



Logo 1



Logo2

Goethe-Universität Frankfurt
Fachbereich 12 - Informatik und Mathematik

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science

Thema: Deep Neural Networks for occluded Image Recognition

Autor: Julius Taylor <s8423760@stud.uni-frankfurt.de>
MatNr. 5210444

Version vom: 17. November 2017

Betreuer: Prof. Dr. Jochen Triesch

Sperrvermerk

Die vorliegende Arbeit beinhaltet interne und vertrauliche Informationen der Firma <Firmenname>. Die Weitergabe des Inhalts der Arbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften - auch in digitaler Form - sind grundsätzlich untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma <Firmenname>.

Abstract

Im Rahmen dieser Bachelorarbeit wurde ein Convolutional Neural Net zur Interpretation von Bildinformationen im anspruchsvollen Szenario der teilweisen Okklusion entwickelt und trainiert. Den Ausgangspunkt dieser Arbeit bietet die Arbeit von Spoerer und Kriegeskorte, welche das Konzept rekursiver Verbindungen in Neuronalen Netzen ausnutzen, um die Performance bei der Erkennung von Bildern mit teilweise fehlenden Informationen zu verbessern. Grundlage ist die Beobachtung, dass im ventralen Visuellen System laterale und rückwärtsgewandte Verbindungen und dadurch rekurrente Dynamiken zum Einsatz kommen, welche in der Bilderkennung bisher kaum oder gar nicht genutzt werden. Spoerer und Kriegeskorte vermuten, dass diese Rekurrenzen die Performance bei teilweise verdeckten Stimuli verbessern können. Um dies zu verifizieren, wurde ein simples Stimulus-Set generiert, welches Ziffern enthält, die wahlweise mit zufällig zerschnittenen Teilen von verschiedenen Zahlen verdeckt werden können. Das zugrundeliegende Problem der Klassifikation ist ein simples und wohl erforschtes. Daher kann sich in der Untersuchung allein auf die zusätzliche Schwierigkeit, die durch das Verdecken besagter Ziffern entsteht, konzentriert werden. Diese Arbeit reproduziert die Ergebnisse, die zeigen, dass rekurrente Dynamik die Klassifikationsrate bei sowohl unokkludierten als auch okkludierten Ziffern verbessert. Dabei werden die rekurrenten Modelle einerseits mit reinen feed forward Architekturen verglichen, die ungefähr in der Anzahl der Parameter mit den rekurrenten vergleichbar sind. Darüber hinaus, stelle ich sie alternativen Modellen gegenüber, die zusätzliche Konvolutions-Layer besitzen, um die Anzahl der Konvolutionen vergleichbar zu machen.

Inhaltsverzeichnis

Abbildungsverzeichnis	5
Tabellenverzeichnis	5
Listingverzeichnis	5
Abkürzungsverzeichnis	6
1 Introduction	7
2 Background	7
2.1 Supervised Learning	7
2.1.1 Linear Models	7
2.2 Neuronale Netze	8
2.3 Convolutional Neural Networks	9
2.4 Recurrent Neural Networks	9
2.5 Unrolling Recurrent Netowrks	9
2.6 Backpropagation	9
3 Materials and Methods	10
3.1 Generatives Modell für Stimuli	10
3.2 Models	10
3.2.1 Implementation	11
3.2.2 Recurrent Convolution Layers	11
3.3 Training	13
3.3.1 Error Measurement	13
3.3.2 Backpropagation Through Time	13
3.4 Truncated Backpropagation Through Time	13
4 Ergebnisse	13
4.1 Performance changes under varying levels of occlusion	14
5 Diskussion	14
6 Danksagung	14
Literaturverzeichnis	15
Anhang	16
Eidesstattliche Erklärung	16

Abbildungsverzeichnis

1 Ein einfaches Perzeptron	9
--------------------------------------	---

Tabellenverzeichnis

Listingverzeichnis

1 Introduction

Among a range of other tasks, image recognition has been a domain dominated by feed forward Convolutional Neural Network architectures. Strong evidence suggests, that the human brains ability to rapidly recognize objects under appearance variation is solved in the brain via a succession of largely feedforward computations. [DZR12, 2012] Recent success of feedforward networks has only supported this hypothesis [KSH12]. However, evidence also suggests, that after the initial recognition process further processing takes place [SYUK99]. This delayed processing might indicate recurrent computations which I want to investigate in this work. Using a range of convolutional architectures, I try to systematically measure the impact of recursion on image recognition tasks.

2 Background

2.1 Supervised Learning

The common ground for all further aspects in this work will be supervised learning, which is most generally speaking simply the task of deriving a function from data. This function should be able to map given inputs or data points to some sort of output. In the supervised setting, data is given with an associative label such, that the inferred function is able to map said input data to the correct labels, without explicitly looking at the given labels. The learning algorithm, when deriving said function, therefore has to use patterns and structure among the data points to learn something about the quality of a given data point. Outputs are generally either of a quantitative type e.g. temperature measurements or changes in stock price or of a qualitative type like distinctive types of flowers. Tasks involving the former kinds of outputs are generally referred to as regression problems whereas the tasks involving the latter are generally called classification problems.

We can roughly articulate the supervised learning problem as follows: for a given input X , an associated label Y and a function $\mathbb{F}(X) = \hat{Y}$, \mathbb{F} makes a good prediction, when \hat{Y} is close to Y . Proximity, in this case, must be defined and can vary depending on the type of data which is present. For a regression problem of e.g. predicting tomorrows temperature, the prediction is better, the more similar the predicted value \hat{Y} and tomorrows true temperature are, measured by the absolute distance of $|Y - \hat{Y}|$. X in this case could be a vector of meteorologic measurements which are regarded highly predictive of short term weather prognosis.

2.1.1 Linear Models

Linear Regression has been a staple in statistics and machine learning and remains a very important and widely used tool [HTF01]. As it serves as the basis for a wide range

of more complex machine learning algorithms like logistic regression and in some sense even Deep Neural Networks, it serves as a good introduction. The linear model

$$\mathbb{F}(X) = \beta + \sum_{i=1}^{|X|} w_i x_i = \hat{y}$$

predicts Y by using a linear combination of all input variables x_i using weights w_i . β is the intercept of the linear decision boundary, often referred to as the bias. Since in this case we are modeling a scalar value, \hat{y} is a single value, but can also be N -vector if we're predicting values of higher dimensionality. To get an idea how good our model is performing, we first need to come up with a way of measuring its prediction quality. The sum of squares is a widely used method and sums each squared difference between any predicted value \hat{y}_i and its corresponding true value y_i . We define the sum of squared errors as E , a function of our parameters \mathbb{W} which we then can minimize. Since it is a quadratic function, its minimum always exists but does not have to be unique.

$$E(\mathbb{W}) = \frac{1}{2} \sum_{i=1}^N (\hat{y}_i - y_i)^2 = \frac{1}{2} \sum_{i=1}^N \left(\sum_{j=1}^{|X|} (w_j x_{ij} + \beta) - y_i \right)^2$$

Minimizing $E(\mathbb{W})$ with respect to \mathbb{W} yields:

$$\frac{\delta E}{\delta \mathbb{W}} = \left(\frac{\delta E}{\delta w_1}, \dots, \frac{\delta E}{\delta w_n} \right) = \left(\sum_{i=1}^N (\hat{y}_i - y_i) x_{i1}, \dots, \sum_{i=1}^N (\hat{y}_i - y_i) x_{in} \right)$$

2.2 Neuronale Netze

Neuronale Netze sind biologisch inspirierte Programmierparadigma, welche es erlauben, iterativ vom Betrachten von Daten zu lernen. Dazu benötigen sie meist kein domänen-spezifisches Wissen, sondern "lernen" Strukturen in den betrachteten Daten selbst zu erkennen. Die Grundeinheit solcher Netzwerke sind dabei künstliche Neuronen, die in Schichten angeordnet sind. Ein Neuron erhält eine Eingabe und propagiert sie über eine Verbindung (Synapse) an andere Neuronen weiter. Typischerweise senden Neuronen ihre Eingabe entlang eines Pfades von der Eingabe zur Ausgabe und haben Gewichte, die bestimmen, wie viel des erhaltenen Signals an folgende Neuronen weitergegeben wird.

Die simpelste Form eines Neuronalen Netzes ist ein einfaches Perzeptron, welches 1958 von Frank Rosenblatt in [Ros58] vorgestellt wurde. Dieses basiert auf einer McCulloch-Pitts-Zelle, einem rudimentären Modell einer Nervenzelle nach [MP43], enthält mehrere Eingaben und produziert eine einzige Ausgabe. Perzeptrons können prinzipiell aus mehreren Schichten solcher Zellen bestehen, die in Reihe geschaltet sind. Dabei gibt eine Zelle ihren berechneten Output an ein Neuron der nächsten Schichten weiter. Solche

Netzwerke heißen Multi-Layer-Perzeptron, im weiteren beschäftigen wir uns jedoch mit einer einzigen Zelle. Das einfache, in Abbildung 1 dargestellte Perzeptron erhält n

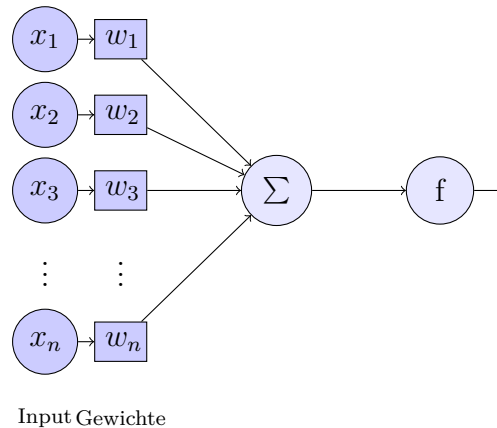


Abbildung 1: Ein einfaches Perzeptron mit n Eingängen

Eingänge $x_1, \dots, x_n \in \mathbb{R}$, die jeweils mit korrespondierenden Gewichten $w_1, \dots, w_n \in \mathbb{R}$ multipliziert werden. Die Summe der gewichteten Eingänge wird dann als Argument der Aktivierungsfunktion verwendet. Die Ausgabe des Perzeptron lässt sich als eine Funktion seiner Eingaben wie folgt beschreiben:

$$\mathbb{P}(\mathbf{x}, \mathbf{w}) = f\left(\sum_{i=1}^n w_i x_i\right)$$

mit der Aktivierungsfunktion

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

womit die Ausgabe als Zugehörigkeit zu einer Klasse, entweder 0 oder 1 zu interpretieren ist.

2.3 Convolutional Neural Networks

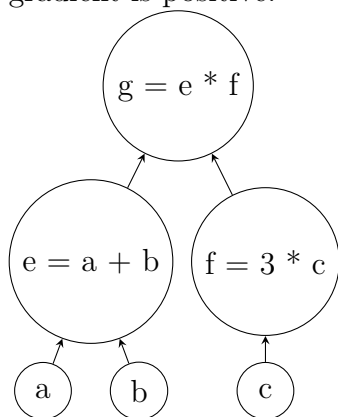
2.4 Recurrent Neural Networks

2.5 Unrolling Recurrent Networks

2.6 Backpropagation

Backpropagation is a widely used technique to train neural networks. It is, in fact, the algorithm that made training deep neural networks tractable in the first place. While being originally introduced in the 1970's it has not been adapted until David Rumelhart, Geoffrey Hinton and Ronald Williams drew a lot of attention to it in their

famous 1986 Paper [RHW⁺88]. At the core of Backpropagation stand partial derivatives like $\frac{\delta C}{\delta w}$ of a cost function C with respect to a weight w . This gradient expresses, at what rate C changes if we tune w . Knowing how the error of the network behaves when changing a parameter can be very helpful, since we then can adjust it in a way, such that our total error decreases. In order to minimize our cost function we therefore have to compute the partial derivatives of the networks cost function with regard to every variable in the network i.e. every weight and bias. We then use those gradients to move "downwards" on our cost function i.e. adding a small positive value to our weight or bias, if the gradient is negative and vice versa adding a small negative value, if the gradient is positive.



3 Materials and Methods

3.1 Generatives Modell für Stimuli

Um die Effekte von Rekursion auf das klassifizieren von okkludierten Objekten zu untersuchen, wurde ein möglichst einfaches Grundproblem betrachtet. Die Klassifizierung von Ziffern ist ein gut untersuchtes Machine Learning Problem, in dem übermenschliche Performance erreicht werden kann. Allgemein gilt das sehr ähnliche MNIST-Datenset gemeinhin als das "Hello world!" des Machine Learning. Durch die Einfachheit der Aufgabe, können die Auswirkungen von Rekursion isoliert von anderen Herausforderungen betrachtet werden.

3.2 Models

Um die Wirkungskraft von Rekursion zu untersuchen, wurden verschiedene CNNs mit variierenden Graden an Rekursion benutzt und systematisch miteinander verglichen. Die Nomenklatur richtet sich am Paper von Spoerer und Kriegeskorte aus. Als Grundlage dient eine Standard feed-forward Architektur (B) mit reinen 'bottom-up' Verbindungen. Da diese jedoch in der Anzahl der Parameter und der Anzahl der durchgeführten Konvolutionen gegenüber seinen Rekursiven Varianten unterlegen ist, wurden

zum Vergleich zusätzlich Modelle entwickelt, die in den entsprechenden Domänen angepasst wurden. Einerseits wurde die Grösse der Konvolutionskernel angepasst und somit die Grösse der erlernbaren Features. Andererseits wurde die Anzahl der Konvolutionen erhöht, die Gewichte in den zusätzlichen Konvolutionen jedoch mit den anderen Konvolutionen geteilt. Somit kann die Anzahl der Faltungsoperationen vergleichbar gemacht werden, ohne die freien Parameter zu erhöhen. Die Architekturen werden im Folgenden BK respektive BKC genannt.

3.2.1 Implementation

Apart from BKC, all models consist of two convolution layers. All bottom up convolutions are implemented as standard convolutions with 1x1 stride and zero padding, which leads to the output pictures being the same size as the input. The output of the convolution is then fed into a parameterized version of the Rectified Linear Unit activation function (PReLU). PReLU works as a generalized form of the ReLU activation function as it controls the output for negative values with a slope which can be learned.

$$f(x_i) = \begin{cases} x_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

If the slope parameter is zero, PReLU results in standard ReLU. If the parameter is a small positive number, PReLU equals Leaky ReLU. The output of the PReLU activation is then normalized with local response normalization (LRN), which tries to account for lateral inhibition which was found in brains. LRN therefore dampens responses which are uniformly large in any given local neighborhood and strengthens activations which are relatively larger than their surroundings and so imposes sparse activations. After normalization, the image is fed into a max-pooling layer with 2x2 stride and a pooling window of size 2x2, thus reducing the image size by half in each dimension. This whole process is repeated for each of the convolutions. After the second pooling, the image therefore is 8x8 in size and is flattened to a 1x64 vector before being fed into a readout layer, which maps the input to 1x10 vector. The sigmoid function is then applied to the final output, yielding values from 0 to 1 which can be interpreted as the probability that each of the responding targets is present in the given input picture.

3.2.2 Recurrent Convolution Layers

The heart of this work are recurrent convolution layers (RCL) whose effect on occluded image recognition is to be investigated. We denote the input of layer l at timestep t with

$I_{l,t}$ and use a vectorized format which contains values across feature maps. The input at layer 0 e.g. $I_{0,t}$ is defined as the input picture. The preactivation of B is given by

$$p_{l,t,m} = K_{l,m} * I_{l-1,t} + b_{l,m}$$

in which $*$ represents the convolution operation, $K_{l,m}$ and $b_{l,m}$ being the convolution kernel and the bias respectively, each for layer l and feature map m .

BL has additional lateral connections and the preactivation is given by

$$p_{l,t,m} = K_{l,m}^b * I_{l-1,t} + K_{l,m}^l * I_{l,t-1} + b_{l,m}$$

where the output of layer l at time step $t-1$ is convolved with the lateral convolution kernel $K_{l,m}^l$ and added to the preactivation of layer l at time step t . Following the same principles, we can construct the preactivation of BT which is given as

$$p_{l,t,m} = K_{l,m}^b * I_{l-1,t} + K_{l,m}^t * I_{l-1,t} + b_{l,m}$$

When combining lateral and top down connections, we are left with BLT which is the full recursive model, yielding a preactivation of

$$p_{t,l,m} = K_{l,m}^b * I_{l-1,t} + K_{l,m}^t * I_{l-1,t} + K_{l,m}^l * I_{l,t-1} + b_{l,m}$$

Each preactivation is then fed into a ReLU and a Local Response Normalization Layer. ReLU is defined as

$$r_{t,l,m} = \max\{0, p_{t,l,m}\}$$

and Local Response Normalization (LRN) (Krizhevsky et al., [KSH12]) is given by

$$lrn(x) = x \left(c + \alpha \sum_{k'=\max(0,k-n/2)}^{\min(n-1,k+n/2)} x^2 \right)^{-\beta}$$

with $n = 5$, $c = 1$, $\alpha = 10^{-4}$, and $\beta = 0.5$ with the sum over n adjacent kernel maps at the same spatial position. Even though ReLU activations do not require for input normalization to prevent saturation, LRN seems to aid generalization by imposing a competition among adjacent neurons by simulating a concept of lateral inhibition. Hence the output of layer l at time step t is

$$\omega_{l,t} = lrn(r(p_{l,t,m}))$$

3.3 Training

3.3.1 Error Measurement

Cross Entropy and training data structure goes HERE

3.3.2 Backpropagation Through Time

Usually in feedforward neural networks, Backpropagation is used for training. This refers to the mathematical method of calculating derivatives of the network function in regard to a given input by using the chain rule. These derivatives are then used to update network weights and hence minimize training error. Since in recurrent networks, the output also depends on network states of earlier time steps, standard Backpropagation cannot be applied here. Backpropagation Through Time is the application of Backpropagation to recurrent neural networks. This works by unrolling the network across all time steps and passing one input to one copy of the network. Errors are calculated for all time steps and summed up. Usually recurrent neural networks deal with some sort of sequence data like a time series, but in this case the sequence is the same input picture presented multiple times. Since these stimuli have no innate sequence nature, we create a virtual sequence by presenting the network with the same picture for a fixed number of timesteps.

3.4 Truncated Backpropagation Through Time

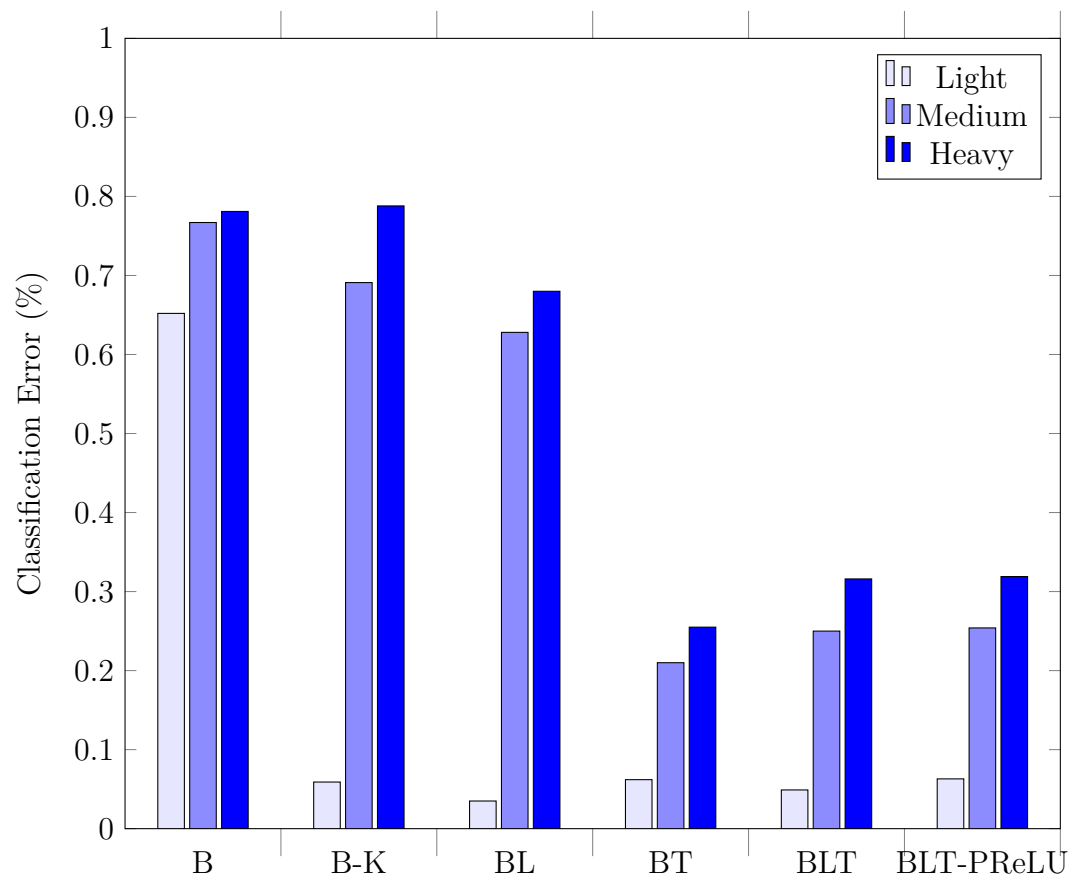
Truncated Backpropagation Through Time (TBPTT) is an adapted version of BPTT where the unrolling of the network is limited to a fixed number of time steps.

4 Ergebnisse

All models were trained and tested to investigate how recursion changes performance when dealing with occluded stimuli. Three different data sets have been used, each containing single digit targets to be recognized. The amount of occlusion varies between 10 fragments (light occlusion), 30 fragments (medium occlusion) and 50 fragments (heavy occlusion) among data sets. Under light occlusion BL seemed to be performing best with a mere 0.035% classification error. Under medium and heavy occlusion BT and BLT outperformed all other networks.

Model	B	B-K	BL	BT	BLT	BLT-PReLU
Light Debris	0.652	0.059	0.035	0.062	0.049	0.063
Medium Debris	0.767	0.691	0.628	0.210	0.256	0.254
Heavy Debris	0.781	0.473	0.680	0.255	0.316	0.319

4.1 Performance changes under varying levels of occlusion



5 Diskussion

6 Danksagung

Literaturverzeichnis

- [DZR12] DiCARLO, James J. ; ZOCCOLAN, Davide ; RUST, Nicole C.: How Does the Brain Solve Visual Object Recognition? In: *Neuron* 73 (2012), Nr. 3, 415 - 434. <http://dx.doi.org/https://doi.org/10.1016/j.neuron.2012.01.010>. – DOI <https://doi.org/10.1016/j.neuron.2012.01.010>. – ISSN 0896-6273
- [HTF01] HASTIE, Trevor ; TIBSHIRANI, Robert ; FRIEDMAN, Jerome: *The Elements of Statistical Learning*. New York, NY, USA : Springer New York Inc., 2001 (Springer Series in Statistics)
- [KSH12] KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; HINTON, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. (2012), 1097–1105. <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [MP43] McCULLOCH, Warren S. ; PITTS, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Nr. 4, S. 115–133
- [RHW⁺88] RUMELHART, David E. ; HINTON, Geoffrey E. ; WILLIAMS, Ronald J. u. a.: Learning representations by back-propagating errors. In: *Cognitive modeling* 5 (1988), Nr. 3, S. 1
- [Ros58] ROSENBLATT, Frank: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386
- [SYUK99] SUGASE, Yasuko ; YAMANE, Shigeru ; UENO, Shoogo ; KAWANO, Kenji: Global and fine information coded by single neurons in the temporal visual cortex. In: *Nature* 400 (1999), Aug, 869 EP -. <http://dx.doi.org/10.1038/23703>

Anhang

Eidesstattliche Erklärung

Eidesstattliche Erklärung zur <-Arbeit>

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäss aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Unterschrift :

Ort, Datum :

