Application of the Lottery Ticket Hypothesis in NLP and Early Pruning (Proposal)

Anwendung der "Lottery Ticket"-Hypothese in NLP und frühem Pruning (Proposal)
Bachelor-Arbeit
Tim Unverzagt
KOM-type-number KOM-B-0666

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachgebiet Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Application of the Lottery Ticket Hypothesis in NLP and Early Pruning (Proposal)
Anwendung der "Lottery Ticket"-Hypothese in NLP und frühem Pruning (Proposal)

Bachelor-Arbeit
Studiengang: Computational Engineering
KOM-type-number KOM-B-0666

Eingereicht von Tim Unverzagt
Tag der Einreichung: 20. Januar 2020

Gutachter: Prof.Dr.-Ing. Ralf Steinmetz
Betreuer/in: Anna Filighera, Tim Steuer

Technische Universität Darmstadt
Fachgebiet Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Erklärung zur Abschlussarbeit gemäß § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Tim Unverzagt, die vorliegende Bachelor-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Bachelor-Arbeit stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Darmstadt, den 20. Januar 2020

Tim Unverzagt

Contents

Abstract

The abstract goes here...

1 Introduction

1.1 Motivation

- LTH has demonstrated extreme pruning on different architectures
- Study of lottery-ticket emergence points might result in a reasoned early pruning approach
- LTH might bring a new and promising pruning approach to NLP

1.2 Problem Statement and Contribution

- Calculate pruning masks earlier in training and check if LTH still holds.
  Observing when lottery-tickets are no longer found might improve understanding of early pruning.
- Research whether the point of lottery-ticket-emergence can be estimated "a-priori".
- Implement an architecture comparable to the ones studied in the lottery-ticket hypothesis performing well on an NLP-task with similar structure.
- Determine whether the LTH holds on said architecture.

1.3 Outline

???

2 Background

To develop this work a common base of concepts is needed. This chapter aims to establish the fundamental ideas necessary to comprehend this thesis.

2.1 Neural Networks

2.1.1 Basics

Neural networks are a part of most major AI-breakthrough in the last decade enabling computers to compete in fields formerly championed by humans.[1] They implement a statistical understanding of AI, which is to say that they try to find a specific model optimizing the likelihood of reproducing input-output pairs similar to some training data. The competing philosophy directly divines behaviour rules, frequently from expert knowledge, and as such is far less dependant from data. [citation needed]
For the former concept its model classes are the essential point of design. A multitude of properties maybe sought after in a model class of which a few important ones are:

- **Richness:**
  The diversity of single models in the class and thus the ability to fit a wide field of different input-output landscapes.[2]
  If a model class is inherently restricted the underlying relation between inputs and outputs might simply be beyond the expressive capabilities of all its models.
  In other words: If a model class is not rich enough all of its models will underfit the given training data.
- **Stability:**
  Tendency of similar models in the class to handle inputs in a similar way.
  If your model class shows unstable behavior defining a sensible way to search it for good models becomes difficult.
- **Interpretability of Models:**
  Ease of formulating knowledge out of any given model in the class.
  As fields exist in which statistical AI outperform experts the extraction of knowledge understandable and applicable by humans is of special interest.
- [citation needed]

If one knows an entity that already performs well on a given task it is a sensible approach to design ones model class to reproduce its decision process. Humans usually are such entities for many tasks of interest to AI research so they are a natural source of inspiration. Neural networks essentially are simplified models of a human central nervous system.

[1]

- 2011: "Watson" of IBM defeats two former grand champions in "Jeopardy!" [LF11]
- 2011: "Siri" enables users to use natural language to interact with their phones [Aro11]
- 2015: A convolutional neural network classifies images from the ImageNet dataset more accurately than human experts [RDS+15] [HZRS15]
- 2016: "AlphaGo" beats Lee Sedol, one of the world's strongest Go players [Gib16] [SSS+17]

[2] More formally the richness of a model class can be described as the amount of different functions from the input-space to the output-space which can be expressed through a model of said class.

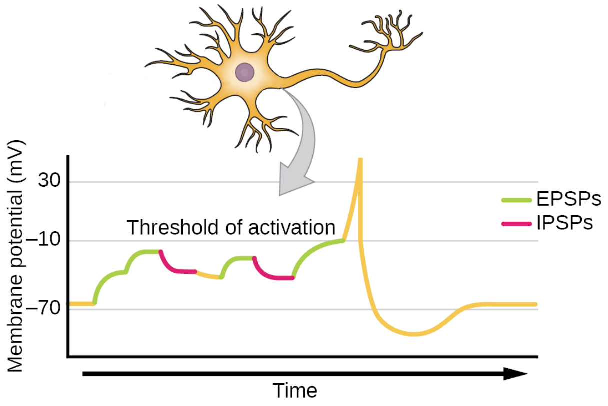

Figure 2.1.: Representation of a biological Neuron [CDC18] edited

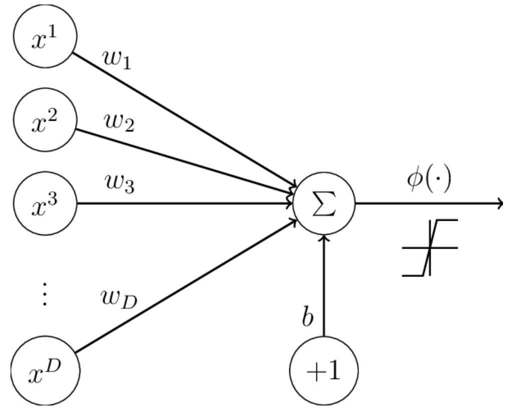

Figure 2.2.: Abstraction of a Neuron [DMK+12]

The most basic building block of the human central nervous system is a neuron which can receive multiple stimuli and is able to produce an output if the combined stimulation exceeds a threshold.[citation needed] One such neuron and its stimulus measure are depicted in 2.1. Another functionality observed in nature is the ability of a neuron to strengthen the connection to any source of stimulus thus giving said source more influence on whether the neuron produces an output. [citation needed]

The canonical mathematical model of a neuron, as seen in 2.2, is defined as:

- **Inputs** x_i **:**
  All stimuli of a neuron are simply referred to as its inputs

- **Weights** w_i **:**
  The ability to assign importances is modelled as weights which are coupled to specific stimuli

- **Combined Weighted Inputs** $\sum_{i=1}^{n} w_i x_i$ **:**
  After the inputs are scaled by their according weight they superpose to form the total excitation of the neuron

- **Activation Function** $\Phi(\sum_{i=1}^{n} w_i x_i)$ **:**
  Correlation between excitation of an neuron and the signal thus produced

- **Bias** b **:**
  Base excitation used to model a neurons sensibility to excitation

2.1.2 Data

In addition to this model a numerical representation of any utilized data is also needed. A single data point is represented as a collection of inputs. Generally two descriptions can be distinguished:

- **One-Hot-Encoding :**
  For each form the data point can assume a single input is modelled. Said input is activated if it fully represents the data otherwise it is not.
  Example:
  If a vocabulary of size n is given its m-th word can be described as a vector with n entries where only the m-th entry is non-zero.

- **Embedding :**
  If the data point can be described through features it can be though of as being embedded in a lower-dimensional more expressive space comparable to one-hot-encoding. An important advantage of this description is the resulting continuity of the input-space.
  Example:
  A sound could be described through its volume, pitch and duration

TODO: multidimensionality

2.1.3 Layers

As an individual neuron is too simple to model any complex relations between inputs and outputs the next step is to aggregate multiple neurons. At the core of neural networks is the idea to collect the signal many different neurons produce for a given input and reuse them as new features for another round of neurons. Such a collection is called a **layer** and especially a **hidden layer** if neither its inputs where original data nor are its outputs the final activations. [3]
A layer is defined by the structure of connections it prescribes. The layers used in this thesis consist of:

- **Input :**
  The numeric representation of data points can be thought of activations a input-layer produces. In applications this layer is commonly used to describe assumptions on the data-points.

- **Fully-Connected | Dense :**
  Every neuron of the layer is connected to every input.

- **Convolution :**
  Every neuron is only connected to a small neighbourhood of features.
  The Parameters are: neighbourhood size, amount of considered neighbourhoods and definition of behaviour at the edges of data points.
  Example:...

- **Pooling :**
  Not a conventional trainable layer but rather a data-processing-step between other layers. Reduces the number of features by condensing a small neighbourhood into a single feature.

- **Flatten :**
  Not a conventional trainable layer but rather a data-processing-step between other layers. Collapses features from multiple dimensions into a single one.

2.1.4 Architectures

The collection of layers used for a given task is called an **architecture**. As multiple architectures are discussed throughout this work a clear system to note them is fundamental. Any architecture description first declares all default assumptions on its layers. Afterwards a list of layers follows defining the type of said layers, their hyper-parameters and especially the dimensionality of their outputs. Additionally and in interest of compatibility the following notation while be close to the functional Keras-API utilized

[3] This hierarchy of abstracts features is essential to the descriptive abilities of neural network. As such any sensible function between input and output can be approximated up to arbitrary precision by a network with at least one hidden layer. [HSW89]

throughout the associated source code.
The following two examples are meant to illustrate the notation:

1

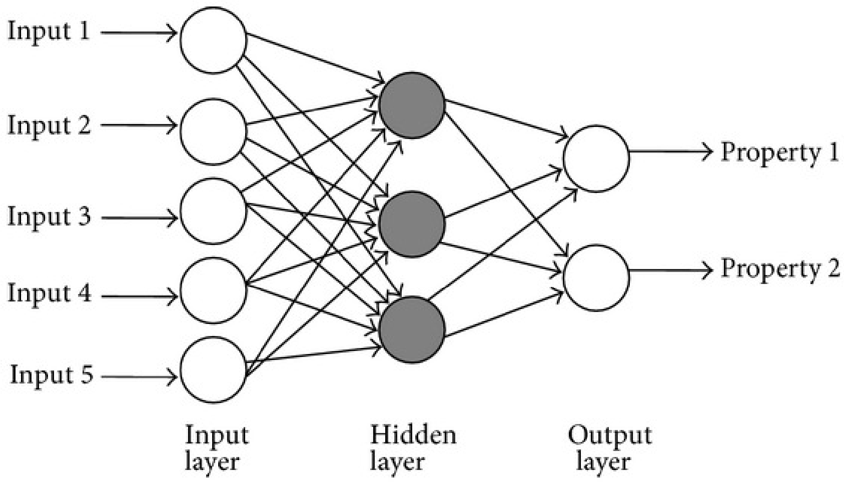

Figure 2.3.: Architecture of a small fully-connected network [Bel18]

Simple-FCN | 2.3

Defaults	Dense: activation	rectified linear unit
Input	output dimension	5
Dense	output dimension	3
Dense	output dimension	2
	activation	softmax

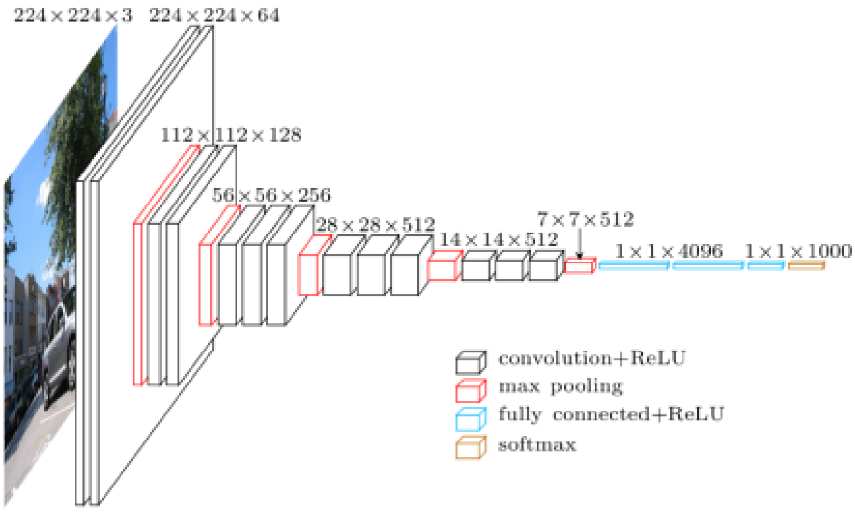

Figure 2.4.: Macroarchitecture of VGG16
[Fro]

VGG-16 | 2.4

Defaults	Convolution: kernel size	[3,3]
	Convolution: stride	[1,1]
	Convolution: paddig	same dimension zero padding
	Softmax: kernel size	[2,2]
	Softmax: stride	[1,1]
Input	output dimension	[224,224|3]
2x **Convolution**	output dimension	[224,224|64]
Softmax	output dimension	[112,112|64]
2x **Convolution**	output dimension	[112,112|128]
Softmax	output dimension	[56,56|128]
3x **Convolution**	output dimension	[56,56|256]
Softmax	output dimension	[28,28|256]
3x **Convolution**	output dimension	[28,28|512]
Softmax	output dimension	[14,14|512]
3x **Convolution**	output dimension	[14,14|512]
Softmax	output dimension	[7,7|512]

VGG-16 | 2.4

Flatten	output dimension	25.088
Dense	output dimension	4096
Dense	output dimension	4096
Dense	output dimension activation	1000 softmax

2.2 Pruning

As the computational power of modern devices increases ever larger architectures become possible. While this allows for more precise models on any given data it is important to recall that pure representation is not the ultimate goal of most applications. Being more concrete, all tasks discussed in this work can be categorized as **supervised learning**, meaning they provide a collection of labelled data points and demand an extrapolation of the implicit labelling process, called **generalization**. It is a well known phenomenon in the field of supervised learning[4] that generalization suffers from over-adaptation to the given data and that said over-adaptation tends to happen more easily in complex architectures.
On the other hand does massive parametrization not only enable us to approximate the labelling process[5] more precisely but also to find such an approximation in a feasible way. [DSD$^+$13].
Pruning is a compromise in which a large model is fitted to a given data set and then truncated as much as possible while maintaining accuracy.

2.3 Preprocessing for Natural Language Processing

In addition the previously mentioned treatment for any dataset there are additional preprocessing steps when handling text-inputs in natural language[6]. The most important ones are **tokenization**, the separation of a text into words and/or sentences, and **stopword removal**, the removal of little to no syntactic or semantic importance.
As the former is almost always necessary to even quantify the numeric representation of the datapoints most frameworks provide datasets already preprocessed in such a manner.
The later is a canonical inclusion into any natural-language-processing data-flow but also is generally not preemptively applied to dataset.

4 called **overfitting**

5 Which can be encoded as a function

6 The term **natural language** describes language written, spoken and otherwise used by humans in contrast to precisely defined languages used for communication between computation devices.

3 Related Work

To quantify the goals previously defined the context of current research is needed. The importance of any work assuming an underlying architecture can not be correctly evaluated without knowledge about the quality of said architecture. As such this section shortly presents state-of-the-art approaches to the tasks relevant to this thesis. Additionally an overview over previous compression methods and their achievements is given.

3.1 State of the art: Image Classification

MNIST and CIFAR-10 are both datasets containing small images which are to be classified according to the object they display. While MNIST contains gray scale images of hand-written digits CIFAR-10 consists of colorful real-world images. State of the art approaches deliver superhuman accuracy on both data sets.
For MNIST Kowasari et al. with their random multi-purpose deep learning ensemble report the nominal highest performance [KHB$^+$18] although many others achieve similar results through varying means.
Already in 2012 Ciresan et al. describe a deep and sparse convolutional architecture that resembles the visual cortex of mammal [CMS12]. Later Sato et al. apply data-augmentation [SNY15], Chang Jia-Ren & Chen Yong-Sheng package whole architectures and treat them like layers [CC15] and Hasanpour et al. carefully design a small and simple convolutional network through the use of structural heuristics [HRFS16] all reproducing the same performance.
In contrast the three best-performing approaches to CIFAR-10 are all published in 2019. Currently an ensemble of auto-encoding transformations claims the highest performance. Wang et al. provide their model with a rich class of transformations to prepare abstraction of the input. [WKLQ19]. Close second and third are Cai et al. with a direct network-architecture-search scheme [CZH18] and Hu et al. with a novel network building block that explicitly models interaction between channels [HSS17].
While Frankle & Carbin do not provide exact values in the LTH-paper their figures indicate that they achieve roughly 98% accuracy on MNIST and 90% on CIFAR-10 [FC18].[1] This result is reproducible with the source code provided alongside this thesis.

Accuracy %	MNIST	CIFAR-10	Published
EnAET		98.0	2019
DirNAS		97.9	2019
Squee		97.88	2019
RMDL	99.82	91.2	2018
Simple	99.8	95.5	2016
BatchNorm	99.8	93.3	2015
APAC	99.8	89.7	2015
Multi-Column	99.8	88.8	2012
Lenet-FCN	~98		LTH
VGG-19		~90	LTH

Figure 3.1.: Performance for Image Classification

[1] State-of-the-Art architectures are presented only if no extra training data was used and as described on https://paperswithcode.com/sota

Accuracy %	20-News	Reuters	Published
Neural BoE	88.1		2019
Graph Star	86.9		2019
RMDL		90.69	2018
multi-scale CNN	86.12		2018

Figure 3.2.: Performance for Topic Classification

3.2 State of the art: Topic Classification

In the field of NLP topic classification is arguably the task most similar to image classification and Reuters-21578 is arguably the most iconic dataset for such a task. Yet neither do its corresponding state of the art architectures compare sensibly to the ones studied by Frankle & Carbin nor is Reuters-21578 structurally akin to MNIST. The essential differences will be covered in section 6.
20-Newsgroup is another NLP data set not only more aligned with MNIST and CIFAR-10 but also with an competitive CNN architecture exists. In their work Pappagari et al. develop an approach integrating the implicit verification objective and learning multiple language models for different channels of their CNN [PVD18]. They come close to state of the art performance on 20-Newsgroup.

3.3 Early Pruning

Beginning around 1990 with M.C. Mozer & P. Smolensky [MS89] as well as LeCun et al. [LDS90] weights were being removed from neural networks after training them for a task. Shortly thereafter the idea of further training a pruned network was proposed [HS93] which became common practice over the next decade. While LeCun et al. describe a network compression factor of $\times 4$, more recent works achieve a factor of $\times 9$ to $\times 16.6$ while loosing no or close to no accuracy [HPTD15] [LWL17]. Frankle & Carbin report pruning rates over 98,5% of weights in one of their networks while maintaining network capabilities which amounts to a compression rate of over. [FC18] $\times 50$

In a recent paper [LSZ+18] Z.Liu et al. observe that if pruned networks are trained with randomly reinitialized weights instead of fine-tuning their previous ones they retain from the original network, the pruned networks keep their capabilities. They conclude that said weights can not be essential to a pruned networks quality, contrary to prior common belief. Thus Z.Liu et al. claim that the architecture of pruned networks is responsible for its capabilities and furthermore that pruning can be interpreted as a kind of network architecture search .
After the effectiveness of pruning is established and its interpretation as network architecture search becomes available there is a legitimate question whether all the weights in a network are really necessary for all of the training. In a paper of Y. Li & W. Zhao & L. Schang from early 2019 [LZS19] they describe a method named IPLT to prune common convolutional network architectures at the filter level and especially before convergence. Thus they do not only compress the networks by a factor of $\times 10$ but also speed up training by a similar magnitude. If the LTH can be applied in such a fashion a speed-up of up to $\times 20$ should be expected.

3.4 Additions to the Lottery Ticket Hypothesis

Even though the Lottery-Ticket-Hypothesis was only proposed earlier this year additional papers on the topic exist. In a paper from June 2019 J. Frankle & M. Carbin et al. [FDRC19] expand their method to find winning tickets on deep convolutional network architectures that proved difficult before. They attribute this achievement to the decision of not returning to the very first state of the network but to

one a few iterations into training. Not only does this mark a lower limit for how early pruning is possible with the LTH but i also implies that a certain structure emerges after little training of the big network. Whether said structure only marks a point for valid reinitialization or rather already one for magnitude-based pruning is part of what this thesis wants to explore.
Additionally H. Zhou et al. [ZLLY19] document an ablation study on the phenomenon of lottery tickets. They reaffirm the initially naive magnitude-based pruning and describe "supermasks" that improve accuracy when applied to the initial network even without additional training. Finally they find that a replacement of all weights in the pruned network by a constant with same sign as said weights does not significantly influence the networks capabilities. As such H. Zhou et al. conclude that the sign of weights are the essential property for such neural networks.

4 Design

4.1 Components

Aim of the Experiment

The performed experiments pursue different goals. At first, the validation of the code base, used in the remaining experiments, is necessary. Next, one experiment on the search for early lottery tickets is conducted and finally, a transfer to an NLP-task is attempted.

Dataset and Preprocessing

Various Datasets are used for this thesis. A more thorough description of each one is given in section 6. If any preprocessing was used it is explained at this point.

Task

For each Dataset, multiple tasks are reasonable. A collection of text, for example, could either be classified by one network or compressed by another. The task informs the structure of the network's output, and possibly its entire design. Different Tasks might also vary greatly in difficulty.

Architecture

The specific shape of a network is called the **architecture**. A precise description of a network's architecture is vital to the reproducibility of any experiment. All necessary parameters are given here. Any parameters that were inferred, because they are indiscernible from the referenced papers, are mentioned here. If I found parameters to be inconsistent or incompatible with each other it is discussed in this subsection.

Pruning

Not all architectures are pruned in the same fashion. In their paper, J. Frankle and M. Carbin used different pruning percentages for different kinds of layers.[FC18] Additionally, their results show that the quality of different architectures degrades at different speeds concerning the number of weights pruned. Thus the number of pruning iterations varies over different experiments, which is discussed here shortly. Finally, the layers in which pruning is applied are enumerated together with their corresponding pruning percentages.

4.2 Reproduction: Dense Network | MNIST-Lenet-FCN

Aim of the experiment

Pruning the most basic architecture examined by J. Frankle and M. Carbin served as a minimal working prototype for the codebase.

Dataset and Preprocessing

For this experiment, I employed the image dataset MNIST. It contains gray-scale images of hand-written digits with a size of 28x28 pixels. [ref section 6.1] No preprocessing was applied.

Task

The network was supposed to classify each image according to the digit it displays.

Architecture and Setup

MNIST-Lenet-FCN

Model	loss	categorical crossentropy
	Optimizer	Adam
Optimizer	learning rate	$1.2 \cdot 10^{-4}$
Defaults	Dense: activation	rectified linear unit
Input	output dimension	[28|28]
Flatten	output dimension	784
Dense	output dimension	300
Dense	output dimension	100
Dense	output dimension	10
	activation	softmax
Training	epochs	50
	batch size	60
Pruning	layers	Dense
	amount	20%
	iterations	25
	initial weights	266.610
	remaining weights	$\sim$1007

4.3 Reproduction: Convolutional Network | CIFAR10-CNN-6

Aim of the Experiment

The first network is the simplest example of architectures discussed by J.Frankle and M.Carbin. The "conv-6", they propose, utilizes an additional popular kind of trainable layer, the convolutional layer, and has an order of magnitude more weights than the previous network. Furthermore, it operates on an arguably more difficult dataset, CIFAR10. In summary: This architecture uses close to all features present in the original paper, which makes it valuable for the validation of the code I produced.

Dataset and Preprocessing

For this task, I utilized the image dataset CIFAR10. In contrast to MNIST, CIFAR10 contains colored images with a size of 32x32 pixels. Additionally, each image is subdivided by gray-scale images for its share of red, blue, and green. The result is the final size of 3x32x32 pixels. [ref Section 6.2] No preprocessing was applied.

Task

The network was supposed to classify the image according to the common real-world objects displayed on them. The ten possible objects include different means of transportation and animals.

Architecture and Setup

J. Frankle and M. Carbin developed the "conv-6" based on the VGG-architectures and only note the parameters necessary to infer the remaining parts of the infrastructure. [cite LTH] I based my implementation on those parameters and the referenced paper of K. Simonyan and A. Zisserman [cite VGG], and the number of weights in the dense part differs from the number reported by J.Frankle and M.Carbin. Because they do not supply an openly accessible implementation of their experiments, it was not possible to cross-validate the code. As the most natural way, to prepare a multidimensional input for a dense layer, is flattening, I assume that J. Frankle and M. Carbin either reported the wrong number of weights or parameters in their description.

CIFAR10-CNN-6

Model	loss	categorical cross entropy
	Optimizer	Adam
Optimizer	learning rate	$3 \cdot 10^{-4}$
Defaults	**Dense**: activation	rectified linear unit
	2D Convolution: activation	rectified linear unit
	2D Convolution: kernel size	[3|3]
	2D Convolution: edge padding	same
	2D Max Pooling: pool size	[2|2]
	2D Max Pooling: strides	[2|2]
Input	output dimension	[32|32|3]
2D Convolution	number of filters	64
	output dimension	[32|32|64]
2D Convolution	number of filters	64
	output dimension	[32|32|64]
2D Max Pooling	output dimension	[16|16|64]
2D Convolution	number of filters	128
	output dimension	[16|16|128]

CIFAR10-CNN-6

2D Convolution	number of filters	128
	output dimension	[16|16|128]
2D Max Pooling	output dimension	[8|8|128]
2D Convolution	number of filters	256
	output dimension	[8|8|256]
2D Convolution	number of filters	256
	output dimension	[8|8|256]
2D Max Pooling	output dimension	[4|4|256]
Flatten	output dimension	4096
Dense	output dimension	256
Dense	output dimension	256
Dense	output dimension	10
	activation	softmax
Training	epochs	36
	batch size	60
Pruning	layers	Dense
		2D Convolution
	amount	20%
		15%
	iterations	25
	initial weights	1.117.194
		1.145.408
	remaining weights	~4220
		~19698

4.4 Early Ticket: MNIST-Lenet-FCN

As this experiment shares an architecture with the reproduction discussed earlier, redundant subsections are omitted.

Aim of the Experiment

In the introduction of this thesis, I remarked that there is no inherent necessity that one defines the structure of lottery tickets after full training of a network. Such a definition is natural, but in the end, J. Frankle and M. Carbin perform network architecture search on the initial network. The trained weights are only used to inform this search. In principle, searching for a performant architecture could be done without any training, using only the initialized weights, but H. Zhou et al. rule out that possibility in their ablation study. [cite Deconstruction] This experiment aims to study the behavior of lottery tickets dependent on the point in training when the weights are used to inform the pruning.

Pruning

The network converges no later than 15 epochs into training. Thus 15 experiments were performed, each being set to another epoch for pruning. To ensure comparability all 15 networks share the same initialization, and each training is run for the full 50 epochs of the original experiment.

4.5 Transfer: Newsgroups-End2End-CNN

Aim of the Experiment

J. Frankle and M. Carbin report a desirable degree of pruning through the search for lottery tickets, but all their results pertain only to the field of image recognition. This experiment aspires to be a proof-of-concept for the search for lottery tickets in natural language applications. To this end, the code reproduces the network of an approach, of R. Pappagari et al., that achieved performance close to the state-of-the-art on a natural language processing task[cite End2End].

Dataset and Preprocessing

The natural language dataset used for this experiment is called "20 Newsgroup". It contains articles of varying lengths in plain text. As networks only handle numerical values, the documents had to be quantified. R. Pappagari et al. one-hot-encoded the documents on a word level, utilizing the vocabulary provided on the 20 newsgroup website. [footnote] While they mention that they used the canonical split of training and test data, this is not enough to accurately define the setup. First, the documents should be stripped of any metadata. Afterward, a tokenizer, of which many different ones exist, is necessary to split the articles into single words. The code provided along this thesis utilizes the word tokenizer supplied by the framework nltk. [footnote] Furthermore, the provided vocabulary does not contain all tokens. For this experiment, all such weights were removed all such tokens as stopwords. Lastly, the input length of a network cannot be variable. While a few documents have an extreme length of over 3000, most of them do not [footnote]. Thus simple zero-padding would overexert the computer memory and over parametrize the architecture. As such, the preprocess truncated all documents after the first 200 words and padded the rest.

Task

For each document, the network has to determine precisely one out of 30 possible topics.

Architecture and Setup

Embedding layers are dense layers with one-hot input and special implementation. As such they are pruned like dense layers

Newsgroup-End2End-CNN

Model	loss	sparse categorical cross entropy
	Optimizer	Adam

Newsgroup-End2End-CNN

Sequential Layers	**Embedding**	input dimension = 61188
		input length = 200
		output dimension = 300
	1D Convolution	filters = 3
	1D Average Pooling	
	Dropout	rate = 0.5
	1D Global Average Pooling	
Input	output dimension	[61188|200]
Sequential A1	input from	Input
	1D Convolution	kernel size = 1
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A2	input from	Input
	1D Convolution	kernel size = 4
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A3	input from	Input
	1D Convolution	kernel size = 7
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A4	input from	Input
	1D Convolution	kernel size = 10
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A5	input from	Input
	1D Convolution	kernel size = 13
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A6	input from	Input
	1D Convolution	kernel size = 16
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential A7	input from	Input
	1D Convolution	kernel size = 19
	1D Average Pooling	pool size = 2

Newsgroup-End2End-CNN

	output dimension	3
Sequential A8	input from	Input
	1D Convolution	kernel size = 22
	1D Average Pooling	pool size = 2
	output dimension	3
Sequential B1	input from	Input
	1D Convolution	kernel size = 1
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B2	input from	Input
	1D Convolution	kernel size = 4
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B3	input from	Input
	1D Convolution	kernel size = 7
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B4	input from	Input
	1D Convolution	kernel size = 10
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B5	input from	Input
	1D Convolution	kernel size = 13
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B6	input from	Input
	1D Convolution	kernel size = 16
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B7	input from	Input
	1D Convolution	kernel size = 19
	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B8	input from	Input
	1D Convolution	kernel size = 22

Newsgroup-End2End-CNN

	1D Average Pooling	pool size = 7
	output dimension	3
Concatenate	input from	Sequential A1
		Sequential A2
		Sequential A3
		Sequential A4
		Sequential A5
		Sequential A6
		Sequential A7
		Sequential A8
		Sequential B1
		Sequential B2
		Sequential B3
		Sequential B4
		Sequential B5
		Sequential B6
		Sequential B7
		Sequential B8
	output dimension	48
Dropout	input from	Concatenate
	rate	0.5
Dense	input from	Dropout
	output dimension	20
	activation	softmax
Training	epochs	10
	batch size	60
Pruning	layers	Embedding
		1D Convolution
		Dense
	amount	20%
		15%
		20%
	iterations	10
	initial weights	293.702.400
		165.648
		980
	remaining weights	~31.536.055
		~32.611

Newsgroup-End2End-CNN

	$\sim$105

5 Implementation

Hint:

This chapter should describe the details of the implementation addressing the following questions:

1. What are the design decisions made?
2. What is the environment the approach is developed in?
3. How are components mapped to classes of the source code?
4. How do the components interact with each other?
5. What are limitations of the implementation?

The section should have a length of about five pages.

5.1 Design Decisions

Tensorflow 2.0 is used as the framework for all experiments presented in this thesis. It enables software development on a high level of abstraction while ensuring code performance. Because some procedures are not naturally compatible with the implementation of Tensorflow 2.0 this thesis needs to employ a few workarounds. Notably pruned weights are not removed from the network but rather set to zero each time a layer is evaluated. As such they do not affect the predictions but are influenced by backpropagation. *The effect on the presented experiments is unclear to me.*

5.1.1 Missing Parameters

Through the related work referenced in this thesis specifications of any one model where incomplete. The following subsection aims to explain how the parameters were inferred or chosen.

- Activation function for most layers
- Activation function for output layers
- Dropout rate
- padding in CNNs
- ...

5.2 Architecture

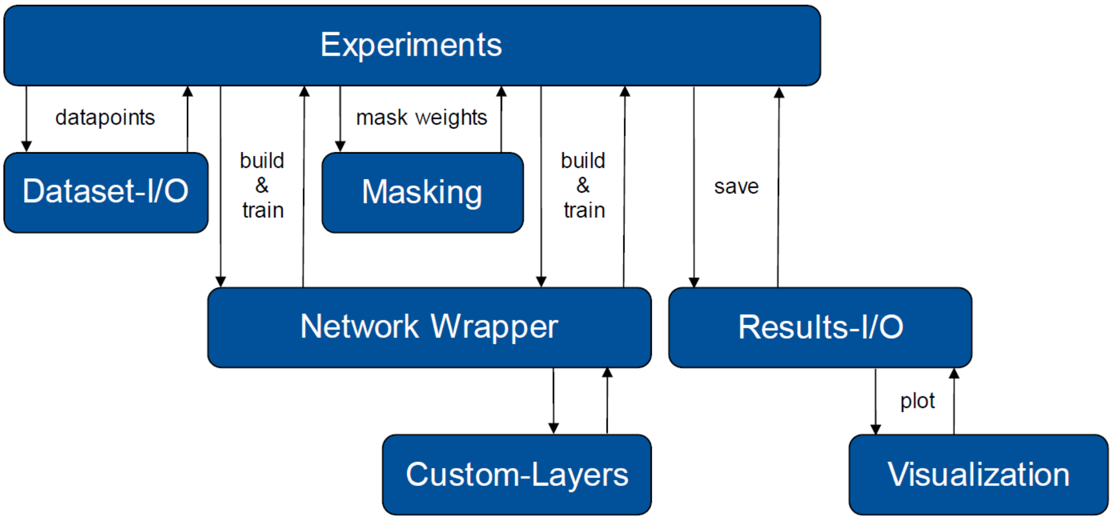

Figure 5.1.: project architecture

5.3 Interaction of Components

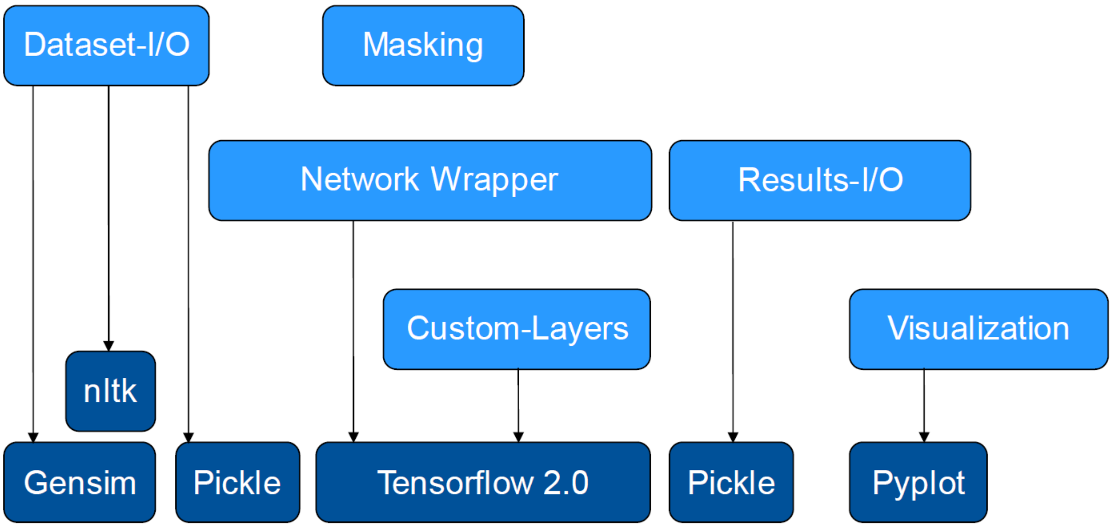

Figure 5.2.: project architecture

5.4 Summary

6 Data Sets

6.1 MNIST

The MNIST-dataset contains 25x25 gray-scale images of handwritten digits padded to 28x28 [YL].

6.2 CIFAR-10

6.3 20-Newsgroup

6.4 Reuters-21578

The Reuters-21578-dataset contains 21578 articles published by the Reuters News Agency in 1987 [Lew]. Reuters-21578 differs from the previous data sets in the sense that it lacks a few fundamental properties. In particular Reuters-21578 is not only multi-class but rather multi-label meaning that any one data point can satisfy multiple categories. Additionally there are categories in Reuters-21578 that have no associated positive example and even for all remaining ones the amount of samples is heavily skewed. In order to restore parts of the missing properties with minimal change to the dataset different subsets of Reuters-21578 have been chosen by different researchers.
F. Debole & F. Sebastiani [DS05] describe those subsets, starting out stating that close to half of the data points are unusable which leaves 12,902 documents. 9,603 are marked for training and 3,299 for validation.[1] They also point out the different groups of categories used for classification:

- **R**(115)
  The group with the 115 categories containing at least one positive training example.

- **R**(90)
  The group with the 90 categories containing at least one positive training and test example.

- **R**(10)
  The group with the 10 categories containing the most examples.

	MNIST	CIFAR-10	20-Newsgroup	Reuters-21578
N. labels	10	10	20	10 to 115
N. datapoints	70.000	60.000	18846	12.902
fixed split	x	x	"bydate"	"ModApté"
shortened			x	x
class imbalance				x
multi-label				x

[1] While different training-splits were used for Reuters-21578 "ModApté" has become the canonical choice

7 Evaluation

Hint:

This chapter should describe how the evaluation of the implemented mechanism was done.

1. Which evaluation method is used and why? Simulations, prototype?
2. What is the goal of the evaluation? Comparison? Proof of concept?
3. Wich metrics are used for characterizing the performance, costs, fairness, and efficiency of the system?
4. What are the parameter settings used in the evaluation and why? If possible always justify why a certain threshold has been chose for a particular parameter.
5. What is the outcome of the evaluation?

The section should have a length of about five to ten pages.

7.1 Goal and Methodology

- Early Pruning:
  Proof of concept
  + trial of select early stopping points

- Transfer to NLP:
  Proof of concept

7.2 Evaluation Setup

- Early Pruning:
  LTH holds for mask with less than 50% of iterations to early stop

- Transfer to NLP:
  Network holds performance to within 1%-point while at least 50% weights are pruned

7.3 Evaluation Results

7.4 Analysis of Results

8 Conclusions

Hint:

This chapter should summarize the thesis and describe the main contributions of the thesis. Subsequently, it should describe possible future work in the context of the thesis. What are limitations of the developed solutions? Which things can be improved? The section should have a length of about three pages.

8.1 Summary

8.2 Contributions

8.3 Future Work

8.4 Final Remarks

Bibliography

[Aro11] Jacob Aron. How innovative is apple's new voice assistant, siri? *New Scientist*, 212(2836):24, 2011.

[Bel18] Soufiane Belharbi. Neural networks regularization through representation learning, 07 2018.

[CC15] Jia-Ren Chang and Yong-Sheng Chen. Batch-normalized maxout network in network. *CoRR*, abs/1511.02583, 2015.

[CDC18] Mary Ann Clark, Matthew Douglas, and Jung Choi. *Biology 2e*. OpenStax, 2018.

[CMS12] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[CZH18] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *CoRR*, abs/1812.00332, 2018.

[DMK+12] Jelena Djuris, Djordje Medarević, Marko Krstić, Ivana Vasiljević, Ivana Aleksić, and Svetlana Ibrić. Design space approach in optimization of fluid bed granulation and tablets compression process. *TheScientificWorldJournal*, 2012:185085, 07 2012.

[DS05] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56(6):584–596, 2005.

[DSD+13] Misha Denil, Babak Shakibi, Laurent Dinh, Marc' Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2148–2156. Curran Associates, Inc., 2013.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018.

[FDRC19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611, 2019.

[Fro] D. Frossard. Macroarchitecture of vgg16. https://www.cs.toronto.edu/ frossard/post/vgg16/.

[Gib16] Elizabeth Gibney. Google ai algorithm masters ancient game of go. *Nature News*, 529(7587):445, 2016.

[HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[HRFS16] Seyyed Hossein HasanPour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *CoRR*, abs/1608.06037, 2016.

[HS93] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

[HSS17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[KHB+18] Kamran Kowsari, Mojtaba Heidarysafa, Donald E. Brown, Kiana Jafari Meimandi, and Laura E. Barnes. Rmdl: Random multimodel deep learning for classification. In *Proceedings of the 2Nd International Conference on Information System and Data Mining*, ICISDM '18, pages 19–28, New York, NY, USA, 2018. ACM.

[LDS90] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[Lew] David D. Lewis. Reuters-21578. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[LF11] Adam Lally and Paul Fodor. Natural language processing with prolog in the ibm watson system. *The Association for Logic Programming (ALP) Newsletter*, 2011.

[LSZ+18] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *CoRR*, abs/1810.05270, 2018.

[LWL17] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[LZS19] Yue Li, Weibin Zhao, and Lin Shang. Really should we pruning after model be totally trained? pruning based on a small amount of training. *CoRR*, abs/1901.08455, 2019.

[MS89] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989.

[PVD18] R. Pappagari, J. Villalba, and N. Dehak. Joint verification-identification in end-to-end multi-scale cnn framework for topic identification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, April 2018.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[SNY15] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. APAC: augmented pattern classification with neural networks. *CoRR*, abs/1505.03229, 2015.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[WKLQ19] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning, 2019.

[YL] Christopher J.C.Burges Yann LeCun, Corinna Cortes. The mnist database. http://yann.lecun.com/exdb/mnist/.

[ZLLY19] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *CoRR*, abs/1905.01067, 2019.

Appendices

A A history of neural networks

- 1. wave: 1955-1970
- 2. wave: 1985-2000 ?
- 3. wave: ???

Figure A.1.: Relative amount of occurences of the word "Perceptron" in published books between 1940 and 2009

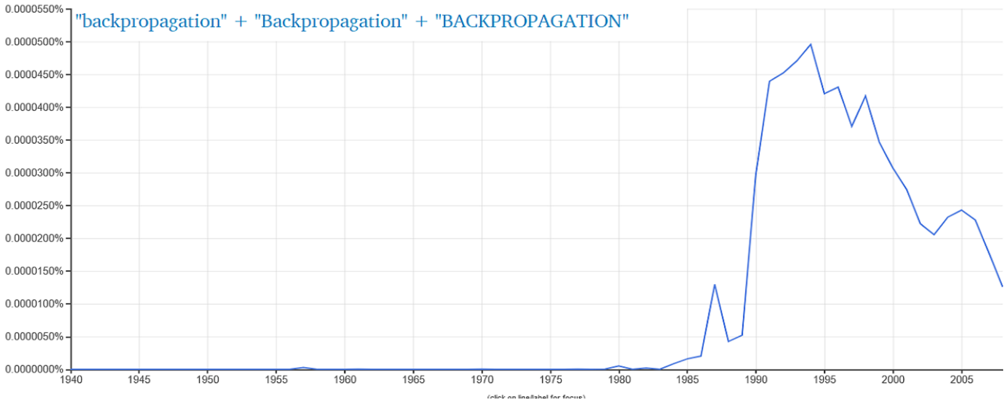

Figure A.2.: Relative amount of occurences of the word "Backpropagation" in published books between 1940 and 2009