



**HOCHSCHULE  
MITTWEIDA**  
University of Applied Sciences

---


# MASTER THESIS

---

Mr.  
**Thomas Davies**, B.Eng., B.Sc.

**Analysis of Attention Learning  
Schemes and the Design of an  
Attention Integration into Learning  
Vector Quantization**

Mittweida, August 2023



Faculty of **Applied Computer Sciences and Biosciences**

---

## **MASTER THESIS**

---

# **Analysis of Attention Learning Schemes and the Design of an Attention Integration into Learning Vector Quantization**

Author:

**Thomas Davies**

Course of Study:

Applied Mathematics

Seminar Group:

MA21w1-M

First Examiner:

Prof. Dr. rer. nat. habil. Thomas Villmann

Second Examiner:

Dr. David Nebel

Submission:

Mittweida, 20.08.2023

Defense/Evaluation:

Mittweida, 2023

Faculty of **Applied Computer Sciences and Biosciences**

---

# **MASTER THESIS**

---

## **Learning Timeseries Data with Attention and Learning Vector Quantization**

Author:

**Thomas Davies**

Course of Study:

Applied Mathematics

Seminar Group:

MA21w1-M

First Examiner:

Prof. Dr. rer. nat. habil. Thomas Villmann

Second Examiner:

Dr. David Nebel

Submission:

Mittweida, 20.08.2023

Defense/Evaluation:

Mittweida, 2023

## **Bibliographic Description:**

Davies, Thomas:

Analysis of Attention Learning Schemes and the Design of an Attention Integration into Learning Vector Quantization. – 2023. – 59 S.

Mittweida, Hochschule Mittweida – University of Applied Sciences, Faculty of Applied Computer Sciences and Biosciences, Master Thesis, 2023.

## **Abstract:**

Machine learning models for timeseries have always been a special topic of interest due to their unique data structure. Recently, the introduction of attention improved the capabilities of recurrent neural networks and transformers with respect to their learning tasks such as machine translation. However, these models are usually subsymbolic architectures, making their inner working hard to interpret without comprehensive tools. In contrast, interpretable models such learning vector quantization are more transparent in the ability to interpret their decision process. This thesis tries to merge attention as a machine learning function with learning vector quantization to better handle timeseries data. A design on such a model is proposed and tested with a dataset used in connection with the attention based transformers. Although the proposed model did not yield the expected results, this work outlines improvements for further research on this approach.

# Contents

<b>Contents</b>	<b>I</b>
<b>Additional Lists</b>	<b>II</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Preliminaries</b>	<b>3</b>
2.1 Machine Learning Concepts . . . . .	3
2.2 Mathematical Concepts . . . . .	11
<b>3 Attention</b>	<b>16</b>
3.1 Neuroscientific Motivation . . . . .	16
3.2 Computational Attention . . . . .	21
3.2.1 Prerequisites . . . . .	22
3.2.2 Attention Model . . . . .	25
3.2.3 Generalized Attention . . . . .	27
3.2.4 Attention Taxonomy . . . . .	34
<b>4 Learning Vector Quantization</b>	<b>38</b>
4.1 Introduction . . . . .	38
4.2 Generalized Learning Vector Quantization . . . . .	39
4.3 Generalized Matrix Learning Vector Quantization . . . . .	41
<b>5 Attention Learning Vector Quantization</b>	<b>43</b>
5.1 Post Attention Downsampled Learning Vector Quantization . . . . .	44
5.2 Pre Attention Downsampled Learning Vector Quantization . . . . .	47
<b>6 Experiments</b>	<b>49</b>
6.1 Dataset . . . . .	49
6.2 Hardware System . . . . .	49
6.3 Data Preparation . . . . .	49
6.4 Experiment Observations . . . . .	54
6.5 Conclusion . . . . .	55
<b>7 summary</b>	<b>59</b>
<b>Bibliography</b>	<b>60</b>
<b>Eidesstattliche Erklärung</b>	<b>67</b>

# Additional Lists

## List of Figures

2.1	Example with $\vec{a} = (1, 3) \in V$ and $\vec{b} = (3, 1) \in V$ and their respective points $a, b \in L = V$ and the vector $\vec{ab} \in V$ . . . . .	12
2.2	$n$ -planes in $\mathbb{R}^2$ . . . . .	14
2.3	Illustration of Simplices . . . . .	15
3.1	Yerkes-Dodson Curve [36] . . . . .	17
3.2	Illustration of the perception of the features <i>color</i> and <i>orientation</i> and how they are processed distinctively. Each feature is associated on a master map to a location [43]. . . . .	20
3.3	Attention Experiment . . . . .	21
3.4	Example 3.13 illustrated in two different principles. The compact illustration displays the recurrence as a memory cell. The unrolled illustration on the other hand shows the information flow over time, where each sequence element $x_i$ is inserted separately. . . . .	24
3.5	Attention Encoder-Decoder Illustration according to 'Attention Based Encoder-Decoder RNN' on page 26 . . . . .	27
3.6	Attention implementation into BRNN according to [2] . . . . .	28
3.7	Illustration of a basic attention mechanism having some query $q$ and scoring it against keys $k_i$ with the alignment function $a(q, k_i)$ . The result is an alignment value $a_i$ . . . . .	31
3.8	Illustration showing the transition of distribution functions with different $\alpha$ parameters. Taken from [20] . . . . .	33
3.9	In the transformer <i>multi-level attention</i> and <i>multi-representational attention</i> can be observed. Since each encoder or decoder layer are stacked $N \times$ times, attention is computed at different abstraction layers, hence <i>multi-level attention</i> . Each encoder or decoder layer makes use of <i>multi-head attention</i> which is categorized as <i>multi-representational attention</i> . Figure is taken from [3]. . . . .	35
3.10	Multi-head attention is displayed. Different layers of <i>Linear</i> transformation and <i>Scaled Dot-Product Attention</i> can be seen. Each layer forms an attention-head performing attention calculations independently. Figure is taken from [3]. . . . .	37
4.1	Attraction and repulsion visualization with LVQ. The randomly chosen data sample $x$ and the closest prototype $w$ are highlighted in yellow. . . . .	39
5.1	Illustration of the processing steps according to 5.3 . . . . .	45
5.2	Processing steps in Post Attention Downsampled Learning Vector Quantization. On the left-hand side the different layers are displayed which have been explained in the according definitions. On the right-hand side some exemplary data illustration with dimensionality highlighting is showed and how data is transformed during the transformation process. . . . .	47
6.1	Overview of Given Class Distributions . . . . .	50
6.2	FFT of All Feature Timeseries from the Aeroelastic Simulations of Wind Turbines Dataset at Full Resolution . . . . .	52

6.3	FFT of All Feature Timeseries from the Aeroelastic Simulations of Wind Turbines Dataset at $\frac{1}{6}$ Resolution . . . . .	53
6.4	Training Metrics . . . . .	54
6.5	Transformer setup in [84] with the special <i>time-window</i> embedding. . . . .	58

## List of Tables

2.1	Exemplary Activation Functions . . . . .	4
6.1	Sizes of an Exemplary Model Configuration with GMLVQ used in the LVQ Layer . .	51
6.2	Experiment Configurations . . . . .	55

## Acronyms

<b>AM</b>	Attention Model
<b>BRNN</b>	bidirectional recurrent neural network
<b>CNN</b>	Convolutional neural networks
<b>COMASA</b>	COmpact Multiple Alignment for Sequence Averaging
<b>CPU</b>	central processing unit
<b>DTW</b>	Dynamic Time Warping
<b>FFT</b>	Fast Fourier Transform
<b>GB</b>	gigabytes
<b>GLVQ</b>	Generalized Learning Vector Quantization
<b>GMLVQ</b>	Generalized Matrix Learning Vector Quantization
<b>GPU</b>	graphics processing unit
<b>LSTM</b>	long short term memory
<b>LVQ</b>	Learning Vector Quantization
<b>NLP</b>	natural language processing
<b>RNN</b>	Recurrent Neural Networks
<b>SGD</b>	Stochastic Gradient Descent
<b>TB</b>	terabytes
<b>VQA</b>	visual question answering
<b>WTA</b>	Winner Takes All

# 1 Introduction

Attention has gained a lot of popularity in recent years since its introduction in 2014 [1, 2]. Especially since the introduction of the transformer networks [3], which utilize attention in their layers, attention has become a viable component in the processing of sequential data. Learning sequential data in conjunction with [Learning Vector Quantization \(LVQ\)](#) learning schemes is a special topic of interest because of the special data structure sequential data provides. Attempts have been in the past to combine learning on sequential data together with [LVQ](#) based algorithms, utilizing techniques such as using recurrent processing [4] or special distance measures embedded into [LVQ](#) [5].

This thesis aims to integrate attention based learning with [LVQ](#). The objective is to keep processing as straightforward as possible to retain interpretability, or at least, to not complicate the interpretation of such a model. Therefore, attention will be discussed in great depth to gain a comprehensive understanding of its vast variety of existing designs. Based on this understanding, a design will be proposed to be set up in conjunction with [LVQ](#).

Preliminary knowledge will be provided in [Preliminaries](#). This includes basic information on topics discussed in this thesis, covered in [Machine Learning Concepts](#) and [Mathematical Concepts](#). When appropriate, these basics will be referenced to facilitate easy navigation for the reader through the thesis. Knowledgeable readers may skip [Preliminaries](#) and proceed directly to the next chapter.

In upcoming chapter, attention will be explored in great depth, as previously mentioned. First, attention will be analyzed from a neurological and psychological (neuroscientific) perspective. Given other models have been inspired by biological models in the past, this section will start by analyzing the potential connection between recent [Attention Model \(AM\)](#) and the biological background. Subsequently, attention will be discussed in regard to its application in machine learning. Attention is probably most known for its use in transformer models. To fully grasp the origins and purpose of attention, an evolutionary presentation into [Recurrent Neural Networks \(RNN\)](#) models and the development of attention will be presented. Having built an introductory understanding on attention and its fundamentals, attention will be presented in its generalized form. There, all parameters will be defined and presented in all its variations. A comprehensive review of attention and its different designs will be presented. The attention chapter will be concluded with a taxonomy on attention.

Next, [LVQ](#) will be explained, along its variations: [Generalized Learning Vector Quantization \(GLVQ\)](#) and [Generalized Matrix Learning Vector Quantization \(GMLVQ\)](#). With all these fundamentals brought together, different variations of an attention-based [LVQ](#) model will be introduced. The proposed model design will be composed out of an attention layer, a down-sampling layer and [LVQ](#) layer forming the backend layer. Various model configurations will be tested with different hyperparameter settings in the experiments chapter. Results and potential enhancements in the model's design will also be discussed there.



Note that throughout this thesis, vectors will be presented boldfaced in lowercase letters, i.e.  $\mathbf{x}$ ,  $\boldsymbol{\alpha}$ , and matrices will be boldfaced in capital letters, i.e.  $\mathbf{A}$ ,  $\boldsymbol{\Omega}$ . Sequences will usually be written as  $(x_i)_{i=1}^n$  for a sequence of length  $n$  beginning with an element at  $i = 1$ <sup>1</sup>. The scalar product of two vectors  $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$  is denoted by  $\mathbf{x}_1^T \mathbf{x}_2$ . Due to the wide range of topics introduced in this thesis, some of will receive only a brief explanation. Interested readers can consult the provided sources for more in-depth information.

---

<sup>1</sup>In the chapter [Conclusion](#) under *Input Dimensionality and Embedding Representation Dynamics*, a sequence will be given as a lowercase lettered matrix  $\mathbf{x}$  to use the notation of the referenced paper.

## 2 Preliminaries

This chapter introduces concepts and definitions which are helpful to get a deeper understanding on upcoming topics. These might be already known to an experienced reader. Therefore, it is not strictly necessary to read this chapter to understand the main topic of this thesis. These preliminaries will be referenced, such that in doubt the reader will be able to backtrack definitions and to get a deeper understanding on the discussed topics. The preliminaries include *machine learning concepts* and *mathematical concepts*. Note, these topics are deeply intermingled since they associate on a wide scale.

### 2.1 Machine Learning Concepts

#### Neural Networks

An artificial neuron is a computing unit inspired by its biological counterpart. For the sake of brevity it will be referred as *neuron*. It takes some weighted input and outputs a scalar value. Usually, a greater number of such neurons are arranged in a way such that they are capable of solving learning tasks. Such constructions are called *neural networks*.

**Definition 2.1** (Neuron)

Let  $\mathbf{x} \in \mathbb{R}^n$  be some input to the neuron [6, p.10] and  $\mathbf{w} \in \mathbb{R}^n$  associated *synaptic weights*. Denote the bias as  $b \in \mathbb{R}$ . The pre-activation computation  $u$  is calculated as

$$\begin{aligned} u &= \sum_{i=1}^n x_i w_i + b \\ &= \mathbf{w}^T \mathbf{x} + b \end{aligned}$$

The resulting calculation is then fed into the *activation function* yielding the output  $y$  of the neuron.

$$y = f(u)$$

When speaking about activation functions usually *non-linear activation functions* are considered. The usage of linear activation functions is only able to separate linearly separable data for any number of used neurons [7, p.168][8]. Further, by Cybenko's theorem it was also shown, that the usage of non-linear activation functions being discriminatory and sigmoidal (monotonically increasing and bounded) can approximate any continuous function with a single hidden layer of finite neurons arbitrarily well [9, 10].

**Definition 2.2** (Activation Function)

An activation function  $f$  is a monotonically increasing mapping

$$f : \mathbb{R} \rightarrow U \subseteq \mathbb{R}$$

Its role is to add non-linearity to solve learning tasks on non-linearly separable data [11].

**Remark 2.3**

If  $U$  is bounded such activation functions are called *squashing functions* [11].

**Example 2.4**

Table 2.1 shows widely known activation functions. More examples and a comprehensive taxonomy can be found in [8, 11].

Name	Function	Range $U \in \mathbb{R}$
Identity	$\text{id}(a) = a$	$(-\infty, \infty)$
Heaviside	$H(a) = \begin{cases} 0 & \text{if } a < 0 \\ 1 & \text{else} \end{cases}$	$\{0, 1\}$
Bipolar	$B(a) = \begin{cases} -1 & \text{if } a < 0 \\ 1 & \text{else} \end{cases}$	$\{-1, 1\}$
Sigmoid	$\sigma(a) = \frac{1}{1+\exp(-a)}$	$(0, 1)$
Hyperbolic Tangent	$\tanh(a) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp-x}$	$(-1, 1)$
Rectified Linear Unit (ReLU)	$\text{ReLU}(x) = \max(0, x)$	$(0, 1)$
Swish	$\text{Swish}(x) = \frac{x}{1+\exp(-\beta x)}$	$(-\infty, \infty)$

**Table 2.1:** Exemplary Activation Functions

**Property 2.5**

Activation functions should hold the following properties to learn non-linear and abstract features [8].

1. Non-linearity
2. Computational Efficiency
3. Smoothness/Differentiability
4. Statistical preservation

## Feedforward Network

Feedforward networks are neural networks where information flows only in one direction. A data point  $x \in \mathbb{R}$  is fed into an input layer of neurons and the computed information flows into succeeding layers, until an output is calculated. The layer, which processes the input, is called the *input layer*. Additionally, the layer, which computes the output, is called the *output layer*, while all the intermediate layers, are called the *hidden layers* [7, pp.163].

## Convolutional Neural Network

Convolutional neural networks (CNN) are feedforward networks, which are widely used in pattern recognition learning tasks. Often these pattern recognition tasks are performed on image data. The special mechanism, they apply, is creating feature extraction maps in a convolutional layer. Convolution is the process of sliding a filter over the pattern data, creating a convolved output. This data is then averaged and subsampled. These processing steps make the pattern recognition insensitive to shifts and distortions [6, p.201].

## Supervised and Unsupervised Learning

In *supervised learning* tasks the dataset is presented in such a way, that each datasample is associated with either a *labeled* or a *target* (classification and regression respectively). The learning scheme is centered around finding a function  $f$ , which estimates  $f(x) = \hat{y}$  with  $\hat{y}$  being the prediction and  $y$  being either the target or class associated with the datasample  $x$  [7, p.102].

*Unsupervised learning*, on the other hand, does not make use of labels, such that these learning algorithms receive only the data  $x \in X$  as input. In this learning scheme the objective is to learn either the data distribution or to cluster the complete dataset into different clusters with similar features [7, p.102].

## Regression

Regression can be considered as a specific type of supervised learning task. A regression model is a mapping  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  for some data sample  $x \in \mathbb{R}^n$ . The predicted value is a scalar  $\hat{y} \in \mathbb{R}$  [7, p.98]. In such learning tasks, we can consider the given data as a set of random variables, where the random variable, to be predicted, is called the *response*. The response is denoted as  $\hat{y}$ . The remaining random variables are considered independent and are called *regressors*. These relate to the features of the data space  $X \subseteq \mathbb{R}^n$  and the input variables  $x \in X$  themselves [6, p.68]. Note, though we can consider the data as a representation of random variables, the values which we learn on are fixed observations.

## Classification

Similar to [Regression](#), a classification is a specific type of learning task. Classification can be learned in a supervised and unsupervised learning scheme. Instead of estimating a value on some given input, the model predicts a class for such input. Let  $\mathbf{x} \in \mathbb{R}^n$  some labeled data sample with  $C$  different labels. A prediction model  $f : \mathbb{R}^n \rightarrow [0, 1]^C$  returns a distribution of probabilities over all given classes  $\{1, \dots, C\}$  [12, p.179]. Models making predictions on class affiliations are also known as *classifiers*. A label is an affiliation of the data sample  $\mathbf{x}$  to a class. The indication function  $c : \mathbb{R}^n \rightarrow \{1, \dots, C\}$  returns the predefined label of some labeled data sample, i.e. the label of  $\mathbf{x}$  is given as  $c(\mathbf{x})$ . The component, with the highest probability returned by the probabilistic prediction model, will represent the given prediction by the model.

Other classifiers can also be defined as  $f : \mathbb{R}^n \rightarrow \{1, \dots, C\}$ , where the classifier returns only the predicted class instead of a probability distribution, hence a correct prediction  $f(\mathbf{x})$  for some arbitrary data sample  $\mathbf{x} \in \mathbb{R}^n$  is given as  $f(\mathbf{x}) = c(\mathbf{x})$  [7, p.97].

## Kernel

A kernel, denoted as  $K : U \times U \rightarrow R$ , defines an inner product within a feature space  $F$ , with the important property that there exists a mapping function  $\phi : U \rightarrow F$ . What makes this property significant is that the kernel value  $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = \phi(\mathbf{x})^T \phi(\mathbf{y})$  can be computed directly within the data space, without requiring explicit knowledge of the mapping to the feature space. Kernels find extensive use in machine learning, particularly for addressing non-linear problems using linear classifiers. They achieve this by transforming the input data into higher-dimensional spaces where it becomes linearly separable. The function  $\phi$  responsible for mapping data into higher-dimensional spaces is referred to as feature map, and the resulting space is known as the feature space. Importantly, these feature maps are inherently non-linear in nature [7, p.137][13].

## Training, Test and Validation sets

To be able to generalize on a dataset with a given learning task, we need to prepare the given data in various ways. One key preparation step is to properly split the dataset into *training*, *test* and *validation sets*. A *training set* is used to adjust parameters for a given machine learning model. Here, the objective is to create a model which is able to generalize from the given training set with respect to the learning task. The *test set* is used to check the quality of the model's ability to generalize on the given learning task [12, p.2-3]. If the model performs poorly on the test set, but very well on the training set, the model is said to *over-fit* [12, p.6]. The more complex, versatile and large the training set is and the better it represents the complexity of the learning task, the more over-fitting can be avoided [12, p.9]. The *validation set* is taken from the training set and is used to compare the model's performance with different hyperparameter configurations. Hyperparameters are pre-configurable parameters in machine learning models [12, p.32].

## Stochastic Gradient Descent

**Stochastic Gradient Descent (SGD)** is a special gradient descent learning scheme for some machine learning model  $f \in \mathbb{F}$ . Gradient descent is an iterative optimization method, where weight vectors  $\mathbf{w} \in \mathbb{R}^n$  at time  $t$  are updated based on a calculated gradient of the given weight vector and the training set with respect to an overall error function  $E$  [14, pp.421]. The error function quantifies the regression or prediction error of the model with respect to its inputs. Further, the error function is required to be differentiable. The overall loss is described over the local losses  $l$  of the given data samples. The error function is given as

$$E(f, X, W) = \frac{1}{n} \sum_{i=1}^n l(f(\mathbf{x}_i), y_i) \quad (\text{Discrete Case})$$

$$= \int_{\mathbf{x}_i \in X} l(f(\mathbf{x}_i), y_i) dP(z) \quad (\text{Continuous Case})$$

with

$f$	... machine learning model
$\mathbf{x} \in X \subseteq \mathbb{R}^n$	... data sample taken from training set
$y \in Y \subseteq \mathbb{R}$	... label or regression value associated with a data sample
$W$	... set of learnable parameters
$l$	... local loss function
$P(z)$	... data distribution
$z$	... a pair of data sample and label $(\mathbf{x}, y)$

The objective is to minimize the loss function by using the gradient to update the weight vectors. The update for some learnable parameter  $\mathbf{w} \in W$  at time  $t + 1$  is given as

$$\mathbf{w}(t + 1) = \mathbf{w}(t) - \alpha \frac{\partial l(X, W)}{\partial \mathbf{w}(t)}$$

The hyperparameter  $\alpha > 0$  denotes the learning rate and the size of the training set is given as  $|X| = n$ . If the learning rate is properly chosen according to the magnitude of the gradient, convergence can be guaranteed [15][16, pp.30].

A properly sized training set is required to properly generalize on a given learning task. With a growing training set, the computational complexity of the gradient descent method grows according to the model's computational complexity. **SGD** is a simplification of the gradient descent method by taking a random sample  $\mathbf{x} \in \mathbb{R}^n$  from the training set and estimating the overall gradient [7, p.147-148][16, p.422]. Here, instead of only a single sample, also a mini batch  $\mathbb{B} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$  with  $m$  being the batch size can be used. The update for learnable parameter  $\mathbf{w}$  for **SGD** is given as

$$\mathbf{w}(t + 1) = \mathbf{w}(t) - \alpha \frac{\partial E(\{\mathbf{x}\}, W)}{\partial \mathbf{w}(t)} \quad (\text{Single Sample Version})$$

$$= \mathbf{w}(t) - \alpha \frac{\partial E(B, W)}{\partial \mathbf{w}(t)} \quad (\text{Batch Version})$$

where the learning rate is for convergence conditioned with [16, p.424]

$$\begin{aligned}\sum_{i=1}^{\infty} \alpha_i^2 &< \infty \\ \sum_{i=1}^{\infty} \alpha_i &= \infty \\ \alpha_i &\geq 0\end{aligned}$$

## Entropy

Entropy will be discussed mostly as an information theoretical concept, as introduced by Shannon [17]. For clarity, we will briefly refer to the thermodynamic branch, where entropy was introduced. In the context of information theory, entropy is a *quantitative measure on information density* [18, p.4]. More strictly speaking, entropy can be interpreted as *the amount of randomness of a random variable, a measure of uncertainty of a random variable or as average quantity of information obtained by observing a random variable* [18, p.20]. Let  $W$  be a channel encoding some input sequence  $(x_i)_{i=1}^n$ , transmitting the sequence and lastly decoding to an output sequence  $(y_i)_{i=1}^n$ . Such a channel can be interpreted as a conditional probability<sup>2</sup> [18, p.6]

$$W(y_1, \dots, y_n \mid x_1, \dots, x_n) = p(y_1, \dots, y_n \mid x_1, \dots, x_n) \quad (2.1)$$

### Definition 2.6 (Shannon Entropy)

Let  $X$  be a discrete random variable over a probability distribution  $P_X$  with  $p_i = P(X = x_i)$  and  $x_i \in X$ . The Shannon entropy [17][18, p.20] of  $X$  is given as

$$\begin{aligned}H(X) &= H(P_X) = E_{P_X}[-\log P_X(X)] \\ &= - \sum_{x_i \in X} P(X = x_i) \log P_X(X = x_i) \\ &= - \sum_{x_i \in X} p_i \log p_i\end{aligned}$$

In statistical mechanics, a related branch of thermodynamics, the probability of a microstate  $i$  with some energy  $\varepsilon_i$  is given as

$$p_i = \frac{\exp\left(\frac{-\varepsilon_i}{k_b T}\right)}{\sum_j \exp\left(\frac{-\varepsilon_j}{k_b T}\right)}$$

<sup>2</sup>The source differentiates between a source encoder/decoder and channel encoder/decoder. For the sake of simplicity, we omit these distinctions. At a later point in this thesis, recurrent neural networks will be introduced. These estimate for an input sequence an output sequence. The given form looks very similar. This is used in distribution functions in sparsemax and entmax later on.

This equation is known as the *Boltzmann distribution* [17]. Here,  $k_b$  is the Boltzmann constant and  $T$  the temperature. The attentive reader will quickly notice, by setting  $k_b T = -1$ , we yield the softmax function. Note,  $p(\varepsilon_i) \propto e^{-\beta \varepsilon_i}$ , which shows the relation to exponential distribution families. From the Boltzmann distribution and the expectation of the energy level  $U = \sum_i p_i \varepsilon_i$  the *Gibbs-Boltzmann-Shannon Entropy* can be derived [19].

**Definition 2.7** (Gibbs-Boltzmann-Shannon Entropy)

Let  $X$  be some discrete random variable over a probability distribution  $P_X$  with  $p_i = P(X = x_i)$ . Then, the Gibbs-Boltzmann-Shannon Entropy [19, 20] is given as

$$\begin{aligned} H^S(X) &= H^S(P_X) = E_{P_X}[-\ln P_X(X)] \\ &= - \sum_{x_i \in X} p_i \ln p_i \end{aligned}$$

*Remark 2.8*

Note, in some literature a more physical representation of the *Gibbs-Boltzmann-Shannon* entropy is given as [17]

$$H^S(X) = -k_b \sum_{x_i \in X} p_i \ln p_i$$

incorporating the Boltzmann-constant  $k_b$ .

*Remark 2.9*

By deriving the *Gibbs-Boltzmann-Shannon* from a thermodynamic perspective, we connected the softmax function to entropy and set  $e$  as the basis of the log function.

Next, we will introduce the *Gini Entropy*, which is also known as the Gini index. It is commonly known as a measure for income inequality in the field of economics [21, p.39]. In machine learning, it is known as cost function used in decision trees for optimally constructing such trees [12, p.681]. We define the *Gini Entropy* according to [20].

**Definition 2.10** (Gini Entropy)

Considering the same assumptions as in [Gibbs-Boltzmann-Shannon Entropy](#), we define the Gini entropy for  $p_i = P(X = x_i)$  as

$$H^G(X) = \frac{1}{2} \sum_{x_i \in X} p_i (1 - p_i)$$



Lastly, we will introduce a generalization<sup>3</sup>. Tsallis generalized on issues like the behavior of self-gravitating systems (black holes, galaxies), neutral plasma, single hydrogen atoms or a single spin in an external magnetic field, where the Gibbs-Boltzmann calculations do not hold [23]. First, we define the  $q$ -logarithm [22].

**Definition 2.11** ( $q$ -logarithm)

Let  $x \in \mathbb{R}^+$  and  $q \in \mathbb{R}$ . We define the  $q$ -logarithm as

$$\ln_q x = \begin{cases} \ln x & \text{if } q = 1 \\ \frac{x^{1-q} - 1}{1-q} & \text{otherwise} \end{cases}$$

**Property 2.12**

Let  $\ln_q$  be the  $q$ -logarithm. We have

$$\lim_{q \rightarrow 1} \ln_q x = \ln x$$

Property 2.12 follows by applying L'Hospital's rule in solving the limit.

**Definition 2.13** (Tsallis Entropy)

Considering the same assumptions as in the previous entropy definitions, we define the Tsallis entropy [22–24] for  $p_i = P(X = x_i)$  as

$$\begin{aligned} H_q^T(X) &= \sum_{x_i \in X} p_i \ln_q \left( \frac{1}{p_i} \right) \\ &= \frac{1}{1-q} \left( \sum_{x_i \in X} p_i^q - 1 \right) \end{aligned}$$

Assuming we do not have an indifferent probability distribution, implying we cannot assign probabilities  $p_i = \frac{1}{n}$  for  $1 \leq i \leq n$ , we are given the task to find probabilities  $p_i$  to some given constraint, i.e. the mean value. This is commonly known as *underdetermined* problem, because there are many probability values which can satisfy the constraint [25, p.38]. The maximum entropy principle maximizes the entropy given such constraint. In other words, we estimate the probabilities for a random variable  $X$  given limited information about the system, while still maximizing the entropy. By solving this task, the optimal set of probabilities  $\mathbf{p}^*$  with respect to the entropy is the least biased solution in terms of making assumptions about the given system [26] [25, p.38-39].

<sup>3</sup>Among others, the *Rényi entropy* generalizes the entropy concept. Since there is no further use for other entropies, we omit an investigation of other entropies. For the sake of completeness, it is mentioned at this point. More on the Rényi entropy and an axiomatization of entropy can be found in [22].

**Definition 2.14** (Maximum Entropy Principle)

Let  $X$  be a random variable with probability  $p_i = P(X = x_i)$ . The *maximum entropy principle* maximizes the entropy  $H(X)$  given a constraint  $E[\phi_\alpha(X)] = \hat{\mu}_\alpha$  with  $\phi_\alpha(X) : X \rightarrow \mathbb{R}$ ,  $\alpha \in \mathbb{N}$  and  $E$  being the expectation value<sup>4</sup>. Hence, the *maximum entropy principle* finds a set of probabilities  $\mathbf{p}^* = [p_1^*, \dots, p_n^*]$ , which maximizes the entropy. This defines the following optimization problem

$$\mathbf{p}^* = \operatorname{argmax}_{\mathbf{p} \in \Delta^d} H(\mathbf{p})$$

subject to the expectation

$$E[\phi_\alpha(X)] = \hat{\mu}_\alpha$$

Note, the probability simplex  $\Delta^d$  will be explained in detail in [Affine Spaces and Simplices](#).

*Remark 2.15*

Assume we have an entropy optimization problem without constraint. Maximizing the entropy  $H(X)$  yields  $p_i = \frac{1}{n}$  and minimizing the entropy yields setting some  $p_i = 1$  [26].

## 2.2 Mathematical Concepts

### Basic Similarity

**Definition 2.16** (Basic Similarity)

The basic similarity [27] is defined for some objects in the object space  $X$  as a mapping

$$\operatorname{sim} : X \times X \longrightarrow \mathbb{R}$$

with  $\forall x, y \in X$

- (1) Minimum Principle:  $\operatorname{sim}(x, x) \leq \operatorname{sim}(x, y)$   
 $\operatorname{sim}(x, x) \leq \operatorname{sim}(y, x)$
- (2) Non-negativity:  $\operatorname{sim}(x, y) \geq 0$

*Remark 2.17*

In [27], a comprehensive taxonomy is built around the concepts similarity and dissimilarity. The properties given in [definition 2.16](#) define the most basic form of similarity. Further similarities add further properties.

<sup>4</sup>Here, the function  $\phi_\alpha$  determines the kind of expectation value computed. For instance, if we choose  $\phi_2(x) = x^2$ , we effectively compute the second moment with the expectation value function [25, p.38].

## Affine Spaces and Simplices

### Definition 2.18 (Affine Space)

Let  $V$  be a vector space and  $L$  a set of points. Then the tuple  $(L, V)$  is called an affine space [28, p.289], if each pair of points in  $L$  can be associated with a vector in  $V$ , hence

$$\forall a, b \in L : \exists \vec{ab} \in V$$

with the following conditions

$$\forall a, b, c \in L : \vec{ab} + \vec{bc} = \vec{ac} \in V$$

$$\forall a, b, c \in L : \exists ! d \in L : \vec{ab} = \vec{cd} \in V$$

$$\forall a, b \in L \text{ and for all scalars } \gamma : \exists ! c \in L : \vec{ac} = \gamma \vec{ab}$$

### Remark 2.19

The order of the points  $a, b \in L$  in the vector  $\vec{ab} \in V$  is important. The point  $a$  is the *initial point* and  $b$  is the *terminal point*.

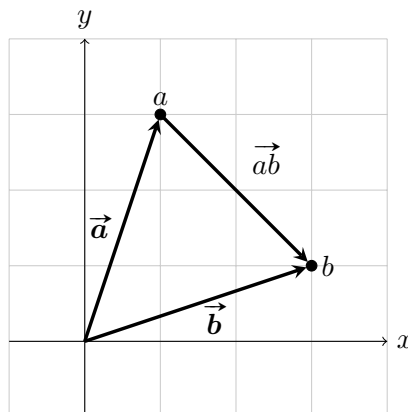
### Remark 2.20

In linear spaces, a vector can be located with respect to its origin. This property is dismissed in affine spaces. Here, only the *magnitude* and *direction* of vectors, built by points, are considered. For this reason affine spaces can be called *uniform* [28, p.291]. Therefore, the order of points, as stated in [remark 2.19](#), is important.

### Proposition 2.21 (Vector Construction by Points)

Let  $V$  be a vector space, then  $(V, V)$  is an affine space by considering  $a, b \in V$  as points and setting  $L = V$ . Then the corresponding vector in  $V$  can be defined by points in  $L$  as [28, p.290]

$$\vec{ab} = b - a$$



**Figure 2.1:** Example with  $\vec{a} = (1, 3) \in V$  and  $\vec{b} = (3, 1) \in V$  and their respective points  $a, b \in L = V$  and the vector  $\vec{ab} \in V$

**Definition 2.22** (Affine Combination)

Let  $(a_i)_{i=1}^n$  be a family of points in  $L$  and  $(\gamma_i)_{i=1}^n$  a family of scalars with  $\sum_{i=1}^n \gamma_i = 1$ . The association of points with weights are *weighted points*  $(a_i, \gamma_i)_{i=1}^n$ . Then

$$\sum_{i=1}^n \gamma_i a_i$$

is called the *barycentric combination*, *affine combination*, *convex combination* or *barycenter of points*  $a_i$  assigned to weights  $\gamma_i$  [29, p.19][30, p.8].

**Definition 2.23** (Geometric Independence)

A set of points  $\{a_i\}_{i=0}^n \subseteq L$  is called *geometrically independent* [31, p.2] if for any scalar  $\gamma_i$  it holds

$$\sum_{i=0}^n \gamma_i = 0 \quad \wedge \quad \sum_{i=0}^n \gamma_i a_i = 0 \quad \implies \quad \gamma_i = 0 \text{ for } 0 \leq i \leq n$$

**Proposition 2.24** (Geometric and Linear Independence)

Let  $\{a_i\}_{i=0}^n \subseteq L$  then we have

$$\{a_i\}_{i=0}^n \text{ is geometrically independent} \iff \{a_i - a_0\}_{i=1}^n \text{ is linearly independent} [31, p.3]$$

*Proof:* Let  $\{(a_i, \gamma_i)\}_{i=0}^n$  be a set of weighted geometrically independent points. We have

$$\begin{aligned} \sum_{i=0}^n \gamma_i a_i &= 0 \\ \sum_{i=1}^n \gamma_i a_i + \gamma_0 a_0 &= 0 \end{aligned}$$

Note,  $a_i$  are geometrically independent, hence  $\gamma_i = 0$  for  $0 \leq i \leq n$ , we can write

$$\begin{aligned} \sum_{i=1}^n \gamma_i a_i - \sum_{i=1}^n \gamma_i a_0 &= 0 \\ \sum_{i=1}^n \gamma_i a_i - \gamma_i a_0 &= 0 \\ \sum_{i=1}^n \gamma_i (a_i - a_0) &= 0 \end{aligned}$$

□

**Proposition 2.25** (Geometric Independence in  $\mathbb{K}^n$ )

At maximum  $n + 1$  points can be geometrically independent in  $\mathbb{K}^n$ . This follows directly from [proposition 2.24](#).

**Definition 2.26** (*n*-Plane)

Let  $\{(a_i, \gamma_i)\}_{i=0}^n$  be a set of geometrically independent weighted points in some affine space  $(L, V)$ . The *n*-plane  $P$  [31, p.3] is the set of points spanned by the affine combination

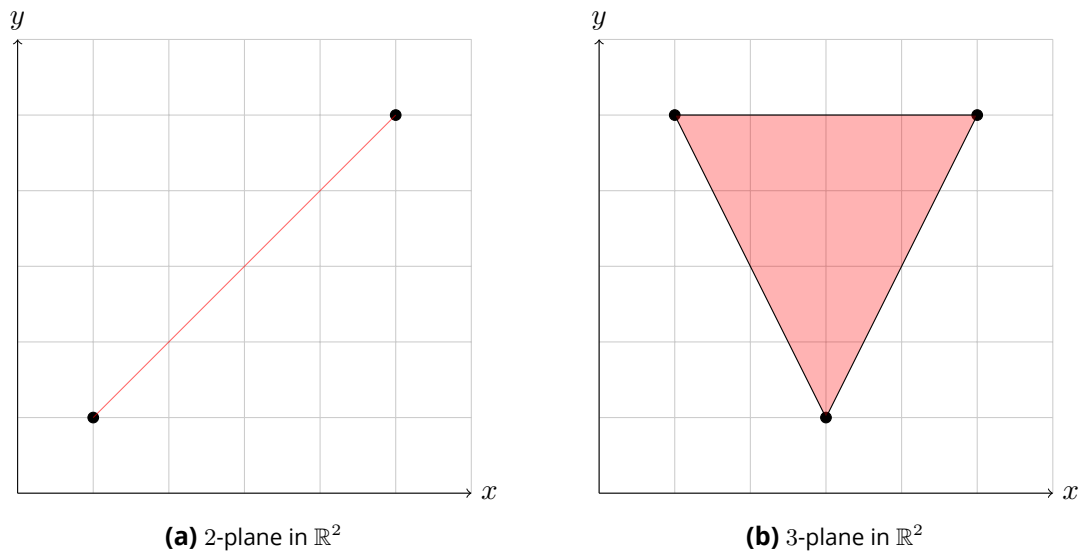
$$P = \left\{ \sum_{i=0}^n \gamma_i a_i \mid \sum_{i=0}^n \gamma_i = 1 \right\}$$

An *n*-plane  $P$  through  $a_0$  parallel to the vectors  $\{a_i - a_0\}_{i=1}^n = \{\overrightarrow{a_0 a_i}\}_{i=1}^n$  is the set

$$P = \left\{ a_0 + \sum_{i=1}^n \gamma_i (a_i - a_0) \mid \sum_{i=1}^n \gamma_i = 1 \right\}$$

**Example 2.27**

Let  $(\mathbb{R}^2, \mathbb{R}^2)$  be an affine space. The set of geometrically independent points  $\{(1, 1), (5, 5)\}$  define a 2-plane and the set of geometrically independent points  $\{(3, 1), (5, 5), (1, 5)\}$  define a 3-plane



**Figure 2.2:** *n*-planes in  $\mathbb{R}^2$

**Definition 2.28** (Standard *n*-Simplex)

Let  $(\mathbb{R}^{n+1}, \mathbb{R}^{n+1})$  be an affine space. The *n*-plane defined by a vector of scalars  $\gamma_i \in \mathbb{R}$  with

$$\Delta^n = \left\{ (\gamma_1, \dots, \gamma_n) \mid \sum_{i=1}^n \gamma_i = 1 \wedge \gamma_i \in [0, 1] \right\}$$

is called the *standard n-simplex* [32, p.103].

**Remark 2.29**

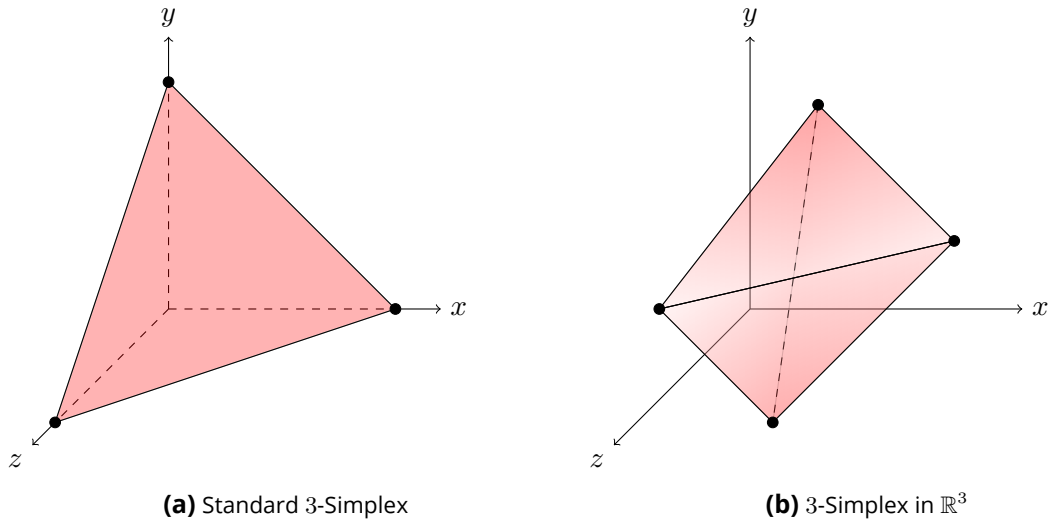
The vertices spanning the *standard n-simplex* are the unit vectors in  $\mathbb{R}^{n+1}$ .

**Definition 2.30** ( $n$ -Simplex)

Let  $\{a_i\}_{i=0}^n$  be a set of *geometrically independent* points in the affine space  $(\mathbb{R}^m, \mathbb{R}^m)$  and  $\gamma_i$  some scalars. The convex hull defined by

$$\Delta^n = \left\{ \sum_{i=0}^n \gamma_i a_i \mid \sum_{i=0}^n \gamma_i = 1 \wedge \gamma_i \in [0, 1] \right\}$$

is called the  $n$ -simplex [31, p.4].



**Figure 2.3:** Illustration of Simplices

**Remark 2.31**

While the barycentric coordinates  $\gamma_i$  of the  $n$ -plane can be negative, they must be non-negative in simplices.

**Remark 2.32**

The standard simplex is also known as the *probability-simplex*. A coordinate  $\mathbf{p} = (p_1, \dots, p_n) \in \Delta^n$  contains probabilities over a discrete probability distribution on a random variable  $X$ , i.e.  $p_i = P_i(X = x_i)$ . Hence, the standard simplex is an  $n$ -manifold of all probability distributions  $\mathbf{p}$  [33, p.7].

## 3 Attention

### 3.1 Neuroscientific Motivation

Before delving into mathematical and technical considerations about the most recent developments of attention [34], we will consider attention from a biological and psychological perspective to get a better grasp on the semantics of the term and how we capture the mechanics of it mathematically.

Hommel published a paper “No one knows what attention is” in 2019 arguing that the term is misused in the field of cognitive science [35]. Hommel further suggests that the term *attention* encapsulates too many concepts which differ vastly or even are mutually exclusive up to some degree. The struggle of neuroscientists to find a uniform definition for attention has been ongoing for the past decades. Three different problems of the understanding attention from a neurological point of view have been given by Hommel.

1. When considering attention as a set of cognitive and neural mechanisms which maximize the efficacy and efficiency regarding a task, we face the issue that almost any cognitive mechanism falls into this point of view. This renders this understanding of attention redundant.
2. Attention is used in different contexts to explain opposing ideas, such as to be the cause as well as to be the result from such an idea.
3. Facing overlapping semantical meaning of definitions with other concepts, i.e. attention and intention, lead to the approach of reducing the semantic overlap by pruning the definitions. The problem of this approach is not taking into consideration that these concepts are nested in a neurological dynamic not allowing having distinct static definitions.

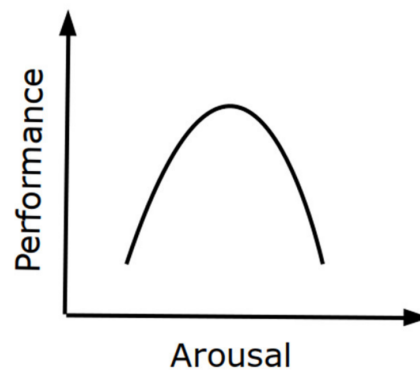
Since there was no success to find a coherent definition for attention, it is hard to pin down the term attention to a specific region of the brain. Though the semantic variations of attention themselves can be linked to brain areas up to some degree of certainty. To enhance the understanding of attention, we introduce some different notions of attention used in neurosciences.

**Proposition 3.1** (Attention by Arousal)

*Attention* is the ability to engage with an environment. The level of attention can be determined by the *arousal, alertness or vigilance* of a subject. The highest level of attention can be found between the lowest and highest level of arousal. This is known as the Yerkes-Dodson-curve, which resembles an inverted U-shape [36].

In [proposition 3.1](#) we find a very generic and basic approach on how to define attention [36]. Here peak attention is balanced between a mental under- and mental overload. This phenomena is described by the Yerkes-Dodson curve shown in [Figure 3.1](#). There the performance is the measure for the level of attention shown. Attention can be further analyzed

by focusing on different sensory inputs. In this neuroscientific motivation the sensory input discussed will be the visual sensory system. Given this restriction we can more easily point out the different mechanics attention can actually involve.



**Figure 3.1:** Yerkes-Dodson Curve [36]

Attention can also be defined by a more biologically focused approach.

**Proposition 3.2** (Attention as system of organs)

Attention is a system of organs with its own unique anatomy, connectivity, neuromodulators, and functions [37].

We further discuss attention with the focus on the visual sensory system. Notice upcoming proposals will still be held general, but can easily be referred to the visual sensory system. Later on the discussion of attention mechanics will be specific to the visual system mechanisms.

There are many ways how to further break down the term attention. In neuroscience we can find alerting, orienting, executive, selective, divided, sustained attention and many more [36–39]. In this introduction to the concepts of attention we will focus on *selective attention*.

**Proposition 3.3** (Selective Attention)

Selective attention is the ability to filter stimuli within a sensory scope with respect to a certain filtering objective [40].

Attention can on the one hand be thought as a person focusing on a certain thing, while on the other hand we can also consider it to be a person switching immediately to some unexpected emerging impulse. These notions are captured by the following propositions.

**Proposition 3.4** (Top-Down Attention)

In top-down attention the subject is given some sort of task where the subject *actively* is trying to localize the requested stimulus [41].



**Proposition 3.5** (Bottom-Up Attention)

In bottom-up attention the subject is experiencing the stimulus *passively*, such that the excitation can happen involuntarily. The cause for such excitation is the appearance of salencies. Salencies are features which stand out in the environment *attracting attention*. These can be described with saliency maps, which specify the features which cause this type of attention [41].

Appending to [proposition 3.5](#) these salencies can be ranked by the level of sensitivity with respect to the excitation, i.e. how low the threshold for excitation is to be found. Such salencies can be color and size ranked as an effective saliency or the number of appearances and closure of objects to be rather weak salencies. A comprehensive list of such saliency ranking can be found in [41].

As previously discussed the term attention can be ambiguous. Though we segment the concept of attention into more refined terms, these can still be semantically overlapping. Given a search task the subject will actively try to localize a certain feature in its visual field. Since the subject is in the mode of top-down attention, it can still be guided by stimuli of bottom-up attention.

**Example 3.6**

Assume the task is to identify a letter of a given color in a text, i.e. a red letter in an otherwise uniformly black text. Though, the subject clearly received a search task, the search can be guided by stimuli in the peripheral vision, such that the eye movement is directed to the salient stimulus [36, 41]. This is phrased as *bottom-up attention guides top-down attention*.

We can now further distinct how attention is being applied. The subject does not necessarily need to look at the given feature of interest to be found by the search task. Hence, we can distinguish by actually looking at the to be found letter and just being aware of its presence without looking at it. These situations can be described by the concepts *overt* and *covert spatial attention*.

**Proposition 3.7** (Covert Attention)

Covert attention can be characterized by selectively focusing on a stimulus *without conscious awareness or overt movement of the sensory system* towards the stimulus. This allows to monitor the environment and react if a stimuli presents itself as relevant [42].

**Proposition 3.8** (Overt Attention)

Contrary to *covert attention* the sensory system focuses on the stimulus causing the excitation. Synonymously this type of attention can be named *directed attention* [42].

Note, covert attention can be justified biologically. The retina's point of sharpest resolution is the fovea. Since it is a limited computational resource it is important to direct this sensor with care [36]. At first glance the concepts of top-down/bottom-up and covert/overt attention seem to be equivalent. But the concepts of these categories can overlap to some degree, while still being different enough to not be equivalent.

### **Example 3.9**

The situation of driving a car illustrates these concepts in a single scenario.

#### **Covert Attention**

A person driving a car needs to be aware of objects in traffic without redirecting its sensory system towards such objects while still being aware of them peripherally. The person needs to emphasize its focus on the location where the car is heading, i.e. the road, which defines the purpose of covert attention. Such covert objects might be pedestrian stepping, guardrails or any other obstacles.

#### **Overt Attention**

The direction of the sensory system to the road and immediate traffic illustrates overt attention. Also considering an ambulance appearing and seeing it in the rear-view mirror is overt attention, since the sensory system, i.e. visual system, is directed towards the ambulance.

#### **Top-Down Attention**

As the driver is presumably heading for a destination, they are actively looking for landmarks, street signs or following directions from the navigation system.

#### **Bottom-Up Attention**

Given some flashing lights appear in the peripheral view of the driver, i.e. ambulance flashing lights, then this can trigger bottom-up attention.

Considering [example 3.9](#) we can notice an overlap between these concepts, while still having distinct considerations. Bottom-up and covert attention share the same property here of an ambulance entering the peripheral vision, while bottom-up attention can be considered to be an automatic experiencing of the stimulus and covert attention a selective choice of a covert stimulus. Still, we have the distinct effects such as that the bottom-up attention guides the covert attention to become aware of the stimulus without redirecting our sensory system towards the stimulus.

We enhance this example slightly, such that the driver is listening to the news on the radio, where the information is shared that an ambulance is supposed to appear to the driver's vision. Then we can expect the driver to perform top-down attention actively searching for the ambulance. If the ambulance's flashlight appear peripherally, they can cause bottom-up attention which then guides the top-down attention to the given search object. This example shows that the distinction of attention concepts can have overlaps, while still being motivated by different mechanics.

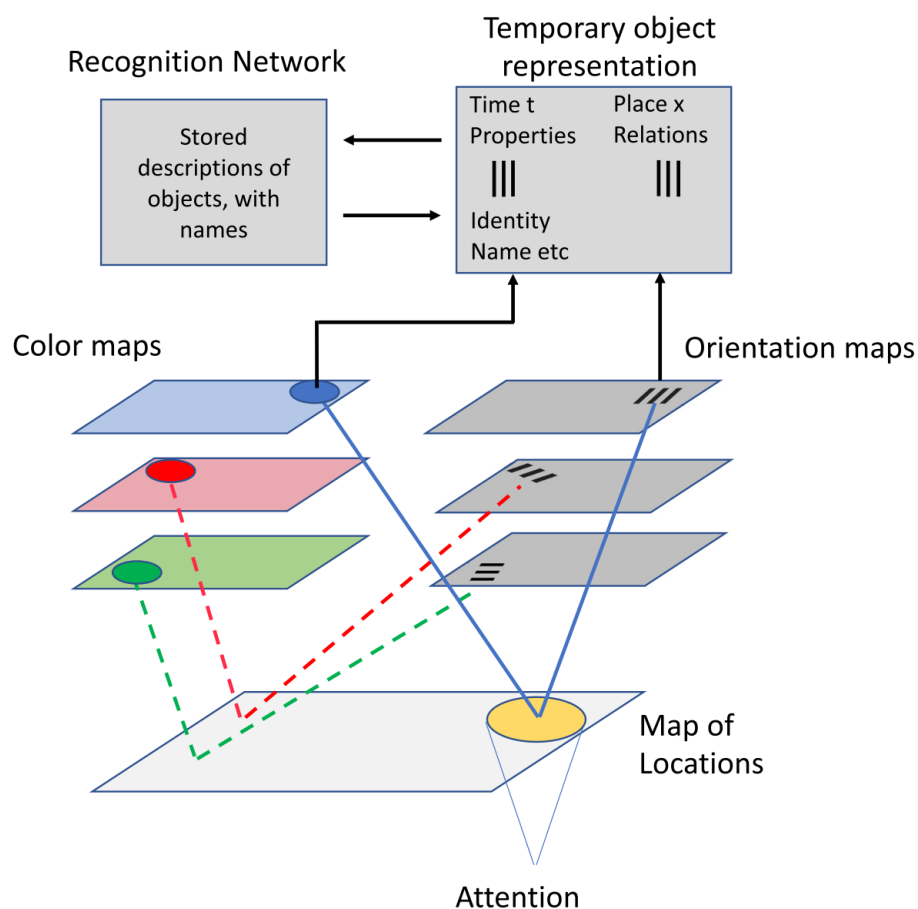
### **Proposition 3.10** (Spatial Attention)

Spatial attention characterizes selectively processing a stimulus causing alertness for attention at a specific location in a person's surrounding space [36].

*Visual spatial attention* can be also be connected with the terms bottom-up/top-down and covert/overt attention as we did previously [36]. Consider [example 3.9](#) where we have the flashlights from the ambulance in our peripheral vision. Since the attention is directed to the location of the ambulance, we could consider it *visual spatial covert bottom-up attention*. Now consider the modification of the example that the person got notified via radio such that it should be possible to see such an ambulance. The person now actively looks for it in specific locations of its view, hence this is *visual spatial overt top-down attention*.

**Proposition 3.11** (Feature Attention)

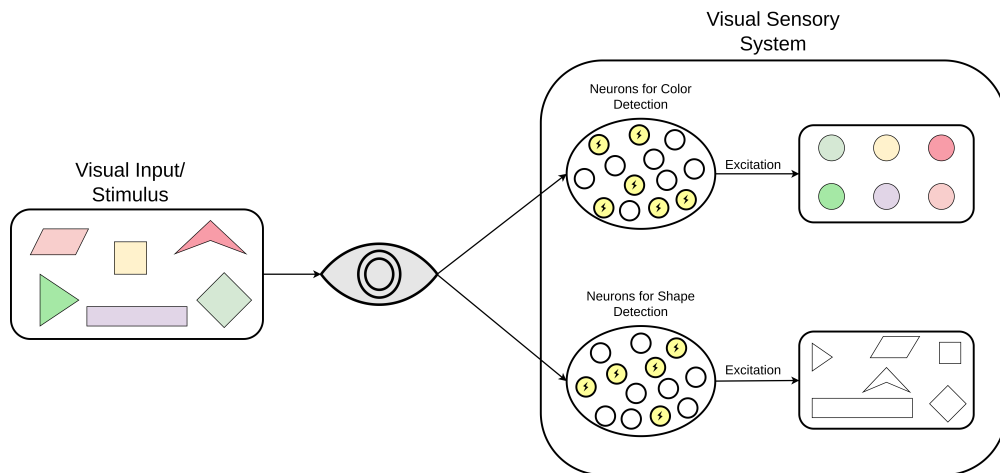
Feature attention also characterizes selectively processing a stimulus with a specific feature/saliency of the sensory domain, allowing for the detection and prioritization of information relevant to the task at hand [36].



**Figure 3.2:** Illustration of the perception of the features *color* and *orientation* and how they are processed distinctly. Each feature is associated on a master map to a location [43].

The same considerations can be done for feature attention as we did with spatial attention. Considering again [example 3.9](#) with the same scenario of having flash lights from the ambulance in the person's peripheral vision. This could also be considered *visual feature covert bottom-up attention* if we emphasize that the person is not focusing on the location but instead on the feature of the salient stimulus. Also for the second scenario we can change it to *visual spatial overt top-down attention* if again the focus is on the feature of the to be searched object instead of its location.

For visual attention we can identify that perception of visual features such as color, shape, letters, words or clock times is each bound to a specific set of neurons [41]. This observation was shown by Treisman and Gelade with the introduction of *Feature-Integration Theory of Attention* (FIT), which among other things states that the first processing of stimuli to the sensory system is done on each feature in parallel [44]. This preprocessing is named preattentive processing. These features are roughly coded as spatial locations into a master map. Though this observation was done on the visual sensory system, it also holds for other sensory systems such as the auditory system [45]. It should be noted that original FIT was adapted to certain aspects, where it did not hold specific conditions [43].



**Figure 3.3:** Attention Experiment

The binding of different features to an object has a possible side-effect of illusory conjunctions. Consider the following experiment, that a subject is given a visual stimulus with colored shapes. [Figure 3.3](#) illustrates the situation with the neural computation. If the subject is asked to name the shapes or colors it is likely that the subject is able to name them. But if the subject is asked to name them in *conjunction* a mix up is common. As [Figure 3.2](#) suggests locally neighboring elements are subject to be mixed up in their conjunction [41].

This section should highlight how diffuse the concept of attention is and that it is necessary to be very precise when speaking about attention in a neuroscientific manner. Also, the manifold of interpretations and views on attention can inspire approaches on how to model neural models involving these concepts. This is only a brief introduction and motivation on the term attention from a neuroscientific view. There are many more aspects which can be considered, possibly leading to new concepts in machine learning.

## 3.2 Computational Attention

In the previous section, we introduced some neuroscientific findings regarding attention. Such findings are subject to creating computational models mimicking the analyzed biological mechanisms and/or psychological behavior. In this thesis, we will foremost discuss attention-based neural networks as they were introduced in encoder-decoder [Recurrent Neural Networks \(RNN\)](#) [1, 2]. Note, there have been attempts to develop attention based models before. For instance, there is a model emphasizing on using feature integration theory. There, a saliency map is utilized to spatially detect objects of interest as it was discussed

in [proposition 3.5](#)[46, 47]. Also, other types of bottom-up approaches have been developed which can be found in [48]. Interestingly, recently it has been discussed that [CNN](#) (see preliminary section [Convolutional Neural Network](#)) perform *selective attention* [49].<sup>5</sup>

Already in 1964, ideas similar to the attention of modern network architectures have been developed. Here, some non-negative function  $\delta$  maps over all data points  $(\mathbf{x}_i)_{i=1}^n$  with a query  $\mathbf{x}_j$  to compute a weight  $\alpha_i$  [50–52].

$$\delta(\mathbf{x}_j, \mathbf{x}_i) = \alpha_i$$

This weight  $\alpha_i$  represents the relevance of the data point  $\mathbf{x}_i$  to the query  $\mathbf{x}_j$ . This is then weighted with the according response  $\mathbf{y}_i$ . Then, as a prediction, we have  $\hat{\mathbf{y}}$  for the query  $\mathbf{x}_j$

$$\hat{\mathbf{y}} = \frac{1}{n} \sum_{i=1}^n \alpha_i \mathbf{y}_i \quad (3.1)$$

There is no biological motivation to be found in the original papers describing this procedure for smooth regression analysis (see [Regression](#)). Yet, interpreting this with the neuropsychological knowledge on attention, this construction resembles the detection of saliencies with respect to some specific observation. Further, it should be noted that it was suggested to find the optimal function  $\delta$  by trial and error.<sup>6</sup>

Ongoing, we will discuss different kinds of attention models and show in which context they were implemented. Certain models will be briefly explained, since they are not of main interest in this thesis. For further reading, sources are given at the introduction of such a model. To give appropriate context to the development of these models, necessary underlying architectures will be introduced briefly.

### 3.2.1 Prerequisites

As mentioned before, the [Attention Model \(AM\)](#) was introduced to improve on encoder-decoder [RNN](#). Basic [RNN](#) are neural networks with a special architecture which can process input data of variable length, i.e. sequential data [7, p.364].

#### Definition 3.12 (RNN)

Let  $(\mathbf{x}_i)_{i=1}^n \subseteq \mathbb{R}^k$  be sequential data of variable length  $n$ . A [RNN](#) is a neural network taking the sequence  $\mathbf{x}$  as input and processing it sequentially. At each time step  $t$  a hidden state  $\mathbf{h} \in \mathbb{R}^l$  is being computed memorizing information of the previously processed inputs. We have

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t, \Theta)$$

where

<sup>5</sup>As a sidenote, [CNN](#) are also used to better interpret the visual system from a biological perspective. Currently, very similar processing is assumed to happen in the biological processing of optic input to the image processing of [CNN](#)[49].

<sup>6</sup>The family of functions  $\delta_n$  eligible for this process has some specific conditions like  $\delta_n$  converges to the Dirac function as  $n \rightarrow \infty$ .

$f$  ... is a non-linear activation function  
 $\Theta$ ... are the learning parameters  
 $t$  ... as index referring to an input  $1 \leq t \leq n$  for  $(\mathbf{x}_i)_{i=1}^n$

Note, this general definition focuses on the recursive architecture without explaining how predictions  $\hat{\mathbf{y}}$  are generated. RNNs can have different constructions; to give a better understanding on RNNs, an example is presented.

### Example 3.13

An RNN could be constructed as follows

$$\begin{aligned}
 \mathbf{a}_t &= \mathbf{b}_1 + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t \\
 \mathbf{h}_t &= \tanh(\mathbf{a}_t) \\
 \mathbf{o}_t &= \mathbf{b}_2 + \mathbf{V}\mathbf{h}_t \\
 \hat{\mathbf{y}}_t &= \text{softmax}(\mathbf{o}_t)
 \end{aligned}$$

with

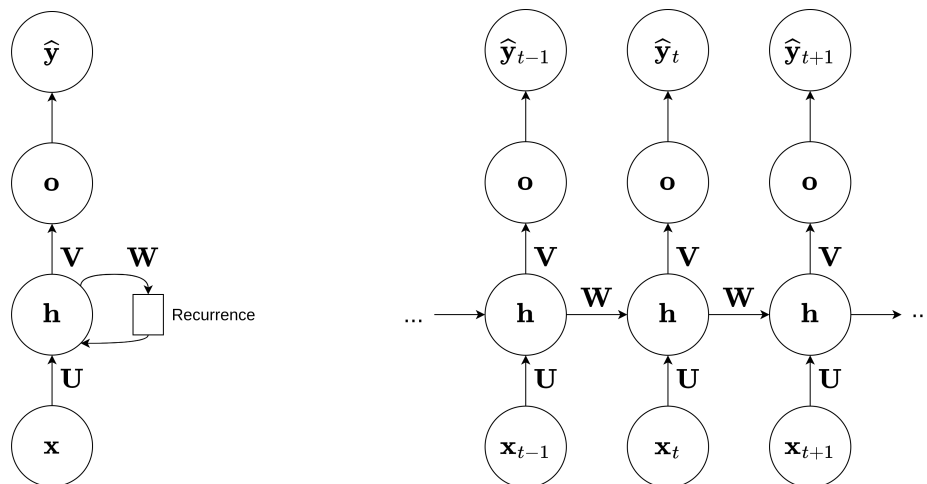
$\mathbf{a}_t \in \mathbb{R}^l$  ... input to the activation function  
 $\mathbf{b}_1 \in \mathbb{R}^l$  ... recursion bias  
 $\mathbf{h}_t \in \mathbb{R}^l$  ... hidden state at time  $t$   
 $\tanh$  ... non-linear activation function  
 $\mathbf{W} \in \mathbb{R}^{l \times l}$  ... recursive linear map  
 $\mathbf{x}_t \in \mathbb{R}^k$  ... input vector at time  $t$   
 $\mathbf{U} \in \mathbb{R}^{k \times l}$  ... input linear map  
 $\mathbf{o}_t \in \mathbb{R}^m$  ... output state at time  $t$   
 $\mathbf{b}_2 \in \mathbb{R}^m$  ... output bias  
 $\mathbf{V} \in \mathbb{R}^{l \times m}$  ... output linear map  
 $\text{softmax}$  ... post-processing to obtain normalized probabilities  
 $\hat{\mathbf{y}}_t \in \mathbb{R}^m$  ... output of normalized probabilities at time  $t$

Note, in this example the learnable parameters are  $\Theta = \{\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{b}_1, \mathbf{b}_2\}$  according to definition 3.12. Figure 3.4 illustrates the given example.

#### Remark 3.14

Given a basic RNN with  $(\mathbf{x}_i)_{i=1}^n$  as input and  $(\mathbf{y}_i)_{i=1}^n$  as output probabilities, such a RNN parameterizes the conditional probability  $p(\mathbf{y}_1, \dots, \mathbf{y}_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n)$  [53]. Hence, at time  $t$  the RNN estimates [54]

$$p(\mathbf{y}_t \mid \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$$



(a) Compact RNN Illustration of Example 3.13

(b) Unrolled RNN Illustration of Example 3.13

**Figure 3.4:** Example 3.13 illustrated in two different principles. The compact illustration displays the recurrence as a memory cell. The unrolled illustration on the other hand shows the information flow over time, where each sequence element  $x_i$  is inserted separately.

There are many kinds of RNN depending on their architectures and their outputs. As standard RNN are very prone to vanishing/exploding gradients due to parameter sharing<sup>7</sup>, they were improved by the introduction of gating cells such as long short term memory (LSTM) cells [55, 56]. Such gating cells control the flow of information in a sophisticated way, such that special learning properties arise. LSTM cells, for instance, are able to learn to forget information over the sequential computation of succeeding cells. Up until this point, network architectures allowed mapping sequential data  $(x_i)_{i=1}^n \subseteq \mathbb{R}^k$  to an output vector  $y \in \mathbb{R}^m$  or to another sequence of the same length  $(y_i)_{i=1}^n$ . To allow a mapping to sequential data of different length from the input length  $n$ , the encoder-decoder architecture was introduced [57].

**Definition 3.15** (Encoder-Decoder RNN)

Let  $f_E$  be a RNN called the *encoder* with

$$h_t = f_E(h_{t-1}, x_t, \Theta_E)$$

and  $\Theta_E$  being the learnable parameters for the encoder [57, 58]. The encoder computes from the hidden state(s)  $h_t$  a context vector<sup>8</sup>  $c \in \mathbb{R}^l$ . In its most basic form, the last hidden state  $h_n = c$ .

Let  $f_D$  be a RNN called the *decoder* with

$$p(y_t | y_1, \dots, y_{t-1}) = f_D(s_{t-1}, y_{t-1}, c, \Theta_D)$$

<sup>7</sup>Parameter sharing refers to the idea of weights being the same variable over all time steps  $t$ , hence, they are shared over all inputs of the sequence.

<sup>8</sup>As encoder-decoder networks were introduced, the last hidden state is set to be the context vector  $h_n = c$  [57]. Note, there are variations on how the context is being computed. If we have a [bidirectional recurrent neural network \(BRNN\)](#), then the context vector can be a concatenation of hidden states of the forward ( $\vec{h}_t$ ) and backward ( $\overleftarrow{h}_t$ ) directed RNN [2], i.e.  $h = [\vec{h}_t^T; \overleftarrow{h}_t^T]^T$ .

calculating the conditional probability of  $\mathbf{y}_t$  given  $\mathbf{y}_1, \dots, \mathbf{y}_{t-1}$  and  $\Theta_D$  being the learnable parameters of the decoder,  $\mathbf{y}_{t-1}$  the generated output of the previous time step and  $\mathbf{s}_{t-1}$  being the hidden state of the decoder at time  $t - 1$  [54]

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}, \Theta_D)$$

Here  $g$  is a non-linear activation function generating desired outputs, i.e. probability distributions.

*Remark 3.16*

An encoder-decoder RNN parameterizes the conditional distribution

$$p(\mathbf{x}_1, \dots, \mathbf{x}_t' \mid \mathbf{y}_1, \dots, \mathbf{y}_t) \quad (3.2)$$

Note, contrary to [remark 3.14](#) where the input length was equal to the output length, here input and output length can be different [59].

The general idea behind the encoder-decoder architecture is to compress important information of the input sequence  $(\mathbf{x}_i)_{i=1}^n$  into a context vector  $\mathbf{c} \in \mathbb{R}^l$ . As shown in [definition 3.15](#), the decoder is a recurrent architecture, processing additionally the context vector and the previous output to generate the current output. Usually, both the encoder and decoder make use of gating cells [54, 57, 58]. Since the context vector  $\mathbf{c}$  holds all the information used to process the output in the decoder, it needs to hold all relevant information with respect to the learning task, such that the decoder can produce viable results.

### 3.2.2 Attention Model

Encoder-decoder RNN face the short-coming of not being able to fully capture the context of longer inputs depending on the size of the context vector  $\mathbf{c}$ , and to model alignment between the input and output sequence [50]. To tackle these issues, the attention model (AM) was introduced.

**Definition 3.17** (Attention Model)

Let  $(\mathbf{h}_i)_{i=1}^n \subseteq \mathbb{R}^l$  be the hidden states of a RNN to the corresponding input sequence  $(\mathbf{x}_i)_{i=1}^n$ . The *attention model* computes for each hidden state  $\mathbf{h}_i$  a corresponding weight  $\alpha_{ij} \in \mathbb{R}$  such that the context vector  $\mathbf{c}_j$  at decoding position  $j$  is given as

$$\begin{aligned} \mathbf{c}_j &= \sum_{i=1}^n \alpha_{ij} \mathbf{h}_i \\ &= \boldsymbol{\alpha}_j^T \mathbf{H} \end{aligned} \quad (3.3)$$

with  $\mathbf{H} = [\mathbf{h}_1; \dots; \mathbf{h}_n]$  being a matrix of concatenated hidden states [50].



The attention model illustrates how the principle of attention is realized in encoder-decoder RNN. [Definition 3.17](#) does not show yet, how the attention weights are determined. To better understand how attention is implemented, we introduce how it was proposed originally [2].

**Proposition 3.18** (Attention Based Encoder-Decoder RNN)

Let  $f_E$  be an encoder-,  $f_D$  a decoder-RNN,  $g$  a generating output function<sup>9</sup> and  $(\mathbf{x}_i)_{i=1}^n \subseteq \mathbb{R}^k$  an input sequence. Then the network architecture can be described as follows

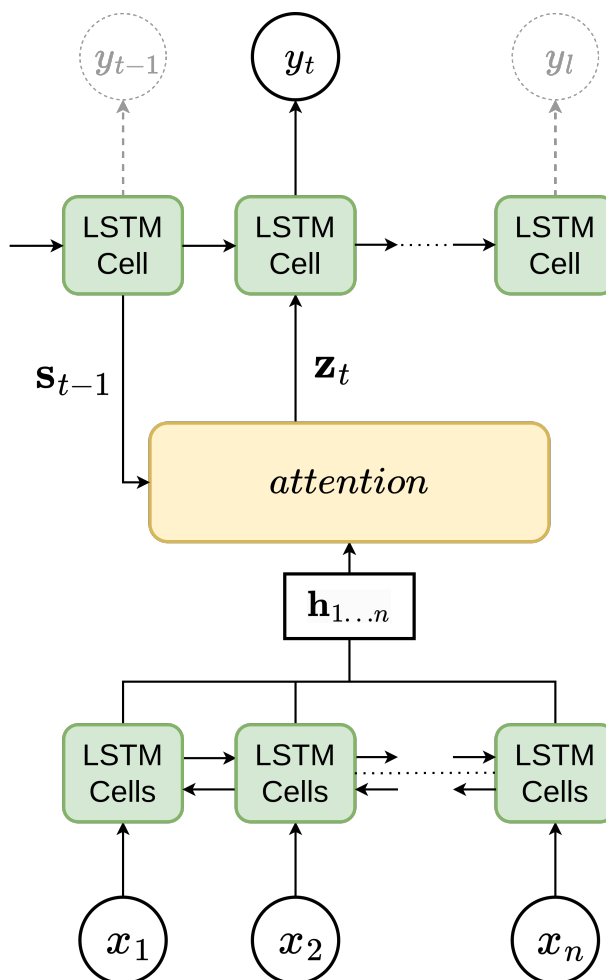
$$\begin{aligned} \mathbf{h}_t &= f_E(\mathbf{x}_t, \mathbf{h}_{t-1}) && \text{(Encoder Hidden State)} \\ \mathbf{s}_i &= f_D(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{c}_i) && \text{(Decoder Hidden State)} \\ \mathbf{y}_i &= g(\mathbf{y}_{i-1}, \mathbf{s}_i, \mathbf{c}_i) && \text{(Output)} \\ \mathbf{c}_i &= \sum_{j=1}^n \alpha_{ij} \mathbf{h}_j && \text{(Context Vector)} \\ \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})} && \text{(Attention Weight)} \\ e_{ij} &= a(\mathbf{s}_{i-1}, \mathbf{h}_j) && \text{(Energy)} \end{aligned}$$

where  $a$  is an alignment model of the form of a feed-forward neural network. The index  $t$  describes the input sequence into the encoder while the index  $i$  describes the input sequence into the decoder.

Contrary to the classical encoder-decoder RNN architecture, we now have for each hidden state  $\mathbf{s}_i$  a corresponding context vector  $\mathbf{c}_i$ . The context vector is computed as a weighted sum over all hidden states  $\mathbf{h}_t$ , with weights being the attention values  $\alpha_{ij}$  (see [equation 3.3](#)). Architecturally speaking, each attention weight  $\alpha_{ij}$  can be interpreted as an alignment between some hidden state  $\mathbf{h}_t$  of the encoder and the hidden state  $\mathbf{s}_{j-1}$  of the decoder [50]. A more global interpretation is that it weighs some input token  $\mathbf{x}_i$  to some output token  $\mathbf{y}_j$ . The attention weights are calculated using the softmax function to determine a probability distribution, such that  $\sum_i \alpha_{ij} = 1$  for some fixed  $j$ . [Figure 3.5](#) illustrates how attention is embedded into encoder-decoder RNN.

As the first attention based encoder-decoder RNN was introduced [2], the encoder was a BRNN (see [figure 3.6](#)). The purpose of a BRNN is to capture relations not only in one direction, but in both directions. Assuming we have as input a sentence, then a BRNN should capture the context in both directions. Take the sentence “John really liked the band, he was at each concert”. Without the information of “concert” at the end of the sentence, the meaning of band can be ambiguous, i.e., some radio wave length or thin piece of material for the purpose of wrapping something. Therefore, capturing the context in both directions allows relating concert to band. Note, the context vector therefor captures relations from both directions.

<sup>9</sup>In [example 3.13](#) the softmax function was used.



**Figure 3.5:** Attention Encoder-Decoder Illustration according to 'Attention Based Encoder-Decoder RNN' on page 26

### 3.2.3 Generalized Attention

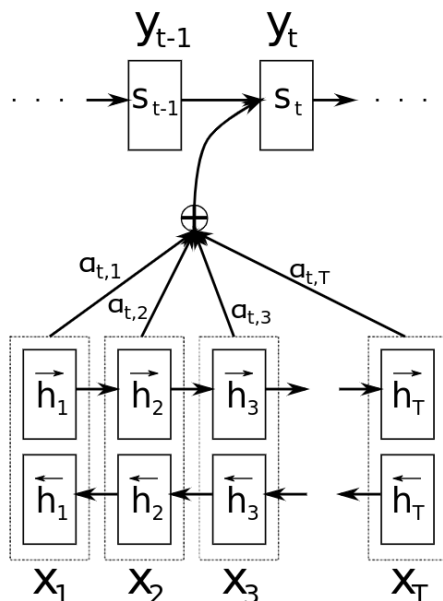
In the previous subsections, we have had a historical presentation on the evolution of attention in modern machine learning architectures. Since the original introduction by [2], other implementations have emerged with transformers being currently the most famous architectures among those [3]. With the manifold of different implementations of attention based models introduced in the last decade, a classification on different attention implementations can be made. First we discuss a generalization of attention. Further, we continue to classify attention.

**Definition 3.19** (Generalized Attention)

Let  $A$  be the attention mapping taking as input a query  $q \in \mathbb{R}^{d_q}$ , a key matrix  $\mathbf{K} = [k_1; \dots; k_n] \in \mathbb{R}^{n \times d_k}$  and a value matrix  $\mathbf{V} = [v_1; \dots; v_n] \in \mathbb{R}^{n \times d_v}$ . The attention mapping returns a weighted sum over a key value pairs  $(k_i, q) \in \mathbf{K} \times \mathbf{Q}$  with respect to the query  $q \in \mathbf{Q}$ . [50, 60]

$$A(q, \mathbf{K}, \mathbf{V}) = \sum_i p(a(k_i, q)) * v_i$$

where the operation  $*$  is some multiplicative operation and



**Figure 3.6:** Attention implementation into BRNN according to [2]

$p...$  a distribution function

$a...$  an alignment function

The functions  $p$  and  $a$  will be explained in [Alignment Function](#) and [Distribution Function](#) respectively.

*Remark 3.20*

Considering the regression analysis introduction highlighted by [equation 3.1](#), the query is represented by the data sample  $q = x_j$ . The key matrix also consists of the data samples  $\mathbf{K} = \mathbf{X} = [x_1, \dots, x_n]$  and the values are represented by the labels  $\mathbf{V} = \mathbf{Y} = [y_1, \dots, y_n]$ . The non-negative function  $\delta$  is responsible for the distribution and alignment at the same time. We conclude

$$A(x_j, \mathbf{X}, \mathbf{Y}) = \hat{y} = \frac{1}{n} \sum_{i=1}^n \delta(x_j, x_i) y_i$$

*Remark 3.21*

Considering the initial implementation of attention into the encoder-decoder RNN networks in [proposition 3.18](#), we have as distribution function the softmax function and the alignment function is a feed-forward network. Further, the value and key matrix consists of the hidden states of the encoder  $\mathbf{V} = \mathbf{K} = \mathbf{H} = [h_1, \dots, h_n]$ . Hence, we have

$$A(s_{i-1}, \mathbf{H}, \mathbf{H}) = c_i = \sum_{j=1}^n p(a(s_{i-1}, h_j)) h_j$$

As [remark 3.20](#) and [remark 3.21](#) illustrate, it is possible to further analyze attention over the implementation of the aggregation and alignment function. Another way of classifying attention is how the variables are chosen. As we can see, the choice of query, key and value can be different.

### Alignment Function

#### Definition 3.22 (Alignment Function)

The alignment function

$$a : \mathbb{R}^{d_k} \times \mathbb{R}^{d_q} \rightarrow \mathbb{R}$$

measures the compatibility or similarity between a key  $\mathbf{k}_i \in \mathbb{R}^{d_k}$  and a query  $\mathbf{q} \in \mathbb{R}^{d_q}$ . The resulting value can be referred as the energy or alignment score between query and key [[50](#), [60](#)].

#### Remark 3.23

For a given key  $\mathbf{k}_i$  we calculate the energy/alignment score

$$e_i = a(\mathbf{k}_i, \mathbf{q}) \tag{3.4}$$

yielding a vector of energies  $e = [e_1, \dots, e_{d_k}]$ .

[Example 3.24](#) gives an overview over common alignment function implementations<sup>10</sup> [[50](#), [60](#)]. There are many ways, how the alignment between query and key can be computed. An illustration can be found in [figure 3.7](#). One straight forward approach is to compute the *dot-product* between key and query [[3](#), [61](#)]. For this to be possible, both query and key must have the same vector space dimensionality  $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$  ([equation 3.6](#) and [equation 3.7](#)). Another category of alignment functions deals with the issue of having key and query vectors not being of the same dimensional vector space dimensionality. These alignment functions are *general* alignment functions. There, some learnable matrix  $\mathbf{W} \in \mathbb{R}^{d_k \times d_q}$  or  $\mathbf{W} \in \mathbb{R}^{d_q \times d_k}$  maps the vectors into the according dimensional representation to calculate the dot-product again [[61](#)–[63](#)]. Hence, we have some expression of the form  $\mathbf{q}^T \mathbf{W} \mathbf{k}_i$  which is called a *bilinear term* [[62](#)] ([equation 3.8](#), [equation 3.9](#) and [equation 3.10](#)).

<sup>10</sup>Note, for the scaled dot product there is a typo in [[60](#)]. Here,  $n_k$  is used as down-scaler in the denominator. As a reminder,  $n_k$  is the number of key vectors. In the original paper in [[3](#)] the dimensionality of the key vectors  $d_k$  is used.

**Example 3.24** (Alignment Functions)

Some examples of alignment functions are listed below and will be explained in the following text.

$$\text{similarity} \quad a(\mathbf{k}_i, \mathbf{q}) = \text{sim}(\mathbf{k}_i, \mathbf{q}) \quad (3.5)$$

$$\text{dot product} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{q}^T \mathbf{k}_i \quad (3.6)$$

$$\text{scaled dot product} \quad a(\mathbf{k}_i, \mathbf{q}) = \frac{\mathbf{q}^T \mathbf{k}_i}{\sqrt{d_k}} \quad (3.7)$$

$$\text{general} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{q}^T \mathbf{W} \mathbf{k}_i \quad (3.8)$$

$$\text{biased general} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{k}_i (\mathbf{W} \mathbf{q} + \mathbf{b}) \quad (3.9)$$

$$\text{activated general} \quad a(\mathbf{k}_i, \mathbf{q}) = \text{act}(\mathbf{q}^T \mathbf{W} \mathbf{k}_i + \mathbf{b}) \quad (3.10)$$

$$\text{generalized kernel} \quad a(\mathbf{k}_i, \mathbf{q}) = \phi(\mathbf{q})^T \phi(\mathbf{k}_i) \quad (3.11)$$

$$\text{concat} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{w}_{\text{imp}}^T \text{act}(\mathbf{W}[\mathbf{q}; \mathbf{k}_i] + \mathbf{b}) \quad (3.12)$$

$$\text{additive} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{w}_{\text{imp}}^T \text{act}(\mathbf{W}_1 \mathbf{q} + \mathbf{W}_2 \mathbf{k}_i + \mathbf{b}) \quad (3.13)$$

$$\text{deep} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{w}_{\text{imp}}^T E^{L-1} + \mathbf{b}^L \quad (3.14)$$

$$E^{(l)} = \text{act}(\mathbf{W}_l E^{(l-1)} + \mathbf{b}^l) \quad (3.15)$$

$$E^{(1)} = \text{act}(\mathbf{W}_1 + \mathbf{W}_0 \mathbf{q} + \mathbf{b}^1) \quad (3.16)$$

$$\text{location-based} \quad a(\mathbf{k}_i, \mathbf{q}) = a(\mathbf{q}) \quad (3.17)$$

$$\text{feature-based} \quad a(\mathbf{k}_i, \mathbf{q}) = \mathbf{w}_{\text{imp}}^T \text{act}(\mathbf{W}_1 \phi_1(\mathbf{K}) + \mathbf{W}_2 \phi_2(\mathbf{K}) + \mathbf{b}) \quad (3.18)$$

A further approach is to use a similarity measure (equation 3.5). In the paper [64] referred to by [50, 60] the cosine measure is used to compute the compatibility of  $\mathbf{q}$  and  $\mathbf{k}_i$

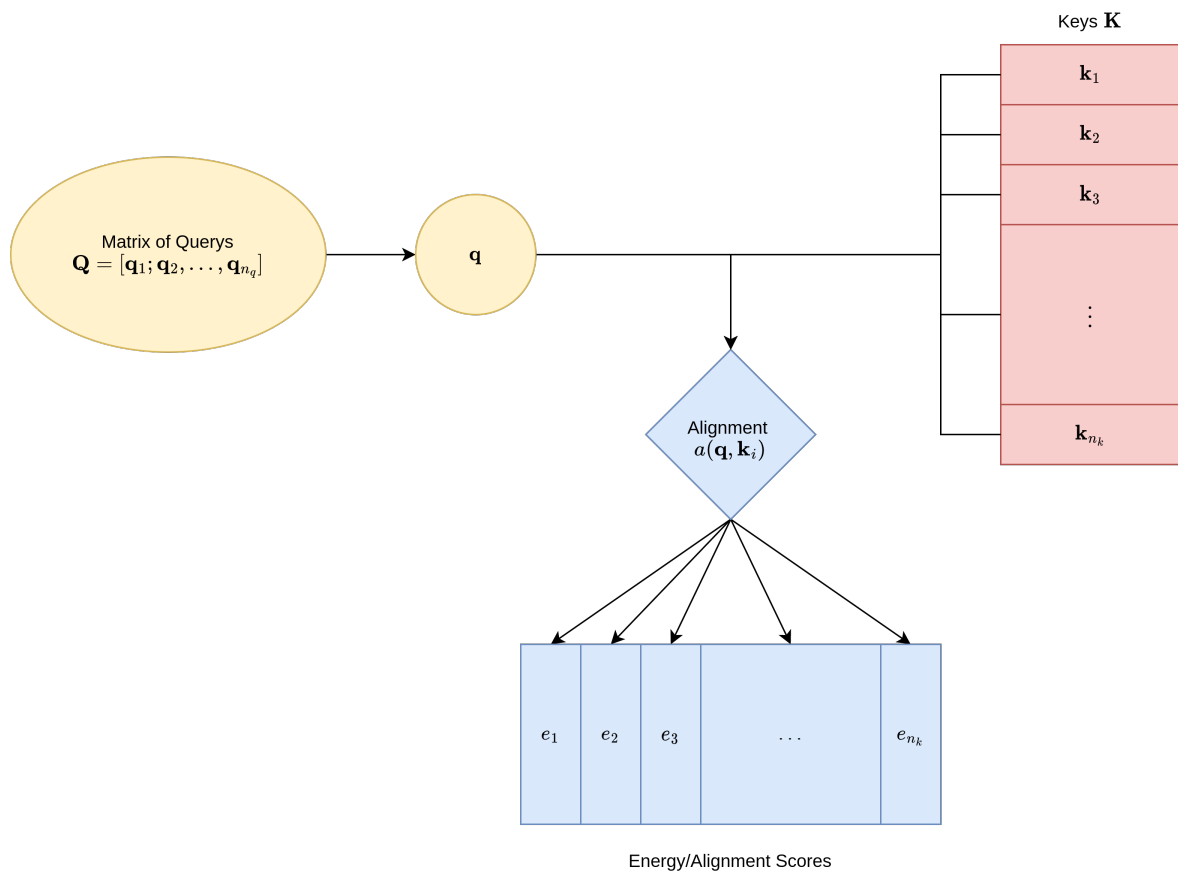
$$\text{sim}(\mathbf{k}_i, \mathbf{q}) = \frac{\mathbf{q}^T \mathbf{k}_i}{\|\mathbf{q}\| \|\mathbf{k}_i\|}$$

with

$$\text{sim} : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow [-1, 1]$$

Though this measure is widely known as *cosine similarity*, it should be noted by some rather recent taxonomy of similarity and dissimilarity measures it can not be considered a similarity measure. Clearly, the cosine measure is neither in agreement with the non-negativity nor with the minimum principle which has been introduced in preliminary section [Basic Similarity](#). The paper [27] provides a comprehensive taxonomy on similarity and dissimilarity measures, which can give inspiration for other approaches to implement alignment functions. Further, also more complex objects could be processed considering the similarity and dissimilarity properties.

The *generalized kernel* method estimates the dot product by mapping  $\mathbf{q}$  and  $\mathbf{k}_i$  into higher dimensional spaces using random feature map kernels [65](equation 3.11). This method was introduced as an improvement on the transformer neural networks, which are called *performers*. Attention usually suffers from its high computational cost for estimating the attention weights. Therefore many heuristic restrictions were made to cut down on the computational cost. The generalized kernel method introduced by performers has been proven



**Figure 3.7:** Illustration of a basic attention mechanism having some query  $\mathbf{q}$  and scoring it against keys  $\mathbf{k}_i$  with the alignment function  $a(\mathbf{q}, \mathbf{k}_i)$ . The result is an alignment value  $a_i$ .

to perform at least equally accurate to transformers in *almost* linear time and sub-quadratic memory consumption, while transformers scale quadratically. A brief introduction on kernels can be found in preliminary section [Kernel](#).

Another category of alignment functions is the *combination* of query and key. The first method is concatenating the query with the key into a unified vector  $[\mathbf{q}; \mathbf{k}_i]$ . The concatenation is then mapped by a learnable matrix  $\mathbf{W}$  and further mapped by an activation function. The resulting vector is then multiplied with the learnable importance weight  $\mathbf{w}_{\text{imp}}^T$  ([equation 3.12](#))[61]. Instead of concatenating, also the separate processing with distinct matrices  $\mathbf{W}_1$  and  $\mathbf{W}_2$  is possible, to then do the same processing as in the previous method([equation 3.13](#)) [2]. Doing this processing of the activated addition multiple times is then referred as the *deep* method ([equation 3.14](#))[66].

Also, it is possible to calculate alignment scores solely from the query itself, which is known as *location-based* attention [61]([equation 3.17](#)). Concerning *feature-based* attention, the improvement is made on the mechanic, that attention usually considers a set of items for scoring values with items having a fixed granularity size. For instance, in a [natural language processing \(NLP\)](#) learning task, there each letter can be an item or token. Clearly, this is in principle not ideal for scoring attention values learning semantics on a character based level.

Hence, feature-based attention learns to score by grouping these items with structural affiliation. As illustrated in [equation 3.18](#), only the key is used to calculate the attention score [\[67\]](#).

### Distribution Function<sup>11</sup>

The distribution function the energies calculated by the alignment function and maps them into a distribution space. We aim to generalize the notion of distribution function with respect to attention, though some used distribution functions do not hold the given definition.

#### Definition 3.25 (Distribution Function)

The distribution function  $p_i$  maps a specific energy  $e_i \in e$  over all energies  $e \in \mathbb{R}^{d_k}$  an importance value or weight with respect to the key  $k_i$  [\[50, 60\]](#)

$$p : \mathbb{R}^{d_k} \longrightarrow \Delta^{d_k}$$

$$p_i : \mathbb{R} \longrightarrow [0, 1]$$

with the property

$$\sum_{e_i \in e} p(e_i) = 1 \quad \text{(Normalization)}$$

with  $\Delta^{d_k}$  being the  $d_k - 1$ -dimensional probability simplex [\[20, 69\]](#) as introduced in the preliminary section [Affine Spaces and Simplices](#).

#### Example 3.26 (Distribution Functions)

Some examples of distribution functions are listed below and will be explained in the following text.

logistic sigmoid	$p_i(e) = \frac{1}{1 + \exp(-e_i)}$	(3.19)
------------------	-------------------------------------	--------

softmax	$p_i(e) = \frac{\exp(e_i)}{\sum_{e_j \in e} \exp(e_j)}$	(3.20)
---------	---	--------

sparsemax	$p(e) = \operatorname{argmin}_{p \in \Delta^{d_k}} \ p - e\ ^2$	(3.21)
-----------	---	--------

entmax	$p_\alpha(e) = \operatorname{argmax}_{p \in \Delta^{d_k}} p^T e + H_\alpha^T(p)$	(3.22)
--------	--	--------

Different distribution functions are listed in [example 3.26](#). Note, *logistic sigmoid* given in [equation 3.19](#) does not hold the normalization property given in [definition 3.25](#) for the general case<sup>12</sup>[\[7, p.177\]\[70\]](#). *Softmax* can be considered the normalized logistic sigmoid in the general case of more than 2 variables. This distribution function can be considered as *dense*

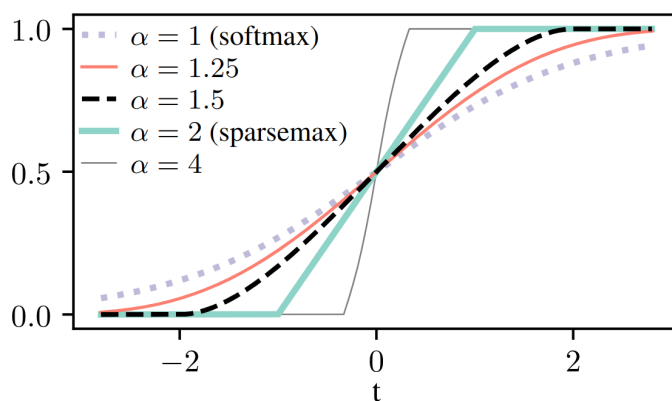
<sup>11</sup>Also referred to as aggregation function in other sources [\[50, 60\]](#). The term *aggregation function* does not actually hold the definition, violating the domain/codomain and non-decreasing property [\[68, p.3\]](#). Though some properties of an aggregation function apply, it is important for clarity to point out the imprecise usage of this term.

<sup>12</sup>Logistic sigmoid can be considered normalized in the use-case of Bernoulli distributions[\[7, p.177\]](#)

*distribution function* [50]. A dense distribution function produces for every possible outcome an output, which we refer to as probability from here on, which is always greater than 0, since<sup>13</sup>

$$\text{softmax} : \mathbb{R}^d \rightarrow (0, 1)^d \subset \Delta^d$$

There are use-cases where this is an undesired property, and it is required to truncate probable outputs. This can be the case for NLP applications, where in certain scenarios specific words or characters are practically impossible. Assume that the first two letters produced by the decoder of an encoder-decoder RNN are “Th”. In English, it’s highly unlikely, if not impossible, for the letter “z” to follow. Dense distribution functions would still compute a probability greater than 0. Though, it is possible to truncate these by a threshold [69], *sparse distribution functions*, i.e. sparsemax and entmax, deal with this issue within the function itself.



**Figure 3.8:** Illustration showing the transition of distribution functions with different  $\alpha$  parameters. Taken from [20]

In the derivation of sparsemax in equation 3.21 and entmax in equation 3.22, the entropy is used as a regularizer such that *maximum entropy principle* can be used. These concepts are introduced in a more comprehensive manner in preliminary section Entropy. In [20], it is shown how to reformulate distribution functions with the help of entropy solving an optimization problem as shown by the maximum entropy principle shown in definition 2.14. The optimization problem can be solved using Lagrangian multipliers<sup>14</sup> with inequality constraints, yielding the suggested sparse distribution functions. This can be seen in appendix A.2 of [20]. Clearly, entmax uses the Tsallis entropy. Just as the Tsallis entropy  $H_q^T = H_\alpha^T$  is able to shift between different entropies through the parameter  $\alpha$ , the entmax distribution function is also capable of shifting between different distribution functions. By setting  $\alpha = 1$ , we yield the softmax function and for  $\alpha = 2$  sparsemax. Hence, by increasing  $\alpha$ , we increase the sparsity of the distribution function as can be seen in figure 3.8

<sup>13</sup>Due to summing over exponential values, the mapping can never become 0 or 1 and will always be in between.

<sup>14</sup>In the derivation, convex analysis is crucial for understanding all the intricacies involving the optimization itself and the requirements to apply such optimization methods. For a more comprehensive analysis for the suggested sparse functions and the reformulation into maximum entropy problems, [20, 30, 71] are recommended resources.



### 3.2.4 Attention Taxonomy

In the previous subsection [Generalized Attention](#), we described how attention can be constructed mathematically. These constructions can be classified considering different properties of the attention model. We conclude the introduction of attention with such a classification. We will consider 4 categories, namely *number of sequences*, *abstractions*, *positions* and *representations*. If not otherwise cited, this taxonomy will refer to [50].

#### Number of Sequences

In the classification of *number of sequences*, we consider how many sequences are fed into an [AM](#). First, we consider **distinctive attention**, where two different sequences are used. One sequence will represent *keys* and the other one will represent *queries*. These can usually be found in translation tasks, where the model tries to learn aligning some input sentence to some output sentence. Bahdanau used this technique when introducing [AM](#) into [RNN](#). [Remark 3.21](#) explains the attention construction. Here, the input is the context vector  $h_j$  created by the sequentially processed input  $x_j$ . The context vector represents the key  $k = h_j$ . It is then aligned to the output sequence generated by the [RNN](#)  $s_{i-1}$ , which is the representation of the query  $q$ . Bahdanau states it as follows: the attention weight “[...] scores how well the inputs around position  $j$  and the output at position  $i$  match.”

In a **co-attention** model, multiple input sequences are used at the same time, learning attention weights together. In the source [72], a [visual question answering \(VQA\)](#) learning task is described where a model is given a question along with a picture and is expected to provide a correct answer in relation to the image. Consider an image with a traffic light; then a question could be given as “What color does the traffic light have?”. In this paper, the authors propose to not only use attention on the image, but also on the input sequence. Therefore, two different input sequences are used for the [AM](#). Both input sequences, image and question, will be aligned, such that the model becomes more robust against different variations of the same question, i.e. the model learns to extract the most important information of the sentence to properly output the correct answer.

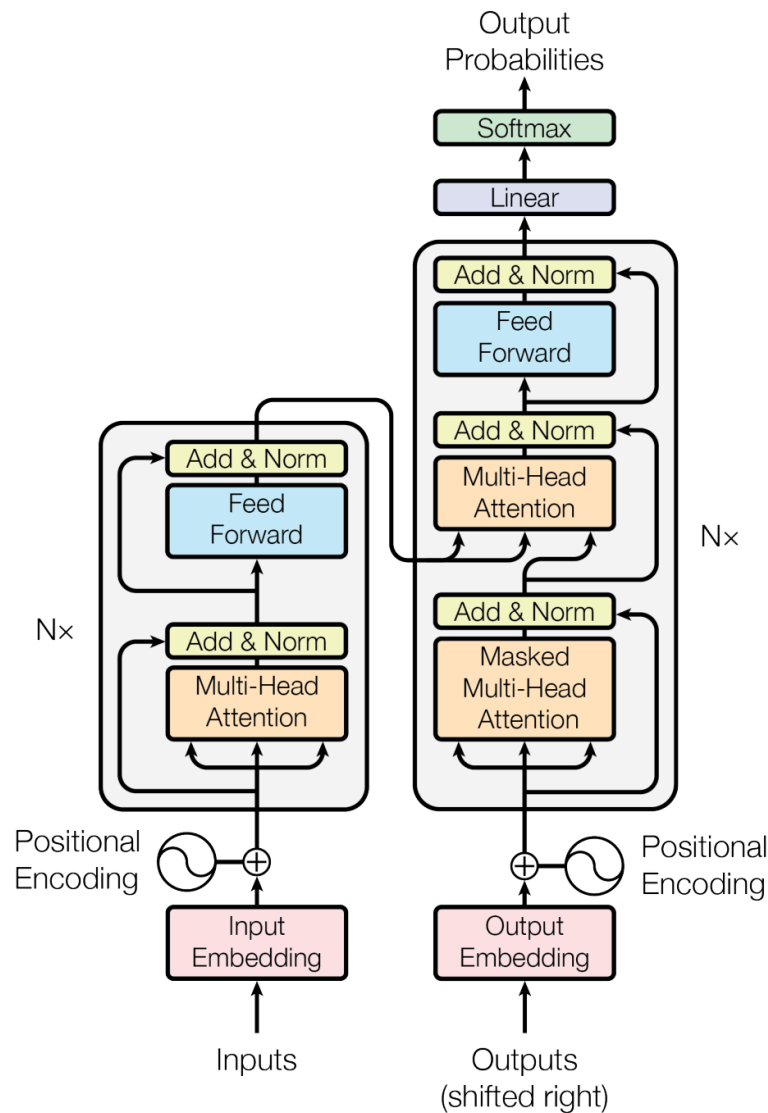
Lastly, there are also **self attention** models (or also inner attention or intra attention). In these models, both *query* and *key* are taken from the same sequence. That is, only one sequence is used for composing such an [AM](#). The most famous model using this category is the transformer network as it was introduced in [3]. Here, multiple attention layers employ linearly transformed representations of the same sequence to determine *queries* and *keys*. Such a model is used to capture dependencies between a sample of a sequence and all the other corresponding sequence elements.

#### Number of Abstraction Levels

Attention can be used on different abstraction layers. Considering the *co-attention* model in [72], the alignment is computed on different abstraction layers, i.e. on a word, phrase<sup>15</sup> and sentence level. In this work, a *hierarchical model* is proposed, where the computed attention

<sup>15</sup>We consider here a phrase to be a sequence of 1,2 or 3 consecutive words in a sentence.

on the word level is combined with the on top sitting abstraction layer, hence the phrase level. The same mechanism applies with the phrase level and the on top sitting sentence level. Such an interdependence in the computation of attention between different abstraction levels is called **multi-level attention**.



**Figure 3.9:** In the transformer *multi-level attention* and *multi-representational attention* can be observed. Since each encoder or decoder layer are stacked  $N \times$  times, attention is computed at different abstraction layers, hence *multi-level attention*. Each encoder or decoder layer makes use of *multi-head attention* which is categorized as *multi-representational attention*. Figure is taken from [3].

Another model, which can be considered applying *multi-level attention*, is the transformer network. In the transformer network, multiple layers of attention are stacked, such that we have a *hierarchical architecture* with different levels of abstraction [3] (See figure 3.9).

If a model computes attention only on a single input sequence, without embedding any other attention abstraction layers, it is called **single-level attention**.

## Number of Positions

Another category considers the positions of the input sequence from which attention is calculated. If generally all elements of the sequence are used for calculating the attention values, it is considered to be **global attention** or also **soft attention**. Since such an approach can be computationally very expensive due to its quadratic memory consumption and time complexity, there is also the approach to use a sliding window over a sequence, such that only a limited part of the sequence is used for the computation of the attention values [61]. Since attention is performed only on a slice of the sequence, it is called **local attention**.

In **hard attention** models, on the other hand, sequence elements are selected by some means for the attention calculation. The scope of calculation decreases and therefore the overall computational complexity decreases significantly due to the quadratic computational complexity of the attention algorithm. In [73], the selection is done stochastically. Due to this procedure of stochastically selecting sequence elements, the model loses its differentiability.

## Number of Representations

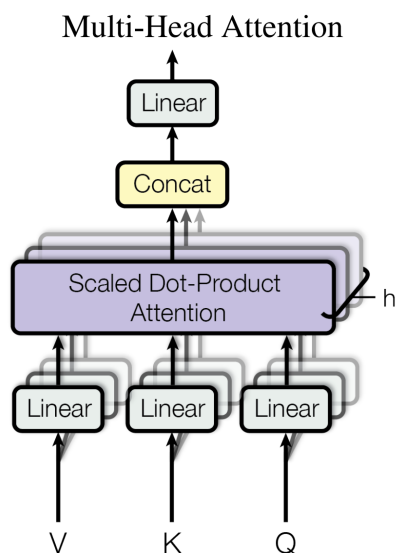
**Multi-representational AM** take different representations of the same input and integrate the corresponding outputs. The input is transformed by some means like linear mappings to extract distinct features, such that the attention algorithm can extract different features. In the transformer network in an attention layer, a *multi-head attention* computation is performed [3]. An attention-head  $H_i$  is an independent set of learnable attention parameters. Consider that an attention layer consists of  $h$  different attention-heads and that  $Q_i, K_i, V_i$  represent the *query*, *key*, and *value* matrices respectively. Then each *attention-head* receives a different set of linearly transformed queries, keys and values by using distinct linear mapping  $W_i^Q, W_i^K$  and  $W_i^V$ . These linear projections are dimensionally reduced queries, keys and values [34]. The inputs to the attention-heads therefore are different abstractions of the original input matrices  $Q, K$  and  $V$ . The resulting vectors are then concatenated and then again projected which represents the integration of the outputs (see figure 3.10).

Another variation is **multi-dimensional** attention. Here attention weights  $\alpha$  are calculated for each feature dimension separately. In [74], the authors proposed modifications to RNN-based attention embeddings. Consider the following attention implementation where a key  $x_i (= k_i)$  is being aligned to some query  $q$

$$A(q, (x_i)_{i=1}^n, w) = \sum_{i=1}^n w^T \sigma(W^{(1)}x_i + W^{(2)}q)$$

where  $W^{(1)}, W^{(2)}$  are linear transformations and  $w^T$  represents the value vector of the general attention scheme. By choosing a linear mapping  $W$  instead of a value vector, the result is a vector  $\alpha$  rather than scalars. In this setup, each component,  $\alpha_i$  of  $\alpha$ , represents an attention weight specific to a dimension.

If none of the multi-representational classifications apply this category is omitted.



**Figure 3.10:** Multi-head attention is displayed. Different layers of *Linear* transformation and *Scaled Dot-Product Attention* can be seen. Each layer forms an attention-head performing attention calculations independently. Figure is taken from [3].

*Remark 3.27*

The original attention based encoder-decoder model according to [proposition 3.18](#) can be categorized into *soft, distinctive, single-level attention*[50].

## 4 Learning Vector Quantization

### 4.1 Introduction

**Learning Vector Quantization (LVQ)** is a prototype based classification machine learning algorithm originally introduced by Teuvo Kohonen in 1990 [75, pp.245]. Given a labeled dataset  $D = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \dots, C\} \mid i = 1, \dots, m\}$  with  $C$  different classes and  $m$  data samples. For each given class of the dataset prototypes  $\mathbf{w} \in \mathbb{R}^n$  are initialized affiliated to the given class, hence the class affiliation is given by  $c(\mathbf{w}) \in \{1, \dots, C\}$ . The objective is to place the prototypes optimally in  $\mathbb{R}^n$  such that they discriminate the data samples with respect to their labels. Prototypes are relocated during the learning process to improve the data representation.

Further we discuss the original **LVQ1** algorithm introduced by Kohonen. It gives a good intuition on how **LVQ** algorithms generally work. If not otherwise noted, information is taken from [75, pp.245].

#### Definition 4.1 (LVQ1 Prediction)

Let  $\mathbf{x} \in \mathbb{R}^n$  be some arbitrary data sample for which the class affiliation  $c(\mathbf{x}) \in 1, \dots, C$  is to be predicted by the **LVQ1** classifier. Further,  $\mathbf{w} \in W \subseteq \mathbb{R}^n$  are prototypes with a fixed class affiliation  $c(\mathbf{w}) \in \{1, \dots, C\}$ . Predicting the class affiliation of  $\mathbf{x}$  is given by

$$\begin{aligned} \mathbf{w}_s &= \arg \min_{\mathbf{w} \in W} d_E(\mathbf{x}, \mathbf{w}) \\ &= \arg \min_{\mathbf{w} \in W} \|\mathbf{x} - \mathbf{w}\|_E \end{aligned}$$

with  $d_E$  being the Euclidean distance,  $\|\cdot\|_E$  being the Euclidean norm and  $\mathbf{w}_s$  being the closest prototype to the given data sample  $\mathbf{x}$ . The data sample is predicted to be  $c(\mathbf{x}) = c(\mathbf{w}_s)$ .

#### Remark 4.2

These kinds of algorithms are generally called **Winner Takes All (WTA)** competitions [6, p.425] [75, p.XI]. Here, all prototypes compete to be closest to a given data sample by some distance measure, to be the winner of the competition. In **definition 4.1**  $\mathbf{w}_s$  is the winning prototype.

For predictions to have a high accuracy, prototypes must discriminate optimally. Therefore, a learning scheme for the prototypes is defined. Over a period of time steps (also known as *epochs*) random data samples are picked from the dataset  $D$  to challenge the prototypes on how well they discriminate input data samples.

#### Definition 4.3 (LVQ1 Learning Scheme)

Let  $\mathbf{x}(t) \in \mathbb{R}^n$  be a data sample at time  $t$  and  $\mathbf{w}_s(t) \in W$  the winning prototype of the **WTA** competition with class affiliation  $c(\mathbf{w}_s(t)) \in \{1, \dots, C\}$ . Then the update to the prototypes

is described as follows

$$\begin{aligned} \mathbf{w}_s(t+1) &= \mathbf{w}_s(t) - \Delta \mathbf{w}_s \\ \Delta \mathbf{w}_s &= -(\mathbf{x} - \mathbf{w}_s)\gamma(\mathbf{x}, \mathbf{w}_s) \end{aligned}$$

with  $\gamma$  being a function returning if the classes of the data sample  $\mathbf{x}$  and  $\mathbf{w}_s$  match.

$$\gamma(\mathbf{x}, \mathbf{w}_s) = \begin{cases} 1 & \text{if } c(\mathbf{w}_s) = c(\mathbf{x}) \\ 0 & \text{else} \end{cases}$$

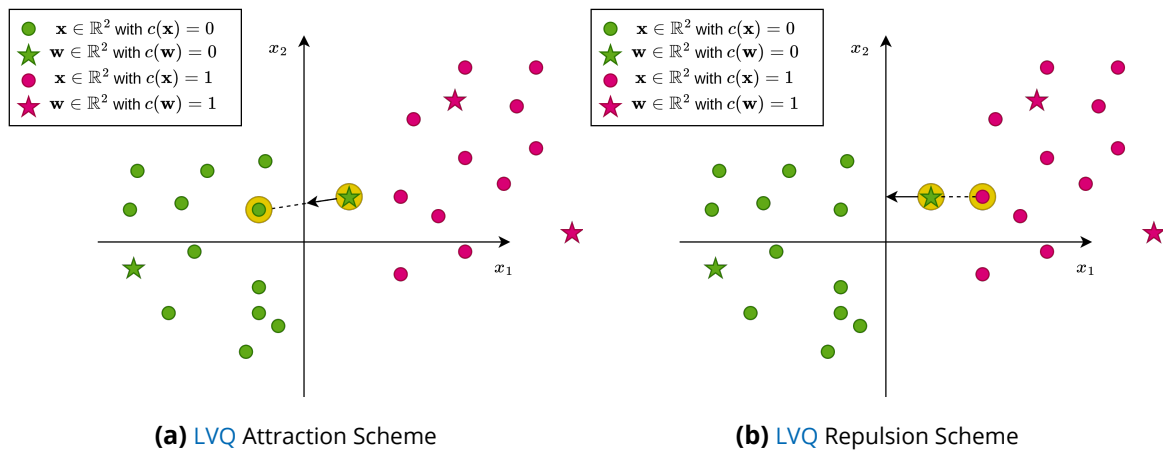
*Remark 4.4*

Contrary to the chapter about attention and the processing of timeseries,  $t$  is the timestep in the learning process. The prototypes will be updated along the learning process, and they have different states at different times  $t$ .

Because the algorithm is a heuristic and there is no differentiable loss function, there is no mathematical guarantee for convergence [76].

*Remark 4.5*

The relocation process is usually differentiated in *repulsion* and *attraction* (see figure 4.1). A prototype is *repelled* if during the learning process for a random data sample  $\mathbf{x}$  and the winning prototype  $\mathbf{w}_s$  we have  $\gamma(\mathbf{x}, \mathbf{w}_s) = 0$ , and it is *attracted* if we have  $\gamma(\mathbf{x}, \mathbf{w}_s) = 1$ .



**Figure 4.1:** Attraction and repulsion visualization with LVQ. The randomly chosen data sample  $\mathbf{x}$  and the closest prototype  $\mathbf{w}$  are highlighted in yellow.

## 4.2 Generalized Learning Vector Quantization

To extend LVQ to achieve the guarantees which SGD provides, Sato and Yamada improved LVQ by integrating a differentiable cost function in 1995 [77]. The resulting algorithm is named Generalized Learning Vector Quantization (GLVQ).

**Definition 4.6** (GLVQ Classifier Function)

Let  $\mathbf{x}, \mathbf{w}_x^+, \mathbf{w}_x^- \in \mathbb{R}^n$  with  $\mathbf{w}_x^+$  being the closest prototype to  $\mathbf{x}$  with the same class affiliation and  $\mathbf{w}_x^-$  being the closest prototype to  $\mathbf{x}$  with a different class affiliation. Further, denote the following distances

$$d^+ = d(\mathbf{x}, \mathbf{w}_x^+) \quad \text{(Distance Function of Same Class)}$$

$$d^- = d(\mathbf{x}, \mathbf{w}_x^-) \quad \text{(Distance Function of Different Class)}$$

The *classifier function*  $\mu$  is given as [77]

$$\mu(\mathbf{x}) = \frac{d^+ - d^-}{d^+ + d^-}$$

With the help of the classifier the following cost function can be given.

**Definition 4.7** (GLVQ Cost Function)

Let  $\phi$  be some monotonically increasing, non-linear activation function (see [Activation Function](#)), then the cost function can be given as [77]

$$E_{\text{GLVQ}} = \sum_{i=1}^m \phi(\mu(\mathbf{x}_i))$$

with

$X \subseteq \mathbb{R}^n$  ... being the training set

$W \subseteq \mathbb{R}^n$  ... being the set of prototypes

$m$  ... being the number of training vectors

$f_{\text{GLVQ}}$  ... being the model determining the winning prototypes

$\phi(\mu(\mathbf{x}_i))$ ... local cost.

By having a differentiable and, with respect to the data samples  $\mathbf{x}$ , separable cost function, we can properly apply [SGD](#). Therefore, the local gradient can be used for giving the update rule of the learning process.

**Definition 4.8** (GLVQ Update Rule)

Given the same conditions as in [GLVQ Classifier Function](#), the update rule for the winning prototypes can be given as

$$\mathbf{w}^\pm = \mathbf{w}^\pm - \alpha \frac{\partial \phi}{\partial \mathbf{w}^\pm}$$

with  $\mathbf{w}^\pm \in \{\mathbf{w}^+, \mathbf{w}^-\}$  and  $\alpha$  being the learning rate. Assuming the distance measure  $d^\pm$  is chosen to be the squared Euclidean distance  $d_E^2$ , hence

$$d^\pm = d_E(\mathbf{x}, \mathbf{w}^\pm)^2 = \|\mathbf{x} - \mathbf{w}^\pm\|_E^2$$

then the update rule can be given as

$$\begin{aligned}\mathbf{w}^+ &= \mathbf{w}^+ + \alpha \frac{\partial \phi}{\partial \mu} \frac{d^-}{(d^+ + d^-)^2} (\mathbf{x} - \mathbf{w}^+) \\ \mathbf{w}^- &= \mathbf{w}^- - \alpha \frac{\partial \phi}{\partial \mu} \frac{d^+}{(d^+ + d^-)^2} (\mathbf{x} - \mathbf{w}^-)\end{aligned}$$

### 4.3 Generalized Matrix Learning Vector Quantization

For these LVQ algorithms, the assumption is made that the data samples are distributed isotropic clusters, meaning the variance of the data samples are roughly the same across all dimensions [78][7, p.66]. A conventional metric is not able properly learn on the inconsistent scaling of such non-isotropic data distributions. In [Generalized Matrix Learning Vector Quantization \(GMLVQ\)](#) a special metric is introduced to counter this issue.

#### Definition 4.9 (GMLVQ Distance)

Let  $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$  and  $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$  being a *positive definite matrix*. The GMLVQ distance is given as

$$d_{\mathbf{\Lambda}}(\mathbf{x}, \mathbf{w}) = (\mathbf{x} - \mathbf{w})^T \mathbf{\Lambda} (\mathbf{x} - \mathbf{w})$$

with  $\mathbf{\Omega} \in \mathbb{R}^{m \times n}$  in the decomposition

$$\mathbf{\Lambda} = \mathbf{\Omega}^T \mathbf{\Omega}$$

For the learning process, the same cost function as in [GLVQ \(definition 4.7\)](#) is used. Next to the prototypes, also the projection matrix  $\mathbf{\Omega}$  has to be learned.

#### Definition 4.10 (GMLVQ Gradients)

Let all parameters be as given in [GMLVQ Distance](#) and [GLVQ Classifier Function](#). The gradient for a matrix component  $\Omega_{ij} \in \mathbf{\Omega}$  is given as

$$\begin{aligned}\frac{\partial \phi}{\partial \Omega_{ij}} &= \frac{\partial \phi}{\partial \mu} \frac{\partial \mu}{\partial d_{\Omega}^{\pm}} \frac{\partial d_{\Omega}^{\pm}}{\partial \Omega_{ij}} \\ &= -2\alpha \frac{\partial \phi}{\partial \mu} \cdot \left( \mu^+(\mathbf{x}) \left( (x_j - w_j^+) (\mathbf{\Omega}(\mathbf{x} - \mathbf{w}^+)) \right)_i \right. \\ &\quad \left. - \mu^-(\mathbf{x}) \left( (x_j - w_j^-) (\mathbf{\Omega}(\mathbf{x} - \mathbf{w}^-)) \right)_i \right)\end{aligned}$$

with

$$\frac{\partial d_{\Omega}}{\partial \Omega_{ij}} = 2(x_j - w_j) (\mathbf{\Omega}(\mathbf{x} - \mathbf{w}))$$



and the gradient for the winning prototypes  $\mathbf{w}^\pm$  [79] is given as

$$\begin{aligned}\frac{\partial \phi}{\partial \mathbf{w}^+} &= 2\alpha \frac{\partial \phi}{\partial \mu^+} \cdot \mu^+(\mathbf{x}) \Lambda(\mathbf{x} - \mathbf{w}^+) \\ \frac{\partial \phi}{\partial \mathbf{w}^-} &= -2\alpha \frac{\partial \phi}{\partial \mu^-} \cdot \mu^-(\mathbf{x}) \Lambda(\mathbf{x} - \mathbf{w}^-)\end{aligned}$$

*Remark 4.11*

To prevent the algorithm from degeneration it is suggested to normalize the matrix  $\Lambda$  with conditioning the diagonal to [78, 79]

$$\sum_i \Lambda_{ii} = 1$$

## 5 Attention Learning Vector Quantization

A successful LVQ model is dependent on a (dis-)similarity measure for prototypes to be optimally placed with respect to the data samples. Timeseries data are difficult to use with the basic principle LVQ algorithms use. First, time series data can be of different lengths. This property makes conventional dissimilarity measures, like the Euclidean distance, practically useless, since they rely on data samples to be of same length, i.e. same dimensionality. Jain and Schultz managed to extend the LVQ algorithm to accommodate for time series of different lengths by making use of [Dynamic Time Warping \(DTW\)](#) as a distance measure<sup>16</sup>[5]. In DTW, points of time are aligned by warping them according to a cost function creating a warping path which accumulates to a cost which is interpreted as a distance between two time series. Hereby, we can identify the principle of dynamic timewarping as aligning two time series sequentially. This procedure is suitable for data which in classification tasks are very similar and differ mostly in their warping properties.

A second difficulty is the interdependency of different time points in time series. As pointed out in [Prerequisites](#), RNN architectures try to capture these beyond the sequential alignment used in DTW. Ravichandran et al. introduced a recurrent architecture in the LVQ learning scheme [4]. Here an RNN network transforms a time series  $(\mathbf{x}_i)_{i=1}^{d_{in}}$  to a vectorial representation of fixed length; i.e.  $Rec((\mathbf{x}_{i=1})^{d_{in}}) = \tilde{\mathbf{x}}$  with  $d_{in}$  being the length of the sequence and  $d_x$  being the dimensionality of the vectors of the sequence. Prototypes are chosen as timeseries aswell and are mapped through the network like the data samples are.

As pointed out earlier, RNN architectures have the issue of data samples being inserted sequentially resulting in slow computation depending on the length of the timeseries. This thesis makes the attempt to utilize findings from transformer networks and the AM. First, we will introduce *positional encoding* which made transformers[3] overcome the limitation of feeding data sequentially to models processing sequential data and then proceed with describing the attention based LVQ models. In RNN models positional information is captured by the recursive computation, hence, the sequential input. In transformers positional information is encoded based on the dimensional component and the position of a vector in the given sequence<sup>17</sup>.

### Definition 5.1 (Positional Encoding)

Let  $\mathbf{X} = (\mathbf{x}_i)_{i=1}^{d_{in}} \subset \mathbb{R}^{d_x \times d_{in}}$  be the matrix representation of a zero-padded data sequence, i.e. setting a fixed length and filling it up with zero vectors  $\mathbf{0} \in \mathbb{R}^{d_x}$ . The *positional encoding*

<sup>16</sup>Strictly speaking, DTW, by theoretical mathematical conditions, is not a distance measure, while empirically holding the given properties. It does not hold the triangle inequality in theory. Empirically, it actually holds the triangle inequality; therefore, it can be interpreted as *loose triangle inequality* [80, p.10][81]

<sup>17</sup>It should be noted that there is research on positional encoding itself. We omit further analysis on positional encoding and utilize the *additional positional encoding* as used in the transformer. For further interest refer to [82, 83].

matrix [34]  $P$  is given as

$$\begin{aligned} P_{(\text{pos}, 2i)} &= \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right) \\ P_{(\text{pos}, 2i+1)} &= \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right) \end{aligned}$$

The positional encoded data sequence  $\mathbf{X}_P \in \mathbb{R}^{d_x \times d_{in}}$  is given as

$$\mathbf{X}_P = \mathbf{X} + P$$

*Remark 5.2*

This type of positional encoding using adding sinus and cosinus mappings is called *sinusoidal position embeddings*. Since the values which are added are fixed this kind of positional encoding is considered *absolute position encoding* [82].

The positional encoded data matrix  $\mathbf{X}_P$  is then used to learn the components needed for the **LVQ AM**. Going forward from here, different attention based **LVQ** models will be introduced. There will be two versions: one **AM** for **GLVQ** and another for **GMLVQ**. Since **AM** suffer from quadratical time and space complexity, and timeseries data can be long in sequence length, the computational burden can grow very fast. Therefore, downsampling layers will be introduced into the models at different points of the algorithm.

## 5.1 Post Attention Downsampled Learning Vector Quantization

**Definition 5.3** (LVQ Attention Layer)

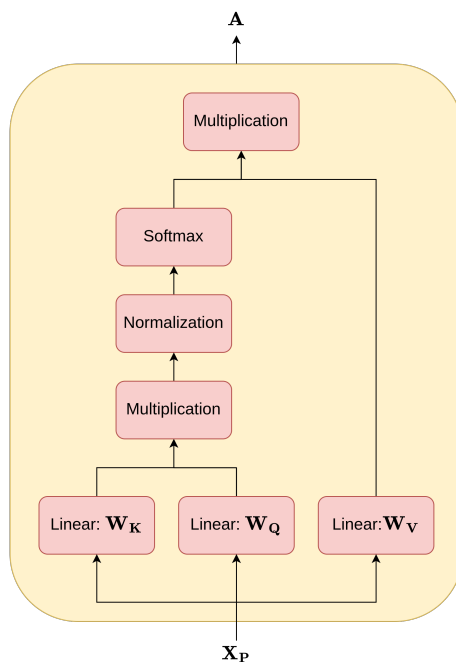
Let  $\mathbf{X}_P \in \mathbb{R}^{d_x \times d_{in}}$  be the positionally encoded data matrix. Further, denote  $\mathbf{W}_K, \mathbf{W}_Q \in \mathbb{R}^{d_k \times d_x}$  and  $\mathbf{W}_V \in \mathbb{R}^{d_v \times d_x}$  the linear transformation matrix for keys, queries and values, where all matrices denoted with  $\mathbf{W}$  are learnable parameters. Further, denote  $\mathbf{b}_K, \mathbf{b}_Q \in \mathbb{R}^{d_k}$  and  $\mathbf{b}_V \in \mathbb{R}^{d_v}$  as the respective learnable biases. Keys, queries  $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{d_k \times d_{in}}$  and values  $\mathbf{V} \in \mathbb{R}^{d_v \times d_{in}}$  are calculated as

$$\begin{aligned} \mathbf{K} &= \mathbf{W}_K \mathbf{X}_P + \mathbf{b}_K \mathbf{1}^T \\ \mathbf{Q} &= \mathbf{W}_Q \mathbf{X}_P + \mathbf{b}_Q \mathbf{1}^T \\ \mathbf{V} &= \mathbf{W}_V \mathbf{X}_P + \mathbf{b}_V \mathbf{1}^T \end{aligned}$$

The attention matrix  $\mathbf{A} \in \mathbb{R}^{d_v \times d_{in}}$  is calculated as

$$\mathbf{A} = \mathbf{V} \cdot \text{softmax}\left(\frac{\mathbf{K}^T \mathbf{Q}}{\sqrt{d_k}}\right)$$

Figure 5.1 illustrates the processing steps. In the upcoming processing, it is planned to stack the entries of the attention matrix into a vectorial representation. Since it can be expected that the attention matrix to be big in dimensionality a downsampling step for dimensionality reduction is included to make the integration into **LVQ** computationally feasible. Especially in the **GMLVQ** version we will initialize another matrix where the input dimensionality will



**Figure 5.1:** Illustration of the processing steps according to 5.3

have input size of the vectorized matrix. This can easily blow up memory requirements to infeasible sizes, which is why downsampling, i.e. dimensionality reduction, is necessary at some point in this algorithm.

**Definition 5.4** (Post Attention Downsampling Layer)

Let  $\mathbf{W}_D \in \mathbb{R}^{d_d \times d_{in}}$  be the downsample matrix with the respective  $\mathbf{b}_D \in \mathbb{R}^{d_d}$  bias which both are learnable parameters. The downsampling is given as

$$\mathbf{D} = \sigma(\mathbf{W}_D \mathbf{A}^T + \mathbf{b}_D \mathbf{1}^T)$$

with  $\sigma$  being some non-linear activation function and  $\mathbf{D} \in \mathbb{R}^{d_d \times d_v}$ . The hyperparameter  $d_d$  configures the downsample dimensionality.

*Remark 5.5*

The attention matrix might hold non-linear information. Given the assumption that these exist, we want to capture the non-linearities by adding the non-linear activation function into the processing steps. It should be noted, that this processing step is expected to lose interpretability to some extent.

Now the downsampled matrix  $\mathbf{D}$  should be vectorized, therefore we define the following function.

**Definition 5.6** (Matrix Vectorization)

Let  $\mathbf{C} \in \mathbb{R}^{n \times m}$  be a matrix with  $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_m]$  and  $\mathbf{c} \in \mathbb{R}^n$ . We define the matrix vectorization as

$$\text{vec} : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{n \cdot m}$$

$$\text{vec}(\mathbf{C}) = \begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_m \end{pmatrix}$$

*Remark 5.7*

In [Matrix Vectorization](#) a matrix is considered as a sequence of vectors. These vectors are stacked in the order they are positioned in the matrix.

At this point, everything is set to perform a [LVQ](#) learning scheme with the preprocessed data.

**Definition 5.8** (LVQ Layer)

Let  $f : \hat{X} \rightarrow \{1, \dots, C\}$  be an [GLVQ](#) or [GMLVQ](#) model with  $\hat{X} \subseteq \mathbb{R}^{d_{lvq}}$  being the embedded data set and  $d_{lvq} = d_d \cdot d_v$  being its dimension. Further, denote  $\hat{\mathbf{x}} \in \hat{X}$  the input to the respective [LVQ](#) model. We define

$$\hat{\mathbf{x}} = \text{vec}(\mathbf{D})$$

with  $\mathbf{D}$  being the downsampled matrix from [Post Attention Downsampling Layer](#). The vector  $\hat{\mathbf{x}}$  is normalized before parsing it into the [LVQ](#) algorithm, such that the last computation step is given as

$$f\left(\frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|_E}\right) = \hat{y}$$

with  $\hat{y}$  being the predicted class to the embedded data sample  $\hat{\mathbf{x}}$ .

*Remark 5.9*

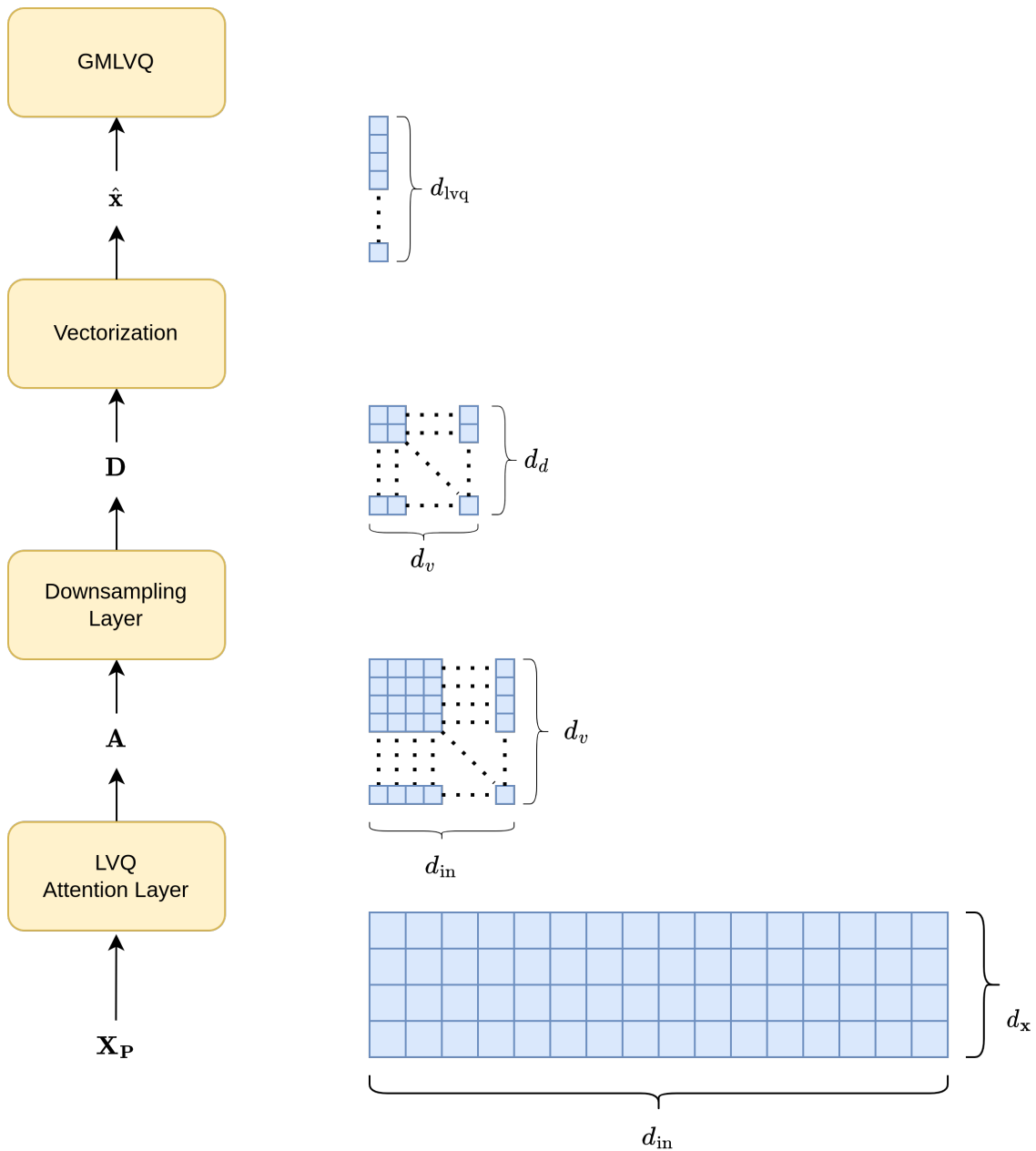
In [LVQ Layer](#) the [GLVQ](#) and [GMLVQ](#) models are initialized and calculated according to the descriptions of sections [Generalized Learning Vector Quantization](#) and [Generalized Matrix Learning Vector Quantization](#). Since all processing steps are continuous transformations, we can apply [SGD](#) to the whole learning process and backpropagate errors.

*Remark 5.10*

Contrary to conventional [LVQ](#) learning schemes the prototypes are not embedded in the same space as the original datapoints  $\mathbf{X} \in \mathbb{R}^{d_{\mathbf{x}} \times d_{in}}$  but rather in the embedding space  $\hat{\mathbf{x}} \in \mathbb{R}^{d_{lvq}}$ .

Generally speaking, Attention LVQ transforms time series of different lengths into a vectorial representation or embedding of fixed length, as can be seen in [figure 5.2](#). The attention layer should try to learn the interdependencies between different points of time in the data sample timeseries to learn patterns which are related over different sections of the series.

As mentioned earlier, [DTW](#) focuses on a sequential alignment, while attention learns to align over all the time points. The processing in [AM](#) comes at the cost that, in full resolution, i.e.  $d_{in} = d_v = d_d$ , we have a quadratic time and space complexity over the length of the time series for each of the matrix multiplications in the attention layer.



**Figure 5.2:** Processing steps in [Post Attention Downsampled Learning Vector Quantization](#). On the left-hand side the different layers are displayed which have been explained in the according definitions. On the right-hand side some exemplary data illustration with dimensionality highlighting is showed and how data is transformed during the transformation process.

## 5.2 Pre Attention Downsampled Learning Vector Quantization

Since the computations are mostly the same, except for the order of computation and the resulting dimensionality, the description will be shortened to only highlight the differences between the algorithm designs. The [Pre Attention Downsampling Layer](#) will be mostly the

same as the [Post Attention Downsampling Layer](#), except the activation function is removed, since [84] has already shown some success without the use of an activation function at this stage of processing. For this reason the pre-attention layer will be defined properly.

**Definition 5.11** (Pre Attention Downsampling Layer)

Let  $\mathbf{W}_D \in \mathbb{R}^{d_a \times d_{in}}$  the downsample matrix with the respective  $\mathbf{b}_D \in \mathbb{R}^{d_a}$  bias. The downsampling is given as

$$\mathbf{D} = \mathbf{W}_D \mathbf{X}_P^T + \mathbf{b}_D \mathbf{1}^T$$

Next, there will be the [LVQ Attention Layer](#) with some changes in the dimensionalities, which are the following

- the input to the attention layer is  $\mathbf{D} \in \mathbb{R}^{d_a \times d_d}$  instead of  $\mathbf{X}_P$
- the linear projections in the attention layer are of dimensionalities
  - $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{d_k \times d_d}$
  - $\mathbf{V} \in \mathbb{R}^{d_v \times d_d}$
  - $\mathbf{A} \in \mathbb{R}^{d_v \times d_d}$

The algorithm steps of the [Pre Attention Downsampled Learning Vector Quantization](#) version is given as

1. the positionally encoded data sample  $\mathbf{X}_P$  is input into [Pre Attention Downsampling Layer](#)
2. the downsampled matrix  $\mathbf{D}$  is input into [LVQ Attention Layer](#)
3. the matrix  $\mathbf{A}$  is being vectorized and the input to [LVQ Layer](#) is  $\text{vec}(\mathbf{A}) = \hat{\mathbf{x}}$

## 6 Experiments

### 6.1 Dataset

The dataset for this work is chosen to be as similar in its data structure to *word embeddings*<sup>18</sup> as possible, since AM models were originally introduced for NLP. In word embeddings, we have timeseries  $(x_i)_{i=1}^n$  with each  $x_i \in \mathbb{R}^k$  being some  $k$ -dimensional vector. Usually,  $k$  ranges from 50 to 1024 [85–88]. Since NLP-driven models have shown promising success using such word embeddings with  $k > 1$  in connection to attention [88], the requirement for a dataset is to have a multivariate timeseries to align closely with the described conditions.

For this reason, the *aeroelastic simulations of wind turbines affected by leading edge erosion* dataset is used [84]. This dataset contains simulated data on erosion of wind turbines, where different sensors in a rotor blade measure different attributes from which erosion should be detected. This erosion can affect the rotor’s structural integrity and/or the rotor’s performance and therefore efficacy. For this matter the data is presented in 10 different classes, each describing different degradation states of the blade and aerodynamic properties. Each degradation class can be interpreted as a severity level of degradation. The simulated timeseries contain 4 different features, i.e. *angle of attack*, *drag coefficient*, *lift coefficient* and *inflow velocity*. Each timeseries consists of 60 001 samples, where the sampling rate is 100Hz. Approximately every 6 days over 20 years, a sample is generated such that 1 200 different data-points are generated on a wind turbine. The authors provide a set of 18 preselected training- and test sets with either ballanced or unballanced class distributions (see figure 6.1). Moreover, transformers were used on this dataset, which provides a good benchmark.

### 6.2 Hardware System

The experiments will be exclusively run on a discrete *graphics processing unit (GPU)*. Specifically, a single NVIDIA A100 80 *gigabytes (GB)* SXM GPU will be used. Preprocessing steps will be run on an AMD Epyc 7713 64-Core Processor with about 2 *terabytes (TB)* of memory available.

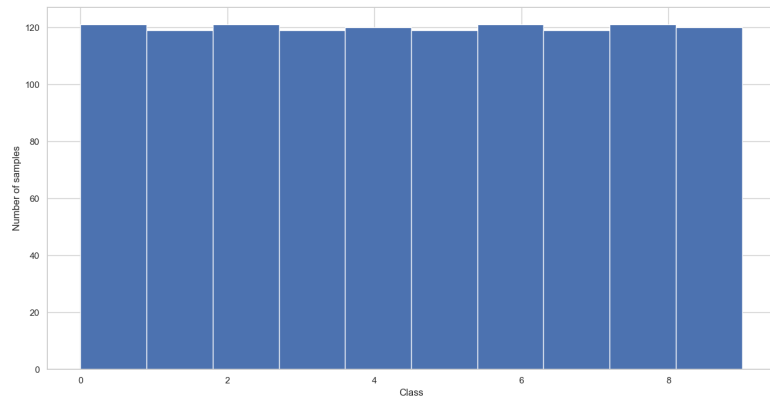
### 6.3 Data Preparation

As highlighted in *Attention Learning Vector Quantization*, attention suffers from quadratic memory and computation growth over the input length  $n$  of the sequence, i.e.  $\mathcal{O}(n^2)$ . Since in some experiments the downsampling layer follows the attention layer, some preprocessing steps need to be taken to manage memory consumption.

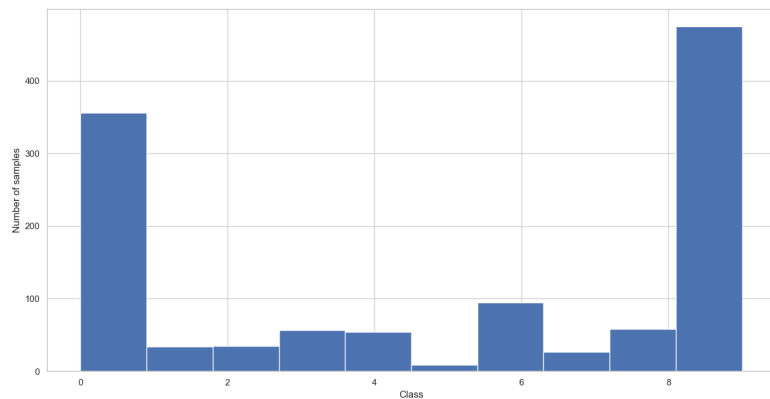
---

<sup>18</sup>Word embeddings are dense vectorial representations of words, where the vector representation has semantic meaning embedded to them [7, p.452].





(a) Balanced Distribution in Training Set 4



(b) Unbalanced Class Distribution in Training Set 2

**Figure 6.1:** Overview of Given Class Distributions

Referring the PyTorch documentation for sizes of datatypes [89], we find that a float is 32 bits or 4 bytes in size, depending on the [central processing unit \(CPU\)](#) architecture used. Consider we choose the parameters of an attention based LVQ model as follows

$$d_{in} = 60\,001$$

$$d_x = 4$$

$$d_k = 1\,024$$

$$d_v = 1\,024$$

$$d_d = 10\,000$$

$$d_{lvq} = 10\,000 \cdot 1024 = 10\,240\,000$$

The memory consumption used just by the weight matrices and the processing matrices is given in [table 6.1](#). Note that the memory requirements for the GMLVQ matrix  $\Omega$  is about 400 TB in this scenario, which renders running such a model on current hardware systems

Parameter	Size in MBytes for Post Attention LVQ	Size in MBytes for Pre Attention LVQ
$W_Q, W_K$	0.016	0.016
$W_V$	0.016	0.016
$W_D$	2288.857	2288.857
$Q, K$	234.379	39.063
$V$	234.379	39.063
$A$	234.379	39.063
$D$	0.153	0.153
$\Omega$	400 000 000	400 000 000
$\hat{x}$	39.063	39.063

**Table 6.1:** Sizes of an Exemplary Model Configuration with GMLVQ used in the LVQ Layer

completely unfeasible. Therefore, it is absolutely necessary to manage the model’s input to reduce dimensionality as much as possible either by the learnable matrix  $W_D$  or by preprocessing it before feeding it into the model.

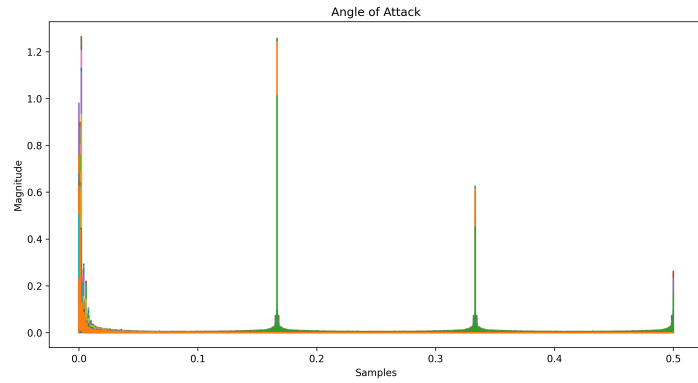
For understanding how the frequencies are distributed and how densely information is represented over the sampling frequency, a [Fast Fourier Transform \(FFT\)](#) is performed. The plots can be seen in [figure 6.2](#). FFT decomposes a signal into its base frequency components and shows the signal’s frequency distribution over a timeframe [90, pp.129, pp.49]. According to the Nyquist-Shannon theorem<sup>19</sup> [90, p.30] it is evident from [figure 6.2a](#), [6.2c](#) and [6.2d](#) that the use of a low-pass filter would erase information from the data, as frequency spikes can be found at the folding frequency<sup>20</sup>. Therefore, in some experiments, no filter is applied, while in others, only each 6-th sample is used. This reduces the length to  $\frac{1}{6}$  of its original size, i.e.  $n = 10001$ . [Figure 6.3](#) displays the impact of downsampling by a factor of 6.

This reduction in length of the timeseries is justified by over the substantial growth of the GM-LVQ’s projection matrix  $\Omega$ , due to the dimensionality of  $d_{lvq} \times d_{lvq}$  with  $d_{lvq} = d_d \cdot d_v$  according to [definition 5.8](#). Since GLVQ is not using the projection matrix  $\Omega$ , a model configuration can be designed that doesn’t exceed the memory constraints of the GPU. Data will be prepared at higher resolution and also lower resolutions to properly determine if learning capabilities can be observed.

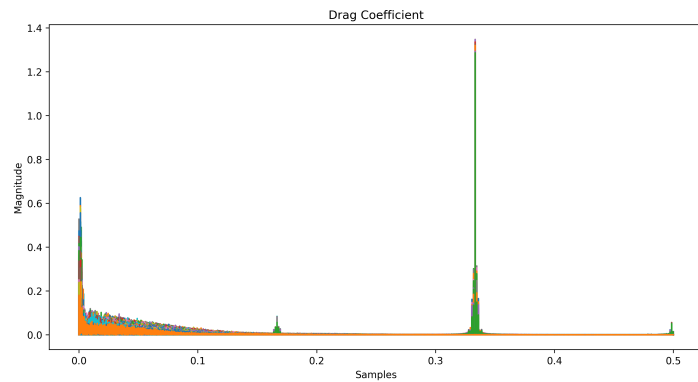
Trainingsets for the model have been chosen as suggested by the authors of the dataset. Additionally, multiple prepackaged trainingsets will be combined to achieve a larger data representation. In this scenario, sets 4 upto 15 will be put together. In any case, the trainingsets have been selected such that all classes are represented as ballanced as possible. For the classification, only classes 0, 1, 6, 9 are used, as the authors also ran experiments with this configuration [84]. Using all 10 classes would introduce an additional source of error and uncertainty regarding the model’s performance. Also, in [84] the smaller set achieved the best results with an accuracy of 96.04%.

<sup>19</sup>The Nyquist-Shannon Theorem states that the sampling frequency must be at least twice the maximum frequency that should be sampled.

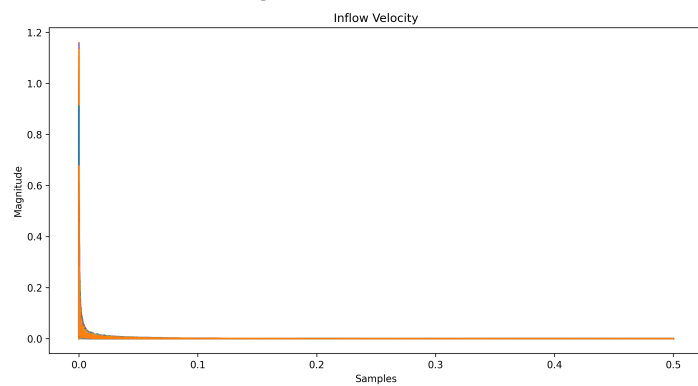
<sup>20</sup>The folding frequency refers to the maximum frequency which can be sampled by a given sampling frequency.



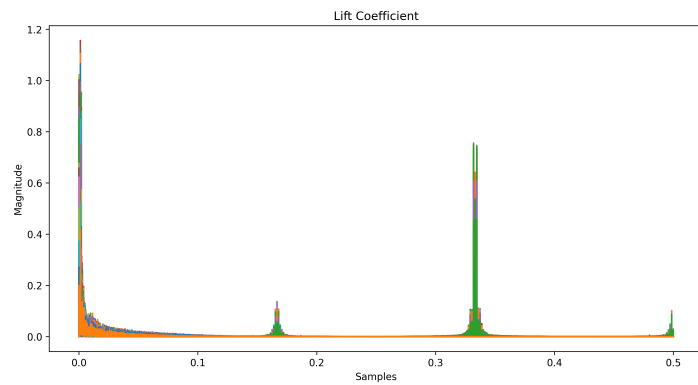
(a) FFT Angle of Attack at Full Resolution



(b) FFT Drag Coefficient at Full Resolution

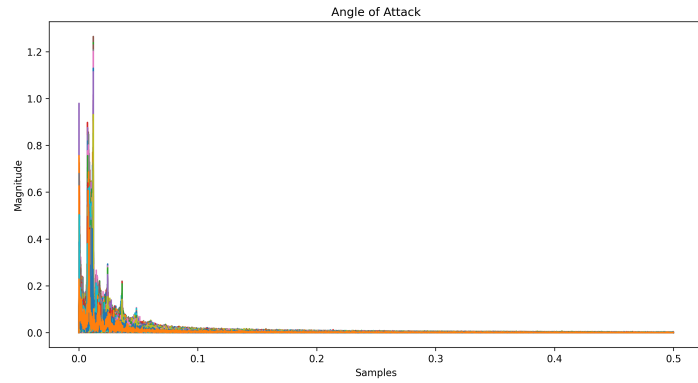


(c) FFT Inflow Velocity at Full Resolution

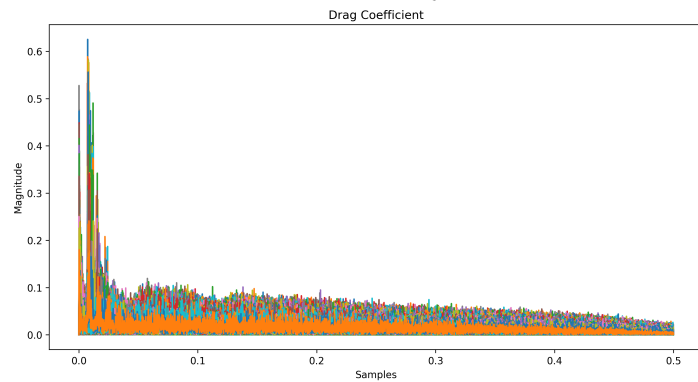


(d) FFT Lift Coefficient at Full Resolution

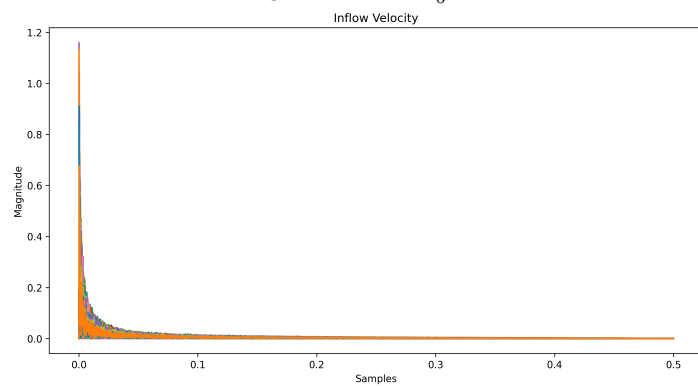
**Figure 6.2:** FFT of All Feature Timeseries from the Aeroelastic Simulations of Wind Turbines Dataset at Full Resolution



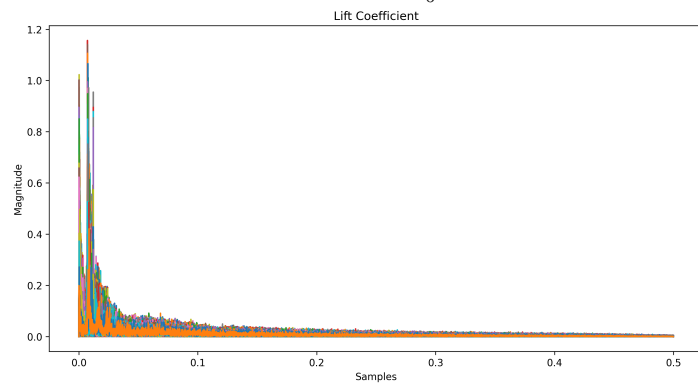
(a) FFT Angle of Attack at  $\frac{1}{6}$  Resolution



(b) FFT Drag Coefficient at  $\frac{1}{6}$  Resolution



(c) FFT Inflow Velocity at  $\frac{1}{6}$  Resolution



(d) FFT Lift Coefficient at  $\frac{1}{6}$  Resolution

**Figure 6.3:** FFT of All Feature Timeseries from the Aeroelastic Simulations of Wind Turbines Dataset at  $\frac{1}{6}$  Resolution

## 6.4 Experiment Observations

All experiments done have yielded the same result of not showing any generalization capability by any means. One exemplary experiment configuration will be explained and displayed. Other experiments have produced very similar results, which is why it is not necessary to go into detail on them. Other configurations will be mentioned to make clear which other configurations produced the same outcomes.

In the exemplary experiment, a *pre downsampled* GLVQ setup is used, such that the input timeseries to the model is unfiltered. For this configuration, the goal is to utilize the 80 GB as much as possible. The parameters are chosen as follows.

$$d_{in} = 60\,000$$

$$d_x = 4$$

$$d_d = 10\,000$$

$$d_k = 4096$$

$$d_v = 2048$$

with the activation function used in the downsample layer. In this scenario, about 65 GB are used in GPU memory. The learning rate for SGD optimizer is 0.25. It is found that any lower learning rate than 0.1 usually gets stuck and does not show any progress at all. Learning rates between 0.1 and 0.25 converge very slowly but show basically the same results if enough epochs are computed. Furthermore, the batch size is set to 8. The training process is performed over 20 epochs. It was observed that performance does not change when about 0.4 accuracy is reached. Figure 6.4 illustrates the metrics of the training process.

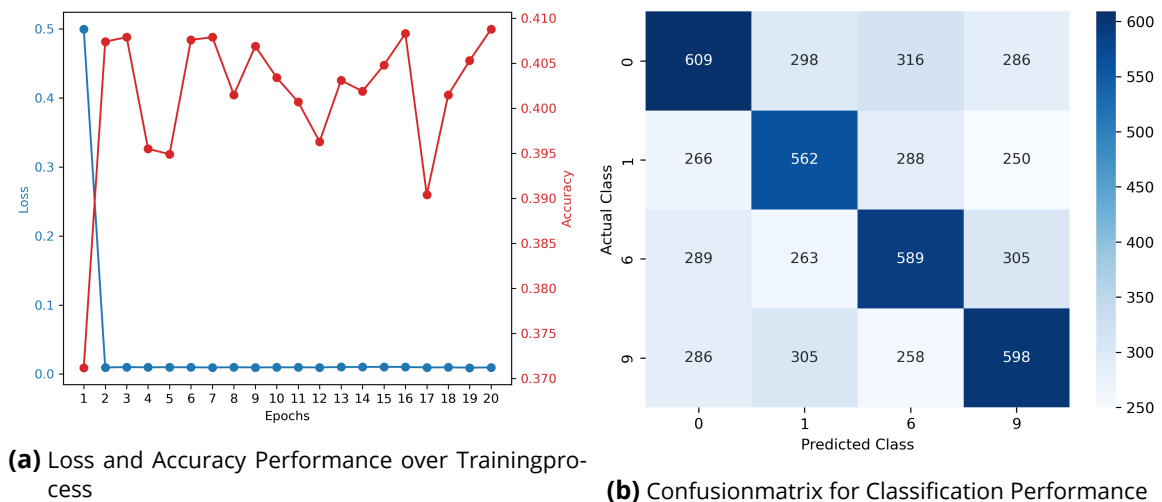


Figure 6.4: Training Metrics

In figure 6.4a, it can be observed that the loss is minimized very quickly and fluctuates around 0.01. This is due to the relatively high learning rate. As pointed out earlier, there are learning rates which converge slower, but these do not improve the classification performance. Figure 6.4b might suggest that the model has tendencies to generalize to a minor

extent. For this matter, the initial timeseries was stacked into a single vector, i.e.  $\text{vec}(\mathbf{X})$ , and trained on an [GLVQ](#) model. It was observed that the metrics were mostly consistent with those illustrated in [figure 6.4](#).

Since all models with pre-/post-attention [GLVQ/GMLVQ](#) performed similarly with different model configurations with accuracies ranging from 0.34 to 0.43, it can be concluded that this model fails to generalize. Other settings tested, which yielded the same results, can be found in [table 6.2](#). Note, to make the memory requirements feasible for some configurations, a 6.0 low-pass filtering/downsampling is applied before data is input into the model; hence,  $d_{\text{in}} = 10\,001$ . The code for the experiments can be found at [\[91\]](#).

Experiment	$d_{\text{in}}$	$d_x$	$d_k$	$d_v$	$d_d$	Pre Downsample	Post Downsample	Downsample Activation	GLVQ	GMLVQ
1	60001	4	1024	512	10000	X			X	
2	60001	4	682	341	5000	X			X	
3	60001	4	512	256	5000	X			X	
4	60001	4	256	256	5000	X			X	
5	10001	4	8192	4096	2500		X	X	X	
6	10001	4	8192	4096	2500		X		X	
7	10001	4	1024	1024	2500		X	X	X	
8	10001	4	1024	1024	2500		X		X	
9	10001	4	256	256	2500		X	X	X	
10	10001	4	256	256	2500		X		X	
11	10001	4	4096	256	180	X		X		X
12	10001	4	4096	256	180	X				X
13	10001	4	2048	256	160	X		X		X
14	10001	4	2048	256	160	X				X
15	10001	4	512	256	128	X		X		X
16	10001	4	512	256	128	X				X
17	10001	4	4096	256	180		X	X		X
18	10001	4	4096	256	180		X			X
19	10001	4	2048	256	160		X	X		X
20	10001	4	2048	256	160		X			X
21	10001	4	512	256	128		X	X		X
22	10001	4	512	256	128		X			X

**Table 6.2:** Experiment Configurations

## 6.5 Conclusion

Though the loss could be minimized in the experiments, the generalization capabilities of the model still appear to be rather random. An objective for this thesis was to design a simple attention based [LVQ](#) model. A clear reason for the failure of this design cannot be determined at the point of writing. There are speculations for the failure of the model's performance, with possible suggestions on how to solve these, which are outlined in this chapter.

### Attention Heads

In this model, only a single attention head was used, as it was an objective to keep the model as simple as possible. Multiple attention heads, contrary to a single attention head, are able to align different positions of the input sequence and extract different features [\[3\]](#). There is the possibility that a single attention head significantly reduces the performance of a model, as [\[92\]](#) suggests. Here, pruning of attention heads in regular transformer networks was

tested to determine 1) the redundancy of certain attention heads and 2) increase the efficiency of predicting by reducing parameters, i.e. attention heads. It was found that in some attention layers, attention heads could be reduced down to a single attention head, whereas in others, multiple attention heads are needed.

It is possible features could not be discriminated by a single attention head, and therefore, multiple attention heads are needed. Furthermore, it is unclear if a single attention layer is enough to extract features properly. As [84] has shown, good results can be achieved with standard transformer architectures using multiple attention heads and multiple attention layers, suggesting that attention-based models can be effective for this dataset. In transformers, multiple attention heads are implemented by attaching a feed-forward neural networks to the output of an attention layer and feeding its input into back into another attention layer again [3]. Unlike transformers, interpretable models should not use feed-forward networks, as these lose interpretability due to their subsymbolic architecture. Hence, designing multiple attention layers in an interpretable fashion remains an open task.

For further design testing, it would be possible to implement multihead attention to increase the ability extracting multiple features.

### Embedded Prototypes

In LVQ models, prototypes are initialized in the space where the datapoints belong to [76]. This model faced the challenge to successfully learn to place prototypes optimally in the embedding space, i.e. the vectorized representation of the matrix of the last layer. The advantage of this approach is that predictions would have lower computational cost, since all prototypes would be needed to be fed through the entire attention network to examine the (dis)similarity to the data point. Initializing prototypes as timeseries presents another issue. If prototypes are initialized randomly, the questions arises regarding which length should be chosen for a suitable random initialization.

A method for initializing prototypes could be to determine the average of a set of timeseries datapoints of the same class. An average for timeseries is known as *Steiner Sequence* [93]. Determining an average of a timeseries is not trivial as it is for points in Euclidean space. Using the average of points as used in Euclidean space would not typically work, since phase shifts over different timeseries could cause cancellation of frequencies and therefore, remove information. In addition to frequency cancellation, the potential difference in length of such time presents another challenge. To determine a possible average for timeseries, an algorithm known as **COmpact Multiple Alignment for Sequence Averaging (COMASA)** was developed [93]. In this method, the average is determined by minimizing the distance of the potential average to the sequences to be averaged. Here, **DTW** is utilized as a distance measure. Therefore, multiple averages can be computed for each class, which then could be the initialized prototypes.

## Learning Embedding and Embedded Prototypes

In the learning process, not only the positioning of the prototypes is being learned, but also the embedding of the datapoints into the LVQ embedding space  $\mathbb{R}^{d_{lvq}}$ . For that, either a single or a set of datapoints are randomly chosen according to (batch) SGD and the optimization is then performed on the given datapoints. This could have the effect that the projection in the attention layers is optimized for the given datapoints with respect to the closest prototypes, but not for the other remaining datapoints. Therefore, it could be possible that the projection for the other datapoint is learned suboptimally, leading to non-covering embedding of the datapoints. Since by assumption, the embedding of the data point disperses to some degree for the non-chosen data points, learning gets stuck. Additionally, the prototypes keep moving while the embedding projections could vary over time due to the dispersion, preventing prototypes from converging to optimal positions for class discrimination. Following this speculated effect, one should visualize it by tracking certain datapoints to see if, on the one hand, the projection is actually optimizing for the chosen datapoints and, on the other hand, the projection disperses for the other datapoints. Note, this issue could be related to 1) the prototypes being initialized in the embedding space and 2) the lack of the feature extraction ability of a single attention head and layer.

Another suggestion on how to proceed about this is to use an alternate learning scheme. Therefore, prototypes should be initialized either distributed with some uniform distance between them or by the suggested COMASA method. Then, as the first part of the alternating learning scheme, the projection by the attention layer should be learned, ensuring that timeseries points will actually get projected close to the placed prototypes. In a second step, the prototypes should be learned to optimally discriminate the datapoint embeddings. This can be done in an alternating fashion until some criterion is met.

## Input Dimensionality and Embedding Representation Dynamics

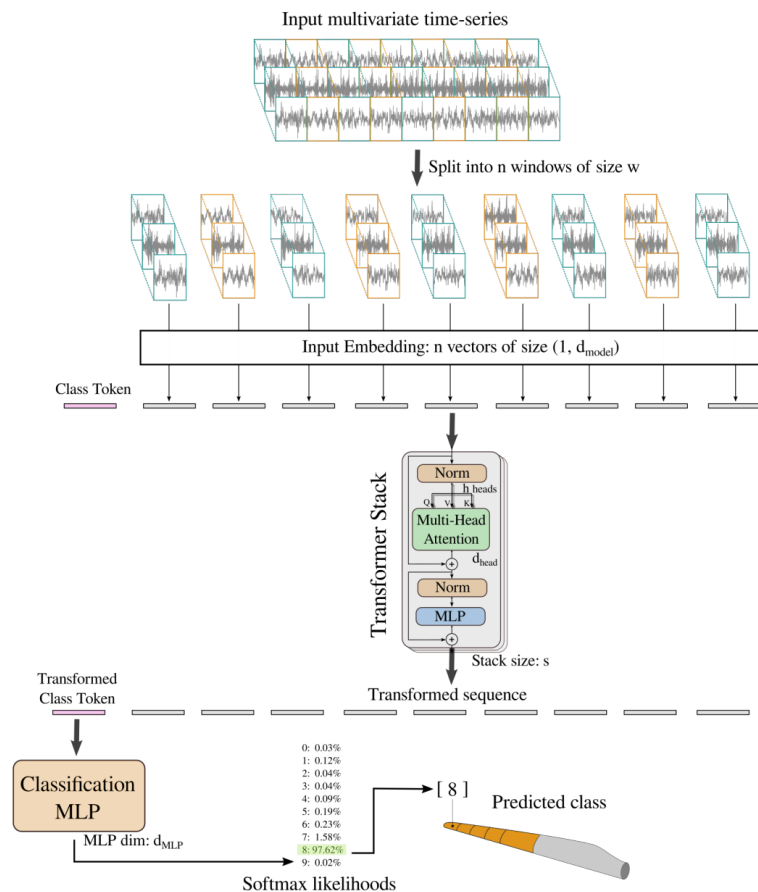
This model faces similar challenges as transformers do due to the quadratic growth of projections and projected matrices with respect to the length of the timeseries. The authors of [84] used the method *time-windowing*, originally introduced in vision transformers [94]. There, the input  $\mathbf{x} \in \mathbb{R}^{k \times n}$  is divided into  $l$  slices  $\mathbf{x}^i \in \mathbb{R}^{k \times l}$  called windows and being reduced in dimensionality by some learnable embedding projection<sup>21</sup> to  $\mathbf{z} \in \mathbb{R}^{d_{\text{model}}}$  with  $d_{\text{model}}$  being the desired model dimensionality. These embeddings are then concatenated with a special class token  $\mathbf{z}_{\text{class}} \in \mathbb{R}^k$  which is a learnable parameter. Hence, we get an embedding

$$\mathbf{Z} = [\mathbf{z}_{\text{class}}, \mathbf{z}_1, \dots, \mathbf{z}_l]$$

This embedding representation is then positionally encoded and parsed into the attention layers. After the attention processing, a representation  $\hat{\mathbf{Z}}$  is obtained. For the classification, the corresponding attention-embedded class token  $\hat{\mathbf{z}}_{\text{class}}$  is used for the classification process. Figure 6.5 illustrates the processing of *time-window* embedding and further processing.

<sup>21</sup>The source [84] does not specify how the learnable embedding projection is constructed.





**Figure 6.5:** Transformer setup in [84] with the special *time-window* embedding.

Not only does this help with reducing the computing and memory requirements in the attention layers, it can also greatly reduce the size of GMLVQ's projection matrix  $\Omega$ , since the complete matrix before the LVQ layer is vectorized. In the time-windowing method, only the class token representation would be used. Further, this provides an alternative to the vectorization  $\text{vec}(\mathbf{A})$  step to test.

## Final Remarks

The above suggestions are not confirmed to actually help the model improve its performance. Most assertions can be linked to observations of similar, successful models. Further research can be done by designing and implementing these suggestions and verify if improvements can be observed. There also are other minor tweaks, which could be implemented, like using different attention principles according to the attention taxonomy, using different alignment or distribution functions or using other LVQ algorithms like Limited Rank GMLVQ [95]. These tweaks should be considered as soon as an attention based LVQ learning scheme could be designed showing any improvement in its performance.

## 7 summary

This thesis analyzed attention from a psychological perspective, tracing its historical roots and the generalization of attention with its different possible configurations. Additionally, some mathematical background for several topics was provided, along with machine learning terminology to help guide the reader through the topics discussed in this thesis. Especially, the topic attention proved to be rich in information and variations, suggesting that there are many possibilities on how to design models that utilize attention.

Having built an understanding on attention, a design on an attention based [LVQ](#) model was proposed and analyzed. For this, one objective was to keep the model as simple as possible to ensure easy interpretability. Multiple variations were proposed for this design to accommodate different input sizes due to varying memory constraints. A multivariate timeseries was used, which had proven to be successful in conjunction with transformer models that heavily rely on attention. This dataset consisted of timeseries from wind blade rotors meant for classification into four different degradation states. Several experiments were conducted under various parametrizations and model designs.

In the experiments, it was shown that the proposed models fail to generalize on the learning task. The reason for the failure could not be determined, but instead reasonable suggestions were given based on experience of similar model designs and the inner workings of the processing.

Further research can be done in testing the suggested improvements on the model's design and examine whether any improvement can be seen in its classification performance. The suggested improvements are significant design changes. If such changes show improvements, other minor improvements, such as changing the alignment function, distribution function, attention design or the [LVQ](#) backend can be tested. Given some successes are found, also the interpretability of the attention matrices can be analyzed and see if they add to the interpretability of [LVQ](#) algorithms.

# Bibliography

- [1] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks", *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015. DOI: [10.1109/TMM.2015.2477044](https://doi.org/10.1109/TMM.2015.2477044).
- [2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1409.0473>.
- [3] A. Vaswani *et al.*, "Attention is all you need", *CoRR*, vol. abs/1706.03762, 2017. arXiv: [1706.03762](https://arxiv.org/abs/1706.03762). [Online]. Available: <http://arxiv.org/abs/1706.03762>.
- [4] J. Ravichandran, M. Kaden, and T. Villmann, "Variants of recurrent learning vector quantization", *Neurocomputing*, vol. 502, pp. 27–36, 2022. DOI: [10.1016/j.neucom.2022.06.035](https://doi.org/10.1016/j.neucom.2022.06.035). [Online]. Available: <https://doi.org/10.1016/j.neucom.2022.06.035>.
- [5] B. J. Jain and D. Schultz, "Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces", *CoRR*, vol. abs/1703.08403, 2017. arXiv: [1703.08403](https://arxiv.org/abs/1703.08403). [Online]. Available: <http://arxiv.org/abs/1703.08403>.
- [6] S. Haykin, *Neural Networks and Learning Machines* (Neural networks and learning machines Bd. 10). Prentice Hall, 2009, ISBN: 9780131471399. [Online]. Available: [https://books.google.de/books?id=K7P361KzI%5C\\_QC](https://books.google.de/books?id=K7P361KzI%5C_QC).
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning series). MIT Press, 2016, ISBN: 9780262035613. [Online]. Available: <https://books.google.de/books?id=Np9SDQAAQBAJ>.
- [8] S. R. Dubey, S. Singh, and B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark", *Neurocomputing*, vol. 503, Jul. 2022. DOI: [10.1016/j.neucom.2022.06.111](https://doi.org/10.1016/j.neucom.2022.06.111).
- [9] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989, ISSN: 1435-568X. DOI: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274). [Online]. Available: <https://link.springer.com/content/pdf/10.1007/BF02551274.pdf>.
- [10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989, ISSN: 0893-6080.
- [11] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions", *CoRR*, vol. abs/2005.00817, 2020. arXiv: [2005.00817](https://arxiv.org/abs/2005.00817). [Online]. Available: <https://arxiv.org/abs/2005.00817>.
- [12] C. M. Bishop, *Pattern recognition and machine learning, 5th Edition* (Information science and statistics). Springer, 2007, ISBN: 9780387310732. [Online]. Available: <https://www.worldcat.org/oclc/71008143>.
- [13] F. Jäkel, B. Schölkopf, and F. Wichmann, "A tutorial on kernel methods for categorization", *Journal of Mathematical Psychology*, vol. 51, pp. 343–358, 2007.

- [14] G. Orr and K. Müller, *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science). Springer Berlin Heidelberg, 2003, ISBN: 9783540494300. [Online]. Available: <https://books.google.com/books?id=VCKqCAAAQBAJ>.
- [15] X. Wu, R. Ward, and L. Bottou, *Wngrad: Learn the learning rate in gradient descent*, 2020. arXiv: [1803.02865](https://arxiv.org/abs/1803.02865) [stat.ML].
- [16] J. Nocedal and S. Wright, *Numerical optimization* (Springer series in operations research and financial engineering), 2. ed. New York, NY: Springer, 2006, XXII, 664, ISBN: 978-0-387-30303-1. [Online]. Available: [http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+502988711&sourceid=fbw\\_bibsonomy](http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+502988711&sourceid=fbw_bibsonomy).
- [17] M. Popovic, "Research in entropy wonerland: A review of the entropy concept", *Thermal Science*, vol. 22, pp. 12–12, Jan. 2018. DOI: [10.2298/TSCI180115012P](https://doi.org/10.2298/TSCI180115012P).
- [18] T. S. Han and K. Kobayashi, *Mathematics of Information and Coding*. USA: American Mathematical Society, 2001, ISBN: 0821805347.
- [19] D. Lairez, *A short derivation of boltzmann distribution and gibbs entropy formula from the fundamental postulate*, 2023. arXiv: [2211.02455](https://arxiv.org/abs/2211.02455) [cond-mat.stat-mech].
- [20] B. Peters, V. Niculae, and A. F. T. Martins, *Sparse sequence-to-sequence models*, 2019. arXiv: [1905.05702](https://arxiv.org/abs/1905.05702) [cs.CL].
- [21] F. A. Cowell, *Measuring inequality* (London School of Economics Perspectives in Economic Analysis), 3td. ed. Oxford University Press, 2011, ISBN: 9780199594030; 0199594031.
- [22] J. M. Amigó, S. G. Balogh, and S. Hernández, "A brief review of generalized entropies", *Entropy*, vol. 20, no. 11, 2018, ISSN: 1099-4300. DOI: [10.3390/e20110813](https://doi.org/10.3390/e20110813). [Online]. Available: <https://www.mdpi.com/1099-4300/20/11/813>.
- [23] C. Tsallis, *Entropic nonextensivity: A possible measure of complexity*, 2000. arXiv: [cond-mat/0010150](https://arxiv.org/abs/cond-mat/0010150) [cond-mat.stat-mech].
- [24] S. Furuichi, "On uniqueness theorems for tsallis entropy and tsallis relative entropy", *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3638–3645, Oct. 2005. DOI: [10.1109/tit.2005.855606](https://doi.org/10.1109/tit.2005.855606). [Online]. Available: <https://doi.org/10.1109/tit.2005.855606>.
- [25] M. Wainwright and M. Jordan, *Graphical Models, Exponential Families, and Variational Inference* (Foundations and trends in machine learning). Now Publishers, 2008, ISBN: 9781601981844. [Online]. Available: <https://books.google.de/books?id=zp5Mo3VsJbgC>.
- [26] R. E. Neapolitan and X. Jiang, "A note of caution on maximizing entropy", *Entropy*, vol. 16, no. 7, pp. 4004–4014, 2014, ISSN: 1099-4300. DOI: [10.3390/e16074004](https://doi.org/10.3390/e16074004). [Online]. Available: <https://www.mdpi.com/1099-4300/16/7/4004>.
- [27] T. Villmann, M. Kaden, D. Nebel, and A. Bohnsack, "Similarities, dissimilarities and types of inner products for data analysis in the context of machine learning", in *Artificial Intelligence and Soft Computing*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., Cham: Springer International Publishing, 2016, pp. 125–133, ISBN: 978-3-319-39384-1.
- [28] I. Shafarevich, A. Remizov, D. Kramer, and L. Nekludova, *Linear Algebra and Geometry* (SpringerLink : Bücher). Springer Berlin Heidelberg, 2012, ISBN: 9783642309946. [Online]. Available: <https://books.google.de/books?id=6Pp2-DT0KWIC>.

- [29] J. Gallier, "Basics of affine geometry", in *Geometric Methods and Applications: For Computer Science and Engineering*. New York, NY: Springer New York, 2011, pp. 7–63, ISBN: 978-1-4419-9961-0. DOI: [10.1007/978-1-4419-9961-0\\_2](https://doi.org/10.1007/978-1-4419-9961-0_2). [Online]. Available: [https://doi.org/10.1007/978-1-4419-9961-0\\_2](https://doi.org/10.1007/978-1-4419-9961-0_2).
- [30] J. Borwein and A. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples* (CMS Books in Mathematics). Springer New York, 2005, ISBN: 9780387295701. [Online]. Available: <https://books.google.de/books?id=TXWzqEkAa7IC>.
- [31] J. Munkres, *Elements Of Algebraic Topology*. CRC Press, 2018, ISBN: 9780429962462. [Online]. Available: <https://books.google.de/books?id=-mdQDwAAQBAJ>.
- [32] A. Hatcher, *Algebraic Topology* (Algebraic Topology). Cambridge University Press, 2002, ISBN: 9780521795401. [Online]. Available: <https://books.google.de/books?id=BjKs86kosqgC>.
- [33] S.-i. Amari, *Information Geometry and Its Applications*, 1st. Springer Publishing Company, Incorporated, 2016, ISBN: 4431559779.
- [34] M. Phuong and M. Hutter, "Formal algorithms for transformers", 2022. DOI: [10.48550/ARXIV.2207.09238](https://arxiv.org/abs/2207.09238). [Online]. Available: <https://arxiv.org/abs/2207.09238>.
- [35] B. Hommel, C. S. Chapman, P. Cisek, H. F. Neyedli, J.-H. Song, and T. N. Welsh, "No one knows what attention is", *Attention, Perception, & Psychophysics*, vol. 81, no. 7, pp. 2288–2303, Oct. 2019, ISSN: 1943-393X. DOI: [10.3758/s13414-019-01846-w](https://doi.org/10.3758/s13414-019-01846-w). [Online]. Available: <https://doi.org/10.3758/s13414-019-01846-w>.
- [36] G. W. Lindsay, "Attention in psychology, neuroscience, and machine learning", *Frontiers in Computational Neuroscience*, vol. 14, 2020, ISSN: 1662-5188. DOI: [10.3389/fncom.2020.00029](https://www.frontiersin.org/articles/10.3389/fncom.2020.00029). [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fncom.2020.00029>.
- [37] M. Posner, M. Rothbart, and M. Rueda, "Developing attention and self-regulation in childhood", in Jan. 2014, pp. 541–569.
- [38] A. Larsen, W. McIlhagga, J. Baert, and C. Bundesen, "Seeing or hearing? perceptual independence, modality confusions, and crossmodal congruity effects with focused and divided attention", *Perception & Psychophysics*, vol. 65, no. 4, pp. 568–574, May 2003, ISSN: 1532-5962. DOI: [10.3758/BF03194583](https://doi.org/10.3758/BF03194583). [Online]. Available: <https://doi.org/10.3758/BF03194583>.
- [39] T. P. Zanto and A. Gazzaley, "Attention and ageing.", in (Oxford library of psychology.), Oxford library of psychology. New York, NY, US: Oxford University Press, 2014, pp. 927–971, ISBN: 978-0-19-967511-1 (Hardcover).
- [40] N. Lavie, A. Hirst, J. Fockert, and E. Viding, "Load theory of selective attention and cognitive control", *Journal of experimental psychology. General*, vol. 133, pp. 339–54, Oct. 2004. DOI: [10.1037/0096-3445.133.3.339](https://doi.org/10.1037/0096-3445.133.3.339).
- [41] J. M. Wolfe, "Approaches to Visual Search: Feature Integration Theory and Guided Search", in *The Oxford Handbook of Attention*, Oxford University Press, Jan. 2014, ISBN: 9780199675111. DOI: [10.1093/oxfordhb/9780199675111.013.002](https://academic.oup.com/book/0/chapter/350821064/chapter-ag-pdf/44422440/book_41256_section_350821064.ag.pdf). eprint: [https://academic.oup.com/book/0/chapter/350821064/chapter-ag-pdf/44422440/book\\_41256\\_section\\_350821064.ag.pdf](https://academic.oup.com/book/0/chapter/350821064/chapter-ag-pdf/44422440/book_41256_section_350821064.ag.pdf). [Online]. Available: <https://doi.org/10.1093/oxfordhb/9780199675111.013.002>.

- [42] M. Carrasco, "183Spatial Covert Attention: Perceptual Modulation", in *The Oxford Handbook of Attention*, Oxford University Press, Jan. 2014, ISBN: 9780199675111. DOI: [10.1093/oxfordhb/9780199675111.013.004](https://doi.org/10.1093/oxfordhb/9780199675111.013.004). eprint: [https://academic.oup.com/book/0/chapter/350821660/chapter-ag-pdf/44422413/book\41256\\\_section\\\_350821660.ag.pdf](https://academic.oup.com/book/0/chapter/350821660/chapter-ag-pdf/44422413/book\41256\_section\_350821660.ag.pdf). [Online]. Available: <https://doi.org/10.1093/oxfordhb/9780199675111.013.004>.
- [43] Á. Kristjánsson and H. Egeth, "How feature integration theory integrated cognitive psychology, neurophysiology, and psychophysics", *Attention, Perception, & Psychophysics*, vol. 82, no. 1, pp. 7–23, Jan. 2020, ISSN: 1943-393X. DOI: [10.3758/s13414-019-01803-7](https://doi.org/10.3758/s13414-019-01803-7). [Online]. Available: <https://doi.org/10.3758/s13414-019-01803-7>.
- [44] A. M. Treisman and G. Gelade, "A feature-integration theory of attention", *Cognitive Psychology*, vol. 12, no. 1, pp. 97–136, 1980, ISSN: 0010-0285. DOI: [https://doi.org/10.1016/0010-0285\(80\)90005-5](https://doi.org/10.1016/0010-0285(80)90005-5). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0010028580900055>.
- [45] M. Hall, R. Pastore, B. Acker-Mills, and W. Huang, "Evidence for auditory feature integration with spatially distributed items", *Attention Perception & Psychophysics*, vol. 62, pp. 1243–1257, Apr. 2012. DOI: [10.3758/BF03212126](https://doi.org/10.3758/BF03212126).
- [46] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 1254–1259, Dec. 1998. DOI: [10.1109/34.730558](https://doi.org/10.1109/34.730558).
- [47] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention", *Vision Research*, vol. 40, no. 10, pp. 1489–1506, 2000, ISSN: 0042-6989. DOI: [https://doi.org/10.1016/S0042-6989\(99\)00163-7](https://doi.org/10.1016/S0042-6989(99)00163-7). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0042698999001637>.
- [48] L. Itti and C. Koch, "Computational modelling of visual attention", *Nature Reviews Neuroscience*, vol. 2, no. 3, pp. 194–203, Mar. 2001, ISSN: 1471-0048. DOI: [10.1038/35058500](https://doi.org/10.1038/35058500). [Online]. Available: <https://doi.org/10.1038/35058500>.
- [49] G. W. Lindsay, "Convolutional neural networks as a model of the visual system: Past, present, and future", *Journal of Cognitive Neuroscience*, vol. 33, pp. 2017–2031, 2020.
- [50] S. Chaudhari, G. Polatkan, R. Ramanath, and V. Mithal, "An attentive survey of attention models", *CoRR*, vol. abs/1904.02874, 2019. arXiv: [1904.02874](https://arxiv.org/abs/1904.02874). [Online]. Available: <http://arxiv.org/abs/1904.02874>.
- [51] G. S. Watson, "Smooth regression analysis", *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, vol. 26, no. 4, pp. 359–372, 1964, ISSN: 0581572X. [Online]. Available: <http://www.jstor.org/stable/25049340> (visited on 03/20/2023).
- [52] E. A. Nadaraya, "On estimating regression", *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964. DOI: [10.1137/1109020](https://doi.org/10.1137/1109020). eprint: <https://doi.org/10.1137/1109020>. [Online]. Available: <https://doi.org/10.1137/1109020>.
- [53] A. Graves, "Generating sequences with recurrent neural networks", *CoRR*, vol. abs/1308.0850, 2013. arXiv: [1308.0850](https://arxiv.org/abs/1308.0850). [Online]. Available: <http://arxiv.org/abs/1308.0850>.



- [54] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: [10.3115/v1/D14-1179](https://doi.org/10.3115/v1/D14-1179). [Online]. Available: <https://aclanthology.org/D14-1179>.
- [55] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [56] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm", vol. 12, Oct. 2000, pp. 2451–71. DOI: [10.1162/089976600300015015](https://doi.org/10.1162/089976600300015015).
- [57] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", *CoRR*, vol. abs/1409.3215, 2014. arXiv: [1409.3215](https://arxiv.org/abs/1409.3215). [Online]. Available: <http://arxiv.org/abs/1409.3215>.
- [58] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: [10.3115/v1/D14-1179](https://doi.org/10.3115/v1/D14-1179). [Online]. Available: <https://aclanthology.org/D14-1179>.
- [59] A. Garg and M. Agarwal, "Machine translation: A literature review", *CoRR*, vol. abs/1901.01122, 2019. arXiv: [1901.01122](https://arxiv.org/abs/1901.01122). [Online]. Available: <http://arxiv.org/abs/1901.01122>.
- [60] A. Galassi, M. Lippi, and P. Torrioni, "Attention in natural language processing", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4291–4308, Oct. 2021. DOI: [10.1109/tnnls.2020.3019893](https://doi.org/10.1109/tnnls.2020.3019893). [Online]. Available: <https://doi.org/10.1109/tnnls.2020.3019893>.
- [61] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation", *CoRR*, vol. abs/1508.04025, 2015. arXiv: [1508.04025](https://arxiv.org/abs/1508.04025). [Online]. Available: <http://arxiv.org/abs/1508.04025>.
- [62] A. Sordani, P. Bachman, and Y. Bengio, "Iterative alternating neural attention for machine reading", *ArXiv*, vol. abs/1606.02245, 2016.
- [63] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification", in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17, Melbourne, Australia: AAAI Press, 2017, pp. 4068–4074, ISBN: 9780999241103.
- [64] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines", *CoRR*, vol. abs/1410.5401, 2014. arXiv: [1410.5401](https://arxiv.org/abs/1410.5401). [Online]. Available: <http://arxiv.org/abs/1410.5401>.
- [65] K. Choromanski *et al.*, *Rethinking attention with performers*, 2022. arXiv: [2009.14794](https://arxiv.org/abs/2009.14794) [cs.LG].
- [66] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos, "Deeper attention to abusive user content moderation", in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1125–1135. DOI: [10.18653/v1/D17-1117](https://doi.org/10.18653/v1/D17-1117). [Online]. Available: <https://aclanthology.org/D17-1117>.
- [67] Y. Li, L. Kaiser, S. Bengio, and S. Si, "Area attention", *CoRR*, vol. abs/1810.10126, 2018. arXiv: [1810.10126](https://arxiv.org/abs/1810.10126). [Online]. Available: <http://arxiv.org/abs/1810.10126>.

- [68] M. Grabisch, J. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions* (Encyclopedia of Mathematics and its Applications). Cambridge University Press, 2009, ISBN: 9781139643221. [Online]. Available: <https://books.google.de/books?id=DbggAwAAQBAJ>.
- [69] A. F. T. Martins and R. F. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification", *CoRR*, vol. abs/1602.02068, 2016. arXiv: [1602.02068](https://arxiv.org/abs/1602.02068). [Online]. Available: <http://arxiv.org/abs/1602.02068>.
- [70] M. Blondel, A. F. T. Martins, and V. Niculae, *Learning with fenchel-young losses*, 2020. arXiv: [1901.02324](https://arxiv.org/abs/1901.02324) [stat.ML].
- [71] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004, ISBN: 0521833787.
- [72] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering", *CoRR*, vol. abs/1606.00061, 2016. arXiv: [1606.00061](https://arxiv.org/abs/1606.00061). [Online]. Available: <http://arxiv.org/abs/1606.00061>.
- [73] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention", *CoRR*, vol. abs/1502.03044, 2015. arXiv: [1502.03044](https://arxiv.org/abs/1502.03044). [Online]. Available: <http://arxiv.org/abs/1502.03044>.
- [74] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding", *CoRR*, vol. abs/1709.04696, 2017. arXiv: [1709.04696](https://arxiv.org/abs/1709.04696). [Online]. Available: <http://arxiv.org/abs/1709.04696>.
- [75] T. Kohonen and M. Schroeder, *Self-Organizing Maps*. Jan. 2001, ISBN: 3540679219.
- [76] D. Nova and P. Estevez, "A review of learning vector quantization classifiers", *Neural Computing and Applications*, vol. 25, Sep. 2014. DOI: [10.1007/s00521-013-1535-3](https://doi.org/10.1007/s00521-013-1535-3).
- [77] A. Sato and K. Yamada, "Generalized learning vector quantization", in *Advances in Neural Information Processing Systems*, D. Touretzky, M. Mozer, and M. Hasselmo, Eds., vol. 8, MIT Press, 1995. [Online]. Available: [https://proceedings.neurips.cc/paper\\_files/paper/1995/file/9c3b1830513cc3b8fc4b76635d32e692-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/1995/file/9c3b1830513cc3b8fc4b76635d32e692-Paper.pdf).
- [78] P. Schneider, M. Biehl, and B. Hammer, "Distance learning in discriminative vector quantization", English, *Neural computation*, vol. 21, no. 10, pp. 2942–2969, Oct. 2009, Relation: <http://www.rug.nl/informatica/organisatie/overorganisatie/iwi> Rights: University of Groningen, Research Institute for Mathematics and Computing Science (IWI), ISSN: 0899-7667. DOI: [10.1162/neco.2009.10-08-892](https://doi.org/10.1162/neco.2009.10-08-892).
- [79] P. Schneider, M. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization", en, *Neural Comput*, vol. 21, no. 12, pp. 3532–3561, Dec. 2009.
- [80] M. Müller, *Information Retrieval for Music and Motion*. Jan. 2007, ISBN: 978-3-540-74047-6. DOI: [10.1007/978-3-540-74048-3](https://doi.org/10.1007/978-3-540-74048-3).
- [81] E. Vidal, F. Casacuberta, J. Benedi, M. Lloret, and H. Rulot, "On the verification of triangle inequality by dynamic time-warping dissimilarity measures", *Speech Communication*, vol. 7, pp. 67–79, Mar. 1988. DOI: [10.1016/0167-6393\(88\)90022-2](https://doi.org/10.1016/0167-6393(88)90022-2).
- [82] P. Dufter, M. Schmitt, and H. Schütze, "Position information in transformers: An overview", *CoRR*, vol. abs/2102.11090, 2021. arXiv: [2102.11090](https://arxiv.org/abs/2102.11090). [Online]. Available: <https://arxiv.org/abs/2102.11090>.



- [83] G. Ke, D. He, and T. Liu, "Rethinking positional encoding in language pre-training", *CoRR*, vol. abs/2006.15595, 2020. arXiv: 2006.15595. [Online]. Available: <https://arxiv.org/abs/2006.15595>.
- [84] G. Duthé, I. Abdallah, S. Barber, and E. Chatzi, *Aeroelastic simulations of wind turbines affected by leading edge erosion: datasets for multivariate time-series classification*, Zenodo, Oct. 2021. DOI: 10.5281/zenodo.5788931. [Online]. Available: <https://doi.org/10.5281/zenodo.5788931>.
- [85] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: <http://arxiv.org/abs/1301.3781>.
- [86] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information", *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, Jun. 2017, ISSN: 2307-387X. DOI: 10.1162/tacl\_a\_00051. eprint: [https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl\\_a\\_00051/1567442/tacl\\_a\\_00051.pdf](https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf). [Online]. Available: [https://doi.org/10.1162/tacl%5C\\_a%5C\\_00051](https://doi.org/10.1162/tacl%5C_a%5C_00051).
- [87] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: <https://aclanthology.org/D14-1162>.
- [88] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding", *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: <http://arxiv.org/abs/1810.04805>.
- [89] *Pytorch datatypes table*, <https://pytorch.org/docs/stable/tensors.html#id4>, Accessed: 3rd August 2023.
- [90] R. G. Lyons, *Understanding Digital Signal Processing*, 1st. Addison Wesley Pub. Co, 1997, ISBN: 0201634678; 9780201634679.
- [91] *Experiment code*, [https://git.hs-mittweida.de/tdavies/attention\\_lvq](https://git.hs-mittweida.de/tdavies/attention_lvq), Accessed: 15th August 2023.
- [92] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?", *CoRR*, vol. abs/1905.10650, 2019. arXiv: 1905.10650. [Online]. Available: <http://arxiv.org/abs/1905.10650>.
- [93] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment", *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, 2012, ISSN: 0304-3975. DOI: <https://doi.org/10.1016/j.tcs.2011.09.029>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S030439751100822X>.
- [94] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *CoRR*, vol. abs/2010.11929, 2020. arXiv: 2010.11929. [Online]. Available: <https://arxiv.org/abs/2010.11929>.
- [95] K. Bunte, "Adaptive dissimilarity measures, dimension reduction and visualization", Thesis fully internal (DIV), Ph.D. dissertation, Groningen, 2011, ISBN: 9789036751865.

## Eidesstattliche Erklärung

Hiermit versichere ich – Thomas Davies – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Mittweida, 09. November 2023

Ort, Datum

Thomas Davies, B.Eng., B.Sc.