Application of the Lottery Ticket Hypothesis in NLP and Early Pruning

Anwendung der "Lottery Ticket"-Hypothese in NLP und frühem Pruning
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
Anwendung der "Lottery Ticket"-Hypothese in NLP und frühem Pruning

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 § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Tim Unverzagt, die vorliegende Bachelor-Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt 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 gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, den 20. Januar 2020

Tim Unverzagt

Contents

1. Introduction **3**
1.1. Motivation . . . 3
1.2. Problem Statement and Contribution . . . 3
1.3. Outline . . . 3

2. Background **5**
2.1. Neural Networks . . . 5
2.1.1. Basics . . . 6
2.1.2. Layers . . . 6
2.1.3. Architectures . . . 7
2.2. Pruning . . . 10
2.3. Non-Numerical Data . . . 10
2.4. Preprocessing for Natural Language Processing . . . 11

3. Related Work **13**
3.1. State of the art: Image Classification . . . 13
3.2. State of the art: Topic Classification . . . 14
3.3. Pruning . . . 14
3.4. Additions to the Lottery Ticket Hypothesis . . . 14

4. Design **17**
4.1. Components . . . 17
4.2. Reproduction: Dense Network | MNIST-Lenet-FCN . . . 17
4.3. Reproduction: Convolutional Network | CIFAR10-CNN-6 . . . 18
4.4. Transfer: Newsgroups-End2End-CNN . . . 20
4.5. Early Ticket: MNIST-Lenet-FCN . . . 21

5. Implementation **23**
5.1. Representation of the components . . . 23
5.2. Execution Flow . . . 23
5.3. Backend . . . 23
5.3.1. Networks . . . 23
5.3.2. Datasets and Preprocessing . . . 24
5.3.3. I/O-Elements . . . 24
5.4. Limitations . . . 24

6. Data Sets **25**
6.1. MNIST . . . 25
6.2. CIFAR-10 . . . 25
6.3. 20-Newsgroups . . . 27
6.4. Reuters-21578 . . . 27

7. Evaluation **31**
7.1. Reproduction . . . 31
7.2. Transfer . . . 34

7.3. Early Tickets 34

8. Conclusions **37**
8.1. Summary 37
8.2. Contributions 37
8.3. Future Work 37

Bibliography **38**

Appendices **43**

A. Detailed Description of 20Newsgroups-End2End-CNN **45**

Abstract

Neural network applications of ever-growing size and their adaptations for a growing number of different and sometimes less powerful devices necessitate pruning, the reduction in network size through the removal of superfluous substructures. The term pruning usually implies that the network in question has absolved training until convergence; techniques for a-priori reductions are generally referred to as network architecture search.
While most pruning techniques retain the weights of the trained network, in their paper "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," J. Frankle and M. Carbin present a novel technique which resets the weights of the pruned network to their pre-training values.

The primary aim of this thesis is to extend their work and research whether the presented algorithm can provide competitive pruning on different datasets, specifically in the field of natural language processing. As the reset to initial values, leaves said algorithm similar to a network architecture search method; this work studies how the pruning develops in quality as a search characteristic over the amount of training provided beforehand.

In pursuit of these two goals, we developed a framework through which we implemented four experiments. The first and second experiments aim to confirm the validity of the codebase. Subsequently, the third one intents to establish an argument for the usage of the pruning method in the field of natural language processing. With the last experiment, we plan to provide explorative data to inform further study on the emergence of actionable experience in the values of a network.

While the first two experiments fail to reproduce the results of J. Frankle and M. Carbin, one of them prunes its associated networks up to the scale contemporary technique, namely a size reduction of 90 percent with little loss in accuracy. Additionally, the second result conforms with their definition of a lottery ticket, achieving the same or even better quality measure, up to the same degree of compression. Finally, the data of the exploratory experiment show low legibility due to the erratic results of single prunings. Nonetheless, we find that no pruning reproduces the same accuracy reached before, even if the latter ones performed enough training for the full training to converge. It suggests that the researched pruning algorithm is highly dependent on a substantial amount of training.

1 Introduction

The thesis on hand discusses the possible extensions of a specific neural network pruning technique presented by J. Frankle and Michael at the beginning of 2019.[FC18]
The next few paragraphs motivate the research, specify its extent, and give an outline for the remaining chapters.

1.1 Motivation

Over the last decade, neural networks have become ever more prevalent in applications of any kind. Not only can they theoretically approximate most mappings between inputs and expected output[1] up to arbitrary precision[2], but in practice, they can also converge on such an approximation via gradient descent. [HSW89]
As computational resources became more available, state-of-the-art approaches featured ever-larger networks. Although they excel at their tasks, it is challenging to execute them on small portable devices such as smartphones. Pruning techniques aim to reduce the size architectures have at runtime through the removal of parts that are no longer necessary after training. Still, there is no consesus on how to identify these superfluous sections.
In their paper "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," J. Frankle and M. Carbin demonstrate an innovative algorithm to remove up to 90 percent of various networks while retaining most or all of their performance. These results are comparable to other pruning techniques while promising new insights into the nature of neural networks.[FC18]
The following chapters of this work explore the transfer of the Lottery Ticket Hypothesis to another field as well as earlier retrieval of lottery tickets on a previously studied architecture.

1.2 Problem Statement and Contribution

In the latter half of their paper, J. Frankle and M. Carbin also discuss the limitations of their work. First and Foremost, they applied their method only to small datasets because iterative pruning is computationally expensive.[FC18]
Both datasets they studied are from the field of image recognition, and it is uncertain whether their results are transferable to other contexts such as natural language processing or voice recognition.
Furthermore, J. Frankle and M. Carbin acknowledge that pruning single connections instead of whole sections of a network does not line up well with modern frameworks, and thus, no actual speed-up is achieved. Finally, they state that they plan to research ways to find lottery tickets earlier or at smaller sizes.[FC18]

1.3 Outline

After this introduction, chapter 2 of this thesis establishes the necessary background for the upcoming descriptions and discussions. Afterward, section 3 gives an overview of the work already done on the Lottery Ticket Hypothesis and the tasks absolved during the experiments. Chapter 4 describes the design

1 The class of networks with at least one hidden layer can approximate any borel-measurable function. Most real-world mappings are implicitly assumed to be measurable functions as long as they deterministically map one specific input to a single specific output.
2 The network in question must be allowed arbitrary size to achieve an approximation of arbitrary precision.

of the experiments without mention of the implementation details, which follow in section 5. Before the discussion of the results, chapter 6 supplies additional information on the datasets utilized in this thesis. Finally, section 7 presents an evaluation, and chapter 8 conludes this work with a summary and a mention of possibilities for future work.

2 Background

A joint base of concepts is necessary for the development of this thesis. This chapter aims to establish the fundamental ideas necessary to comprehend this thesis.

2.1 Neural Networks

Neural networks are a part of most breakthroughs in artificial intelligence over the last decade enabling computers to compete in fields formerly championed by humans.[1] They implement a statistical understanding of **supervised learning**, 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. In contrast to other approaches, which directly divine behavior rules from expert knowledge, they are far more dependant on any training data.[HTF09]
Apart from the provided data, the essential point of the design is the class of all possible models the approach could represent after training. A multitude of properties are usually sought 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]
  A model class lacking richness is inherently restricted, and thus the underlying mapping between inputs and outputs might 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 fail to represent the given training data appropriately. This phenomenon is called **underfitting**.
- **Stability:**
  The tendency of similar models in the class to handle inputs similarly. If a model class shows unstable behavior, defining a sensible way to search it for good models becomes difficult.
- **Interpretability of Models:**
  Interpretability expresses the ease of formulating knowledge about a task given a well-performing model of the class. As fields exist in which statistical approaches outperform experts, the extraction of knowledge understandable and applicable by humans is of particular interest.

If one knows an entity that already performs well on a given task, it is a sensible approach to design one's model class to reproduce its decision process. Humans usually embody such entities for many tasks of interest in real-world applications, so they are a natural source of inspiration. In a nutshell, neural networks 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 humans [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 any model of said class.

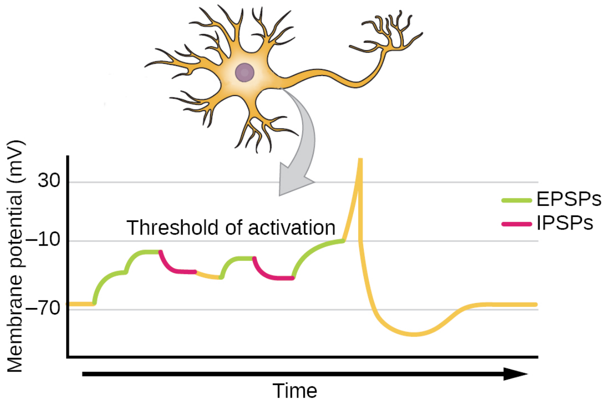


Figure 2.1.: Representation of a biological Neuron [CDC18] edited

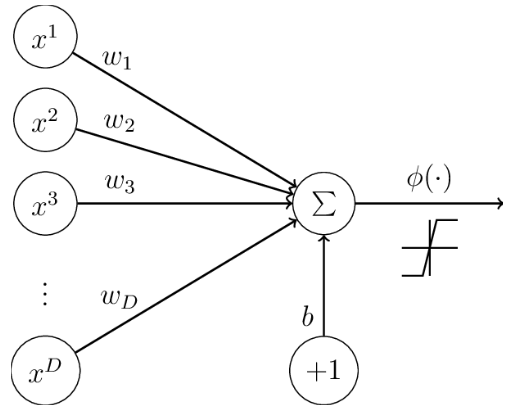


Figure 2.2.: Abstraction of a Neuron [DMK+12]

2.1.1 Basics

The most fundamental building block of the human central nervous system is a neuron that can receive multiple stimuli and can produce an output if the combined stimulation exceeds a threshold. Figure 2.1depicts one such neuron and its stimulus measure. 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.[BH00]

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

- **Inputs** x_i **:**
  All stimuli of a neuron are either referred to as its inputs or as the features of the data.

- **Weights** w_i **:**
  The ability to assign importance to specific stimuli is modeled as weights.

- **Combined Weighted Inputs** $\sum_{i=1}^{n} w_i x_i$ **:**
  A summation superposes the inputs, scaled according to their respective weight, to form the total excitation of the neuron.

- **Activation Function** $\Phi(\sum_{i=1}^{n} w_i x_i)$ **:**
  The monotonously ascending activation function defines the Correlation between excitation of the neuron and its output signal.

- **Bias** b **:**
  As different neurons display varying thresholds of activation, the previously mentioned summation includes an input independent base term. This term effectively shifts the activation of the modeled neuron.

2.1.2 Layers

An individual neuron is too simple to model any complex relations between inputs and outputs. However, certain aggregations of neurons possess the ability to universally approximate functions between input and output.[HSW89]

At the core of of conventional 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 it is neither the input representation nor last in a given network. A layer is defined by the structure of connections it prescribes. The layers used in this thesis consist of: 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 as outputs of an input-layer. In applications, this layer primarily describes assumptions on the shape of the data points.

- **Fully-Connected | Dense :**
  In a fully connected layer, each neuron receives all features of the input, which renders it the most general layer. In theory, it can learn to effectively drop any number of connections, thus reducing to any kind of the subsequent layers, but establishing such behavior through training might be unfeasible. Besides, it would produce enormous computational overhead. The need for more specialized layers motivated this way, is confirmed by their success throughout various fields. Additionally and analogous to the naming of the matrices which implement the computation of neural networks, a fully-connected layer is also called dense.

  The sole parameter of a dense layer is the number of neurons it contains. Nonetheless, the parameters of the neurons are still necessary during implementation.

- **Convolution :**
  Inspired by the convolution kernels used in image processing, a convolutional layer combines features of a relatively small neighboorhood into a single output. This process needs a specification of the data's dimensionality[footnote] and places an implicit bias on local features.

  Convolutions primarily differ in the neighborhood size they define and through the fashion in which they handle missing values at the edges of their data. For example, points beyond the edge of an image could either be defined to be black, white, or even equal to the mean or median of the available part of the convolution. Additionally, contemporary networks utilize striding, where the neighborhods of some features are skipped.

- **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.

2.1.3 Architectures

The collection of layers, and the parameters that define them, is called architecture. In contrast to the term network, an architecture does generally not include the specific trainable weights, but weights close to zero can change the effective shape of an architecture. As this work aims to enable the reproduction of its results and discusses multiple architectures, a transparent system to note them precisely is fundamental. The architecture description we deploy first declares all default assumptions on its layers. Afterward,

a list of layers follows defining the type of said layers, their remaining parameters, and especially the dimensionality of their outputs. Additionally and in the interest of compatibility, the notation remains close to the functional Keras-API utilized throughout the associated source code.
The following two examples illustrate the notation:

1

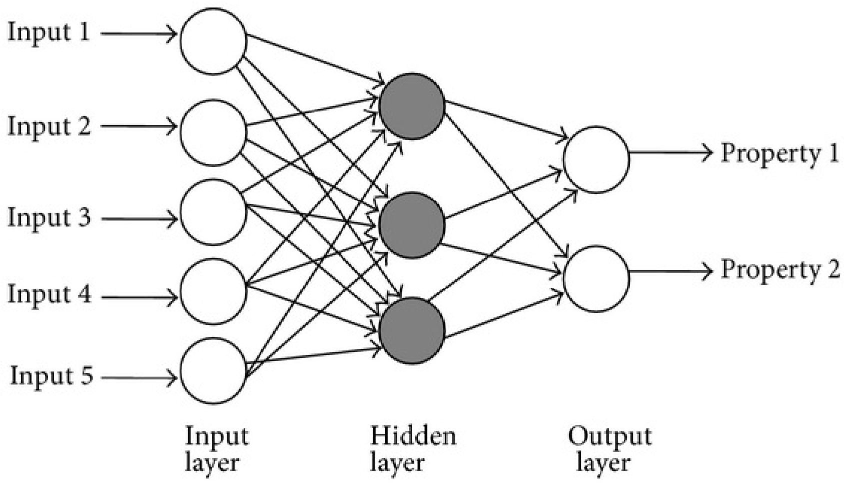


Figure 2.3.: Architecture of a small fully-connected network [Bel18]

Simple-FCN | 2.3

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

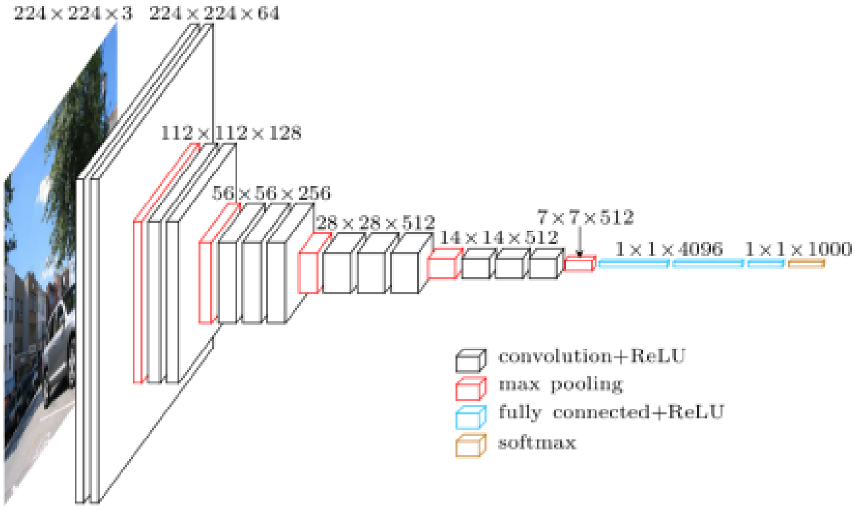


Figure 2.4.: Macroarchitecture of VGG16
[Fro]

VGG-16 | 2.4

Defaults	Convolution: kernel size	[3,3]
	Convolution: stride	[1,1]
	Convolution: padding	same dimension zero padding
	Convolution: activation	rectified linear unit
	Dense: activation	rectified linear unit
	Softmax: kernel size	[2,2]
	Softmax: stride	[1,1]
Input	output dimension	[224,224|3]
2x **Convolution**	output dimension	[224,224|64]
Max-Pooling	output dimension	[112,112|64]
2x **Max-Pooling**	output dimension	[112,112|128]
Max-Pooling	output dimension	[56,56|128]
3x **Convolution**	output dimension	[56,56|256]
Max-Pooling	output dimension	[28,28|256]
3x **Convolution**	output dimension	[28,28|512]
Max-Pooling	output dimension	[14,14|512]

VGG-16 | 2.4

3x **Convolution**	output dimension	[14,14|512]
Max-Pooling	output dimension	[7,7|512]
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.
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, also called **generalization**. It is a well known phenomenon in the field of supervised learning[3] that generalization suffers from over-adaptation to the given data, which tends to happen more easily in bigger architectures.
On the other hand, massive parametrization does not only enable us to approximate the labeling process more precisely but also to find such an approximation feasibly. [DSD+13].
Pruning describes the removal of parts from a trained model, which are estimated to be superfluous after learning. Many researchers have developed pruning methods that achieve a significant reduction in network size with little impact on its performance. Chapter 3 presents a few such techniques.

2.3 Non-Numerical Data

While many phenomena under research, such as digital images or sound recordings, are inherently well representable in numeric terms, others, such as words or sentences in natural language, are not. Before a network can process such data points, they have to be encoded Generally, there are two ways to do so:

- **One-Hot-Encoding :**
  A one-hot-encoding contains a feature for each possible data point. The value in the corresponding dimension is non-zero if, and only if, the feature fully represents the data. If, for example, 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 :**

  Inspired by the way humans can describe various through the use of numerical values in relatively few feature-dimensions, embeddings assign each data point a numerical vector in a space with arbitrary features. If its dimension is significantly lower than the size of the dataset, the embedding describes the data much more efficiently. However, there is no inherent guarantee that this

[3] called **overfitting**

representation retains the information present; In contrast to the one-hot-encoding, which at least remains humanly legible, implying the presence of said information. To avoid its loss, researchers have applied different restrictions to the way data points are distributed in this arbitrary space. In contemporary work about natural language processing, embeddings, where a neural network learned the distribution of words in such a space, have become a useful tool for preprocessing. [Ron14]

2.4 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[4]. 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] 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]

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 **??**.
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 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 $\times 50$. [FC18]
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 it 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

The following section describes the process of this thesis from a high-level point of view. First, all tasks performed before, during, or after the experiments are described. Afterward, these components are used to describe the four experiments carried out for this thesis. For each experiment, a subsection specifies its components and explains any choices made concerning hyperparameters.

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 and Architecture

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. 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 parameters needed to implement the network in our framework are given here. Any parameters that were inferred, because they are indiscernible from the referenced papers, are mentioned here. If we found parameters to be inconsistent or incompatible with each other it is discussed in this subsection.

Experimental Setup

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, we 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	~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 we 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]

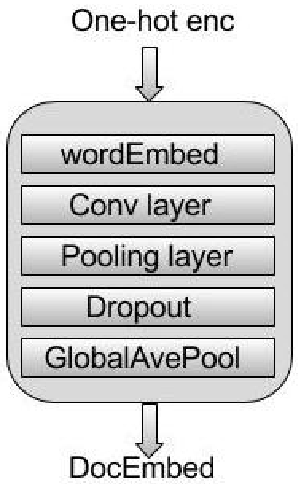
CIFAR10-CNN-6

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

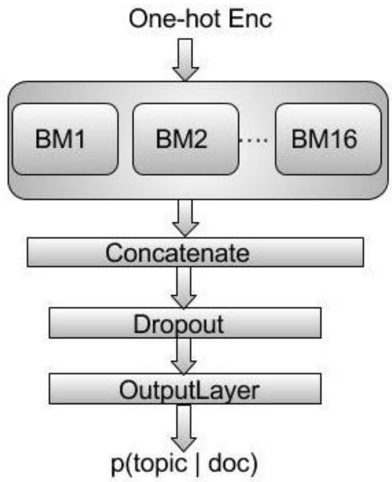
4.4 Transfer: Newsgroups-End2End-CNN

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].



Structure of one sequential subnetwork



Architecture of the whole combined network

Dataset and Preprocessing

The natural language dataset used for this experiment is called "20 Newsgroups". 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.

Architecture and Setup

4.5 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.

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

5 Implementation

Up to this point, this paper presents discussions on a conceptual level. In contrast, this chapter examines the actual execution of the previously presented experiments.

5.1 Representation of the components

In chapter 4, an experiment is defined mainly by its architecture, its training parameters, and its pruning setup. Figure 5.1 clarifies where those components can be found in the framework. Additionally, figure 5.2 describes which datasets are available and which were preprocessed. The natural language dataset, Reuters-21578, consists of legacy code that was not used in any of the presented experiments. It was implemented during the research phase of this thesis.

5.2 Execution Flow

The framework differentiates three layers of abstraction. Any single module should only ever use data on the same level of abstraction. The highest layer defines the training setup, chooses parameters for the remaining layers, collects the resulting training histories, and saves them. Optionally the results can be visualized, either directly or from saved files. On the next layer, the framework loads the dataset and instantiates the network wrapper. Afterward, it trains the network while collecting metrics into histories, and prunes it when appropriate. The lowest layer of the framework forms the interface to the neural network backend. Here, architectures are implements, models are trained models, and weights are masked to model the pruning of connections. Figure 5.3 depicts a scheme of the framework during one of the experiments. The highlighted pieces define the execution flow.

5.3 Backend

5.3.1 Networks

Tensorflow 2.0 supplies a functional API capable of implementing all networks described in this thesis and many more.[cite] Also, it brings execution to other devices than the regular CPU core. Of particular interest to this work is the speed-up achieved through the usage of GPUs.

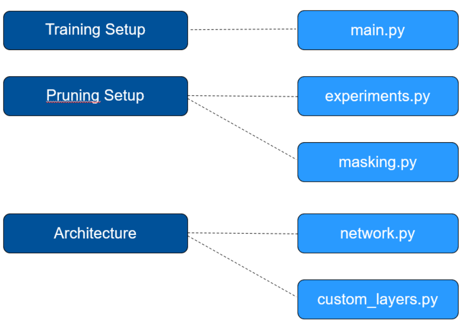


Figure 5.1.: Representation of the main components in the framework

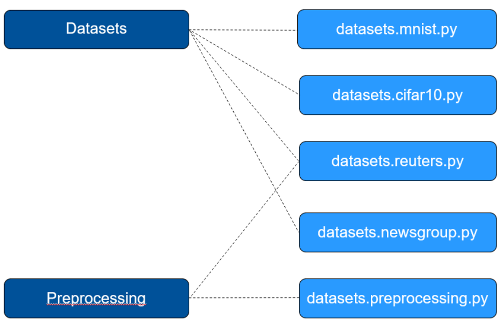


Figure 5.2.: project architecture

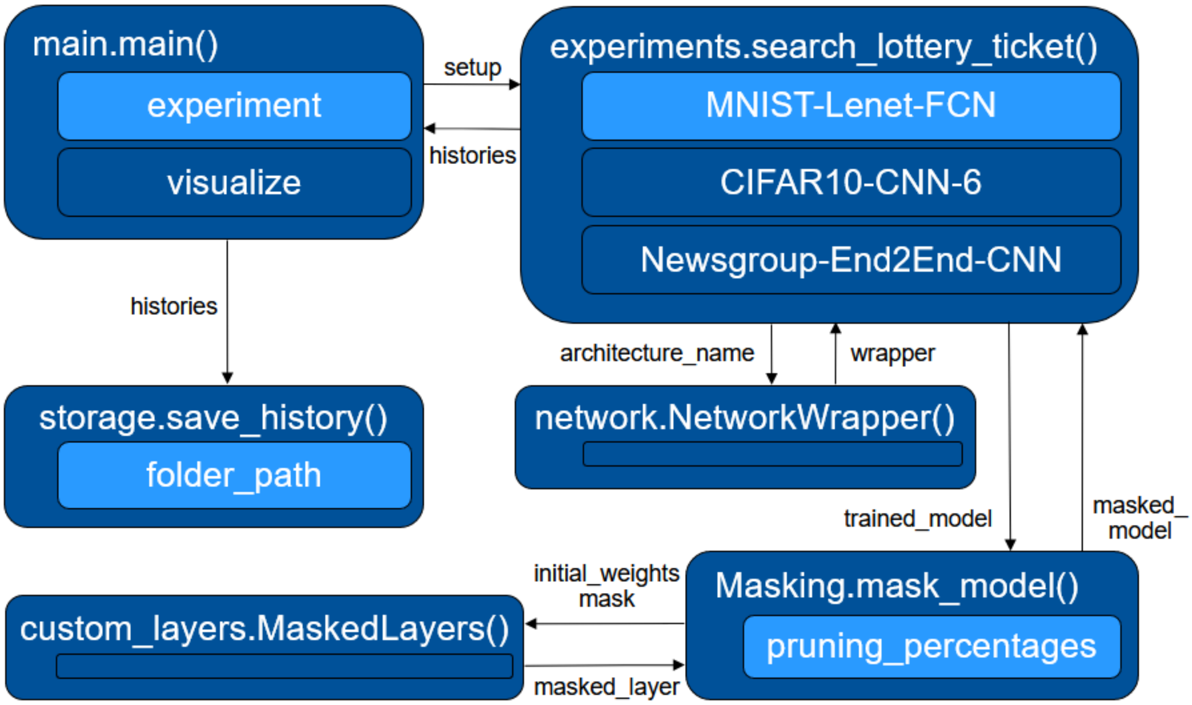


Figure 5.3.: Scheme of the framework during an example experiment

5.3.2 Datasets and Preprocessing

MNIST and CIFAR10 have an interface in Tensorflow 2.0 [cite]. The corresponding modules of the supplied framework read out these interfaces, bring the data into the expected shape, and then split them canonically into training and test datapoints. For 20Newsgroup and Reuters-21578, the backend integrates NLTK, an open-source natural language framework. NLTK also supplies a Word Tokenizer, which splits a raw text into word-like tokens.

5.3.3 I/O-Elements

The supplied framework employs the object-serialization package pickle to save the history of an experiment directly. This type of storage is a solution for internal use only! On their website, the packages authors explicitly warn of untrusted pickle files.

5.4 Limitations

While the code available alongside this thesis is generally capable of unraveling and reproducing any architecture described in the functional API of Tensorflow 2.0, it only supports the masking of the layers described in chapter 4. Additionally, Tensorflow 2.0 itself does not support the pruning of single neural connections because they are bundled together in the tensors used to model layers. As it can only flag whole tensors as non-trainable, a different solution is necessary. While it is less efficient, this thesis' code base resets each pruned weight in a masked layer to zero each time it is called.

6 Data Sets

This thesis deals with three different datasets: MNIST, CIFAR10, and 20Newsgroup. The associated framework contains an additional one, Reuters-21578. While the following table includes primary data about them all, the upcoming sections aim to provide a further description.

Key Attributes of the Datasets

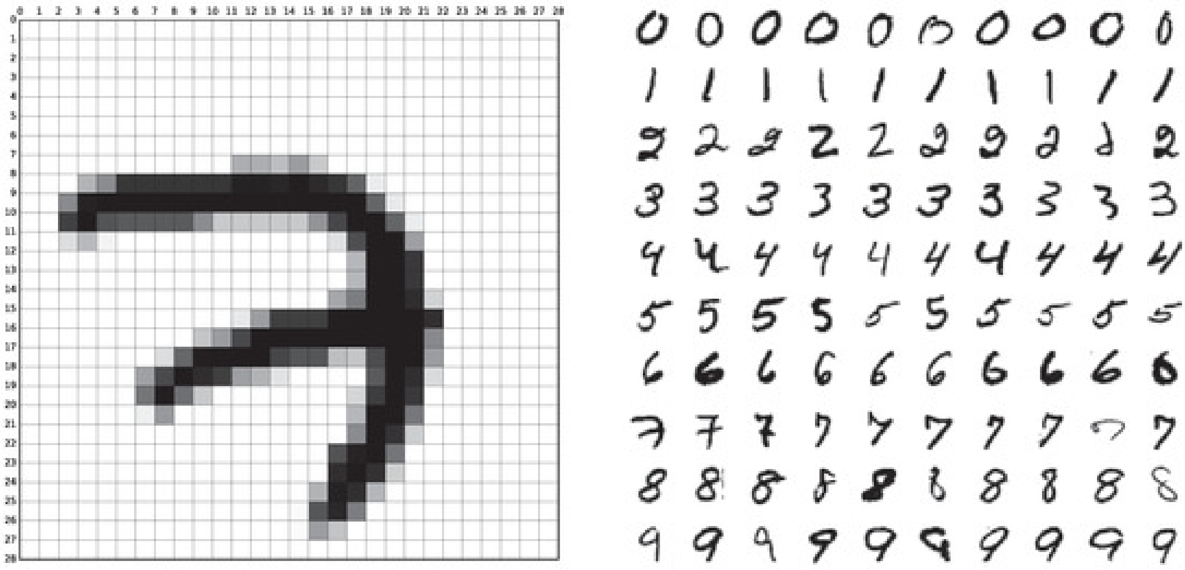
	MNIST	CIFAR-10	20-Newsgroup	Reuters-21578
number of datapoints	70.000	60.000	18846	12.902
number of labels	10	10	20	10 - 115
fixed split	yes	yes	"bydate"	"ModApté"
has metadata	no	no	yes	yes
variable length	no	no	yes	yes
class imbalance	no	no	no	yes
multi-label	no	no	no	yes

6.1 MNIST

MNIST is a collection of gray-scale images depicting handwritten digits collected by Y. LeCun, C. Cortes, and C. J. C. Burges. The authors produced it through the recombination of SD-1 and SD-3, training set and test set of their earlier dataset NIST. They recommend MNIST as an introductory dataset for the study of image recognition methods[YL].
In 1998, Lecun et al. apply different architecture to the image classification task on MNIST, the Lenet-FCN studied in this thesis included. The simplest model, a neural network with a single dense layer, achieved 92,6 percent accuracy.[LBB+98] Figure 6.1 displays a few example images.

6.2 CIFAR-10

CIFAR10 contains colored images of everyday objects that A. Krizhevsky, V. Nair, and G. Hinton collected out of the 80 'million tiny images' dataset. They chose datapoints with mutually exclusive labels and also provide CIFAR100, a related dataset with 100 categories. Figure 6.2 shows 100 images from CIFAR10. [KH+09]
In 2014, T. Chan et al. discuss a simple deep learning network which they intend to be used as a baseline for tasks like object recognition. The architecture achieves an accuracy of 78,7 percent, more than four times the benchmark provided by A. Krizhevsky, V. Nair, and G.Hinton.[CJG+14]



(a) MNIST sample belonging to the digit '7'.

(b) 100 samples from the MNIST training set.

Figure 6.1.: Example images from the MNIST dataset
Source: https://www.mdpi.com/applsci/applsci-09-03169/article_deploy/html/images/applsci-09-03169-g001-550.jpg

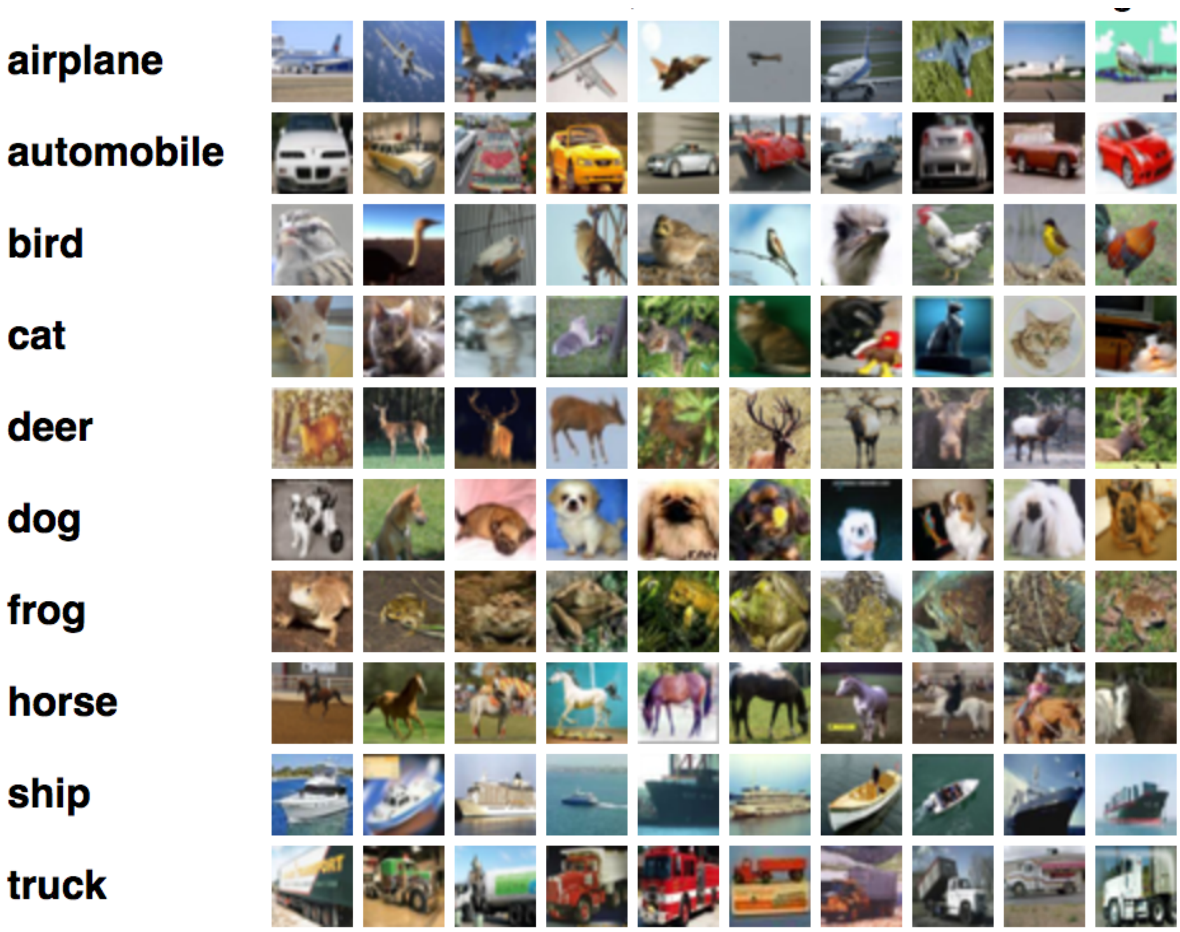


Figure 6.2.: Example images from the CIFAR10 dataset
Source: https://www.cs.toronto.edu/~kriz/cifar.html [Kri]

6.3 20-Newsgroups

Around 1980, a TCP-network named Usenet was established. It distributes articles[1] in a decentralized manner. Any author categorizes his articles according to a specified topic hierarchy whereafter all users of Usenet, who have subscribed to said topic, receive a copy of the document.[2] The text collection defined by such a topic is called a newsgroup. [LF02]
The dataset, 20 Newsgroups, consists of about 20.000 such articles spread over 20 topics. As Usenet defines only eight main categories, the labels in the dataset have to share some of them. The following table displays the relevant hierarchy, and figure 6.3 shows an example text. [Ren]

Topic of articles in 20Newsgroups		
comp.	sys.	imb.pc.hardware
	sys.	mac.hardware
	os.	ms-windows.misc
	windows.	x
	graphics	
rec.	sport.	baseball
	sport.	hockey
	autos	
	motorcycles	
talk.	politics.	misc
	politics.	guns
	politics.	mideast
	religion.	misc
sci.	crypt	
	electronics	
	med	
	space	
soc.	religion.	christian
alt.	atheism	
misc.	forsale	

6.4 Reuters-21578

The Reuters-21578 dataset contains news articles published by the Reuters News Agency in 1987. 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

[1] Transmitions obey a certain template specified by Networks News Transfer Protocol https://tools.ietf.org/html/rfc3977

[2] A flooding algorithm distributes the documents. Each user who receives a copy forwards it to each linked user except the original sender. Applied versions of this algorithm generally execute additional steps to avoid loops. One straightforward possibility would be to restrict any user to send each document only once

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.[3] 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.

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

```
From: pjsinc@phoenix.oulu.fi (Petri Salonen)
Subject: Re: What does the .bmp format mean?

Michael Panayiotakis (louray@seas.gwu.edu) wrote:
: In article <robertsa@unix2.tcd.ie> (Andrew L. Roberts) writes:
: >What exactly does the windows bitmap format look like? I mean, how is
: >the data stored: width, height, no. of colours, bitmap data? I couldn't
: >find anything in ths user manual, is there any other reference material
: >which would give me this information?

: Well, this is *only* a guess:  If it goes by the "true" meaning of "bit
: map", then it holds (x,y,c) where x pixel number in th ex-direction, y:
: pixel-number in the y-dir, c: colour.

Come on fellows! The format is quite plainly explained in the manuals.
It's in the "Programmer's Reference, Volume 3: Messages, Structures,
and Macros" (MSC-Dev.kit for 3.1, should be also in the Borland's
manuals) pages 232-241 (depending what you need).

First there is the BITMAPFILEHEADER-struct then the BITMAPINFO which
contains the BITMAPINFOHEADER and the RGBQUAD and then the bitmap
data. AND there is also a example among the example files (MS_SDK).
Hope this helps....

-------------------------------------------------------------------------------
    ########################## | Yes, I do have some prior knowledge in this.
   ##########################  | There is nothing dangerous in these dragons,
  ####   / ///   /             | they are totally harmless... But my opinion
 ####   /  /    / /// ///      | is that kicking them might not be the right
#### /// ///   / / / /// /     | way to test it. So shut up and RUN!
-------------------------------------------------------------------------------
pjsinc@sunrise.oulu.fi  pjsinc@phoenix.oulu.fi  pjsinc@tolsun.oulu.fi
If it's possible that there are some opinions above, they must be all MINE.
```

Figure 6.3.: An example article from the 20Newsgroups dataset
Source: http://strehl.com/diss/node105.html

7 Evaluation

The following chapter presents the results achieved throughout this thesis and aims to explain their evaluation. First, the intended goal of each experiment is stated, including a formulation in terms of validatable benchmarks. Subsequently, the process which extracts legible data from the framework is developed. The visualized data follows, and finally, the results are analyzed.

7.1 Reproduction

Goal and Benchmarks

As the latter sections evaluate explorative experiments, the validity of the framework which supports them is crucial. We trained the first two models described in sections [4] and [5] under the conditions J. Frankle and M. Corbin describe in their paper. The mismatch of weights previously described forms the only known difference. J. Frankle and M. Corbin primarily report the accuracy achieved by their implementations at the epoch of a simple stopping criterion. They executed each experiment five times and plot the mean alongside the minimum and maximum, which are represented through error bars. Additionally, they reperformed the same experiments ten times, but applied the masks, found after each epoch, to randomly reinitialized networks. The results were visualized in the same manner. [cite LTH] On the left side of [figure 6.1] and [figure 6.2], these measures are presented for the MNIST-FCN and the CIFAR10-CNN-6, respectively. Both plots are taken from the Lottery Ticket Hypothesis paper, but we cleaned up the latter one and brought it up to scale for improved legibility.[footnote] The goal of this reproduction is to produce results within the reported confidence intervals of accuracy in the Lottery Ticket Hypothesis paper.

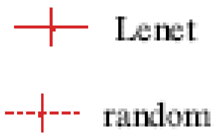
Evaluation Setup

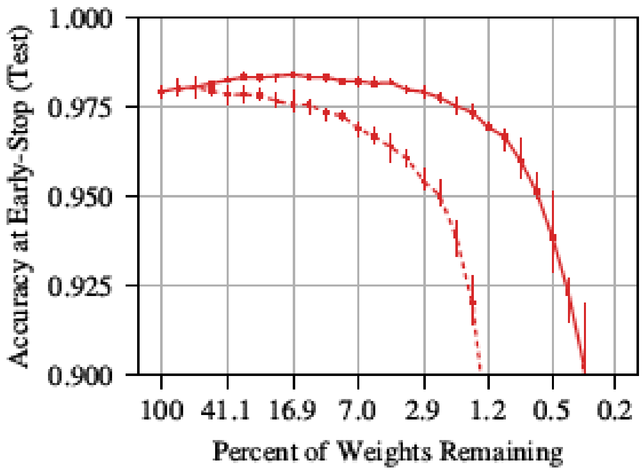
As the produced framework does not provide any early stopping criterion, we display the full range of accuracy a given architecture achieves after convergence. A line visualizes the mean, while a gray band denotes the interval between maximal and minimal achieved accuracy. As a positive side-effect, this setup increases the breadth of visualized data in the following explorative experiments because it is parameter-agnostic concerning the early stopping criterion. Finally, as J. Frankle and M. Corbin present training accuracies for the MNIST-Lenet-FCN architecture, we visualize the same measure for additional points of comparison.[1]

Evaluation Results

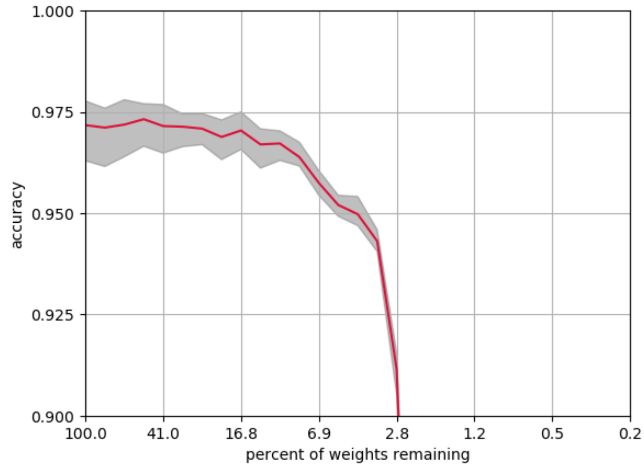
As seen in figure [6.1], the MNIST-Lenet-FCN implementation we provide achieves a lower accuracy than reported by J. Frankle and M. Corbin. Additionally, it does not show an interim improvement and degrades significantly faster at advanced pruning iterations. Said degradation is qualitatively similar to the behavior of the randomly reinitialized networks. The comparison in figure [6.2] yields similar results. While the CIFAR10-CNN-6 implementation produces the same accuracy as a full network, it degrades as quickly as J. Frankel and M. Corbin's reinitialized networks. Most of its graph falls into the deviation intervals they visualized.

[1] For the Conv6 architecture, the effective pruning rate per iteration amalgamate its dense and convolutional pruning rate. We remind that the disagreement in the number of weights in the dense layers between our framework and the Lottery Ticket Hypothesis paper also results in a different pruning rate per epoch.

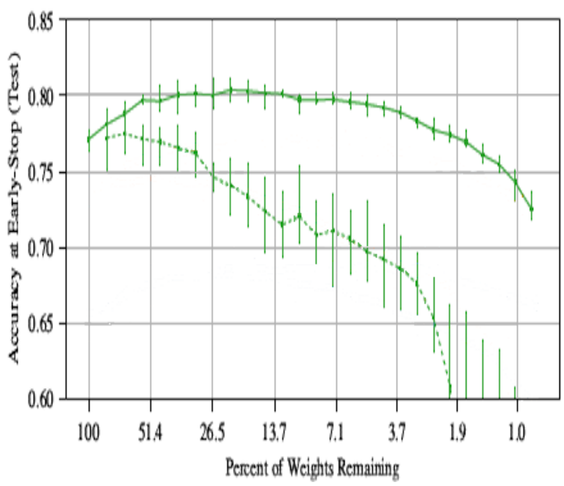




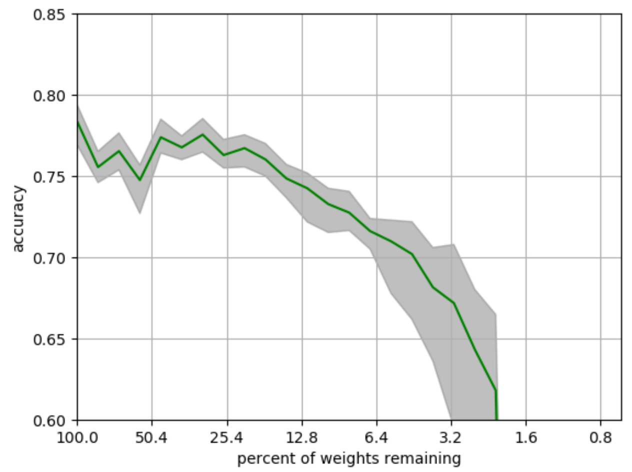
LTH-paper: Accuracy after convergence



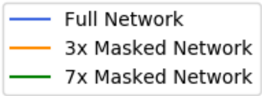
Thesis-Framework: Accuracy after convergence

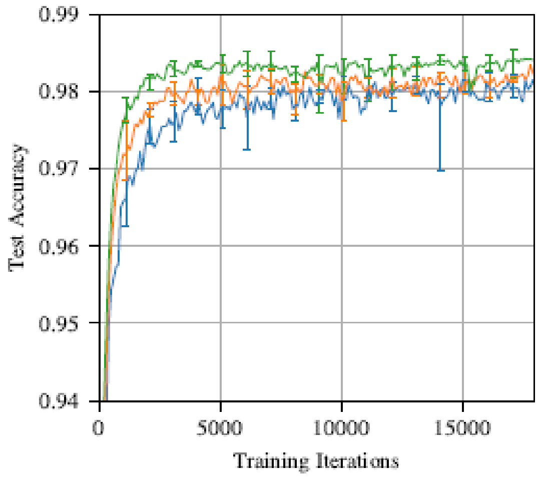


LTH-paper: Accuracy after convergence

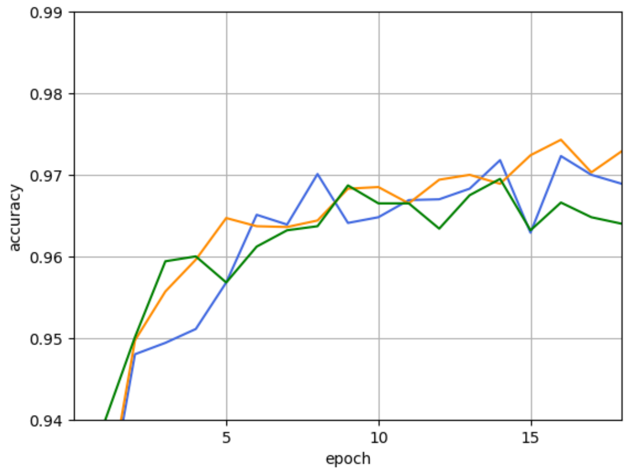


Thesis-Framework: Accuracy after convergence

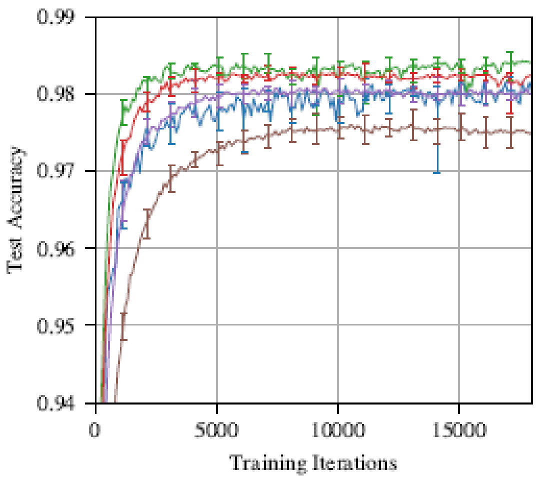




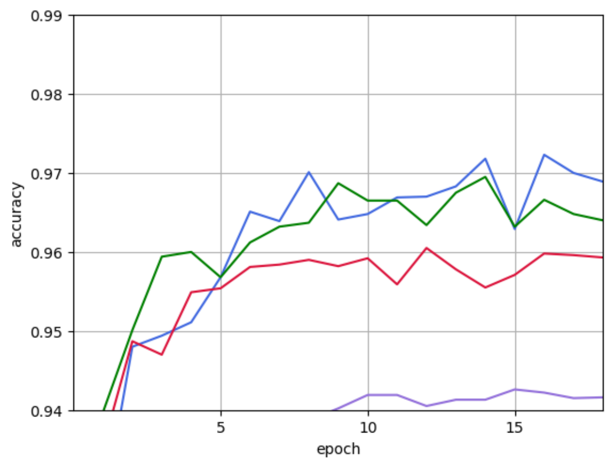
LTH-paper: pruned 0|3|7 times



Thesis-Framework: pruned 0|3|7 times



LTH-paper: pruned 0|7|12|15|18 times



Thesis-Framework: pruned 0|7|12|15|18 times

Analysis of Results

The implementations of the provided framework do not reproduce the results presented in the Lottery Ticket Hypothesis paper, meaning that the framework is not validated. This poses the question of how to proceed with the remaining experiments, which we discuss in the corresponding "Goal and Benchmarks" subsections. While the framework did not fulfill the primary goal of the experiment, two phenomena remain unclear and intriguing: The MNIST-Lenet-FCN architecture produces through our implementation of the Lottery Ticket algorithm degrades only by one percentage point under compression of up to ten times. Such a result is comparable to various pruning methods discussed in chapter [3].

7.2 Transfer

Goal and Benchmarks

While the reproduction experiments did not validate the framework, it still pruned the MNIST-Lenet-FCN to a degree comparable to contemporary work through, and it did so through the masking of networks with frozen initialization. As such, the transfer to another field still might yield another pruning tool for additional tasks and applications. If the framework prunes about 90 percent of weight without the sacrifice of prediction quality, we consider the transfer succesful.

Evaluation Setup

While figure [6.3] presents the accuracies collected in the same manner as before, fewer pruning iterations are displayed. Due to the sheer size of the 20Newsgroups-End2End architecture and the way our framework interacts with TensorFlow 2.0 to produce pruned networks severely limited the number of pruning iterations, a single experiment could afford to run without overflowing the memory of our computing devices. We managed to run experiments with ten pruning iterations, the minimal number to prune a network to a competitive degree.

Evaluation Results

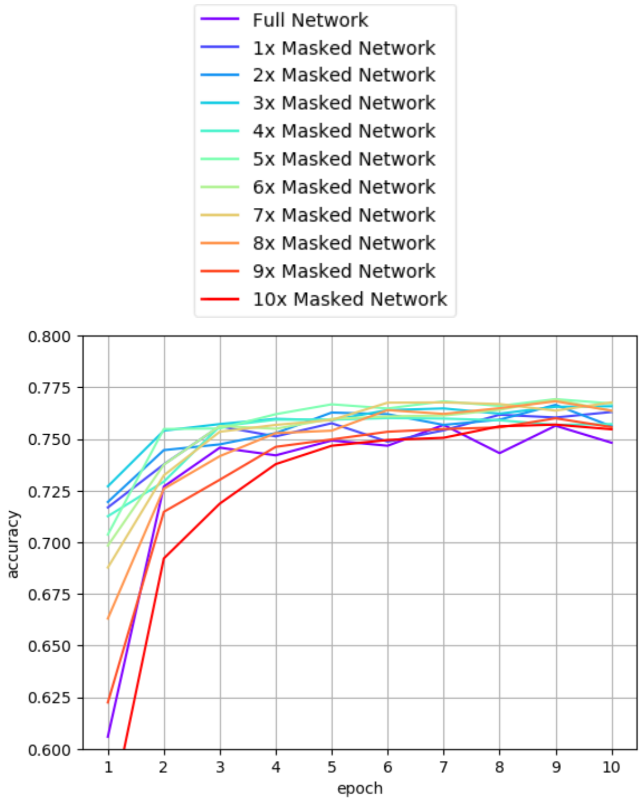
The right side of figure [6.4] shows that the 20Newsgroups-End2End architecture retains its accuracy even if about 90 percent of its weights are pruned. Additionally, the networks of the intermediate pruning epochs show an accuracy improvement of at least one percentage point.

Analysis of Results

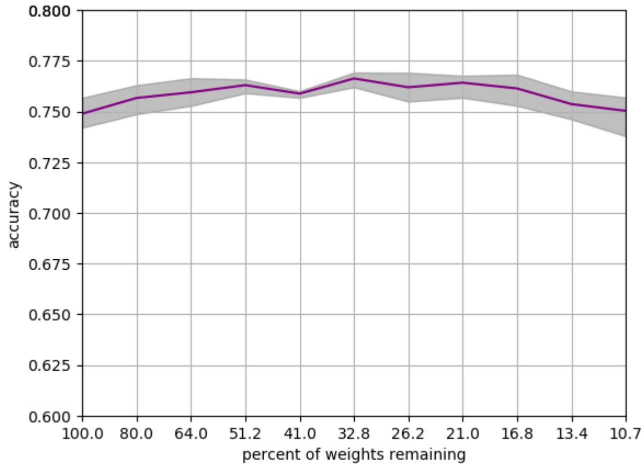
According to the previously defined goal, the transfer was successful, which shows that non-image-recognition architectures can contain efficient subnetworks upon initialization.

7.3 Early Tickets

Independent of the ability to recover actual lottery tickets, the pruning implementation supplied by our framework finds small subnetworks in the initialization which are trainable to a nontrivial accuracy. H. Zhou et al. findings confirm that the utilization of trained weights, to calculate the pruning mask, is essential.[cite] Information on the development of the quality of said weights is still of interest.



Accuracy on 20Newsgroups pruned 0-10 times



Accuracy on 20Newsgroups after convergence

Goal and Benchmarks

The aim of this experiment is purely explorative. If any patterns are recognized, they may inform experiments implemented with valid frameworks.

Evaluation Setup

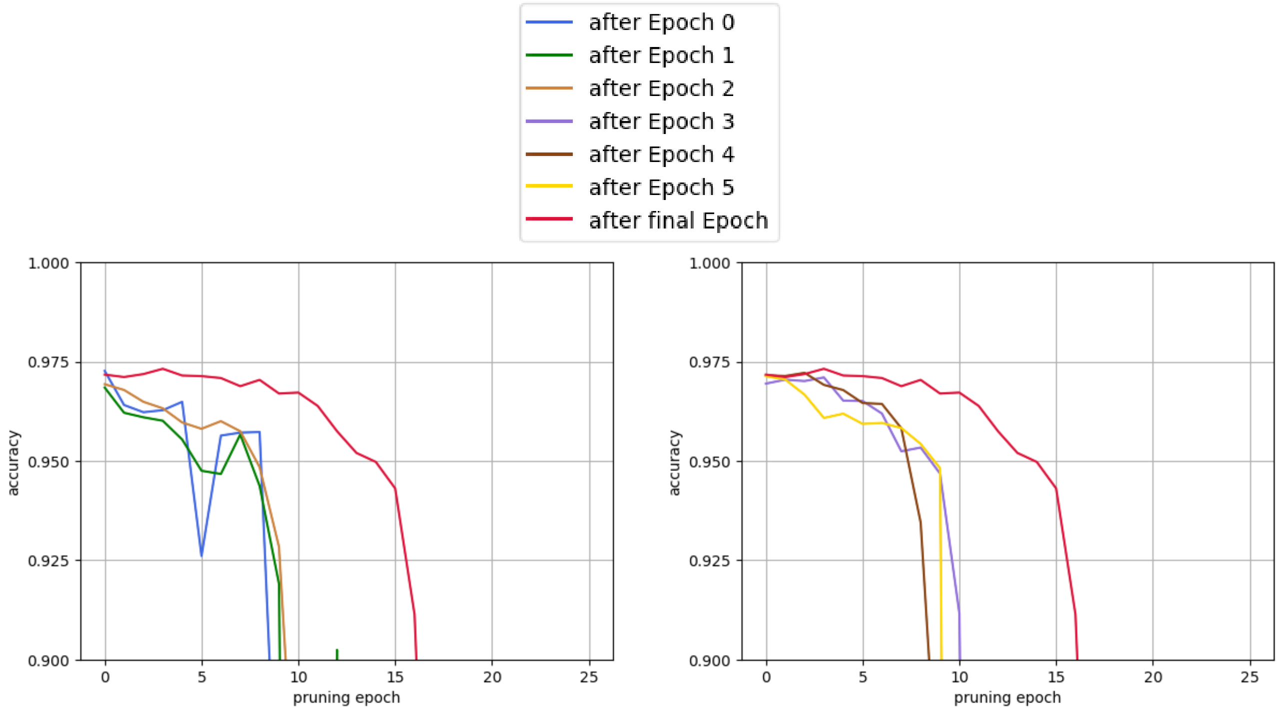
Figure [6.4] and [6.5] plot the mean accuracy achieved by implementations set to prune at the n-th epoch of training. To avoid visual clutter, the interval representing minimal and maximal values are omitted.

Evaluation Results

Because the accuracy of multiple graphs behaves erratically, it is challenging to discern a development over the training depth. The only certain observation is that all trials of earlier pruning degrade significantly faster than the original approach.

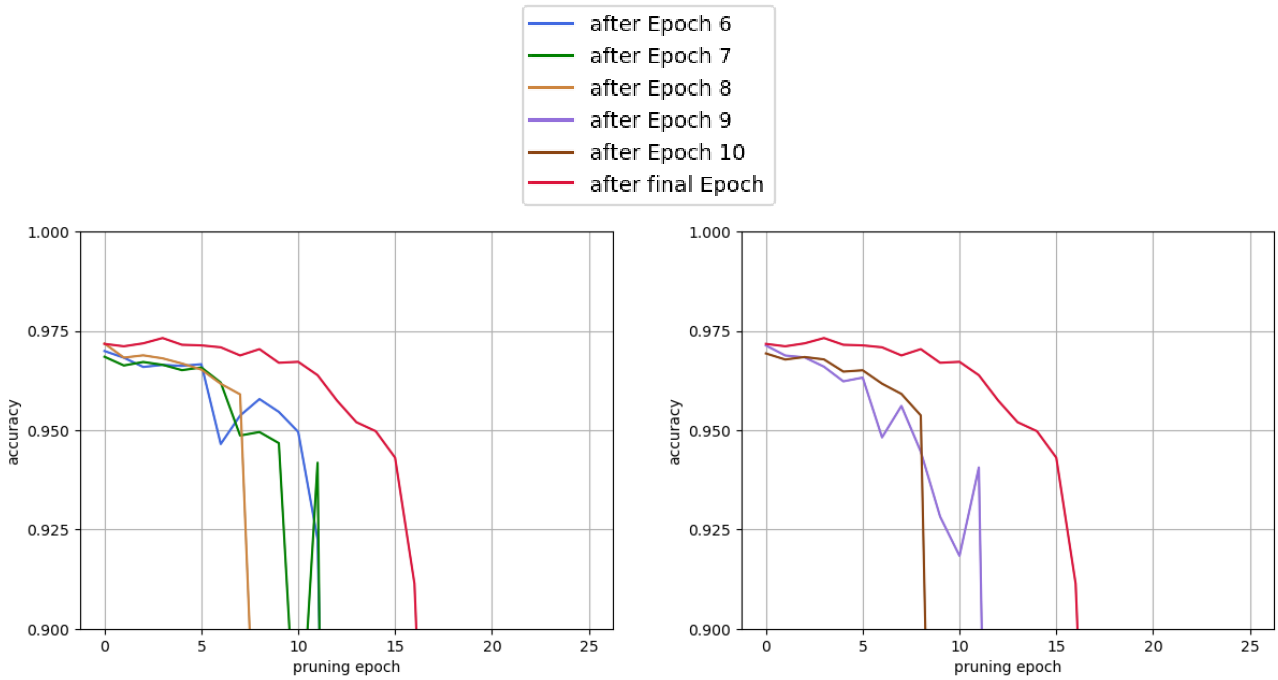
Analysis of Results

While the original network converges in about ten epochs the same amount of training does not seem to suffice to achieve the full efficiency of the pruning algorithm implemented in our framework



Ticket candidates pruned at epochs 0|1|2

Ticket candidates pruned at epochs 3|4|5



Ticket candidates pruned at epochs 6|7|8

Ticket candidates pruned at epochs 9 and 10

8 Conclusions

To conclude this thesis, the following sections summarize the accomplished work, assure results and record possibilities for future work

8.1 Summary

Inspired by the paper J. Frankle and M. Corbin published at the beginning of 2019, we decided to research two applications of the pruning algorithm they proposed. With the help of Tensorflow 2.0, the natural language toolkit nltk, scikit-learn, and a few additional backend applications, we developed a framework to search different architectures for lottery tickets. Afterward, we designed four experiments: two to validate the framework, one to extend this kind of pruning to new datasets, and the last one to explore the emergence of actionable training experience in an architecture. The reproduction was not successful, but the framework still pruned the MNIST-Lenet-FCN architecture by a factor of ten while loosing less than a percentage point. The second experiment was a definite success, and the exploration of early pruning possibilities yielded no starting points for further study.

8.2 Contributions

The primary contribution of this thesis is the openly available framework we developed. While it failed the reproduction experiments, it achieved competitive pruning results on at least two architectures. Furthermore, the availability of source code enables any future researcher to check and rerun the tests themselves. Additionally, the success of the second experiment makes a good case for the application of the presented kind of pruning algorithm in the field of natural language processing.

8.3 Future Work

The developed framework could be a convenient tool to prune various networks, but at the moment, it is heavily limited by two factors:

Firstly, some part of the workflow fills up the working memory once per pruning iteration. Experiments with massive architectures or great pruning depth may cause a memory failure resulting in the loss of all training data. Our best guess for a cause is the Tensorflow 2.0 backend. During the restoration of weight after one training procedure has finished, it might integrate the newly pruned model into its old execution graph instead of creating a new one.

Secondly, the lack of a fourth layer of abstraction overloads the main module. As described in chapter [5], the framework function at three such layers, but none of them should handle the control flow between and visualization. The result is an inefficient way to interface with the framework.

Finally, the success of the transfer experiment shows that the implemented pruning methodology can extend to non-image datasets. The next sensible point of order is a proof-of-concept that the method can also be extended to architectures more common in the natural language processing field, such as LSTMs.

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.

[BH00] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. Neural Computting in Micrbiology.

[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.

[CJG+14] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *CoRR*, abs/1404.3606, 2014.

[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.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer, 2009.

[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.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[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.

[Kri] Alex Krizhevsky. The cifar-10 dataset. https://www.cs.toronto.edu/ kriz/cifar.html.

[LBB+98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[LF02] Christopher Lueg and Danyel Fisher. *From Usenet to CoWebs*. Springer London, 2002.

[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.

[Ren] Jason Rennie. 20-newsgroups. http://qwone.com/ jason/20Newsgroups/.

[Ron14] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.

[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 Detailed Description of 20Newsgroups-End2End-CNN

Newsgroup-End2End-CNN

Model	loss	sparse categorical cross entropy
	Optimizer	Adam
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

Newsgroup-End2End-CNN

	output dimension	3
Sequential A7	input from	Input
	1D Convolution	kernel size = 19
	1D Average Pooling	pool size = 2
	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

Newsgroup-End2End-CNN

	1D Average Pooling	pool size = 7
	output dimension	3
Sequential B8	input from	Input
	1D Convolution	kernel size = 22
	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

Newsgroup-End2End-CNN

	initial weights	293.702.400
		165.648
		980
	remaining weights	~31.536.055
		~32.611
		~105