described as a special type of quivers (see appendix A.1), so called *network quivers*.

Definition 2.1.1 (Network Quiver)**.** A *network quiver* is a quiver *arranged by layers* that fulfills two additional conditions [68]:

1. There are no loops on source or sink vertices.
2. Each hidden vertex is equipped with exactly one loop.

Source vertices in neuronal networks are the *input* (definition 2.1.8) and *bias* (definition 2.1.7) vertices. Sink vertices are most commonly the output vertices. If too many hidden vertices are sinks, the neuronal network is not able to learn any more. These conceptualities are schematically shown in figure 2.1.

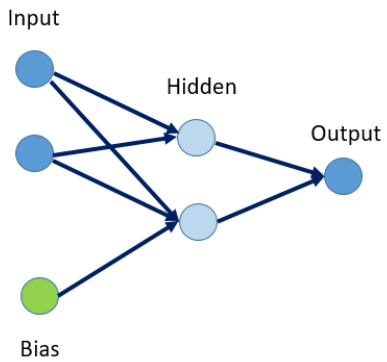


Figure 2.1: Source, sinks and hidden vertices.

The requirement for the network quiver to be arranged by layers is imposed by the *information flux* within a (deep) neural network. A neural network is called deep if it has more than three layers, or, equivalently, more than one hidden layer. A layer is defined as a columnwise arrangement of vertices $\mathcal{V}$, where there are no oriented edges from vertices on the right to vertices on the left.

For *recurrent neural networks* and other structures with temporal dependencies [63], this applies with some structural modifications as well.

2.1.1 Constituents of a Neuronal Network

The vertices $\mathcal{V}$ of a network quiver are its calculation units - the *neurons*.

Definition 2.1.2 (Neuron)**.** A neuron, "node" or "unit" is a quadruple $(\mathbf{x}, \mathbf{W}, f, \mathbf{y})$ consisting of an input tensor $\mathbf{x}$ with entries x, a (*weights*) tensor $\mathbf{W}$ with entries w, and a nonlinear *activation function* f that defines an output tensor $\mathbf{y}$ via

$$\mathbf{y} = f(\mathbf{W}^T\mathbf{x}), \tag{2.1}$$

where the activation function f is applied element-wise on the product $\mathbf{W}^T\mathbf{x}$.

For now, the tensors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{W}$ as well as the activation function f are just abstract quantities needed for the definition of a neuron as a calculation entity. Their meaning and interpretation with respect to concrete neuronal networks will be explained below. Also, it is shown in figure 2.3, that equation (2.1) is modified in practice to include an additive *bias term*.
A *forward pass* within a neuronal network is the - layerwise - activation of *all* [1] its neurons. As the structure of a neuronal network - the quiver - without its action on input data is just an abstract graph, the definition of neuronal networks cannot be thought without the notion of *quiver representations*.

Definition 2.1.3 (Quiver Representation)**.** A representation for a finite quiver Q has the following two components [13] [68]:

1. To each vertex $v \in \mathcal{V}$ there is associated a K-vector space W_v.
2. To each edge $e \in \mathcal{E}$ between two vertices v and v' there is associated a K-linear map $W_e : W_v \longrightarrow W_{v'}$.

The entity over all edges, i.e. the second component within the above definition 2.1.3, can be associated to the weights tensor $\mathbf{W}$ in definition 2.1.2. To use the above definition for neuronal networks, one has to include requirements to assure the desired information flux in a neuronal network. These are elaborated in appendix A.1.

Definition 2.1.4 (Neuronal Network)**.** A **neuronal network** over a network quiver Q is a pair (W, f); consisting of a *thin quiver representation* W and an activation function f [68].

In the above definition, a thin quiver representation just means that the quiver representation W takes values in $\mathbb{C}$ for each vertex (definition A.1.1). Furthermore, the network quiver representation is considered without loops in definition 2.1.4, to make sure that it reduces to the activation state before the forward pass. The forward pass is constituted by the one-directed, successive, layerwise activation of the weights. Neuronal networks including only forward passes are called *feedforward neura networks*. They are dags. In the following, they are our main subject of interest. Counter-examples are given by neuronal networks processing temporal dependencies, such as recurrent ones.

Definition 2.1.5 (Weights)**.** The **weights** of a neuronal network, $\mathbf{W}$ are given by the entity of complex numbers $W_{\mathcal{E}}$ defining the inter vertex-maps associated to an edge $e \in \mathcal{E}$ [68].

The weights are the true power of a neuronal network, their adjustment is what constitutes its *training* (definition 2.1.13); they raise an assembly of units to a network applicable to complex tasks such as speech and image recognition, time series segmentation, data augmentation and generation and ever increasing possibilities of use. A unit i in a layer k is connected to the unit j in layer $k+1$ via the weight θ_{ij}. The input x to the unit i is multiplied with θ_{ij} in order to obtain the input to unit j in layer $k+1$. To obtain the output of unit j in layer $k+1$ and thus the input to a unit r in layer $k+1$, one needs to apply the activation function on it.

[1] Sometimes it is favorable to not use all neurons for computational reasons - *dropout*.

Definition 2.1.6 (Activation Function)**.** An activation function f is a complex nonlinear function $f : \mathbb{C} \to \mathbb{C}$ in one variable, that is differentiable except in a set of measure zero [68].

The activation of a single node is illustrated in figure 2.2.

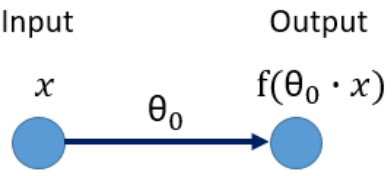


Figure 2.2: The activation of one neuron with activation function f and input x.

Some examples of commonly used activation functions can be found A.2.1. Many activation functions hit the $(0, 0)$ point in the two-dimensional coordinate system. To prevent that too many units in layer $k+1$ give back output zero and as a consequence there is no input to the subsequent layer and thus no information flow, one introduces *bias terms*.

Definition 2.1.7 (Bias Term)**.** A bias term is a constant value added to the input of a node j in a layer $k + 1$, as illustrated in figure 2.3.

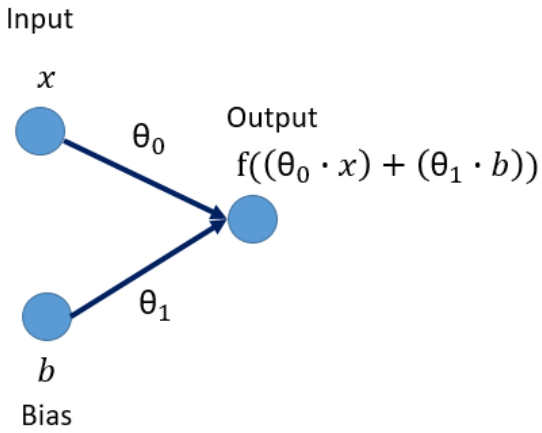


Figure 2.3: The activation of one neuron with activation function f, input x and bias b.

Figure 2.3 also consitutes a simplified illustration of the workwise of every neuron within a feedforward neural network.

2.1.2 Calculations within a Neuronal Network

Having the basic constituents of neuronal networks at hand, the question arises on how to actually use them. In deep learning, it's all about the data. The demand on neuronal networks is to transform the *input* in a way such that the resulting *output* contains (or better: *represents*) more information. Hereby, the terminus information has to be defined properly with respect to the task one intends to use the neuronal network for. Roughly speaking, one can discriminate between three classes of tasks [86]:

- **Classification and Regression**: Let (A, B) be a set of *input* and *output* samples and $f : A \to B$ a learning function. One speaks of classification when B is a finite set and regression when B is real-valued.

- **Generative Models**: These are constructed in order to be able to learn the distribution of a dataset. The task is to generate new samples close to a given dataset.
- **Reinforcement Learning**: The problems are formulated in a way such that an "agent" learns by feedback dranw from an "environment".

In the autoencoder context, the output shall be equivalent to the input. It could be described as an auto-regressor. What really interests us there is the intermediate representation of the data. However, before we turn to the specialities of autoencoders, we give the basic constituents of a feedforward (classifier or regressor) network. Of course, the fun begins when the above classes of networks interbreed, but this we save ourselves for later. [2]

Definition 2.1.8 (Input)**.** The input to a neuronal network is the vector of input tensors $(\mathbf{x}_1, ..., \mathbf{x}_d)^T$ that is fed into the neurons of the first layer. The input space is called $\mathcal{I}$.

An illustration of definition 2.1.8 is given on the left side of figure 2.4. The tensors $\{\mathbf{x}_i\}$, $i \in \{1, ..., d\}$ can have various contents. Prominent examples are the rgb pixel values of an image, any kind of scientific measurement data, time series, audio signals or even video sequences. If the input exhibits a histographical ordering, it is called *labelled*. The label l_i denotes the affiliation to a histogram or "class" within the input, i.e. the labeld input tensor consists of pairs $\mathbf{x}_{i,l} = \{\mathbf{x}_i, l_i\}$.

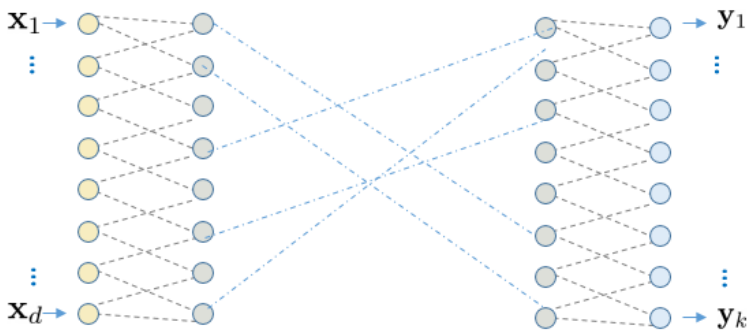


Figure 2.4: **Left:** The input to a neuronal network with sample connections to the subsequent layer. **Right:** The output of a neuronal network with sample connections from the preceeding layer and intermediate sample connections

Definition 2.1.9 (Output)**.** The output of a neuronal network is the vector of output tensors $(\mathbf{x}_1, ..., \mathbf{x}_k)^T$ that is given back by the neurons of the last layer. The space of all outputs is denoted by $\mathcal{O}$.

An illustration of definition 2.1.9 is given on the right side of figure 2.4. The output of the neuronal network depends on the task that was assigned to it. If it had to deal with a binary classification task, it is 0 or 1, depending on whether a given input belongs to a given class or not and accordingly for more complex classification tasks. On the contrary, regression-like tasks require the output to be a vector of the same structure as the input vector. For autoencoders, we have the special case that the output vector has exactly the same dimension as the input vector.

The accentuated role of the first and last layer within a neuronal network led [68] to the definition of a *double-framed* thin quiver representation. They use this concept to define stable double-framed thin quiver representations (see definition 7.12. in [68], definition A.1.1), which basically constates that the neuronal network should transport all the input information through

[2] "Farewell the tranquil mind; farewell content" as Othello would say.

the network and give it back within the output (although this might be by a different presentation). They use the change of basis group $\tilde{G}$ of double-framed thin quiver representations to define their so-called *moduli space* - the space of isomorphism classes of stable double-framed thin quiver representations over the action of the change of basis group $\tilde{G}$ of neuronal networks. However, as the consequences from this ansatz are rather sparse - comprising a nice interpretation of *pruning* and the notion that all possible outputs constitute a subspace of the moduli space whose topology should give us *some* information about the neuronal network training - we take a different perspective in our work. Rather than taking the neuronal networks architectural perspective we follow the data *through* the network. We do this by describing the input by a suitable topos and pursuing the information contained in it in terms of homotopy and homology theory. The impact of the architectural structure reduces to the study of potential internal groups in the later to be defined layer topoi $\mathcal{E}^l$.
To be elaborated in Chapter 3

Assigning tasks to neuronal networks immediately imposes the need to evaluate their performance. For this, one needs a *target*.

Definition 2.1.10 (Target)**.** The target of a neuronal network is the codomain of the functional relation $\Psi : \mathcal{I} \to \mathcal{T}$, $\Psi(\mathbf{x}) = \mathbf{t}$, that the neuronal network is imposed to approximate. Herein, $\mathcal{T}$ denotes the target space.

If the targets are known (at least for a certain part of the data) they are associated to the aforementioned *labels*. This setting is known as *supervised learning* - in contrast to the opposite case *unsupervised learning*. Somewhere in between one could settle *reward-learning*. Herein, we have no labels, but a reward that evaluates Ψ on the run; a perspective that is for example taken in reinforcement learning.

Still missing now is a metric to measure the distance between the output and target vector. This metric is induced by the *error* or *loss function*.

Definition 2.1.11 (Loss Function)**.** The loss or 'error' function is a functional relation that measures the "distance" between the output state and the target state. As the error corresponds to information about Ψ that is lost through the layerwise network computations,it is also called the *loss term* $\mathcal{L}$. It is defined as:

$$L(\theta, b) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i; \theta, b), \tag{2.2}$$

where θ denotes the parameters (weights) of the neuronal network, b denotes the bias, $\{x_i\}$ denotes the corpus of feature vectors and $\{y_i\}$ denote the accompanying labels respectively. In eq. (2.2), $l(x_i, y_i, \theta)$ measures how well the neuronal network with parameters θ predicts the label of a data sample, whose cardinality is m. It denotes the *error metric*. [46]

A neuronal network consists of large number of parameters θ, thus the loss function defines a high-dimensional space (whose dimensions are defined by the $\theta_{l,n}$, whereby l measures the layer and n the node position), that is not visualizable. To circumvent this, [46] and others introduced methods to find one or two dimensional reasonable visualizations. The reader might check [98] to find some impressive examples.

Definition 2.1.12 (Loss Landscape)**.** The loss landscape is a two dimensional visualization of $L(\theta)$.

Examples for the error metric l include the maximum norm, the supremum norm, the L_2 norm, the mean squared error metric, the cross entropy and many others. This metric has to be chosen with care in order to make sure that the neuronal network is pushed towards the aim underlying the task given to it. Sometimes, several loss terms are used in combination (e.g. in [82]). An example where the choice of loss term went completely wrong is Microsoft's search engine Bing.

This was optimized to maximize searches per session, but while the metrics improved, the quality of search results degraded. Why? Because more searches per session means people get more and more desperate to find things, certainly not a desirable property of a search engine. Hence, thinking carefully about the data, the task and its implications is absolutely essential for *training* a neuronal network.

Definition 2.1.13 (Training)**.** To train a neuronal network means to minimize the difference between the output and the target state and iteratively update the weights (and biases) until a (local) minimum in the *landscape* [46] of the loss is attained.

One instance of training as described in the definition above is also called a *backward pass*. After each backward pass, the performance of the network is evaluated on the loss landscape and the weights are adjusted accordingly. However, one does not want the neuronal network to 'see' all of the data for training. Thus for the training, the available data consisting of the initial states and targets, $\{\mathbf{x}_i, \mathbf{t}_i\}$, is split into three sets: the training, the validation and the test data set. The split ratio depends on the size of the whole dataset, but is often around $70\% \sim 20\% \sim 10\%$. The training of a neuronal network for one particular split of the dataset is called a *run*. It is common to repartition the dataset and initiate several runs (for example for *k-fold cross validation* [18]).

There are several algorithmic choices for the iterative process of updating the weights in definition 2.1.13. The most popular one is *backpropagation*.

Definition 2.1.14 (Backpropagation)**.** Backpropagation is an algorithm that computes - via the chain rule - the gradient in weight space of a feedforward neural network one layer at a time. Hereby, the gradient is calculated with respect to a particular choice of loss function.

There are various methods for calculating the gradients in defintion 2.1.14. Examples comprise mini-batch gradient descent, stochastic gradient descent or momentum based methods. More details can be found in appendix A.2.3. Based on the calculated gradient, the weights θ are updated as follows:

$$\theta \to \theta - \eta \frac{\partial L(\theta, \mathbf{b})}{\partial \theta} x, \tag{2.3}$$

where η denotes the *learning rate*, that measures how fast the weights tensor θ iteratively approaches a (local) minimum in the loss landscape.

Summing up the functional principle of feedforward neuronal networks, one can say that they **forward transform (input) states and backward transform the loss**. This is illustrated in figure 2.5.

2.1.3 Hyperparameters

The selection of the neuronal network architecture, in particular the quiver representation, is crucial for the success of a particular data driven artificial neural network (ANN) model. However, even if the architecture remains stable, the proper choice of *hyperparameters* can significantly boost or degrade the overall performance of an ANN [50].

Definition 2.1.15 (Hyperparameters)**.** Hyperparameters are defined as the particular choices of entities that influence an ANN's performance apart from the quiver representation.

Examples for hyperparameters include the batch size, the choice of optimization algorithm, the activation function, the validation split, the learning rate or the amount of dropout. More details can be found in appendix A.2.

2.1.4 Possible Pitfalls

Definition 2.1.16 (Overfitting)**.** Overfitting describes a neuronal network that has not only learned the *structure* of the training data, but also its statistical noise.

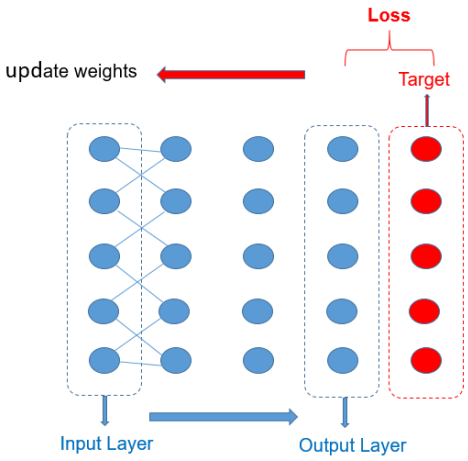


Figure 2.5: **From left to right:** The data in the input layer is transformed layer by layer as exemplary shown for one node in figure 2.3. **From right to left:** The loss is calculated as in equation (2.3) and propagated backwards to update the weights.

A neuronal network suffering from overfitting is not able to generalize from the training dataset to unknown datasets. Converseley, *overparametrization* describes the case where a neural network constitutes a model that has far more parameters than are necessary to obtain a satisfactory representation of the data. To circumvent overparametrization, several techniques have been introduced to remove "unneccessary" weights. These are called *network pruning*. However, these sparsified networks can be much harder to train than the original ones. In [53] the authors hypothesized that for each (randomly-initialized) deep neural network there exists a pruned version with comparable test accuracy:

Hypothesis 2.1.1 (The Lottery Ticket Hypothesis)**.** A randomly-initialized, dense neural network contains a subnetwork initialized in a way that it matches the test accuracy of the original neural network when trained for at most the same number of iterations. [53]

However, that this sparsified network exists does not mean that it is found a priori by the hyperparameter search process.

2.2 Autoencoders

Autoencoders are - in their vanilla form - feedforward neural networks with a so-called *bottleneck architecture*. This means that they consist of two basic parts, as depicted in figure 2.6 - namely the *encoder* and the *decoder*. The decoder is also called the *generative model*. The encoder and decoder share one layer - the latent space, at the center of the bottleneck. Thus autoencoders accomplish transformations between three 'spaces' in general: the input space $\mathcal{I}$, the output space $\mathcal{O}$ and the latent space $\mathcal{L}$. At this place we consider the autoencoder as an abstract algorithm and hence do not describe these spaces any further yet. Let e denote the encoder morphism mapping the input space to the latent space and d be the decoder mapping the latent space to the output space. Then the autoencoder as a whole is given by the morphism

$$a := d \circ e. \tag{2.4}$$

In its vanilla form, the node columns of the encoder and decoder are assorted mirror-symmetrically such as in figure 2.6.

A non-exhaustive overview of further autoencoder architectures and their applications can be found in [70].

Any autoencoder is an algorithm that is trained to minimize the error between its input and

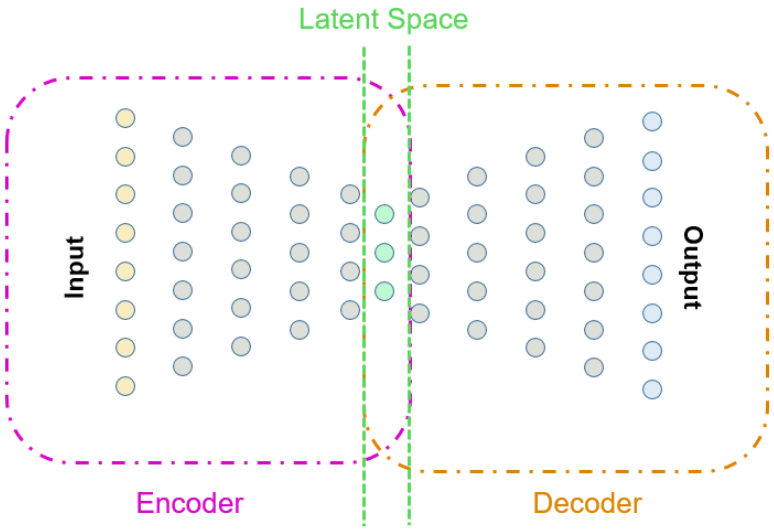


Figure 2.6: The arrangment of nodes for a vanilla autoencoder. The left part is called the encoder, the right part is the decoder. In between, they share the latent space.

equidimensional output (in this case equal to the target $\mathbf{t}$). Thus autoencoders are examples of *self-supervised* neural networks. The training error would be zero, if the autoencoder just constituted the identity transformation. This would be achievable by just selecting enough nodes to transport the data through the architecture unchanged (and circumvent biases). However, this is not the task that autoencoders are hoped to accomplish. Due to the bottleneck architecture, the reconstruction has to be performed from a compressed version of the input - the *latent representation*. As was motivated in chapter 1, the assumption is that this actually constitutes a *model* of the data - a conceptual framework from which the decoder produces the reconstruction in a - possibly cognitive [11] - way. Parts of this hope are reflected within the famous *manifold hypothesis* [34] and the *generalization capacity* [44] of neural networks.

2.3 Categorical Perspectives on Neuronal Nets

Now that the basic constituents and workwise of feedforward neuronal networks have been presented, we want to introduce categorical perspectives that have been taken to algebraically describe them. The papers shared below have in common that they focus on the (functorial) description of the forward and backward transforms within feedforward neuronal networks without characterizing them further. The characterization of the information flow within feedforward neuronal networks is rather investigated with the methods of algebraic topology, most commonly persistence homology as will be discussed in BlaBla.

2.3.1 Spivaks Compositional Perspective

We start with Spivak et al. who introduced in [52] a compositional perspective on supervised learning. They do this by defining a category of update functions as the target of a monoidal functor from the category of (hyper)parametrized functions. For A and B two sets, they define a supervised learning algorithm (*learner*) to be a tuple:

- $I : P \times A \to B$;
- $U : P \times A \times B \to P$;

- $r : P \times A \times B \to A$;

consisting of an implementation function I, an updater U and a request function r. The implementation function describes a "forward pass", the updater U then updates the weight according to the loss attained and the request function is a helper function needed to define functorial composition between two learners. Doing so, [52] define a category in which the morphisms are equivalence classes of learners:

Proposition 2.3.1 (The Category **Learn**)**.** There exists a symmetric monoidal category **Learn** whose objects are sets and whose morphisms are equivalence classes of supervised learning algorithms.

The main theorem in [52] establishes a functorial connection between this category and their category **Para**:

Definition 2.3.1. The strict symmetric monoidal category **Para** has Euclidian spaces as objects and its morphisms $\mathbb{R}^n \to \mathbb{R}^m$ are equivalence classes of differentiable parametrised functions $\mathbb{R}^n \to \mathbb{R}^m$.

Using the two categories defined above, the main theorem of [52] reads as follows:

Theorem 2.3.1 (Backpropagation as a Monoidal Functor)**.** For a real number $\epsilon > 0$ (to be interpreted as the learning rate) and $e(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ differentiable such that $\frac{\partial e}{\partial x}(x_0, -)$ is invertible for each $x_0 \in \mathbb{R}$ (to be interpreted as the error function) one can define a faithful, symmetric on objects, strong monoidal functor as follows:

$$L_{\epsilon,e} : \mathbf{Para} \to \mathbf{Learn}. \tag{2.5}$$

This sends each parametrized function $I : P \times A \to B$ to the leaner (P, I, U_I, r_I) defined as:

$$U_I(p, a, b) := p - \epsilon \nabla_P E_I(p, a, b); \tag{2.6}$$

and

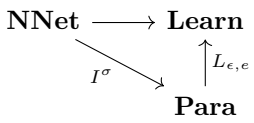
$$r_I(p, a, b) := f_a(\nabla_a E_I(p, a, b)). \tag{2.7}$$

Herein, $E_I(p, a, b) = \sum_j e(I_j(p, a), b_j)$ and f_a is the component-wise application of the inverse to $\frac{\partial}{\partial x}(a_i, -)$ for each i.

Equation (2.6) resembles the gradient descent operation in equation (**??**). This is for a good reason: the functor in equation (2.5) is also called by the authors of [52] the *gradient descent* or *backpropagation* functor that turns parametrized functions into learning algorithms. Equation (2.3) herein encodes the backpropagation value. To bring neuronal networks into the pictures, the authors first define a category **NNet**:

Definition 2.3.2 (The Category **NNet**)**.** The category of neuronal networks **NNet** has as objects natural numbers and as morphisms $m \to n$ neuronal networks of type (m, n). Composition is given by the concatenation of neuronal networks and the identity morphism is given by the 0-layer neuronal network.

In the above definition, a neuronal network of type (m, n) denotes a network with size m input layer and size n output layer. Using this category, the authors show that there is a functor of **NNet** $\to$ **Learn** that factorizes through **Para**:



In the above diagram, I^σ denotes the implementation functor, where σ is the activation function. Thus through the activation function a functor from **NNet** to **Para** is defined. Formally, it is the same as finding a thin quiver representation for a particular network quiver. One can run neural networks in parallel or add multiple inputs and duplicate outputs to each node (concatenation or juxtaposition). Thus neural networks both naturally exhibit a monoidal and bimonoidal structure. A generalization of **NNet** is given by **IDAG**, the category of interfaced directed acyclic graphs [33]. Implementing neuronal networks as parametrized functions even factors through **IDAG**:

$$\mathbf{NNet} \longrightarrow \mathbf{IDAG} \longrightarrow \mathbf{Para}$$

Abstract idags together with the operations of concatenation and juxtaposition form a symmetric monoidal category. In [33], an initial-algebra semantics for dag structure is developed that formalizes operations on dags via product and permutative categories.

2.3.2 Related Work

The main result of [52], integrating backpropagation into a categorical framework, found high response in the literature. In the following, some works building upon it will be presented.

2.3.2.1 Connection to the Lens Category

In [51], the authors show that there is a faithful, identity-on-objects symmetric monoidal functor embedding a category of asymmetric lenses into **Learn**, and also such a functor embedding **Learn** into the category of symmetric lenses. Lenses are a formalization of bidirectional transformations, meaning information is allowed to flow bidirectionally.They form a monoidal category.

Definition 2.3.3 (Lenses)**.** Let $\mathcal{C}$ be a Cartesian category. Then the category $\mathbf{Lens}(\mathcal{C})$ is defined as follows:

- the objects are pairs (A, A'), where both A and A' are objects in $\mathcal{C}$;
- a map from an object (A, A') to an object (B, B') consists of a pair (f, f^*) where $f : A \to B$ and $f^* : A \times B' \to A'$;
- the composite of $(f, f^*) : (A, A') \to (B, B')$ and $(g, g^*) : (B, B') \to (C, C')$ is given by a combination of *get* and *put* [3].
- the identity on (A, A') is the pair $(1_A, \pi_1)$.

The connection to (feedforward) neural networks is not far to seek, as we already noted in section 2.1.2 that neuronal networks forward transform states and backward transform losses. The **Lens** category is suitable as morphisms and their composites can be expressed via graphical calculus (such as was used in [52] to derive the compositionality of backpropagating algorithms).

The connection of **Learn** to lenses is elaborated further in [74], where the authors describe neuronal networks via *parametrized lenses*. More precisely, [74] interpret neuronal networks as morphisms in the category of parametrized lenses over a certain cartesian category, chosen such that it nicely describes the layerwise weight structure.

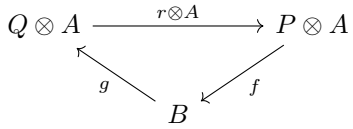
Definition 2.3.4 (Parametric Lenses)**.** The category $\mathbf{Para}(\mathbf{Lens}(\mathcal{C}))$ of parametric lenses on $\mathcal{C}$ has the same objects as the category of lenses. Morphisms from (A, A') to (B, B') consist of a choice of parameter pair (P, P') and a lens

$$(f, f^*) : (A, A') \times (P, P') \to (B, B') \tag{2.8}$$

such that $f : P \times A \to B$ and $f^* : P \times A \times B' \to P' \times A'$.

[3] These two morphisms can also be interpreted as view and update and satisfy certain concatenation axioms.

In the above definition, f propagates values forward in that it takes as input a parameter value P and a datapoint A and gives back the prediction B [86]. On the other hand, the map f^* propagates errors backwards.Thus f and f^* describe a parametrized feedforward neuronal network. The authors of [74] also fit loss maps, learning rates and optimisers into their framework. More concretely, 1-cells are given by tuples (P, f, f^*) consisting of a choice of a parameter P and a lens $(f, f^*) : P \times A \to B$ [86]. Between two 1-cells (P, f, f^*) and (Q, g, g^*), a 2-cell is given by a reparametrization lens $(r, r^*) : P \to Q$ satisfying the following commuting triangle condition [86]:



It is shown in [74] that a variety of optimizers in machine learning can be interpreted as 2-cells in this category, yet convergence properties are not included, which would be certainly necessary for a full categorical description of ANNs. The graphical language used in $\mathbf{Lens}(\mathcal{C})$ can be transferred to $\mathbf{Para}(\mathbf{Lens}(\mathcal{C}))$, thus allowing to use the composition laws of $\mathbf{Lens}(\mathcal{C})$ (which makes the request function of [51] obsolete).

2.3.2.2 Consequences

The main point about the categorical perspective on neuronal networks expressed by **NNet** is that neuronal networks can be composed in a functorial manner. Furthermore, at least for the supervised case, [52] showed that backpropagation is functorial for most error functions. As we will see below, autoencoders fall into the class of *self-supervised* algorithms and thus these results are transferable. @Uli: You see, here is some potential for further work. I like the graphical calculus, that Liam also introduced for his Temperley Lieb Categories. It's quite cool. Yet, not as cool as algebraic topology. But this might be combinable (actually I am quite sure it is), but I still have to mediate on the how part...

2.3.3 Categorical Probability

Reading the sections above, the reader could have got the impression that neuronal networks are solving optimisation problems. In fact, this is not the case because of uncertainty. More precisely, there are two types of uncertainty that hinder neuronal networks from being optimizers [86]:

- **Epistemic Uncertainty**: This describes the uncertainty due to hidden information, that is reduced by collecting more precise data.
- **Aleatoric Uncertainty**: This describes the random relationship of the data and its model, i.e. unknowns that differ each time we run the same experiment.

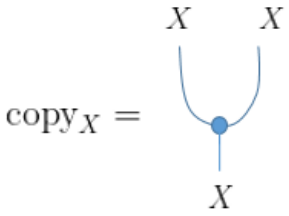
Thus neuronal networks are not pure function approximators; merely distribution approximators dependent on the amount of uncertainty. As such, they are not deterministic, but rather probabilistic learners.While in the ansatzes presented above, the categorical description of neuronal networks focussed on the differential structure of the respective categories (**Learn**), the categories considered here shall model probabilistic structure. More precisely, consider we have a neuronal network with an initial parameter (or weights set) θ_0 that we iteratively train by forward passing states and backward passing losses. What is the probability that our final result, i.e. the result after the last training step, represents the "true state of nature"? Statistically, this is a question that was introduced in 1951 by D. Blackwell and C. Stein ([2]). Their results are now known as the classical Blackwell-Sherman-Stein Theorem (BSS-Theroem) on the comparison of statistical experiments. An experiment is a mathematical structure composed of [8]:

- A set θ, called the parameter set.
- A probability measure P_θ for each $\theta \in \theta$ on the σ-field $\mathcal{A}$ of subsets of a set X.

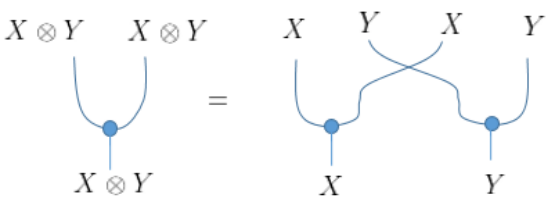
The idea behind the statistical comparison of experiments is that there exists a "true state of nature" that has effect on the observation of a random variable $(X, \mathcal{A})$, modeled by an observing statistician via the probability measure P_θ [8]. This is applicable to sequential experiments, where the stopping rule has previously been chosen [8]. As was discussed above, the forward-backward pass in a feedforward neuronal network is sequential and the stopping criterion is the reaching of a (local) minimum in the loss landscape. For autoencoders, as their target is their input value, the reaching of a loss landscape minimum has - under the manifold hypothesis - more than only (hyper)parametrical significance. Autoencoders could thus by interpreted as auto-observing experiments in the above framework. This generates the need to quantify the confidence we can put into the latent representation generating this observation via the decoder. If we have two runs of training an autoencoder, each of which reached a local minimum in the loss landscape - which result should we trust more? This is statistically measured via second order stochastic dominance, i.e. a partial order on the expected losses. The classical Blackwell-Sherman-Stein Theorem (or *dilation criterion* states that the informativeness order $\succeq$ for statistical experiments coincides with the second-order stochastic dominance order of the so-called *standard measures* on the *distribution space* $P\theta$ [58]. The original criteria developed by [2] to discriminate between experiments involve also sufficiency, informativeness, second order stochastic dominance, mean-preserving spread and martingalizability, of which he also showed mutual equivalences [8].

In [58] the authors are the first to introduce the classical Blackwell-Sherman-Stein Theorem into a categorical theory of probability. Categorical probability aims to develop a category-theoretical foundation for probability theory and mathematical statistics. There are two main ansatzes to categorical probability: *Markov categories* and the *category of Quasi-Borel spaces*. In the following, we will focus on the first, also because quasi-Borel spaces are claimed to be an instance of Markov categories [86]. [4]

Definition 2.3.5 (Markov Category)**.** A Markov category $\mathcal{C}$ is a semicartesian symmetric monoidal category where every object $X \in \mathcal{C}$ is equipped with a distinguished morphism, called the *copy* morphism: as well as a unique morphism $\mathrm{del}_X : X \to I$ making X into a commutative comonoid



such that: for all $X, Y \in \mathcal{C}$. [58]



The copy map can be used to decide whether a morphism $f : X \to Y$ in $\mathcal{C}$ is deterministic, namely if applying f to two independent copies of the input results in the same pair of outputs as when applying f directly to one input and copying the resulting output. Examples of Markov categories include **BorelStoch**, the category of standard Borel spaces and measurable Markov

[4] And also because the main developer Tobias Fritz resides at the University of Innsbruck - how cool is this?

kernels and **FinStoch**, the category of finite sets and stochastic matrices. **BorelStoch** can also be interpreted as the Kleisli category of the Giry monad on the category of standard Borel spaces and measurable maps [58]. The diagram categories of Markov categories are again Markov categories [58]. In view of the categorical treatment of the Blackwell-Sherman-Stein Theorem (parametrized by priors) the authors of [58] introduce *parametric Markov categories*.

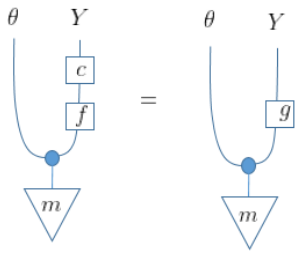
Definition 2.3.6 (Parametric Markov Category)**.** A parametric Markov Category $\mathcal{C}_P$ consists of a Markov category $\mathcal{C}$ and has the same objects as $\mathcal{C}$. The morphisms of $\mathcal{C}_P$ are defined via the morphisms $B \otimes A \to X$. Namely

$$C_W(A, X) := C(W \otimes A, X), \tag{2.9}$$

where W can be interpreted as a parameter space indexing a family of morphisms $A \to X$.

Generally, the comparison of two experiments f and g is about which is more informative about a parameter set θ that labels the distinct hypotheses one aims to discern by performing the experiment [58]. Classicaly, an experiment f is more informative than g if there exists a map c such that $cf = g$; whereas c can be interpreted as a post-processing of the data generated by experiment f in a way that produces the data of experiment g [58]. Within the Markov categorical setting, this can be rewritten as follows:

Definition 2.3.7 (Preorder of Informativeness)**.** Given morphisms $m : I \to \theta$, $f : \theta \to X$ and $g : \theta \to Y$ in a Markov category $\mathcal{C}$, f is m-a.s. more informative than g if there exists a morphism $c : X \to Y$ such that $cf =_{m-a.s.} g$:



The authors us these notions, as well as representability for the definition of standard experiments and standard measures and the existence of conditionas to obtain an abstract notion of Bayesian inference, to arrive at a Markov categorical formulation of the classical BSS Theorem.

Further elaborate on:

- independence of encoder/decoder ⇒ Lemma 12.11 in [57] ⇒ Chapter 6;
- connect topological structure functor with probability perspective and investigate whether AEs are deterministic morphisms, use [48]; graphical framework in [26]
- Fongs causal theory ([29]) as a strict symmetric monoidal category generated by a directed acyclic graph and equipped with a comonoidal structure on each object ⇒ he shwoed equivalence of a functor from a causal theory into **Stoch** to Bayesian network ⇒ transferability to AE
- Shiebler ([85]) built a generalized framework for training neural networks that have stochastic processes as layers. (Elaborated on differences between compositionality of Para and statstical Para. Also implemented compositionability with scipy ⇒ for AE functoriality: which degree of uncertainty of noise lets the AE functorial fall into the "probabilistic" pattern?)
- Bayesian lenses ([65])

2.4 Autoencoders and Data Structure

As with zero loss, $\mathcal{L} = 0$, autoencoders would reconstruct exactly the input data from the latent representation and the goal of training is to minimize the reconstruction loss, autoencoders should have to "learn" as much structure as possible from the data. Thus the idea is not far fetched that autoencoders process data in a functorial manner or - at least - should do so in order to come up to the above mentioned expectations. Herein, "an autoencoder" denotes the architecture depicted in figure 2.6, without a particular choice of hyperparameters and in particular activation functions. To "process data in a functorial manner" has to be defined in the following as well as the structure of the data on which this is done. In general, functoriality as a map between categories that preserves identity morphisms and morphism composition is a perspective, that should, in a most abstract way, allow to identify commonalities between different algorithms, derive extensions that preserve functoriality and identify modifications that break it [64]. For doing this, one could take either the data perspective and investigate how the data - or better - the information *within* the data is transformed by the algorithm. Or one could take a more global and algorithmical perspective and consider the hyperparametrized and dated algorithms themselves. This perspective was described abstractly for neuronal networks via an optimisation as well as probabilistic perspective by the authors cited in section 2.3. And I am elaborating and adjusting this for autoencoders ⇒ different architectures and amount of noise. To investigate the relationships between transformations of datasets and the autoencoders that were trained on these datasets, the first step is to look at the data and define what "structure of the data" should mean. Functorial unsupervised learning algorithms can be classified by either having one of the following two aims [86]:

- Finding the probability distribution that the data was from, or
- the low dimensional manifold on which the data is assumed to lie on.

For the autoencoder context, the claim for data to lie on a manifold is too restrictive for the layerwise description of the data and especially the latent space. (⇒ [78] Here: show and compute easy example where this is the case.) Thus, our goal is to find a way to investigate the structure transformed sequentially by the layers of an autoencoder without having to rely on manifolds. Although in [34], the authors claim to have created an algorithm in order to "test" whether a concrete (and also noisy) dataset lies on a manifold or not, this is not applicable to autoencoders trained on real world datasets. bash Fefferman more with conrete facts about why this is not the case ⇒ measures of complexity In order to define an appropriate autoencoder structure functor $\mathcal{F}_{AE,\text{struct}}$, we first need to restrict the potential class of input data.

2.4.1 The Input Data Space

In the following, the *input data space* denotes the presentation of the data before being processed within an algorithm.

Assumption 2.4.1 (Input Data)**.** The input data is a vector in $\mathbb{C}^D$, D being any integer; so each input tensor to and output tensor from a neuron is one-dimensional in $\mathbb{C}$ (as compared to definition 2.1.8 and definition 2.1.9).

For many applications, input data emerges as a *point could* in a (potentially high-dimensional) Euclidian space $\mathbb{R}^D$.

Definition 2.4.1 (Point Cloud)**.** A point cloud is a finite set of points equipped with a distance function.

For example, spatial sensor data will fall into this predicament. In fact, however, most data observed from physical processes are measured as *time series*.

Definition 2.4.2 (Discrete Time Series)**.** A scalar process x_t is called a discrete time series if x_t is a random variable and the time index t is countable.

An *observed* time series $\{x_t\}$ is a realization of the underlying stochastic process. For many applications, e.g. automated sensory measurements, the time index is equally spaced and the resulting time series denotes the set of N measurements

$$x_{t_n} = x(t_0 + n\tau_s) \quad n \in 0, .., N, \tag{2.10}$$

where t_0 is the starting time and τ_s denotes the sampling interval. Even if the time domain is a continuum, real-world time series become discrete through observation and computational processing. With respect to the paragraph above, it has further to be noted that also time series data can be transformed in a way (by Takens embedding [4]) such that it is presented as a point cloud to the algorithm.

In conclusion, we can thus constate that an autoencoder's input lives in a *finite metric space*.

2.4.2 Structure within the (Transformed) Data

A point cloud has no structure. According to definition 2.4.1 it is just a set. Mathematically, structure is *imposed* onto a set by assigning additional mathematical objects to it such as metric structures (geometries), equivalence relations, orders, topologies, algebraic structures (groups, fields, etc.; and also higher structures such as ring spectra) or measures. Hence for the (algorithmic) study of data, the algebraic objects have to be chosen accordingly.
In this work we consider the following question: '**Are Autoencoders able to learn the shape of data**'? Shape, in a loose mathematical sense, could be understood as the conceptual conglomerate of 'closeness', 'vicinity' and 'convergence' [5]. Properties of shape we could ask for include geodesic distances and curvatures, intrinsic distances, normals, dimension, spectra of differential operators such as the intrinsic Laplacian and topological invariants [14].The subset of these properties reasonable for our problem is restricted by two factors: First of all the algorithmic autoencoder setting itself and, secondly, the type of data we want to process with it.

The above mentioned question has two main dimensions:

1. How does an autoencoder process data and how can be turn this into an algebraically well defined question?

2. How can we practically test any hypothesis drawn upon 1.)?

Autoencoders are trained to minimize the error between its input and equidimensional output (in this case equal to the target $\mathbf{t}$) which would be zero for identity. Due to the bottleneck architecture, the reconstruction has to be performed from a compressed version of the input - the *latent representation*. As was motivated in chapter 1, the hope is that this actually constitutes a *model* of the data - a conceptual framework from which the decoder produces the reconstruction in a - possibly cognitive [11] - way. Parts of this hope are reflected within the famous *manifold hypothesis* [34] and the *generalization capacity* [44] of neural networks.

Definition 2.4.3 (Generalization)**.** Generalization denotes a trained network's capacity to adapt to unseen datasets.

Ideally - i.e. with zero loss $\mathcal{L}$ - the autoencoder would reconstruct exactly the input data from the latent representation. Thus the idea is not far fetched that the autoencoder processes data in a functorial manner or - at least - should do so in order to come up to the above mentioned expectations. Hence to address the first question, we represent an autoencoder functor. In fact, as a composition of functors that shall allow us to investigate to what extent *information* is processed within that. Clearly defining what this means leads over to the second question: we have to investigate **the data space(s)** as well as **the mathematical abstract information** we want to retrieve from them.

However, the main question '**Are Autoencoders able to learn the shape of data**' has also a third and higher level of abstraction. In general, neuronal networks are called *data driven models*. They are quivers of nodes that are trained (definition 2.1.13) in order to achieve a certain weight (definition 2.1.5) configuration corresponding to at least a *local minimum* of the loss landscape (definition 2.1.12). Thus, it is also desirable to compare algorithms and the data they are interactively trained on an abstract level. In figure 2.7 the three main dimensions related to our research question are depicted graphically. The questions arising in each level have to be formulated in an algebraic way such that the hypotheses set within the respective framework are testable.

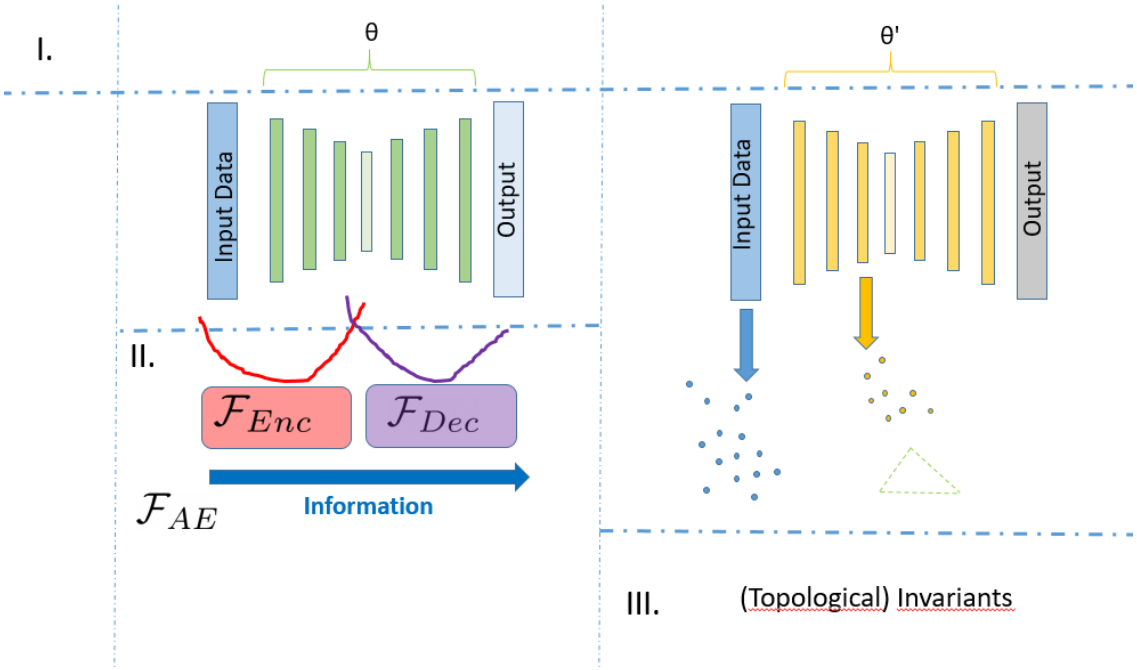


Figure 2.7: The three levels of abstraction arising with the question '**Are autoencoders able to learn the shape of data?**'. On the first level, **I.**, we pose ourselves the question how the entity of a specific autoencoder architecture and the data that is fed into it can be compared and quantitatively investigated. On the second level, **II.** we construct a functorial representation of the autoencoder algorithm. This can be split into the subquestions of whether the encoder and decoder are functorial respectively, and how to compare their processing of the input and latent data. The third level, **III.** deals with the point cloud and its layerwise transformation itself. Related to this are the questions of finding an appropriate triangulation and descriptive algebraic invariants.

In the literature, **persistence homology** has been used in various fashions on all abstraction levels, which will be presented below.

2.4.3 The First Level - I.

The first level in figure 2.7 is also the most abstract level. The goal herein is to compare specific autoencoder architectures with respect to a given input data space as a *model entitity*. However, I would like to use higher category theory to do that :D

2.4.3.1 Usage of Persistent Homology on Abstraction Level I

1. **Neural Persistence** In [47] the authors introduce a *neural persistence* as a measure for the generalization capabilities of deep neural networks via their structural complexity. In order

to calculate persistence diagrams, they introduce a weight-based filtration to the stratified graph (definition A.1.2).

Definition 2.4.4 (Weight-based Filtration after [47])**.** Given the set of weights θ **for one training step**, define $\theta_{\max} := \max_{\theta\in\theta} |\theta|$. To circumenvent scaling effects, the weights are transformed as $\theta' := \{|\theta|/\theta_{\max} | \theta \in \theta\}$, such that $1 = \theta'_0 \geq \theta'_1 \geq ... \geq 0$. A filtration for the kth layer G_k is thus defined as $G_k^{(0)} \subseteq G_k^{(1)} \subseteq ...,$ where $G_k^{(i)} := (V_k \sqcup V_{k+1}, \{(u,v)|(u,v) \in E_k \wedge \phi'(u,v) \geq \theta'_i\})$ and $\phi'(u,v) \in \theta'$ denotes the transformed weight per edge. As neurons with high weight have a large impact on the neural network calculation, this filtration allows an analysis of the neural network.

The calculations in the resulting persistence diagrams allow only for zero-dimensional topological information, as the filtration contains at most 1-simplices. Thus, in fact, a maximum spanning tree is calculated for the weights.

Definition 2.4.5 (Neural Persistence)**.** For the kth layer G_k, the neural persistence $NP(G_k)$ is the p-norm of the persistence diagram $\mathcal{D}_k$ coming from the weight-based filtration:

$$NP(G_k) := ||\mathcal{D}_k||_p := \left(\sum_{(c,d)\in\mathcal{D}_k} \text{pers}(c,d)^p \right)^{\frac{1}{p}}. \tag{2.11}$$

For $p = 2$, this caputes the Euclidian distance of points in $\mathcal{D}_k$ to the diagonal. [47]

The hypothesis set by [47] is, that topological complexity should increase as the neural network is learning, i.e. over the training process. Furthermore, the authors conjecture that 'assessing dissimilarities of neural networks by means of persistence diagrams while making use of higher-dimensional topological features will lead to further insights about their generalization and learning abilities [47].

2. **Shape of Activation Space** In [54] the authors take a very similar approach to [47]. However, instead of filtrating the neural networks weighted graph at every training step, they consider the induced graph after training.

3. **One Dimensional-PH and Complexity Analysis of Trained DNNs** Unlike [47], the authors of [91] use one-dimensional persistence homology for the complexity analysis of trained DNNs (FCN and CNN architectures). The authors claim that by using one-dimensional persistence homology, collaborations between neurons are revealed which is not the case for the zero dimensional analysis.

4. **Measuring the Similarity of NNs** In [75] the authors compare trained neural networks (as a whole, not only layerwise as in [47]) using persistence homology upt to dimension 3.

5. **Intrinsic Dimension** [87] conjectured that the trajectories of the weights during training in the weight space follow fractal trajectories and that the intrinsic dimension of this trajectories measures the generalization capacity of the ANN. In [71] developed a computational method based on persistent homology to calculate this dimension.

6. **Path Homologies** In [49] the authors use path and directed flag complex (DFC) homology in order to sort out pruned networks according to hypothesis 2.1.1. However, they consider at most networks with three layers, programed in matlab, so the appicability of their method to more interesting architectures is questionable.

2.4.4 The Second Level - II.

Level II. is concerned with the (functorial) architecture of the ANN.

2.4.4.1 Usage of Persistent Homology on Abstraction Level II

1. **Topological Layer** In [59] the authors propose a differentiable layer extracting the persistence landscapes of the data in order to force the neural network to learn topological features. The layer can be placed anywhere in the neural network.

2. **Measure of Topological Complexity** In [92] the authors calculate the persistence landscapes layerwise. They conjecture that the topological complexity increases over the whole network and does not decrease from one layer to the other.

3. **Measure of Overfitting** In [90] the authors construct a measure of overfitting in a dense, fully connected neural network by layerwise constructing clique complexes and analysing the persistence diagrams on them.

2.4.5 The Third Level - III.

On level III., the layerwise transformed data or the generated embedded submanifolds are directly investigated.

2.4.5.1 Usage of Persistent Homology on Abstraction Level III

1. In [93], the authors introduce a new metric to evaluate the amount of disentanglement of the data submanifolds within the latent space of a generative model. To do this, they use a vectorized notion of persistent homology barcodes of the conditioned submanifold embeddings - the relative living times (RLTs).

How to do this?
Keywords and Ideas

1. First of all, the (trained) weight configuration in the quivers can be compared using topological invariants (in particular Betti numbers). For example, a high amount of *dropout* (see definition A.2.2) within the layers allows to draw conclusions about the interaction between a particular dataset and a neuronal network architecture on an abstract level. Especially for deep neuronal networks, not necessarily autoencoders, this can grant insight into the data processing and thus into the applicability of a particular network architecture to a dataset - independently of the hyperparameter space. An approach similar like this has already been taken to understand and model neural networks in the brain in [60]. Herein, the authors propose that a sufficient amount of topological complexity (measured by non-trivial homology) is necessary for computation. This could also be the case for deep neural networks. Thus, for unsupervised learning, homological invariants help to sort out particular ANN models for a given complex and high dimensional dataset. Note to Uli: This is quite practical and also implementable, however, results would be empirical. I plan to advertise this as a students project and possible Master thesis at the DFKI, but would not like to perform too much computational experiments myself.

2. **neural persistence** In [47] the authors introduce a *neural persistence* as a measure for the generalization capabilities of deep neural networks via their structural complexity. In order to calculate persistence diagrams, they introduce a weight-based filtration to the stratified graph (definition A.1.2).

   Definition 2.4.6 (Weight-based Filtration after [47])**.** Given the set of weights θ for one training step, define $\theta_{\max} := \max_{\theta\in\theta} |\theta|$. To circumenvent scaling effects, the weights are transformed as $\theta' := \{|\theta|/\theta_{\max} | \theta \in \theta\}$, such that $1 = \theta'_0 \geq \theta'_1 \geq ... \geq 0$. A filtration for the kth layer G_k is thus defined as $G_k^{(0)} \subseteq G_k^{(1)} \subseteq ...,$ where $G_k^{(i)} := (V_k \sqcup V_{k+1}, \{(u,v)|(u,v) \in E_k \wedge \phi'(u,v) \geq \theta'_i\})$ and $\phi'(u,v) \in \theta'$ denotes the transformed weight per edge. As neurons with high weight have a large impact on the neural network calculation, this filtration allows an anlysis of the neural network.

The calculations in the resulting persistence diagrams allow only for zero-dimensional topological information, as the filtration contains at most 1-simplices. Thus, in fact, a maximum spanning tree is calculated for the weights.

Definition 2.4.7 (Neural Persistence)**.** For the kth layer G_k, the neural persistence $NP(G_k)$ is the p-norm of the persistence diagram $\mathcal{D}_k$ coming from the weight-based filtration:

$$NP(G_k) := ||\mathcal{D}_k||_p := \left(\sum_{(c,d)\in\mathcal{D}_k} \text{pers}(c,d)^p \right)^{\frac{1}{p}}. \tag{2.12}$$

For $p = 2$, this caputes the Euclidian distance of points in $\mathcal{D}_k$ to the diagonal. [47]

The hypothesis set by [47] is, that topological complexity should increase as the neural network is learning, i.e. over the training process. Furthermore, the authors conjecture that 'assessing dissimilarities of neural networks by means of persistence diagrams while making use of higher-dimensional topological features will lead to further insights about their generalization and learning abilities [47].

3. **Intrinsic Dimension** [87] conjectured that the trajectories of the weights during training in the weight space follow fractal trajectories and that the intrinsic dimension of this trajectories measures the generalization capacity of the ANN. In [71] developed a computational method based on persistent homology to calculate this dimension.

4. Higher category theory and neuronal models: Matthias had an idea to use higher category theory to model ... in group theory. I really like higher category theory and would like to learn more about it, but the transferability of this idea is questionable and the outcome vague...

N0000000000000000000tes
From [60]:

- sophisticated formalism of cohomological information theory introduced by Baudot and Bennequin, [8].
- In §9.1 we return to the question of how to better represent the wiring structure of the network in the functorial assignment of resources, beyond the conservation laws at vertices introduced in §6 by taking categorical equalizers.
- In §10 we introduce the formalism of finite information structures and cohomological information theory developed in [8], [81]. We construct a functorial mapping from the category of codes introduced in §7 and the category of information structures and we show that the resulting Gamma-network that maps a network and its subsystems to the respective chain complexes of information cohomology satisfies an inclusion-exclusion property.
- In the case of feedforward networks we show that the topology of the same image is essentially trivial. These two scenarios show that the situations where this family of homotopy types can be nontrivial and significant is where the network is neither a simple feedforward architecture nor a completely random wiring =¿ Can I use this for autoencoders as subcategory of feed forward networks?
- In §12 we introduce integrated information, in the form defined using information geometry as in [69]. We construct a cohomological version of integrated information in the same setting of cohomological information theory used in the previous section, and we show that it vanishes on feedforward networks as expected.
- We conclude with a proposal and some questions about a possible enrichment of the cohomological information theory construction in the form of a sheaf of spectra and the associated generalized cohomology.

So as to construct an autoencoder structure functor

$$\mathcal{F}_{AE,\text{struct}} : \mathcal{C} \to \mathcal{D} \tag{2.13}$$

between appropriate categories $\mathcal{C}$ and $\mathcal{D}$, we have to investigate **the data space(s)** as well as **the mathematical abstract information** we want to retrieve from them.

Regarding the algorithmic setting we constate our aim of using the autoencoder as an unsupervised (physical) model builder. For this reason, shape properties relying on prior knowledge are not suitable. In addition to that, shape properties should be computable not only in theory and not only for dummy examples, but real world datasets.

Intrinsic to the design of an autoencoder is the hypothesis that it is capable of segmentating *significantly* different structures. Significantly different in the autoencoder realm does a priori not mean significantly different from a (physical) model builder's perspective. This has to be investigated with the methods we present below. Independently from the result we can extract already the need for shape descriptors with a broad range of applicability. Therefore properties relying on a manifold structure like intrinsic and geodesic distances, curvatures, normals and dimension seem less preferable.

Having started with the input data space being a finite *metric* space, the nonlinear calculations within neuronal networks [78] make clear that this will not be the case for the layerwise transformed *data spaces* and so also probably not for the output data space. Thus topological spaces arise as an opportunity. Although the definition of a topological space is set theoretic, starting with Poincaré, a combinatorial approach to topology emerged whose basic study objects are *(simplicial) complexes*.

2.5 Complexification

Before we turn to the question of how information is processed within an autoencoder, we need to quantify this information. (Simplicial) complexes are one tool to ascribe structure to a set of points and perform calculations on it. They constitute a bridge between topological spaces that patch together by diffeomorphims to a combinatorial and algebraic perspective on topology. Properties of complexes are given by their inter-incidental relationships. Their field of study - i.e. *combinatorial topology* - is split into three parts [1]:

1. The theory of cycles and boundaries that can be summarized to *homology theory*.
2. The theory of sections between two or more cycles in a manifold.
3. The theory of the fundamental group.

Among these, particularly *persistence homology* is an easy, computable and interpretable tool that is widely used in (high dimensional) data analysis and also has been applied recently to deep learning in various constellations (e.g. [71], [79], [55]).
There is a broad range of *computable* complex constructions [35]. In the following, we will present the abstract process of complexifying a point cloud as a functorial one. This is a perspective that has been introduced by Carlsson and Memoli in 2010 ([22]) and been further developed by Culbertson et al. ([40]) and Shiebler ([64]).

Before we turn to the relevant functors, we will first define the required categories.

2.5.1 Simplicial Complexes

2.5.1.1 Abstract Simplicial Complexes

Definition 2.5.1 (Simplicial Complex)**.** Let V be a set of vertices. An (abstract) simplicial complex K is a collection of non-empty finite subsets of V (called *simplices* or *faces*), such that any non-empty subset of a simplex is a simplex itself.

In the following, by a "simplicial complex" we will mean an abstract simplicial complex if not denoted otherwise. If a simplicial complex is generated by the cliques in a graph, it is called a *flag complex*. A clique in a graph means an edge between connected vertices.

Definition 2.5.2 (**SCpx**)**.** Finite abstract simplicial complexes compose a category **SCpx**. The morphisms between the simplicial complex S_X with vertex set X and S_Y with vertex set Y are called *simplicial maps*. These are functions $f : X \to Y$ such that if the vertices $x_1, ..., x_n$ span an n-simplex in S_X, then the vertices $\{f(x_1), ..., f(x_n)\}$ span an m-simplex in S_Y with $m \leq n$. [64]

The process of associating an abstract simplicial complex to a set of points is functorial [45]:

$$Simp : \mathbf{Set}^{(op)} \to \mathbf{SCpx}^{(op)}. \tag{2.14}$$

It is one part of the composite definition of the singular nerve functor, definition 2.5.43.

2.5.1.2 Geometric Simplicial Complexes

A geometric simplex is the convex hull of affinely independent point sets.

Definition 2.5.3 (Geometric Simplicial Complex)**.** A *geometric simplicial complex* is a polyhedral complex in $\mathbb{R}^d$, whose cells are geometric simplices. [6]

To a given geometric simplicial complex Γ one can associate a finite abstract simplicial complex $\Delta(\Gamma)$ via the family of extreme point sets of cells in Γ. Conversely, to a finite d-dimensional abstract simplicial complex there exists geometric simplicial complexes Γ in $\mathbb{R}^{2d+1}$ such that $\Delta(\Gamma) \simeq \Delta$. The underlying space $\cup\Gamma$ is unique up to isomorphism and called the *geometric realization* of Δ, denoted by $||\Delta||$. On the other hand, Γ is called the **triangulation** of $||\Delta||$ and any space isomorphic to it. Abstract and geometric simplicial complexes are thus interchangable in the finite case.

2.5.2 Covers

In order to represent a given set of points, the triangulation or *complexification* has to cover it in a suitable way. This is ensured by the *flagification* process, that ensures that maximal simplices within the resulting complex are uniquely determined by the 2-element subsets of their elements [64].

Definition 2.5.4 (Non-Nested Flag Cover)**.** Given a set X, a non-nested flag cover C_X of X is a cover of X with the following two properties [64]:

1. If $A, B \in C_X$ and $A \subseteq B$, then $A = B$.
2. The simplicial complex whose vertices correspond to the elements of X and faces are all finite subsets of the sets in C_X is a flag complex.

The covers resulting from the flagification process now itself form a category - **Cov** - which we need as the codomain category for defining (overlapping) clustering algorithms as functors.

Definition 2.5.5 (**Cov**)**.** The category **Cov** has pairs of non-nested flag covers C_X and the corresponding finite sets X as objects. The morphisms between (X, C_X) and (Y, C_Y) are given by functions $f : X \to Y$ such that for any set S in C_X there exists some set S' in C_Y such that $f(S) \subseteq S'$. [64]

Given a non-nested flag cover (X, C_X) in **Cov**, flagification can be described as the iterated application of a functor S_{fl} : **Cov** $\to$ **SCpx**, while removing all non-maximum simplices from the cover. The functor S_{fl} maps (X, C_X) to the simplicial complex S_X whose n-simplices are the n-element subsets of sets in C_X. It has an adjoint:

$$Flag : \mathbf{SCpx} \to \mathbf{Cov}, \tag{2.15}$$

that maps the simplicial complex S_X to (X, C_X), where C_X is the flagification of the covering formed by the simplices of S_X.

2.5.2.1 Clustering Functor

Non-nested flag covers of an input space X are obtained by clustering algorithms.

Definition 2.5.6 (Clustering Function)**.** A *clustering function* is any function f that takes a distance function d on a set of points X and returns a partition Γ of X.

A partition means that two elements x and y are indistinguishable in X (i.e. assigned to a common cluster). Hence, with respect to a proper distance function d, we have that $d(x, y) = 0$. As a result, d cannot be a metric, but a (extended) pseudo metric, that is sometimes also called an uber-metric. The uber-metric spaces form a category:

Definition 2.5.7 (**UMet**)**.** Denote the category of uber-metrics spaces **UMet** as the category whose objects are finite uber-metric spaces. These are metric-space-like objects that permit $d(x_1, x_2)$ to be infinite and $d(x_1, x_2) = 0$ does not imply $x_1 = x_2$. In **UMet**, morphisms are given by non-expansive maps, i.e. functions $f : X \to Y$ such that $d_Y(f(x_1), f(x_2)) \leq d_X(x_1, x_2)$. [19]

This category is the domain category of the functor that describes the clustering process:

Definition 2.5.8 (Flat Clustering Functor)**.** A flat clustering functor is a functor $C : \mathbf{UMet} \to \mathbf{Cov}$ that is the identity on the underlying set.

This functor is non-trivial if there exists a *clustering parameter* $\delta_C \in \mathbb{R}_{\geq 0}$ where $C(\Lambda^1_\epsilon)$ is two singleton clusters for any $\epsilon > \delta_C$ and a single two point cluster for any $\epsilon \leq \delta_C$. Intuitively this means that the functor decides according to a certain distance parameter obtained by the pseudo metric whether two points belong to the same cluster or not.

2.5.3 Fuzzy Simplicial Complexes

Fuzzy simplicial complexes (a simplification of Spivak's fuzzy simplicial sets introduced in [19]) allow to describe how points in an uber-metric space group together at different scales.

Definition 2.5.9 (Fibered Fuzzy Simplicial Complex)**.** A fibered fuzzy simplicial complex is a functor $F_X : I^{op} \to \mathbf{SCpx}$ such that for any morphism $a \leq a'$ in I^{op}, the simplicial map $F_X(a \leq a')$ acts as the identity on 0-simplices. [64]

The fibered fuzzy simplicial complexes form a category **FSCpx**, whose morphisms are given by the natural transformations.

They can be flagified in a functorial way in order to obtain a fuzzy non-nested cover of the uber-metric space X.

Definition 2.5.10 (The Functor $FlagCpx$)**.** The functors $(S_{fl} \circ -) : \mathbf{FCov} \to \mathbf{FSCpx}$ and $(Flag \circ -) : \mathbf{FSCpx} \to \mathbf{FCov}$ can be composed to give a functor $FlagCpx = (S_{fl} \circ -) \circ (Flag \circ -)$, which converts any fuzzy simplicial complex into a fuzzy simplicial flag complex. [64]

The fuzzy non-nested covers resulting from the functor $FlagCpx$ form a category **FCov**, whose objects are functors $F_X : I^{op} \to \mathbf{Cov}$, such that $S_fl \circ F_X$ is a fibered fuzzy simplicial complex. Again, the morphisms are given by natural transformations.

2.5.4 The Spivak-McInnes Adjoint Pair

On the basis of the definition of his simplicial sets, in [19], Spivak first introduced the notion of an adjoint pair of functors to switch between fuzzy simplicial representations to uber-metric spaces in a similar way as geometric and simplicial complexes are interchangable or simplicial sets and their topological realizations are adjointly connected (definition real and definition 2.5.43). This notion has further been elaborated by McInnes et al. in [61].

Via the introduction of the functor $Pair : \mathbf{UMet} \to \mathbf{FSCpx}$, Shiebler adapted the Spivak-McInnes adjoint pair to *fibered* fuzzy simplicial complexes.

Definition 2.5.11 (The Functor $Pair$)**.** The functor $Pair : \mathbf{UMet} \to \mathbf{FSCpx}$ sends the uber-metric space $(X, d_X) \in \mathbf{UMet}$ to the fibered fuzzy simplicial complex $F_X : I^{op} \to \mathbf{FSCpx}$. For $a \in (0, 1]$, $F_X(a)$ is a simplicial complex whose set of 0-simplices is X and whose 1-simplices are the pairs $\{x_1, x_2\} \subseteq X$ such that $d_X(x_1, x_2) \leq -\log(a)$. The morphism $f : X \to Y$ is mapped to the natural transformation whose component at a is f. [44]

The corresponding fuzzy singular set functor $S(\cdot)_{fin}$ is then defined as [64]:

$$S(\cdot)_{fin} = FlagCpx \circ Pair. \tag{2.16}$$

For $a \in (0, 1]$, $S(\cdot)_{fin}(a)$ is a flag complex in which the set $\{x_1, ..., x_n\} \subseteq X$ forms a simplex if all pairwise distances between points in the set are less than $-\log(a)$. This is the *Vietoris-Rips Complex* of (X, d_X) at $-\log(a)$.

Definition 2.5.12 (The Functor $|\cdot|_{fin}$)**.** The functor $|\cdot|_{fin} : \mathbf{FSCpx} \to \mathbf{UMet}$ maps a fibered fuzzy simplicial complex $F_X : I^{op} \to \mathbf{SCpx}$ with vertex set X to (X, d_X) such that $d_X(x_1, x_2) = \inf\{-\log(a) | a \in (0, 1], \{x_1, x_2\} \in F_X(a)[1]\}$. A natural transformation $\mu : F_X \to F_Y$ is sent by $|\cdot|_{fin}$ to the function determined by μ's actions on the vertex set of F_X.

The finite realization of F_X is an uber-metric space where the distance between the points x_1, x_2 is determined by the strength of the 1-simplex connecting them.

2.5.5 Kleinbergs Impossibility Theorem and Hierarchical Clustering

There are three basic properties one would like to require from a clustering algorithm: *scale invariance*, *consistencey* and *richness*.

Definition 2.5.13 (Scale Invariance)**.** For any distance function d and any $\alpha > 0$, $f(d) = f(\alpha d)$. [10]

Definition 2.5.14 (Richness)**.** Let $Range(f)$ denote the set of all partitions Γ, such that $f(d) = \Gamma$ for some distance function d. Then a clustering function f is called rich, if $Range(f)$ is equal to the set of all partitions of X.[10]

Definition 2.5.15 (Consistency)**.** Suppose that the partition Γ arises from the distance function d. If we introduce a new distance function d' by reducing distances within the clusters and enlarging distances between the clusters then d' should lead to the same partition Γ. [10]

In 2002, Kleinberg proved the following impossibility theorem about the above three properties:

Theorem 2.5.1 (Impossibility Theorem for Clustering)**.** For each $n \geq 2$, there is no clustering function f that is scale-invariant, rich and consistent. [10]

One way to deal with the Kleinberg impossibility is to take a persistential approach to clustering and consider clusterings at various scales. In this realm, one obtains a (non-trivial) *hierarchical clustering functor*.

Definition 2.5.16 (Hierarchical Clustering Functor)**.** A non-trivial hierarchical clustering functor H introduces a hierarchical scale parameter $a \in (0, 1]$ into the definition of the flat clustering functor, such that we obtain a non-trivial flat clustering functor with clustering parameter $\delta_{H,a}$ for all $a, H(-)(a) : \mathbf{UMet} \to \mathbf{Cov}$. [64]

2.5.6 Maximal and Single Linkage

Maximal and single linkage clustering are non-trivial hierarchical clustering fucntors that are obtained by taking the maximal simplices and the connected components of the resulting complex respectively.

Definition 2.5.17 (The Connected Components Functor π_0)**.** The functor π_0 sends S_X to the tuple (X, C_X), where C_X is the set of connected components of S_X.[64]

Using this, the maximal and single linkage clustering functors are given by:

$$\begin{aligned} \mathcal{ML} &= (Flag \circ -) \circ S(\cdot)_{fin}, \\ \mathcal{SL} &= (\pi_0 \circ -) \circ S(\cdot)_{fin}. \end{aligned} \tag{2.17}$$

Culbertson et al. ([40]) state that any non-trivial clustering functor refines the output of sinlge linkage clustering for a given clustering parameter, while its output is refined by maximal linkage clustering. So in this sense, non-trivial clustering functors lie "in between". This statement is refined to a proposition about corresponding (parametrized) natural transformation by [64]:

Proposition 2.5.1. Suppose $H : \mathbf{UMet} \to FCov$ is a non trivial hierarchical clustering functor with respective clustering parameter $\delta_{H,a}$ for the respective clustering functor $H(-)(a) : \mathbf{UMet} \to \mathbf{Cov}$. Define a parameter $W_H(a) = e^{-\delta_{H,a}}$. Then this parametrizes natural transformations with inclusion maps as components from $\mathcal{ML}(-)(W_H(-))$ to H and from H to $\mathcal{SL}(-)(W_H(-))$.

In conclusion, clustering algorithms can thus be described as functors between the categorie of simplicial complexes, uber-metric spaces and coverings.

2.5.7 Complexification/Triangulation Algorithms

In the following, the complexification methods being the basis of implementations for common persistent homology calculations will be presented and discussed. Regarding the implementations, the author favors *giotto-tda* ([88], a very usable *python* library for topological calculations that was designed to bridge the gap between topological data analysis and machine learning. Other implementations include the *GUDHI* library ([37]), *scikit-tda* ([56]), *Dionysus 2* ([94]) and also the *julia* platform *Eirene* ([80]). In the following, the most common algorithms will be presented, based on their practical usage within the above mentioned implementations. Basically these comprise Vietoris-Rips complexes and modifications thereof, Čech complexes as well as Delaunay triangulations and modifications.

2.5.7.1 Data within the Layers of a Neuronal Network

For our purposes and the autoencoder case, we can *assume* that the input data is a vector in the Euclidian space $\mathbb{R}^n$. Claiming functoriality, we can *hope* that the output vector does so too. But what happens in between? Layer for layer, the data is changed in a nonlinear way. At least for *relu*, a really common activation function (see appendix A.2.1), [78] described how topological spaces in $\mathbb{R}^n$ would be changed. They demonstrate on simple geometrical sample sets how the topological operations act on the topological input data space: it is scaled, translated, rotated, reflected, bended and quotiented. So a number of problems arises when one wants to get a mathematical grab on the data processed within neuronal networks and - in particular- autoencoders.

1. The output of a layer does not have to be a vector in a d-dimensional Euclidian space any more. Can we assume this for the autoencoder or more generally, reconstructive networks? (In contrast to classification architectures)

2. Do we deal with data manifolds after all? Manifold reconstruction is a challenging task for noisy data by itself. The manifold hypothesis is quite strong by itself and does not necessarily leads to more insight into the data processing within a neural network.

3. In order to perform topological calculations on a point cloud data set and extracting the respective information out of it, the data set needs to be triangulated. One might be interested in doing this layerwise. Within the autoencoder setting, this is driven by the interest to compare the embedding process of the encoder with the reconstruction within the decoder. Firstly, this can become computationally cumbersome.

4. Secondly one has to ask oneself carefully, which triangulation makes sense for the layerwise deformed input data point cloud. The wish for applicability within real world scenarios, being subject to uncertainty, noise and high dimensionality, forces one to overthink which triangulations are meaningful in order to obtain reliable topological (homotopical or homological) information. This is not clear a priori.

2.5.7.2 Nerve

The nerve theorem presented below implies a really strong statement about the homotopy of a point set and its triangulation. Hence complexifications stemming from a nerve construction are favorable.

Definition 2.5.18 (Nerve)**.** Suppose $\mathcal{U}$ is a collection of subsets of a simplicial complex X. The nerve of $\mathcal{U}$ is given by the simplicial complex $\mathcal{N}(\mathcal{U})$ defined by the following two conditions:

1. The set of vertices of $\mathcal{N}(\mathcal{U})$ consists of all non-trivial elements of $\mathcal{U}$;
2. a finite subset $\sigma \subset \mathcal{U}$ is a simplex of $\mathcal{N}(\mathcal{U})$ iff $\cap_{U\in\sigma} U \neq \emptyset$,

whereby (2) implies (1) [89].

Theorem 2.5.2 (Nerve Theorem)**.** Let X be a triangulable space and $(A_i)_{i\in I}$ be a locally finite family of open subsets such that $X = \cup_{i\in I} A_i$. If every nonempty intersection $A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_n}$ is contractible, then X and the nerve $\mathcal{N}(A_i)$ are homotopy equivalent. [6]

If $\mathcal{U}$ is a cover, the abstract simplicial complex associated with X through the nerve construction is called the covers nerve complex [36]. Conversely, one wants to have covers that lead to an application of the nerve theorem. These are defined as *good covers*.

Definition 2.5.19 (Good Cover)**.** Let X be a topological space. A family of contractible open subspaces U_i forms a *good (open) cover* if every nonempty intersection $U_{i_1} \cap ... \cap U_{i_n}$ is contractible.[42]

The parts of a data cloud being coverable in this way determines a topological property of this data, quantified within the *(strict) covering type*.

Definition 2.5.20 ((Strict) Covering Type)**.** The *strict covering type* of a topological space X is the minimum number of elements within a good cover of X. The *covering type* of X, $ct(X)$ is the minimum of strict covering types of spaces X' that are homotopy equivalent to X. [42]

For example, contractible spaces like the sphere have $ct(X) = 1$, a disjoint union of contractible spaces has $ct(X) = 2$ and so forth.
For the realization $||K||$ of a simplicial complex K there is a canonical good cover given by the collection of subsets consisting of *stars of vertices* [42]:

Definition 2.5.21 (Star)**.** If v is a vertex in the realization $||K||$ of K, the *star* of v is the union of all open simplices whose closures contain v. On the contrary, the *open star* is the union of the interiors of the simplices that contain v. This is illustrated in figure 2.8. [36]

Definition 2.5.22 (Open Star Cover)**.** There is a partially ordered covering of the realization $||K||$ of a finite simplicial complex K. The partial order of this covering is induced by the partial order of the vertices of the simplicial complex, that is indeed a total order of vertices within each individual simplex. A finite collection of sets in the cover intersect iff the corresponding vertices span a simplex. This is called the open star cover of a simplex k, $stars(K)$. [7]

Using the open stars cover $stars(K)$, each simplicial complex K can naturally be expressed as a nerve. In fact, the function $v \to star(v)$ defines an isomorphism of simplicial complexes $K \to \mathcal{N}(stars(K))$ [7].

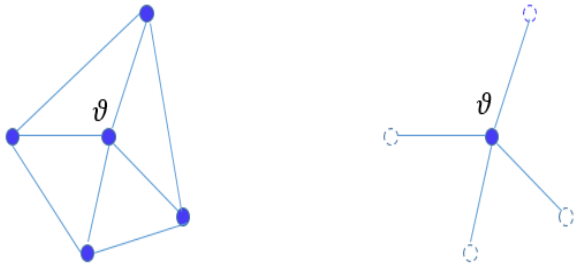


Figure 2.8: **Left:** The star of a vertex v (in the middle). **Right:** The open star of a vertex v (in the middle). Figure adapted from [36].

2.5.7.3 Vietoris Rips Complex

Let $\mathcal{D}$ be a discrete subset of the Euclidian space $\mathbb{R}^n$ representing our point cloud data. Let $\epsilon > 0$ be a constant.

Definition 2.5.23 (Vietoris Rips Complex)**.** The Vietoris Rips complex of scale ϵ on $\mathcal{D}$, $VR_\epsilon(\mathcal{D})$ is the simplicial complex whose simplices are all those finite collections of points in $\mathcal{D}$ of pariwise-distance $\leq \epsilon$. [35]

The degree to which a Vietoris Rips complex captures the topology of $\mathcal{D}$ is not clear, there might not even be homotopy equivalence between $VR(\mathcal{D})$ and the projection of its convex hull in $\mathbb{R}^n$ [35]. Hence, the Vietoris Rips complex might not be representative for the point cloud topology. In fact, [23] showed that the projection map induced by the Vietoris Rips complex does not preserve fundamental group data for point sets in dimension larger than three. Yet, the Vietoris Rips complexification seems to capture large-scale holes (Betti numbers for large ϵ) correctly [35].

The main advantage of Vietoris-Rips complexes is that they are easy to compute, as they rely only on the 1-skeleton. In topological data analysis, Vietoris Rips complexes have been used to recover the topology of a dataset sampling a specific shape [32].
Hence they build the basis of homology computations in many implementations, in particular giotto-tda.

The Vietoris Rips complexification can be expressed as a functor from the category of finite pseudo metric spaces **UMet** to the category of fuzzy simplical complexes coming from a cover, i.e. fuzzy simplizial flag complexes:

$$\begin{aligned} &VR : \mathbf{UMet} \to \mathbf{FSCpx}, \\ &VR_\epsilon(X, d_X) \approx S(X, d_X)_{\text{fin}}(\epsilon). \end{aligned} \tag{2.18}$$

Herein, the approximative equivalence stems from the fact that $Pair$ is defined via the $-log$ distance of a constant in definition 2.5.11, but one could use other distance measures as well. Let K be an abstract simplicial complex, V be the collection of its vertices and U be the collection of all (sub)simplices of K. Then the Vietoris Rips complex of U equals K, hence to each abstract simplicial complex K one can associate a Vietoris complex [89].

2.5.7.4 Čech Complex

Closely related to the Vietoris Rips complex is the Čech complex.

Definition 2.5.24 (Čech Complex)**.** Let X be a finite set of points in $\mathbb{R}^n$. The Čech complex is the abstract simplicial complex whose k-simplices are given by the subsets of $k+1$ points that

can be enclosed in a ball of radius ϵ [32]:

$$C(X,\epsilon) = \{\sigma | \emptyset \neq \sigma \subset X, Rad(\sigma) \leq \epsilon\}. \tag{2.19}$$

On the other hand, a k-simplex consisting of vertices $\{v_0, ..., v_k\}$ belongs a Čech complex iff the $k+1$ closed Euclidian balls $B(v_i, \epsilon)$ have non-empty common intersection [32]. The Čech complex is the nerve of the collection of balls centered at its vertices, $\{B(v,\epsilon)|v \in V\}$ and thus the nerve theorem 2.5.2 implies homotopy equivalence of the Čech complex and the union of these balls [32].

2.5.7.5 Weak and Strong Delauny Complexes

To define weak and strong Delaunay complexes, one first needs the notion of *Voronoi cells* and the associated covering, the *Voronoi diagram*.

Definition 2.5.25 (Voronoi Cell)**.** Let A be a set of points in $\mathbb{R}^n$. The set A determines a decomposition of $\mathbb{R}^n$ into *Voronoi cells* via [16]:

$$V_a = \{x \in \mathbb{R}^n | |x-a| \leq |x-b| \quad \forall b \in A\} \tag{2.20}$$

These cells define a covering of $\mathbb{R}^n$ called Voronoi diagram [16]:

$$Vor(A, \mathbb{R}^n) = \{V_a | a \in A\}. \tag{2.21}$$

Now the *strong Delauny complex* $Del(A, \mathbb{R}^n)$ is the nerve of the Voronoi covering. It is an abstract simplicial complex on A defined via [16]:

$$\sigma \in Del(A, \mathbb{R}^n) \iff V_\sigma \neq \emptyset, \tag{2.22}$$

for each nonempty finite subset $\sigma \subset A$. Then a nonempty finite subset σ of A is called a *simplex with vertices in* A. A p-simplex is a simplex $\sigma = \{a_0, a_1, ..., a_p\}$ of cardinality $p+1$. If A is in general position, i.e. all the faces of the convex hull of A are simplices, then $Del(A, \mathbb{R}^n)$ is an n-dimensional geometric simplicial complex which triangulates the convex hull of A [16]. The complex $Del(A, \mathbb{R}^n)$ may contain simplices of dimension larger than n, hence $Del(A, \mathbb{R}^n)$ does not need to be an embedding [16]. On the other hand, for the definition of *weak Delauny complexes* one needs the notion of *witness points*.

Definition 2.5.26 (Strong Witnesses)**.** Let σ be a simplex with vertices in A. A point $x \in \mathbb{R}^n$ is a *strong witness* for σ with respect to A, if $|x-a| \leq |x-b|$ whenever $a \in \sigma$ and $b \in A$ [16].

Definition 2.5.27 (Weak Witnesses)**.** Let σ be a simplex with vertices in A. A point $x \in \mathbb{R}^n$ is a *weak witness* for σ with respect to A, if $|x-a| \leq |x-b|$ whenever $a \in \sigma$ and $b \in A$ σ [16].

Based on the notion of strong and weak witnesses, one can define the *weak* and *strong Delaunay complex*.

Definition 2.5.28 (Strong Delaunay Complex)**.** The *strong Delaunay complex* $Del(A, \mathbb{R}^n)$ is the abstract simplicial complex on A defined via [16]:

$$\sigma \in Del(A, \mathbb{R}^n) \iff \sigma \quad \text{has a strong witness in } \mathbb{R}^n. \tag{2.23}$$

Similarly, one can define the weak Delauny complex via weak witnesses:

Definition 2.5.29 (Weak Delaunay Complex)**.** The *weak Delaunay Complex* is the abstract simplicial complex on A defined via [16]:

$$\sigma \in Del^w(A, \mathbb{R}^n) \iff \text{every subsimplex } \tau \leq \sigma \quad \text{has a weak witness in } \mathbb{R}^n. \tag{2.24}$$

In 2008, [16] proved the following theorem:

Theorem 2.5.3 (Weak Witness Theorem)**.** Let $A \subset \mathbb{R}^n$ and let $\sigma \subset A$ be a simplex. Then σ has a strong witness in $\mathbb{R}^n$ iff every subsimplex $\tau \leq \sigma$ has a weak witness in $\mathbb{R}^n$.

This means that the abstract simplicial complexes obtained via strong and weak witnesses are equivalent.

Postulate 2.5.4. The weak witness theorem also holds - under the following four assumptions - in non-Euclidian geometries [16]:

1. X is a topological space, A is a set.
2. For each $a \in A$, there is a continuous function $d(a, -) : X \to \mathbb{R}$.
3. Each unordered pair $x, y \in X$ is contained in a connected set $\gamma(x, y) \subset X$.
4. For each $a, b \in A$, the set

$$R(a, b) = \{x \in X | d(a, x) \leq d(b, x)\} \tag{2.25}$$

is convex, meaning that $\gamma(x, y) \subset R(a, b)$, whenever $x, y \in R(a, b)$.

Of the above four postulates, the first two guarantee the definition of weak and strong witnesses and Delaunay complexes, respectively. The third postulate imposes convexity on X and the fourth asserts that the Voronoi cells in X are convex. Under these four assumptions, the weak witness theorem also holds within non-Euclidian spaces. One can also constate a weak witness theorem for approximate Delaunay complexes. These are complexes where the postulates 2.5.4 are formulated up to an uncertainty parameter ϵ, which also induces a filtration of the complexes respectively.

2.5.7.6 Witness Complexes

Witness complexes were constructed in order to reduce computational complexity within the calculation of Vietoris Rips complexes. They are constructed by choosing a (well-distributed) set of *landmark points* $\mathcal{L}$ within a subset B of an (Euclidian) space and subsequently building the Delaunay complexes $Del(\mathcal{L}, B)^{(w)}$ using the ambient metric [12]. One can also use approximate Delaunay complexes. When X is a submanifold of Euclidian space, the question arises of how well this can be reconstructed via Delaunay constructions on a sample set B. So the question is whether there exists a plausible chain of equivalences:

$$X = Del(\mathcal{L}, X) = Del^w(\mathcal{L}, X) = Del^w(\mathcal{L}, B). \tag{2.26}$$

This problem is circumvented if one is only interested in homological invariants of X. The filtrations of approximate Delaunay complexes $\mathbf{Del}(\mathcal{L}, B)$ and $\mathbf{Del}(\mathcal{L}, B)$ can be shown to encode the homology of X in their persistent homology under suitable sampling conditions [16]. Persistent homological calculations also do not make it necessary to show that an individual complex in the filtration $Del(\mathcal{L}, B; \epsilon)$ is homotopy equivalent to X [16].

2.5.7.7 α-Complexes

Alpha complexes are a family of subcomplexes of the Delaunay complex, obtained via a radius constraint [24]. Alpha complexes were introduced as a substitution of Čech complexes that become huge at large r and also contain redundant information. For X a finite set of points in $\mathbb{R}^d$, the alpha complex $Alpha(X, r)$ is much smaller, geometrically realizable in $\mathbb{R}^d$ and delivers the correct homotopy type. Let p be in X and intersect the r-ball around p with its Voronoi region:

$$R_r(p) = B_r(p) \cap V_p. \tag{2.27}$$

The resulting sets are convex and their union equals X_r. The alpha complex of X and r is the nerve of the collection of these sets:

$$Alpha(X, r) = \{\sigma \subseteq X | \cap_{p \in \sigma} R_r(p) \neq \emptyset\}. \tag{2.28}$$

The equivalence of homotopy types of $Alpha(X, r)$ and X_r follows from the application of the nerve theorem. The alpha complex has dimension at most equal to the ambient dimension and we have that $Alpha(X, \infty) = Del(X)$. The free parameter within the alpha complexification is r, it may be varied in order to obtain a filtration.

2.5.7.8 Tangential Delaunay Complexes

Tangential Delaunay complexes were introduced to reconstruct a k-dimensional manifold embedded in d-dimensional Euclidian space based on a finite point set $\mathcal{P}$. To define them, we need the notions of weighted points and weighted Delaunay triangulation.

Definition 2.5.30 (Weighted Point)**.** A weighted point is a pair consisting of a point p of $\mathbb{R}^d$, called *center*, and a non-negative real number $\omega(p)$, called the *weight*. [21]

Given a set of points $\mathcal{P} = \{p_1, ..., p_n\} \subseteq \mathbb{R}^d$, a *weight function* on $\mathcal{P}$ is a non-negative real-valued function $\omega : \mathcal{P} \to [0, \infty)$. This can be used to define a *weighted Voronoi cell* of $p \in \mathcal{P}$ as

$$\begin{aligned} Vor^{\omega}(p) = \{x \in \mathbb{R}^d \quad : \quad ||p - x||^2 - \omega^2(p) \\ \leq ||q - x||^2 - \omega^2(p), \forall q \in \mathcal{P}\}. \end{aligned} \tag{2.29}$$

The weighted Voronoi cells and their k-dimensional faces form the weighted Voronoi diagram of $\mathcal{P}$, that decomposes $\mathbb{R}^d$ into convex polyhedral cells. For τ a subset of $\mathcal{P}$, the collection of all simplices such that $Vor^{\omega}(\tau) \neq \emptyset$ constitutes the weighted Delaunay triangulation $Del^{\omega}(\mathcal{P})$ [21]. Now let $Del^{\omega}_{p_i}(\mathcal{P})$ be the weighted Delaunay triangulation of $\mathcal{P}$ restricted to the tangent space T_{p_i}. The star of p_i in $Del^{\omega}_{p_i}$ is denoted as $star(p_i)$.

Definition 2.5.31 (Tangential Delaunay Complex)**.** The tangential Delaunay complex $Del^{\omega}_{T}(\mathcal{P})$ is defined as the simplicial complex $\{\tau, \tau \in star(p), p \in \mathcal{P}\}$. [21]

Let π be any map that maps $Del^{\omega}_{T}(\mathcal{P})$ to its closest point on the manifold $\mathbb{M}$ to be reconstructed. Then π defines an isotopy between $Del^{\omega}_{T}(\mathcal{P})$ and $\mathbb{M}$.

2.5.8 The Codomain Category of $\mathcal{F}_{AE,\mathbf{struct}}$

Intrinsic to the design of an autoencoder is the hypothesis that it is capable of segmentating *significantly* different structures. Significantly different in the autoencoder realm does a priori not mean significantly different from a physical model builder's perspective. This has to be investigated with the methods we present below. Independently from the result we can extract already the need for shape descriptors with a broad range of applicability. Therefore properties relying on a manifold structure like intrinsic and geodesic distances, curvatures, normals and dimension seem less preferable.

Having started with the input data space being a finite *metric* space, the nonlinear calculations within neuronal networks [78] make clear that this will not be the case for the layerwise transformed *data spaces* and so also probably not for the output data space. Thus topological spaces arise as an opportunity. Although the definition of a topological space is set theoretic, starting with Poincaré, a combinatorial approach to topology emerged whose basic study objects are *(simplicial) complexes*.

Definition 2.5.32 (Simplicial Complex)**.** Let V be a set of vertices. A simplicial complex K is a collection of non-empty finite subsets of V, such that any non-empty subset of a simplex is a simplex itself.

Properties of complexes are given by their inter-incidental relationships. Their field of study - i.e. combinatorial topology - is split into three parts [1]:

1. The theory of cycles and boundaries that can be summarized to *homology theory*.
2. The theory of sections between two or more cycles in a manifold.
3. The theory of the fundamental group.

Homology theory herein is by far the easiest as it deals - in contrast to the theory of sections - only with one given complex and needs - in contrast to the theory of the fundamental group - only finite abelian groups. Furthermore homology theory cannot only be applied to the cell decomposition of a manifold but to arbitrary complexes.

(Simplicial) complexes immediately fulfill two of our above mentioned needs: First of all there is a broad range of *computable* complex constructions [35]. As we obtain our input data in the form of a point cloud and need to triangulate it in order to introduce *structure* into it, simplicial constructions constitute a natural extension. To describe this structure, we need to quantify it. The way to do this is via Secondly, *topological invariants*, for which simplicial complexes offer a basis for calculation.

2.5.8.1 Topological Invariants

Definition 2.5.33 (Topological Invariant)**.** A topological invariant is any property of a topological space X that is invariant under homeomorphisms.

Elementary examples for topological invariants of a topological space X are given by the set $\pi_0(X)$ of *path components* of X and its *fundamental group* $\pi_1(X, x)$.

Definition 2.5.34 (Path Components)**.** The path components of X are given by the quotient of X by an equivalence relation $\simeq$, namely $x \simeq y$ if there exists a continuous path $p : [0, 1] \to X$ satisfying $p(0) = x$ and $p(1) = y$.

This kind of topological invariant was already used for the definition and implementation of the *topological autoencoder* architecture [82]. Whether they provide an advantage over classical architectures is task-dependent: For classification tasks, it might not be advantagous if path connected paths of the data remain vicinous. Already standard classification datasets such as MNIST ([96]) exhibit topologically similar structures (6,9,0), that yet we want to separate in our latent representation obtained by the autoencoder.

Definition 2.5.35 (Fundamental Group)**.** To any topological space X equipped with a base point $x \in X$ one can associate the fundamental group $\pi_1(X, x)$. Its elements are given by the homotopy classes of continuous paths $p : [0, 1] \to X$ satisfying $p(0) = x = p(1)$.

The set $\pi_0(X)$ and the fundamental groups $\{\pi_1(X, x)\}$ can be combined to a single invariant $\pi_{\leq 1}(X)$ - the *fundamental groupoid* of X.

Definition 2.5.36 (Fundamental Groupoid)**.** The fundamental groupoid $\pi_{leq1}(X)$ of a topological space X is a category whose objects are the points of X. Morphisms between objects x and y of $\pi_{leq1}(X)$ are given by a homotopy class of continuous paths $p : [0, 1] \to X$ satisfying $p(0) = x$ and $p(1) = y$.

The path components of X can then be recovered as the set of isomorphism classes of objects of $\pi_{leq1}(X)$ and each fundamental group $\pi_1(X, x)$ is identified with the automorphism group of the point x.

2.5.8.2 Homotopy

Due to the (training) error, not even the input and output space of an autoencoder are homeomorphic. Thus the strongest equivalence one can ask for in the context of autoencoders - under the assumption that the related spaces are topological ones - is homotopy equivalence. This is a

basic equivalence relation on the set of continuous functions from one topological space to another that allows the study of intrinsic algebraic invariants and also gives rise to new ones for topological spaces and continuous functions [20]. In the category of topological spaces **Top** *CW complexes* are the objects of study for homotopy theory.

Definition 2.5.37 (CW Complex)**.** A CW complex is a topological space X inductively constructed as follows:

- The starting point is a discrete set X^0, whose points are regarded as 0-cells.
- The **n-skeleton** is defined inductively from X^{n-1} by attaching n-cells e^n_α via maps $\phi_\alpha : S^{n-1} \to X^{n-1}$. Thus X^n is the quotient space of the disjoint union $X^{n-1} \amalg_\alpha D^n_\alpha$ of X^{n-1} with a collection of n-disks D^n_α under the equivalence relation $x \sim \phi_\alpha(x)$ for $x \in \partial D^n_\alpha$. As a set, X^n equals $X^{n-1} \amalg_\alpha e^n_\alpha$, where each e^n_α is an open n-disk.

This induction can either be stopped at a finite stage, setting $X = X^n$ for some $n < \infty$, or continued indefinitly, setting $X = \cup_n X^n$. In the latter case, X is equipped with the weak topology: A set $A \subset X$ is open, respectively closed, iff $A \cap X$ is open, respectively closed in X^n $\forall n$. [9]

The 'C' in CW complexes stands for *closure finiteness*, meaning that the closure of each cell meets only finitely many other cells. The 'W' symbolizes the above mentioned property of X having the *weak topology* with respect to the set of skeleta $\{X^n\}$. In fact, every simplicial complex is a CW complex and every n-simplex is an n-cell since every n-simplex is homeomorphic to the unit ball in Euclidian space, E^n [20].

2.5.8.3 Simplicial Sets

simplicial sets and quick introduction to simplicial structures enriched with probabilistic data in [60]

However, following Poincar´es combinatorial approach to topology and homotopy theory, one does not really want to work directly on the cells or simplicial complexes. Rather, one wants to model topological spaces via the inter-incidential relationships between their basic constituents, or complexes. For this, one needs the definition of simplicial objects.

Definition 2.5.38 (Simplicial Objects)**.** Let $\mathcal{C}$ be an arbitrary category and Δ be the category of finite ordinal numbers $[n]$ with order-preserving maps between them. A *simplicial object* in $\mathcal{C}$ is a contravariant functor from Δ to $\mathcal{C}$. A cosimplicial object in $\mathcal{C}$ is a covariant functor from Δ to $\mathcal{C}$. [62]

Let $X : \Delta^{\mathrm{op}} \to \mathcal{C}$ be a functor. Then for every ordered set $[n] \in \Delta$ we have an object $X([n]) =: X_n$ in $\mathcal{C}$. All morphisms in Δ can be described as a composite of *face maps* $\delta_i s$ and *degeneracy maps* $\sigma_j s$. Hence we need to know what the maps $X(\delta_i) =: d_i : X_n \to X_{n-1}$ and $X(\sigma_j) =: s_j$ do. We need to understand the sequence of objects $X_0, X_1, ...$ and morphisms d_i, s_j in $\mathcal{C}$. These satisfy the following relations [62]:

$$
\begin{aligned}
d_i \circ d_j &= d_{j-1} \circ d_i \quad i < j, \\
s_i \circ s_j &= s_{j+1} \circ s_i, \quad i \leq j, \\
d_i \circ s_j &= \begin{cases} s_{j-1} \circ d_i, & i < j, \\ 1_{[n]}, & i = j, j+1, \\ s_j \circ d_{i-1}, & i > j+1. \end{cases}
\end{aligned}
\tag{2.30}
$$

Simplicial objects in a category $\mathcal{C}$ form a category where the morphisms are natural transformations of functors. This category is denoted as $\mathbf{s}\mathcal{C}$. One example is given by **sSet**, the category of simplicial sets.

Definition 2.5.39 (Simplicial Set)**.** A simplicial set is a contravariant functor $X : \Delta^{\mathrm{op}} \to \mathbf{Sets}$.

A *simplicial map* $f : X \to Y$ of simplicial sets is a natural transformation of contravariant set-valued functors defined on Δ. In **sSet** one can define *standard n-simplices* Δ^n which can be classified via the Yoneda embedding through simplicial maps $\Delta^n \to Y$. These will be needed later for the construction of the *simplex category* $\Delta \downarrow X$ of a simplicial set X.

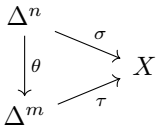
Definition 2.5.40 (Standard n-Simplex)**.** The standard n-simplex in the category **sSet** is defined as

$$\Delta^n = \hom_\Delta(\cdot, [n]). \tag{2.31}$$

Historically, the category **sSet** denotes the first fully algebraic model for homotopy theory [17].

Simplicial sets have a functorial relationship with topological spaces via the *realization functor*. Let **Top** denote the category of topological spaces. The realization functor $|\cdot|$ is a functor from the category of simplicial sets **sSet** to **Top**, the category of topological spaces. It is constructed via the use of the simplex category $\Delta \downarrow X$ of a simplicial set X.

Definition 2.5.41 (Simplex category)**.** The *simplex category* $\Delta \downarrow X$ of a simplicial set X has maps $\sigma : \Delta^n \to X$ as objects (*simplices*). An arrow of $\Delta \downarrow X$ is a commutative diagram of simplicial maps:



Herein, θ is induced by a unique ordinal number map $\theta : [m] \to [n]$ [17].

This can be used to define the realization of a simplicial set X.

Definition 2.5.42 (Realization)**.** In the category of topological spaces, the realization $|X|$ of a simplicial set X is defined as the colimit

$$|X| = \varinjlim_{\Delta^n \to X} |\Delta^n|, \tag{2.32}$$

where the limit is taken in $\Delta \downarrow X$ [17].

For each simplicial set X, the realization is a CW-complex [17]. The realization functor is left adjoint to the singular functor $\mathcal{S}(T)$ for a topological space T.

Definition 2.5.43 (Singular Functor)**.** The singular (simplicial) set $\mathcal{S}(Y)$ of a topological space Y is a functor $\Delta \to \mathbf{Set}$ that assigns $[n]$ to the set $\hom_{\mathbf{Top}}(|\Delta^n|, Y)$, the set of all continuous maps from $|\Delta^n|$ to Y [76].

Proposition 2.5.2. The realization functor is left adjoint to the singular functor. There is an isomorphism

$$\hom_{\mathbf{Top}}(|X|, Y) \cong \hom_{\mathbf{SimpSet}}(X, \mathcal{S}(Y)), \tag{2.33}$$

where $\hom_{\mathbf{Top}}$ denotes continuous maps of topological spaces and $\hom_{\mathbf{sSet}}$ denotes morphisms of simplicial sets [76].

For a topological space Y, the singular simplicial set $S(Y)$ is a Kan complex (definition B.2.3). In fact, every Kan complex can be constructed in this way up to homotopy equivalence [97]. More precisely, the unit map $u_X : X \to S(|X|)$ is a homotopy equivalence for any Kan complex X and a weak homotopy equivalence for every simplicial set X. This is useful because the category of simplicial sets **sSet** has a closed model structure (appendix B.1) that is combinatorial in nature, whose combinatorial homotopy theory is equivalent in a strong sense to the ordinary gemoetrical homotopy theory of topological spaces. Thus the geometric realization functor $X \to |X|$ induces a fully faithful embedding of *homotopy categories* (definition B.1.4):

$$h\mathrm{Kan} \to h\mathrm{Top}, \tag{2.34}$$

whose essential image consists of those topological spaces having the homotopy type of a CW complex (definition 2.5.37). The combinatorial homotopical structure of simplicial sets is described via Kan fibrations and Kan complexes and as a consequence of proposition 2.5.2 the realization of a Kan fibration is a Serre fibration (example B.1.1). Hence, as a codomain category of $\mathcal{F}_{AE}$, we choose the one that is most suitable for the algebraic study of homotopic properties of our data *topoi* over the layerwise autoencoder structure. Inspired by *Grothendieck's Homotopy Hypothesis* (hypothesis B.6.1) we choose the category of simplicial sets, **SimpSet**, thus:

$$\mathcal{F}_{AE} : \mathbf{FinMet} \to \mathbf{SimpSet}. \tag{2.35}$$

3

Data Topoi

4

Homotopical and Homological Investigations

5

A Categorical Notion of Stochastic Optimization

6

Stochastic Optimization within Autoencoders

Appendix A

Neuronal Networks

Neuronal networks were defined as thin quiver representation together with a choice of activation function in definition 2.1.4. However, this alone does not constitute the artificial data driven model - in short the ANN - yet. Thus in section A.2 a non-exhaustive overview of hyperparametrical choices influencing the *performance* of an ANN will be presented.

A.1 Quivers

A.1.1 Thin Quiver Representation

Definition A.1.1 (Stable double-framed thin quiver representation)**.** Let $\tilde{W}$ be a thin quiver representation of the delooped hidden quiver (i.e. the quiver reduced by the input and output column) $\tilde{Q}$. A **stable** double-framed thin quiver representation satisfies the following two conditions:

1. The only sub-representation U of $\tilde{W}$ that is contained in the kernel of the output function h is the zero sub-representation.
2. The only sub-representation V of $\tilde{W}$ that contains the image of the input function l is $\tilde{W}$ itself.[68]

Herein, the input and output function are the conglomerate of intra-layer weight connections for the input and output layers respectively.

A.1.2 Stratified Graph and Layers

Even more, a given feedforward neural network with an arrangment of neurons and their connections (edges) E constitutes a stratified graph [47]. Let θ denote the respective weight configuration, that is changing during training. Thus one requires a function $\phi : E \to \theta$ that maps a specific edge to a weight [47]. Fixing a specific activation function, the connections form a *stratified graph* [47]:

Definition A.1.2 (Stratified Graph and Layers)**.** A stratfified graph is a multipartite graph $G = (V, E)$ satisfying $V = V_0 \sqcup V_1 \sqcup ...$, such that for $u \in V_i$ and $v \in V_j$ and $(u, v) \in E$ we have that $j = i + 1$. As a result, edges only occur between an adjacent vertex set. A kth layer of a stratified graph is deifined as the unique subgraph $G_k := (V_k \sqcup V_{k+1}, E_k := E \cap \{V_k \times V_{k+1}\})$.

A.2 Hyperparameters

A.2.1 Activation Functions

Table A.1 displays a non exhaustive overview of common activation functions.

Activation Functions	
Name	**Definition**
Identity	$g(x) = x$
Linear	$g(x) = ax, a \in \mathbb{R}$
Step Function	$g(x) = H(x)$, where $H(x)$ is the Heaviside function
Sigmoid	$g(x) = \frac{1}{1+e^{-x}}$
Hyperbolic Tangent	$g(x) = \tanh(x)$
Rectified Linear Unit (ReLU)	$g(x) = \max(x, 0)$
Leaky ReLU	$g(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases}$
Parametric ReLU	$g(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases}$
Exponential Linear Unit (ELU)	$g(x) = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{otherwise} \end{cases}$
Softsign	$g(x) = \frac{x}{1+\lvert x\rvert}$
Softmax	$g(x_i) = \frac{e^{x_i}}{e^{\mathbf{x}}}$

Table A.1: Examples for activation functions g commonly used in neural networks.

A.2.2 Dropout

Basically training means to adjust θ (definition 2.1.5) in a way such that $L(\theta)$ is small (definition 2.1.11). However, especially for deep neuronal networks, this is a cumbersome task with a large number of potential pitfalls. One of these is overfitting (definition 2.1.16). One workaround for overfitting could be to fit all possible neuronal networks to the dataset in question and then average the predictions. This, of course, is not practicable. Thus *dropout* was introduced:

Definition A.2.1 (Dropout)**.** Dropout means the process of randomly droping units (along with their connections) from a neuronal network during training. [38]

The process of dropout is illustrated in figure A.1. Basically dropout means that nodes within the quiver are removed as well as the corresponding interrelative edges. This leads to "thinned" networks, from which dropout samples during training. While training, a node is active with a probability p, whereas in the subsequent testing, the node is always present but its weight is multiplied with p [38], as illustrated in figure A.2. Doing this, the above mentioned effect of averaging the predictions of many network architectures can be approximated.

A.2.3 Optimization Function

In the neuronal network context, optimization is performed mainly with respect to the choice of weights (definition 2.1.5) in order to find at least a local minimum in the loss landscape (definition 2.1.12). The most popular optimizer is the *gradient descent* algorithm, which layerwise calculates the gradient of the loss function with respect to the weights within this layer.

Example A.2.1 (Gradient Descent for Quadratic Loss)**.** For a quadratic loss function, the gradient would be calculated as follows:

$$\frac{\partial L(\theta, \mathbf{b})}{\partial \theta_k} = \frac{\partial}{\partial \theta_k} \left(\sum_x \left(f_{\theta,\mathbf{b}}(x) - \Psi(x) \right) \right)^2, \tag{A.1}$$

herein, θ denotes the tensor of weight choices for the complete quiver representation and θ_k denotes the weights for one layer k. The bias tensor is given by $\mathbf{b}$, and f is the thin quiver

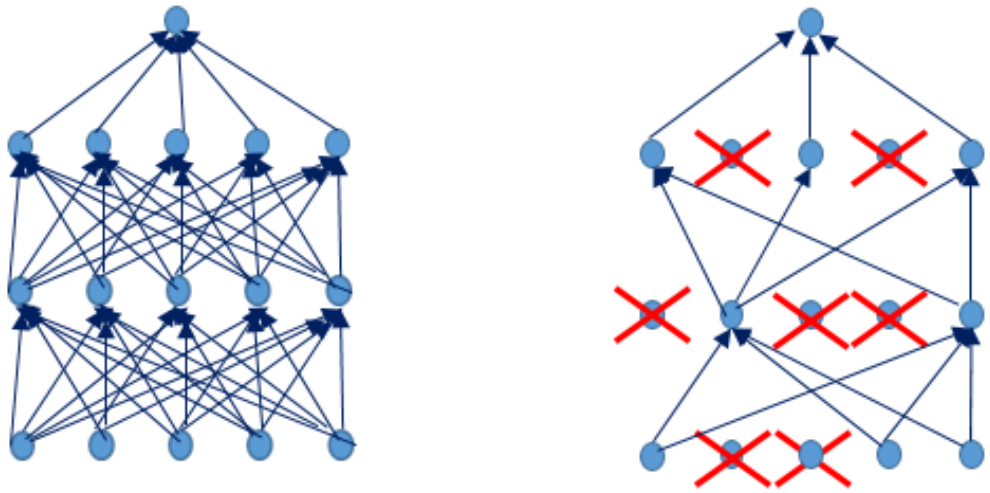


Figure A.1: **Left:** Normal neuronal network. **Right:**Neuronal network with dropout nodes. Figure adapted from [38].

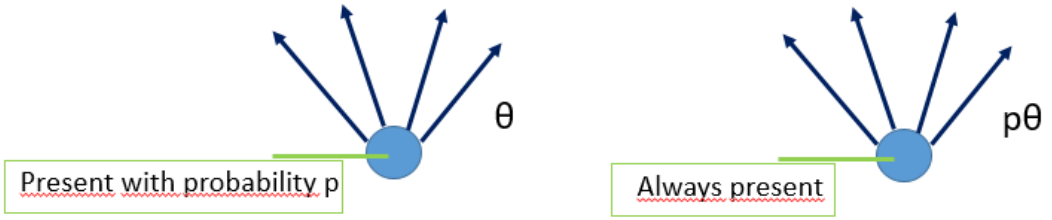


Figure A.2: **Left:** Node during training. **Right:**Node during test time. Figure adapted from [38].

representation enriched with activation functions interpreted as a single functional relationship. For the layer k under consideration x represents the given input. The functional relationship Ψ to be approximated by the neuronal network is defined as in definition 2.1.10.

The problem with the standard gradient descent as presented exemplary above is that it uses the whole training sets for the calculations of the layerwise gradients. One modification to workaround this is *stochastic gradient descent*. Instead of using the whole input x and comparing with $\Psi(x)$, one uses one particular (labeled) sample point $\{x, y\}$ instead.

Example A.2.2 (Stochastic Gradient Descent for Quadratic Loss)**.** Again, for a quadratic loss function, stochastic gradient descent would be calculated as:

$$\frac{\partial L(\theta, \mathbf{b})(\{x, y\})}{\partial \theta_k} = \frac{\partial}{\partial \theta_k} \left((f_{\theta,\mathbf{b}}(x) - \Psi(x)) \right)^2 . \tag{A.2}$$

For stochastic gradient descent, the calculations are much faster, but at the cost of producing more noise and not necessarily targeting a (local) minimum.
In between the standard gradient descent and stochastic gradient descent stands *mini-batch gradient descent*. Instead of using the whole training set or just one labeled sample, it uses a specific part - or mini-batch m - or the training set. For the quadratic loss function, this is calculated as follows:

Example A.2.3 (Mini-batch Gradient Descent for Quadratic Loss)**.** For a quadratic loss term, the layerwise gradient for mini-batch gradient descent calculates as follows:

$$\frac{\partial L(\theta, \mathbf{b})}{\partial \theta_k} = \frac{\partial}{\partial \theta_k} \left(\sum_m \left(f_{\theta,\mathbf{b}}(x) - \Psi(x) \right) \right)^2. \tag{A.3}$$

One problem with stochastic gradient descent was that it does not need to aim a (local) minimum. Hence, *momentum* was introduced as a further hyperparameter to force the algorithm into the direction of (local) minima. This is done by introducing a pseudo-velocity v_τ that is modified over the training epochs. For a training epoch $\tau \in T$:

$$v_\tau = \gamma \cdot v_{\tau-1} + \eta \frac{\partial L(\theta, \mathbf{b})}{\partial \theta_k} x, \tag{A.4}$$

where γ is the **momentum**. The weights are then updated via

$$\theta_k \to \theta_k - v_\tau. \tag{A.5}$$

There are many more modifications of the above mentioned algorithms and ideas. Examples include:

- Nesterov accelerated gradient descent ([39]);
- Adagrad ([27]);
- AdaDelta ([30]) and
- Adam ([43]).

A.2.4 Learning Rate

In (2.3) the learning rate η was fixed, but there are many techniques to adapt it in order to ameliorate the performance of the ANN. In fact, the (initial) learning rate often is the "single most important hyper-parameter" [28]. Examples for adaptation schemes include:

- time-based adaption;
- step-based adaption and
- exponential adaption.

A.2.5 Batch Size

The formulation used for definition 2.1.13 implies that the whole part of the data reserved for training is used at once. This is computationally cost intensive and thus the training data is split into small portions called *batches* for training. This thought also underlies *stochastic mini-batch gradient descent* presented above. Using batches not too small even allows for the parallelization of the involved computation and thus a significant speed up in model training.

Appendix B

About Kan Complexes and Grothendiecks Homotopy Hypothesis

In the following, we are working our path from model categories over Kan complexes to Grothendiecks homotopy hypothesis (hypothesis B.6.1), which will help us to retrieve and describe the homotopy theory of topological spaces purely algebraically; without actually having to deal with their nasty geometries. To do this, we want to switch from the category of topological spaces **Top** to the category of simplicial sets **sSet**. This is kind of a modern formulation of a conjecture (conjecture B.6.2), that Grothendieck gave in [77] and which is still a topic of ongoing research ([25], [31]). Originally, in [77], Grothendieck states that there are two ways to associate to a category $\mathcal{C}$ an algebraic object:

- either we associate to $\mathcal{C}$ the (classifying) topos $\mathcal{C}^\wedge$;
- or we associate to $\mathcal{C}$ a (semi-) simplicial set via the *nerve functor* $N(\mathcal{C})$.

From the latter construction, one is able to obtain a topological space associated to $\mathcal{C}$ via definition 2.5.42. Figure B.1 depicts an overview over the concepts that we will need to follow Grothendiecks thoughts up to his conjecture B.6.2 in its modern formulation characterized by John C. Baez ([15]). Underlying is Grothendiecks idea that "presumably, the category of ∞-groupoids is a 'model category' for the usual homotopy category" [77]. Grothendieck constructs a fundamental ∞-groupoid functor Π_∞ from the category of topological spaces to the category ∞-**Gpd** of Grothendieck ∞-groupoids (of which a simplified definition is explained in [25]). Grothendieck conjectures that this functor induces an equivalence of categories between the homotopy category of topological spaces and an appropriate localization of ∞-**Gpd**, a conjecture that still remains unproven [31].

$$\begin{array}{ccccc} \{\text{Categories}\} & \xrightarrow{N(\cdot)} & \{\infty-\text{Categories}\} & \supset & \{\text{Kan Complexes}\} \\ & & \cap & & \uparrow{\scriptstyle S(\cdot)} \\ & & \{\text{Simplicial Sets}\} & & \{\text{Topological Spaces}\} \end{array}$$

Figure B.1: Categories are related to ∞-categories via the nerve construction, $N(\cdot)$. Every Kan complex is an ∞-category. A simplicial set is a nerve iff it satisfies a special variant of the Kan extension property. Furthermore, every topological space X determines a simplicial set $S(X)$ with the property that each $S_n(X)$ is the collection of continuous maps from the topological n-simplex into X. [97]

The Baez formulation of this conjecture will end up in an equivalence of homotopy categories for $\mathbf{Top}_{\text{Quillen}}$ and $\mathbf{sSet}_{\text{Kan}}$. This will allow us to work in the *combinatorial* model structure of $\mathbf{sSet}_{\text{Kan}}$ instead of the non-combinatorial $\mathbf{Top}_{\text{Quillen}}$ model structure. (Combinatorial basically means that the underlying category is locally presentable in this context.)

B.1 Model Categories

Definition B.1.1 (Model Category)**.** A model category is a category $\mathcal{C}$ together with three classes of morphisms

- weak equivalences $W_{\mathcal{C}}$,
- fibrations $\text{Fib}_{\mathcal{C}}$,
- cofibrations $\text{Cof}_{\mathcal{C}}$,

which are all closed under composition [17].

A morphism that is both a fibration and a weak equivalence is an *acyclic fibration*, dually, a morphism that is both a cofibration and a weak equivalence is an *acyclic cofibration*. This notation is following [69], in other resources this kind of morphisms is called a *trivial* (co)fibration respectively.

The category $\mathcal{C}$ together with these morphisms must satisfy the following axioms [69]:

MC1. $\mathcal{C}$ has all small limits and colimits. There exists an initial object $\emptyset$ and a terminal object $*$.

MC2. If f and g are morphisms such that fg is defined and two out of these three morphisms are weak equivalences, then so is the third. This is called the *2-out-of-3 property*.

MC3. Weak equivalences, fibrations and cofibrations are closed under retracts.

MC4. Given the following commutative diagram

$$\begin{array}{ccc} A & \xrightarrow{f} & X \\ {\scriptstyle i}\downarrow & \overset{l}{\nearrow} & \downarrow{\scriptstyle p} \\ B & \xrightarrow[g]{} & Y \end{array}$$

a lift l exists when either i is a cofibration and p is an acyclic fibration or when i is an acyclic cofibration and p is a fibration.

MC5. Each morphism f in $\mathcal{C}$ can be factored in two ways:

(a) $f = pi$, where i is a cofibration and p is an acyclic fibration.

(b) $f = pi$, where p is a fibration and i is an acyclic cofibration.

Definition B.1.2 (Fibrant, Cofibrant and Bifibrant Objects)**.** For $\mathcal{C}$ a model category, we call an object $X \in \mathcal{C}$ ([69]):

- *fibrant*, if the unique morphism $X \to *$ is a fibration;
- *cofibrant*, if the unique morphism $\emptyset \to X$ is a cofibration;
- *bifibrant* if it is both fibrant and cofibrant.

Definition B.1.3 (Quillen Adjunction)**.** Suppose that $\mathcal{C}$ and $\mathcal{D}$ are model categories. A pair

$$F : \mathcal{C} \leftrightarrows \mathcal{D} : U \tag{B.1}$$

of adjoint functors with F the left adjoint is a *Quillen adjunction* if the following equivalent conditions are met:

- F preserves cofibrations and acyclic cofibrations;
- U preserves fibrations and acyclic fibrations;
- F preserves cofibrations and U preserves fibrations;
- F preserves acyclic cofibrations and U preserves acyclic fibrations [69].

The reason one needs Quillen adjunctions (i.e. adjunctions that respect the model structure of the respective categories) is that one wants to obtain equivalences of the associated *homotopy categories*.The homotopy category for a given model category $\mathcal{C}$ is the category one obtains by formally inverting the weak equivalences in $\mathcal{C}$.

Definition B.1.4 (Homotopy Category)**.** Let $\mathcal{C}$ be a model category and $\mathcal{C}_{cf}$ be the full subcategory of bifibrant objects. Then the homotopy category h$\mathcal{C}$ is - up to equivalence of categories, the category whose

- objects are the objects of $\mathcal{C}_{cf}$;
- morphisms are the homotopy classes of morphisms in $\mathcal{C}$ [69].

Definition B.1.5 (Quillen Equivalence)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be two model categories equipped with a Quillen adjunction. Then $\mathcal{C}$ and $\mathcal{D}$ are *Quillen equivalent* if the *derived adjunction* (i.e. the adjunction of the respective homotopy categories) $\mathbb{F} : h\mathcal{C} \leftrightarrows h\mathcal{D} : \mathbb{U}$ is an equivalence of categories [69].

B.1.1 Top

Example B.1.1 (**Top**)**.** In the category of topological spaces **Top**, a model structure $\mathbf{Top}_{\text{Quillen}}$ is given by

- W: the weak homotopy equivalences,
- fibrations: the Serre fibrations,
- cofibrations: the retracts of relative cell complexes.

By the model category axioms one can always find a weakly equivalent cofibrant object (via cofibrant replacement) for every object in **Top**. This means that every topological space as an approximation by CW complexes which is weakly equivalent to it.

B.1.2 sSet

There are many model structures on the category **sSet**. The most famous ones are the Kan-Quillen model structure (for the formation of the homotopy function complexes) and the Joyal model structure (convenient framework for $(\infty, 1)$-categories). The various model structures on **sSet** encode different homotopy theories, examples of which are the model structure on simplicial algebras or models on simplicial cylinders [69]. The Kan-Quillen model structure provides a convenient combinatorial framework for the study of topological spaces.

Example B.1.2 (Kan-Quillen model structure on **sSet**)**.** There is a combinatorial model structure on the category of simplicial sets, **sSet**, where a morphism $f : X \to Y$ of simplical sets is a

- weak equivalence if its geometric realization $|f| : |X| \to |Y|$ is a weak homotopy equivalence in **Top**
- fibration if it is a Kan fibration, i.e. if it has the right lifting property for all horn inclusions:

$$\begin{array}{ccc} \Lambda^n_k & \longrightarrow & X \\ \downarrow & \nearrow & \downarrow f \\ \Delta^n & \longrightarrow & Y \end{array}$$

- cofibration if it is a monomorphism.

The Kan-Quillen model structure is denoted as $\mathbf{sSet}_{\mathrm{Kan}}$ [69].

The fact that the homotopy theory of simplicial sets coincides with the homotopy theory of topological spaces, i.e. that simplicial sets are - up to homotopy - a combinatorial model for topological spaces, can be expressed by the following Quillen equivalence [69]:

$$|-| : \mathbf{sSet}_{\mathrm{Kan}} \leftrightarrows \mathbf{Top}_{\mathrm{Quillen}} : S(-). \tag{B.2}$$

B.2 Kan Complexes

Kan complexes are the fibrant objects in the category of simplicial sets **sSet**.

Definition B.2.1 (kth Horn)**.** The kth horn $|\Lambda^n_k|$ on the n-simplex $|\Delta^n|$ is the subcomplex of $|\Delta^n|$ that is obtained by removing the interior of $|\Delta^n|$ and the interior of the face $d_k\Delta^n$. The associated simplicial set is referred to as Λ^n_k. [76]

Definition B.2.2 (Kan Extension Property)**.** Let Y_S be a simplicial set. For $0 \leq k \leq n$, every map

$$\sigma_0 : \Lambda^n_k \to Y_S \tag{B.3}$$

admits an extension

$$\sigma : \Delta^n \to Y_S, \tag{B.4}$$

where Δ^n denotes the standard n-simplex, def. 2.5.40.

Definition B.2.3 (Kan Complex)**.** A simplical set Y_S is a Kan complex iff every map $\sigma_0 : \Lambda^n_k \to Y_S$ may be extended to a map defined on Δ^n in the sense that there is a commutative diagram

$$\begin{array}{ccc} \Lambda^n_k & \xrightarrow{\sigma_0} & Y_S \\ \downarrow & \nearrow & \\ \Delta^n & & \end{array}$$

Then Y_S is also called a *fibrant* simplicial set. [17]

Thus a simplicial object Y_S is satisfying the Kan extension property if any morphism can be extended in the way given within the above commutative diagram. It means that every horn on an n-simplex within Y_S can be filled such that the rest of the simplex is there too. In a Kan complex, every morphism is invertible up to homotopy.

Definition B.2.4 (The Homotopy Category of Kan Complexes)**.** The homotopy category of Kan complexes, hKan is defined as follows:

- The objects of hKan are Kan complexes.

- If X and Y are objects of hKan, then

$$\hom_{\mathrm{hKan}} = [X, Y] = \pi_0(\mathrm{Fun}(X, Y)) \tag{B.5}$$

  is the set of homotopy classes of morphisms from X to Y.

- For three objects X, Y, Z of hKan, the composition law

$$\circ : \hom_{\mathrm{hKan}}(Y, Z) \times \hom_{\mathrm{hKan}} \to \hom_{\mathrm{hKan}}(X, Z) \tag{B.6}$$

  is characterzid by the formula $[g] \circ [f] = [g \circ f]$.

B.3 ∞-Categories

We now get to know another special type of simplicial sets satisfying certain conditions listed below, ∞-categories. The relation to Kan complexes is as follows: Every Kan complex is an ∞-category and moreover, every ∞-category $\mathcal{C}_\infty$ contains a largest Kan complex obtained by removing all non-invertible morphisms. Furthermore, for every pair of objects $X, Y \in \mathcal{C}_\infty$, one can associate a Kan complex $\hom_{\mathcal{C}_\infty}(X, Y)$, which is referred to as the *space of maps* from X to Y. Also, the collection of all Kan complexes can be organized into and ∞-category S, the ∞*-category of spaces*.

Definition B.3.1 (Weak Kan Extension Property)**.** A simplicial set Y_S satisfies the weak Kan extension property if for $0 < k < n$, every map

$$\sigma_0 : \Lambda^n_k \to Y_S \tag{B.7}$$

admits a unique extension

$$\sigma : \Delta^n \to Y_S. \tag{B.8}$$

Intuitively, this means that only the *inner* horns have (unique) fillers. If we have fillers for all inner horn inclusions, we call the corresponding object a *quasi-category*.

Definition B.3.2 (∞-Category)**.** A simplicial set Y_S satisfying the weak Kan extension property is called an ∞-category.

An ∞-category does not only encode information about objects and morphisms between the, but also about homotopies between the latter.
For hypothesis B.6.1, we also need the notion of the category ∞**Type** to be defined below:

Definition B.3.3 (Homotopy n-type)**.** A homotopy n-type is a CW complex whose homotopy groups above the nth vanish for all basepoints.

Definition B.3.4 (∞**Type**)**.** ∞**Type** is the ∞-category of homotopy types with [15]:

- CW complexes as objects;
- continuous maps as 1-morphisms;
- homotopies as 2-morphisms;
- ....

B.4 $(\infty,1)$-Categories and Fundamental Groupoids

Definition B.4.1 ((n,k)-Category)**.** An (n,k)-category is a structure consisting of

- objects;
- morphisms;
- 2-morphisms (morphisms between morphisms);
- ...;
- n-morphisms

together with suitable compositions and identities and for which all the i-morphisms are invertible for $i > k$ [69].

In this sense, ∞-categories are (∞,∞)-categories, i.e. no conditions on invertibility are taken.

Definition B.4.2 (Fundamental n-Groupoid)**.** The fundamental n-groupoid of a topological space X, $\Pi_{\leq n}X$ has the following structure [69]:

- objects are the points of X;
- morphisms are paths in X;
- 2-morphisms are homotopies of paths;
- 3-morphisms are homotopies of homotopies;
- ...and so forth.

B.5 Getting $(\infty,1)$-Categories from Model Categories

To any model category one can assign an $(\infty,1)$-category via *simplicial localization*.

Definition B.5.1 (Localization of a Category)**.** Let $\mathbf{W}$ be a subcategory of a category $\mathcal{C}$. Then the *localization* of $\mathcal{C}$ with respect to $\mathbf{W}$, $\mathcal{C}[\mathbf{W}^{-1}]$ has the same objects as $\mathcal{C}$. It is obtained by formally inverting the maps of $\mathbf{W}$. [3]

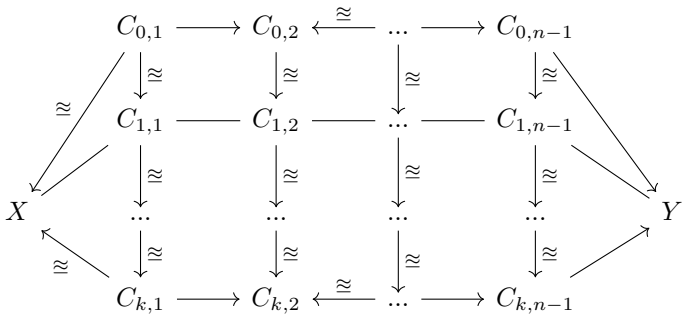
For a homotopical category $\mathcal{C}$ we take $\mathbf{W}$ to be the class of weak equivalences. The *simplicial localization* $L\mathcal{C}$ of $\mathcal{C}$ then is an $(\infty,1)$-category realized concretely as a simplicially enriched category, such that the original category injects into it, $\mathcal{C} \hookrightarrow L\mathcal{C}$, and every weakly equivalent morphism in $\mathcal{C}$ becomes a full equivalence in $L\mathcal{C}$ [95].

For a (combinatorial) model category $\mathcal{C}$, define $L(\mathcal{C},W)$ to be a *simplicially enriched* category with the same objects as $\mathcal{C}$. Firstly, simplicially enriched means that for every two objects X,Y in $\mathcal{C}$, the maps $X \to Y$ form a simplicial set $L\mathcal{C}(X,Y)$ [3]. Secondly, $L\mathcal{C}(X,Y)$ has the localization $\mathcal{C}[\mathbf{W}^{-1}]$ as its category of components, meaning that for every two objects $X,Y \in \mathcal{C}$,

$$\pi_0 L\mathcal{C}(X,Y) \approx \mathcal{C}[\mathbf{W}^{-1}](X,Y), \tag{B.9}$$

i.e. the components of $L\mathcal{C}(X,Y)$ are in $1-1$ correspondence with the maps $X \to Y \in \mathcal{C}[\mathbf{W}^{-1}]$ [3].

One way to construct this simplicial set of morphisms for a category with weak equivalences $(\mathbf{W})$ is the *hammock localization*. Herein, the simplicial set $L^H\mathcal{C}(X,Y)$ has k-simplices given by *reduced k-hammocks* which are commutative diagrams of the form:



with the following properties [69]:

- $n \geq 0$;
- all vertical morphisms are in the class $\mathbf{W}$;
- columnwise the morphisms go in the same direction, if they point to the left they are in the class $\mathbf{W}$;
- we have horizontal zig-zags, i.e. morphisms in adjacent columns go in different directions;
- no column contains only identity morphisms.

For three objects X, Y, Z in $\mathcal{C}$ the composition morphism

$$L^H\mathcal{C}(X,Y) \times L^H\mathcal{C}(Y,Z) \to L^H\mathcal{C}(X,Z) \tag{B.10}$$

is given by horizontally concatenating hammock diagrams as in fig. B.5. Now the crucial point about simplicial localization is that there is an equivalence of homotopy categories

$$h\mathcal{C} \simeq hL^{(H)}(\mathcal{C}, \mathbf{W}) \tag{B.11}$$

for a model category $\mathcal{C}$ and its associated quasi-category $L^{(H)}(\mathcal{C}, \mathbf{W})$ [69]. Even more, for $\mathcal{C}$ and $\mathcal{D}$ two Quillen equivalent model categories, there is an associated equivalence of $(\infty, 1)$-categories [69]:

$$L^{(H)}(\mathcal{C}, \mathbf{W}) \simeq L^{(H)}(\mathcal{D}, \mathbf{W}). \tag{B.12}$$

For a simplicial model category, the set of bifibrant objects of $\mathcal{C}$ form a full subcategory, which is a model for the $(\infty, 1)$-category presented by $\mathcal{C}$ and is up to equivalence the same as the one obtained by hammock localization.

Example B.5.1 (**Top** and **sSet**)**.** The model category **Top** gives the $(\infty, 1)$-category ∞**Type**, the model category **sSet** gives the $(\infty, 1)$-category ∞**Gpd** [15].

B.6 Grothendiecks Homotopy Hypothesis

We now arrive at the fundamental statement of this chapter: **Grothendieck's Homotopy Hypothesis**. Under this terminus, we understand the following:

Hypothesis B.6.1 (Homotopy Hypothesis for dimension ∞)**.** There is an equivalence of ∞-categories

$$\Pi_\infty : \infty\mathbf{Type} \to \infty\mathbf{Gpd}. \tag{B.13}$$

In fact, this formulation was characterized by John C. Baez, who uses Kan complexes as definition of ∞-groupoids [15]. This is also one of the definitions of ∞-groupoids where hypothesis B.6.1 is actually a proven theorem. A theorem on which we will fundament the functorial formulation of our autoencoder. For our concerns, Π_∞ then is actually a functor between categories, that is obtained via the Quillen adjunction in B.2.

However, originally, Grothendieck gave another construction of ∞-groupoids in his letter to Daniel Quillen in 1983, which constitutes the take-off of his famous work "À la poursuite des Champs" [77]. In fact, in a letter to Tim Porter, Grothendieck claimed, that the definition of ∞-groupoids over Kan complexes, "does not in any way meet with what [he] [was] really looking for" [25]. In fact, Grothendieck used a globular understanding of combinatorics instead of relying on simplexes as within the Kan formulation. He conjectured, that

Conjecture B.6.2 (Grothendieck's Conjecture (weak form))**.** "For every Gr-coherator $\mathcal{C}$, the localization of the category of ∞-$\mathcal{C}$-groupoids by the ∞-equivalences is equivalent to the homotopy category of CW-complexes" [25].

The definition, construction and homotopy theory of ∞-gropoids is still a subject of ongoing research ([31], [41], [25]), but shall not be our concern due to our application driven restricted usage of hypothesis B.6.1.

Bibliography

[1] Paul Alexandroff and Heinz Hopf. *Topologie*. Die Grundlehren der mathematischen Wissenschaften. Springer Verlag, 1935.

[2] David Blackwell. "Equivalent Comparisons of Experiments". In: *The Annals of Mathematical Statistics* 24.2 (1953), pp. 265–272. ISSN: 00034851. URL: http://www.jstor.org/stable/2236332.

[3] W.G. Dwyer and D.M. Kan. "Simplicial Localizations of Categories". In: *Journal of PUre and Applied Algebra* 17 (1980), pp. 267–284.

[4] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by David Rand and Lai-Sang Young. Berlin, Heidelberg: Springer, 1981, pp. 366–381.

[5] Klaus Jaenich. *Topology*. Undergraduate Texts in Mathematics. Springer Verlag, 1984. ISBN: 0-387-90892-7.

[6] R.L. Graham, M. Grötschel, and L. Lovasz, eds. *Handbook of Combinatorics. Volume 2*. New York, Amsterdam, Lausanne: Elsevier, 1995.

[7] *Novikov Conjectures, Index Theorems, and Rigidity*. Vol. 2. London Mathematical Society Lecture Note Series. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511629365.

[8] L. Le Cam. "Comparison of Experiments: A Short Review". In: *Lecture Notes-Monograph Series* 30 (1996), pp. 127–138. ISSN: 07492170. URL: http://www.jstor.org/stable/4355942.

[9] Allen Hatcher. *Algebraic Topology*. Cambridge: Cambridge University Press, 2002.

[10] Jon Kleinberg. "An Impossibility Theorem for Clustering". In: *Adv Neural Inform Process Syst (NIPS)* 15 (Jan. 2003).

[11] M. J. Healy and T. Caudell. "Neural Networks, Knowledge and Cognition: A Mathematical Semantic Model Based upon Category Theory". In: 2004.

[12] Vin de Silva and Gunnar Carlsson. "Topological estimation using witness complexes". In: *SPBG'04 Symposium on Point - Based Graphics 2004*. Ed. by Markus Gross et al. The Eurographics Association, 2004. ISBN: 3-905673-09-6. DOI: 10.2312/SPBG/SPBG04/157-166.

[13] Ibrahim Assem, Andrzej Skowronski, and Daniel Simson. *Elements of the representation theory of associative algebras. Volume 1. Techniques of Representation Theory*. Cambridge: Cambridge University Press, 2006.

[14] Facundo Memoli and Guillermo Sapiro. *Computing with Point Cloud Data*. Ed. by Hamid Krim and Anthony Jr. Yezzy. Boston, Basel and Berlin: Birkhäuser Boston, 2006.

[15] John C. Baez. *The Homotopy Hypothesis*. Jan. 2007.

[16] Vin de Silva. "A weak characterisation of the Delaunay triangulation". In: *Geometriae Dedicata* 135 (2008), pp. 39–64.

[17] Paul G. Goerss and John F. Jardine. *Simplicial Homotopy Theory*. Berlin, Heidelberg: Springer, 2009.

[18] Payam Refaeilzadeh, Lei Tang, and Huan Liu. "Cross-Validation". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 532–538. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_565. URL: https://doi.org/10.1007/978-0-387-39940-9_565.

[19] David I. Spivak. "METRIC REALIZATION OF FUZZY SIMPLICIAL SETS". In: 2009.

[20] Martin Arkowitz. *Introduction to Homotopy Theory*. Springer Verlag, 2010.

[21] Jean-Daniel Boissonnat and Arijit Ghosh. "Manifold Reconstruction Using Tangential Delaunay Complexes". In: *Discrete and Computational Geometry* 51 (June 2010). DOI: 10.1145/1810959.1811013.

[22] Gunnar Carlsson and Facundo Memoli. *Classifying Clustering Schemes*. 2010. arXiv: 1011.5270 [stat.ML].

[23] Erin Chambers et al. "Vietoris–Rips Complexes of Planar Point Sets". In: *Discrete Computational Geometry* 44 (July 2010), pp. 75–90. DOI: 10.1007/s00454-009-9209-8.

[24] Herbert Edelsbrunner and John L. Harer. *Computational Topology. An Introduction*. Rhode Island USA: American Mathematical Society, 2010.

[25] Georges Maltsiniotis. *Grothendieck ∞-groupoids, and still another definition of ∞-categories*. 2010. arXiv: 1009.2331 [math.CT].

[26] Bob Coecke and Robert W. Spekkens. "Picturing classical and quantum Bayesian inference". In: *Synthese* 186.3 (Mar. 2011), pp. 651–696. ISSN: 1573-0964. DOI: 10.1007/s11229-011-9917-5. URL: http://dx.doi.org/10.1007/s11229-011-9917-5.

[27] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2121–2159.

[28] Yoshua Bengio. *Practical recommendations for gradient-based training of deep architectures*. 2012. arXiv: 1206.5533 [cs.LG].

[29] Brendan Fong. "Causal Theories: A Categorical Perspective on Bayesian Networks". In: *arXiv: Probability* (2012).

[30] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: 1212.5701 [cs.LG].

[31] Dimitri Ara. "On the homotopy theory of Grothendieck -groupoids". In: *Journal of Pure and Applied Algebra* 217.7 (July 2013), pp. 1237–1278. ISSN: 0022-4049. DOI: 10.1016/j.jpaa.2012.10.010. URL: http://dx.doi.org/10.1016/j.jpaa.2012.10.010.

[32] Dominique Attali, André Lieutier, and David Salinas. "Vietoris-Rips Complexes also Provide Topologically Correct Reconstructions of Sampled Shapes". In: *27th Annual Symposium on Computational Geometry* 46 (May 2013). DOI: 10.1145/1998196.1998276.

[33] "Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky". In: ed. by Bob Coecke, Luke Ong, and Prakash Panangaden. Springer, 2013. Chap. The Algebra of Directed Acyclic Graphs.

[34] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. *Testing the Manifold Hypothesis*. 2013. arXiv: 1310.0425 [math.ST].

[35] Robert Ghrist. *Elementary Applied Topology, ed. 1.0*. Createspace, 2014.

[36] Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Waltman, USA: Elsevier, 2014.

[37] Clément Maria et al. "The Gudhi Library: Simplicial Complexes and Persistent Homology". In: *ICMS*. 2014.

[38] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[39] Aleksandar Botev, Guy Lever, and David Barber. *Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent*. 2016. arXiv: 1607.01981 [stat.ML].

[40] Jared Culbertson et al. *Consistency constraints for overlapping data clustering*. 2016. arXiv: 1608.04331 [cs.LG].

[41] Simon Henry. *Algebraic models of homotopy types and the homotopy hypothesis*. 2016. arXiv: 1609.04622 [math.CT].

[42] Max Karoubi and Charles Weibel. *On the covering type of a space*. 2016. arXiv: 1612.00532 [math.AT].

[43] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[44] Behnam Neyshabur et al. *Exploring Generalization in Deep Learning*. 2017. arXiv: 1706.08947 [cs.LG].

[45] Christian Rüschhoff. *Lecture notes: Simplicial Sets*. 2017.

[46] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: 1712.09913 [cs.LG].

[47] Bastian Rieck et al. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *CoRR* abs/1812.09764 (2018). arXiv: 1812.09764. URL: http://arxiv.org/abs/1812.09764.

[48] Kenta Cho and Bart Jacobs. "Disintegration and Bayesian inversion via string diagrams". In: *Mathematical Structures in Computer Science* 29.7 (Mar. 2019), pp. 938–971. ISSN: 1469-8072. DOI: 10.1017/s0960129518000488. URL: http://dx.doi.org/10.1017/S0960129518000488.

[49] Samir Chowdhury et al. "Path Homologies of Deep Feedforward Networks". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (Dec. 2019). DOI: 10.1109/icmla.2019.00181. URL: http://dx.doi.org/10.1109/ICMLA.2019.00181.

[50] Matthias Feurer and Frank Hutter. "Hyperparamter Optimization". In: *Automated Machine Learning*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Springer, 2019. Chap. Hyperparameter Optimization, pp. 3–35.

[51] Brendan Fong and Michael Johnson. "Lenses and Learners". In: *CoRR* abs/1903.03671 (2019). arXiv: 1903.03671. URL: http://arxiv.org/abs/1903.03671.

[52] Brendan Fong, David Spivak, and Remy Tuyeras. "Backprop as Functor: A compositional perspective on supervised learning". In: (June 2019).

[53] Jonathan Frankle and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. arXiv: 1803.03635 [cs.LG].

[54] Thomas Gebhart, Paul Schrater, and Alan Hylton. *Characterizing the Shape of Activation Space in Deep Neural Networks*. 2019. arXiv: 1901.09496 [cs.LG].

[55] Soheil Rostami, Walid Saad, and Choong Seon Hong. *Deep Learning with Persistent Homology for Orbital Angular Momentum (OAM) Decoding*. 2019. arXiv: 1911.06858 [eess.SP].

[56] Nathaniel Saul. *scikit-tda*. 2019. URL: https://scikit-tda.org/ (visited on 02/10/2022).

[57] Tobias Fritz. "A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics". In: *Advances in Mathematics* 370 (Aug. 2020), p. 107239. ISSN: 0001-8708. DOI: 10.1016/j.aim.2020.107239. URL: http://dx.doi.org/10.1016/j.aim.2020.107239.

[58] Tobias Fritz et al. "Representable Markov Categories and Comparison of Statistical Experiments in Categorical Probability". In: *ArXiv* abs/2010.07416 (2020).

[59] Kwangho Kim et al. "Efficient Topological Layer based on Persistent Landscapes". In: *CoRR* abs/2002.02778 (2020). arXiv: 2002.02778. URL: https://arxiv.org/abs/2002.02778.

[60] Yuri I. Manin and Matilde Marcolli. "Homotopy Theoretic and Categorical Models of Neural Information Networks". In: *CoRR* abs/2006.15136 (2020). arXiv: 2006.15136. URL: https://arxiv.org/abs/2006.15136.

[61] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[62] Birgit Richter. *From Categories to Homotopy Theory*. Cambridge: Cambridge University Press, 2020.

[63] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306.

[64] Dan Shiebler. "Functorial Clustering via Simplicial Complexes". In: *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*. 2020. URL: https://openreview.net/forum?id=ZkDLcXCP5sV.

[65] Toby St Clere Smithe. "Bayesian Updates Compose Optically". In: *arXiv: Category Theory* (2020).

[66] Mary Adkisson et al. *Autoencoder-based Anomaly Detection in Smart Farming Ecosystem*. 2021. arXiv: 2111.00099 [cs.CR].

[67] Stephon Alexander et al. *The Physics of Machine Learning: An Intuitive Introduction for the Physical Scientist*. 2021. arXiv: 2112.00851 [cond-mat.dis-nn].

[68] Marco Antonio Armenta and Pierre-Marc Jodoin. *The Representation Theory of Neural Networks*. 2021. arXiv: 2007.12213 [cs.LG].

[69] Scott Balchin. *A Handbook of Model Categories*. Cham, Switzerland: Springer Nature Switzerland AG, 2021.

[70] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2021. arXiv: 2003.05991 [cs.LG].

[71] Tolga Birdal et al. *Intrinsic Dimension, Persistent Homology and Generalization in Neural Networks*. 2021. arXiv: 2111.13171 [cs.LG].

[72] Florencia Canelli et al. *Autoencoders for Semivisible Jet Detection*. 2021. arXiv: 2112.02864 [hep-ph].

[73] S. V. Chekanov and W. Hopkins. *Event-based anomaly detection for new physics searches at the LHC using machine learning*. 2021. arXiv: 2111.12119 [hep-ph].

[74] Geoff S. H. Cruttwell et al. "Categorical Foundations of Gradient-Based Learning". In: *CoRR* abs/2103.01931 (2021). arXiv: 2103.01931. URL: https://arxiv.org/abs/2103.01931.

[75] David Pérez Fernández et al. "Determining Structural Properties of Artificial Neural Networks Using Algebraic Topology". In: *CoRR* abs/2101.07752 (2021). arXiv: 2101.07752. URL: https://arxiv.org/abs/2101.07752.

[76] Greg Friedman. *An elementary illustrated introduction to simplicial sets*. 2021. arXiv: 0809.4221 [math.AT].

[77] Alexander Grothendieck. *Pursuing Stacks*. 2021. arXiv: 2111.01000 [math.CT].

[78] Mustafa Hajij and Kyle Istvan. *A Topological Framework for Deep Learning*. 2021. arXiv: 2008.13697 [cs.LG].

[79] Mustafa Hajij, Ghada Zamzmi, and Fawwaz Batayneh. *TDA-Net: Fusion of Persistent Homology and Deep Learning Features for COVID-19 Detection in Chest X-Ray Images*. 2021. arXiv: 2101.08398 [cs.CV].

[80] Gregory Henselman-Petrusek. *Eirene.jl*. 2021. URL: https://github.com/Eetion/Eirene.jl (visited on 02/10/2022).

[81] Christoph J. Hönes et al. *Automatically detecting anomalous exoplanet transits*. 2021. arXiv: 2111.08679 [cs.LG].

[82] Michael Moor et al. *Topological Autoencoders*. 2021. arXiv: 1906.00722 [cs.LG].

[83] Sheetal Rajpal et al. *Deep Learning Based Model for Breast Cancer Subtype Classification*. 2021. arXiv: 2111.03923 [cs.LG].

[84] A. M. Samarakoon et al. *Integration of Machine Learning with Neutron Scattering: Hamiltonian Tuning in Spin Ice with Pressure*. 2021. arXiv: 2110.15817 [cond-mat.other].

[85] Dan Shiebler. "Categorical Stochastic Processes and Likelihood". In: *Compositionality* 3 (Apr. 2021), p. 1. ISSN: 2631-4444. DOI: 10.32408/compositionality-3-1. URL: http://dx.doi.org/10.32408/compositionality-3-1.

[86] Dan Shiebler, Bruno Gavranovic, and Paul W. Wilson. "Category Theory in Machine Learning". In: *CoRR* abs/2106.07032 (2021). arXiv: 2106.07032. URL: https://arxiv.org/abs/2106.07032.

[87] Umut Şimşekli et al. "Hausdorff dimension, heavy tails, and generalization in neural networks*". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (Dec. 2021), p. 124014. ISSN: 1742-5468. DOI: 10.1088/1742-5468/ac3ae7. URL: http://dx.doi.org/10.1088/1742-5468/ac3ae7.

[88] Guillaume Tauzin et al. *giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration*. 2021. arXiv: 2004.02551 [cs.LG].

[89] Žiga Virk. "Rips Complexes as Nerves and a Functorial Dowker-Nerve Diagram". In: *Mediterranean Journal of Mathematics* 18.2 (Feb. 2021). ISSN: 1660-5454. DOI: 10.1007/s00009-021-01699-4. URL: http://dx.doi.org/10.1007/s00009-021-01699-4.

[90] Satoru Watanabe and Hayato Yamana. "Overfitting Measurement of Deep Neural Networks Using No Data". In: *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. 2021, pp. 1–10. DOI: 10.1109/DSAA53316.2021.9564119.

[91] Satoru Watanabe and Hayato Yamana. "Topological Measurement of Deep Neural Networks Using Persistent Homology". In: *CoRR* abs/2106.03016 (2021). arXiv: 2106.03016. URL: https://arxiv.org/abs/2106.03016.

[92] Matthew Wheeler, Jose Bouza, and Peter Bubenik. *Activation Landscapes as a Topological Summary of Neural Network Performance*. 2021. arXiv: 2110.10136 [cs.LG].

[93] Sharon Zhou et al. *Evaluating the Disentanglement of Deep Generative Models through Manifold Topology*. 2021. arXiv: 2006.03680 [stat.ML].

[94] Dmitriy Morozov. *Dionysus 2*. 2022. URL: https://mrzv.org/software/dionysus2/ (visited on 02/10/2022).

[95] nLab authors. *simplicial localization*. http://nlab-pages.s3.us-east-2.amazonaws.com/nlab/show/simplicial+localization. Jan. 2022.

[96] Yann LeCun and Corinna Cortes and Christopher J. C. Burges. *The MNIST Database*. http://yann.lecun.com/exdb/mnist/. Jan. 2022.

[97] *Kerodon.net: An online resource for homotopy-coherent matheatics*.

[98] *Loss Landscape. Exploring the Landscape*. https://losslandscape.com/. Accessed: 2022-02-20.