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

Fachbereich 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
Fachbereich 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 Thesis 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

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, the pruning can be interpreted as a search dependant on the weights of the trained network. This work also studies how this search develops in quality over the amount of training provided beforehand.

In pursuit of these two goals, we developed a framework through in we implemented four experiments. The first two experiments aimed to confirm the validity of the codebase. Subsequently, the third one intended to establish an argument for application of their 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 failed to reproduce the results of J. Frankle and M. Carbin, one of them pruned its associated network up to the scale of contemporary technique. It achieved a size reduction of 90 percent with little loss in accuracy. Additionally, the third result conforms with their definition of a lottery ticket, producing 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, none of the early pruning trials achieve the same level of accuracy displayed by our previous experiment, even if the latter trials pruned as late as epoch 10, an amount of training for which the full network already converged. These results suggest that, during the pruning iterations, the pruning algorithm has to submit a given network to more training than necessary for primary convergence.

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. They can theoretically approximate most mappings between inputs and expected output[1] up to arbitrary precision[2][HSW89], and in practice, they achieve results unobtainable with previous technologies.
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 estimated to no longer be necessary after training. At the moment there is no consensus 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

J. Frankle and M. Carbin's algorithm might become a powerful pruning tool for neural network applications, but additional research needs to be done, as they also noted in the latter part of their paper.
First and Foremost, J. Frankle and M. Carbin applied their method only to small datasets due to the computational expense of iterative pruning.[FC18] Additionally, 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 acknowledged that pruning single connections instead of whole sections of a network does not line up well with modern frameworks, and thus achieves no actual speed-up. Finally, they stated that they plan to research ways to find lottery tickets earlier or at smaller sizes.[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.

1.3 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 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 concludes this work with a summary and a mention of possibilities for future work.

[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.

2 Background

A joint base of concepts is necessary for the discussion of our work. 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 approach to **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 the availability of 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. 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.

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 2.1 depicts 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 figure 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.

[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.

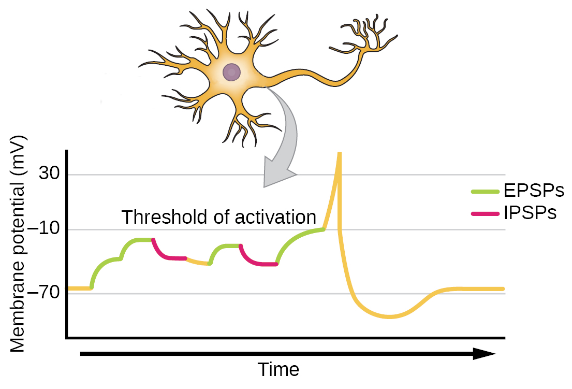


Figure 2.1.: Representation of a biological Neuron

Source:
"Biology 2e" [CDC18]

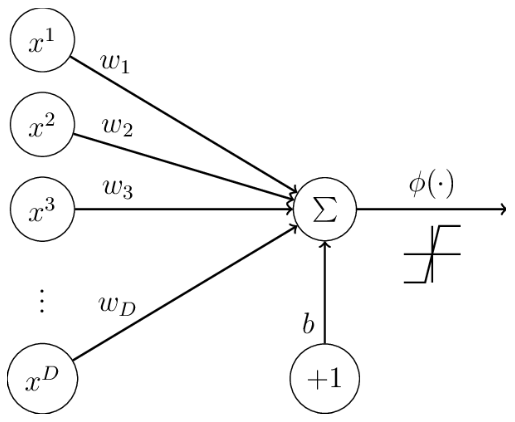


Figure 2.2.: Abstraction of a Neuron

Source:
"Design Space Approach in Optimization of Fluid Bed Granulation and Tablets Compression Process" [DMK+12]

- **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 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:

- **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 neighborhood into a single output.[3] This process needs a specification of the data's dimensionality 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 neighboorhods of some features are skipped.

- **Pooling**
  A pooling layer contains no trainable weights and functions more like a data-processing-step between other layers than anything else. It operates much like a convolutional layer with fixed weights and striding, replacing a whole neighborhood by a single output, thus reducing the size of the data.

- **Flatten**
  Similar to the pooling, a flatten layer specifies a data-processing-step inside a network. It collapses all values of a data point into a single dimension and is the canonical method to prepare multidimensional data for a dense layer. Usually, no parameters are necessary to define a flatten layer.

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. However, weights having values 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 Tensorflow-API utilized in the backend of our framework.
The following two examples illustrate the notation:

1

Simple-FCN | 2.3

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

[3] In practice, many convolutional layers employ multiple kernels at the same time, meaning that one neighborhood produces numerous outputs. This procedure extends the dimension of the processed data by an additional axis, often called **channels**. These channels frequently receive special treatment.

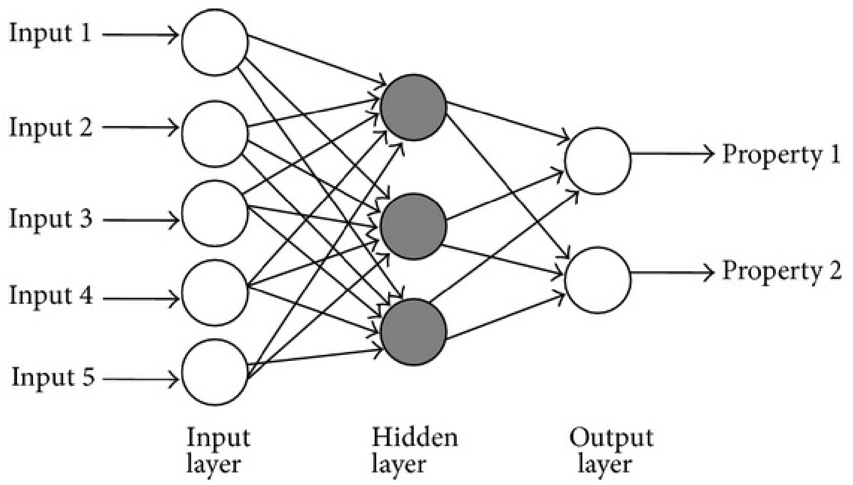


Figure 2.3.: Architecture of a small fully-connected network

Source:
"Neural Networks Regularization Through Representation Learning" [Bel18]

2

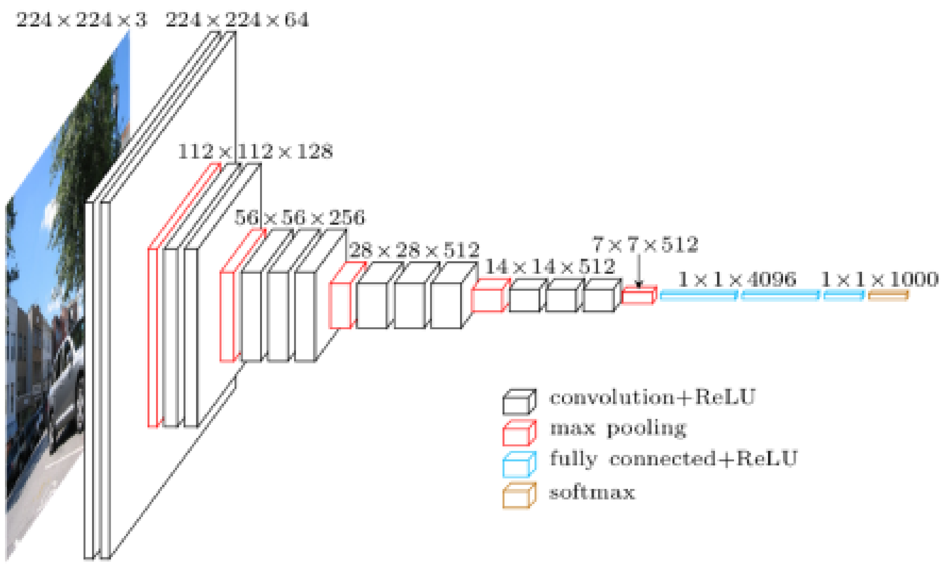


Figure 2.4.: Macroarchitecture of VGG16

Source:
https://www.cs.toronto.edu/ frossard/post/vgg16/

VGG-16 | 2.4

Defaults	Convolution: kernel size	[3,3]
	Convolution: stride	[1,1]
	Convolution: padding	same dimension

VGG-16 | 2.4

		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]
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	1000
	activation	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 that generalization suffers when a network adapt too much to the given data. This process, called **overfitting**, 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 refers 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 items 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, such an embedding describes the data much more efficiently. However, there is no inherent guarantee that this representation retains the information previously present. One-hot-encoding 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, 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

In addition to 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

The context of current research is needed to assess the utility of any results achieved during this thesis. As such, this section shortly presents the state-of-the-art of the relevant tasks and compares them to the approaches that are covered in the following chapters.[1] Additionally, it gives an overview of previous pruning methods and their achievements.

3.1 State of the art: Image Classification

MNIST and CIFAR-10 are the two image datasets covered in this thesis, containing small images displaying grey-scale images of digits and colored images of real-world objects, respectively. Chapter 6 describes them in more detail. The primary task on both of them demands a classification of their images into the ten corresponding classes. State of the art approaches delivers human or even superhuman accuracy on both data sets.
In their paper "RMDL: Random Multimodel Deep Learning" Kowsari et al. combined dense neural networks, convolutional neural networks, and recurrent neural networks into an ensemble, that achieves an accuracy of 99.82 percent for MNIST image classification as well as good benchmarks on other tasks, including CIFAR10 image classification and natural language processing tasks.[KHB$^+$18]
The researchers of the next few approaches report accuracies of 99.8 percent, which equates to only two fewer correctly classified images.
Back in 2012, D. Ciresan et al. described a deep architecture that alternates between convolution and softmax layers before it closes with a short dense classification structure, similar to the VGG16 architecture described in chapter 2. Their paper "Multi-column Deep Neural Networks for Image Classification" improved on the state-of-the-art of image classification over multiple datasets, including MNIST. They also recorded that their networks are structurally similar to the neural connection between the retina and the visual cortex of mammals such as macaque monkeys.[CMS12]
Three years later, Later Sato et al. polished approaches that use self-generated augmented data to increase the transformation invariance of their architecture in their paper "APAC: Augmented PAttern Classification with Neural Networks". They achieved the same results on MNIST image classification.[SNY15]
In the same year, C. Jia-Ren and C. Yong-Sheng published "Batch-normalized Maxout Network in Network", a paper in which they explain how they adopted the newly developed "Network in Network" structure to avoid vanishing gradients.[CC15]
They also share second place alongside S. Hossein Hasanpour et alia.[HRFS16]

[1] The rankings presented in this chapter rely primarily on the website: "Papers with Code". https://paperswithcode.com/sota

In contrast to MNIST, the three best-performing approaches for CIFAR-10 image classification were all published in the last year. Currently, an automated reinforcement learning approach claims the highest performance of 98,3 percent.[2] In their paper "Fast AutoAugment", S. Lim et al. improved the result of multiple existing architectures through automatic data enhancement. They remark that such automation was previously posed as an important open question because the efficiency of any single data enhancement strategy is highly dependant on the dataset at hand.[LKK+19]
The second most accurate approach also utilizes data augmentation in addition to other techniques for the reduction of data dependency. Under the title "EnAET: Self-Trained Ensemble AutoEncoding Transformations for Semi-Supervised Learning", X. Wang et al. produced an approach which is very adept at handling tasks with little available data and very competitive when supplied by the same data as contemporary architectures. It achieves 98,01 percent accuracy.[WKLQ19]
With their publication "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware", Cai et al. supplied a competitive technique much more closely related to the content of this thesis than any of the previously mentioned. They addressed issues of computational expense in the previous network architecture search algorithm allowing them to increase the search space drastically. As a result, they find competitive networks capable of accuracies as high as 97.02 and smaller than those presented in previous state-of-the-art publications.[CZH18]

To complete the picture of performance measures the following table summarizes the state-of-the-art accuracies, as well the ones J. Frankle and M. Carbin, report for the two networks studied in this paper. While they do not provide exact values in the Lottery Ticket Hypothesis paper, their figures indicate that their Lenet-FCN and Conv-6 architectures achieve roughly 98 percent accuracy on MNIST and 80 percent on CIFAR-10, respectively. [FC18] Additionally, it displays the best available estimates for human accuracy on both datasets. The authors of MNIST supplied an estimate in one of their papers, and T. Ho-Phuoc conducted a study on human proficiency for CIFAR10 image recognition.[LJB+95][Ho-18]
The architectures studied by J. Frankle and M. Carbin fall behind any state-of-the-art approaches by a significant margin, but they noted that they primarily studied smaller architectures due to the computational expense of their approach.[FC18] We support the validity of their experiments, as their results and conjectures are based solely on the relative performance of the networks they pruned compared to their complete counterpart.

Contemporary performance for image classification.

MNIST	RMDL	99.82	2018	[KHB+18]
	Human	~99.8	–	[LJB+95]
	Multi-Column	99.8	2012	[CMS12]
	APAC	99.8	2015	[SNY15]
	Maxout Network	99.8	2015	[CC15]
	LTH Lenet FCN	~98	2019	[FC18]
CIFAR10	AutoAugment	98.3	2019	[LKK+19]
	EnAET	98.0	2019	[WKLQ19]
	Direct NAS	97.88	2019	[CZH18]
	Human	~94	–	[Ho-18]
	Maxout Network	93.3	2015	[CC15]
	RMDL	91.2	2018	[KHB+18]
	Multi-Column	88.8	2018	[CMS12]
	LTH Conv-6	~80	2019	[FC18]

3.2 State of the art: Topic Classification

Document topic classification is arguably the task most similar to image classification in the field of natural language processing. While Reuters- 21578 is arguably the most iconic dataset for this task, the architectures commonly applied to it are not related to the ones studied by J. Frankle and M. Carbin, nor is Reuters-21578 structurally akin to MNIST. Section 6 covers

[2] Some researchers achieved better benchmarks but used additional data. Because most researchers, including J.Frankle and M. Carbin, did not use additional data on CIFAR10, we omitted those results.

a few differentiating properties. 20-Newsgroups is another natural language data set utilized for document classification which soime contemporary researches approached with architectures primarily composed of dense and convolutional layers instead of recurrent networks more common in the field of natural language processing.
In their work Pappagari et al. develop an approach that integrates multiple self-learned word-level language embeddings into a simple ensemble.[PVD18] While newer architectures form the current state-of-the-art, their work achieved the best performance of its time on 20-Newsgroups. The following table summarizes the available measures for document topic classification on 20Newsgroups.

Contemporary performance for document topic classification.

20Newsgroups	Neural Bag of Entities	88.1	2019	[YS19]
	RMDL	87.91	2018	[KHB$^+$18]
	Graph Star	86.9	2018	[HHYX19]
	End to End CNN	86.12	2018	[PVD18]
	RMDL	87.91	2018	[KHB$^+$18]

3.3 Pruning

Beginning around 1990 with M. C. Mozer and P. Smolensky as well as LeCun et al., researchers removed weights from neural networks after training them for a task.[MS89][LDS90]
Shortly after, B. Hassibi and D. G. Stork proposed further training of a pruned network [HS93], which became standard practice over the next decade. While LeCun et al. reported that they compressed a network by the factor of four, more recent works achieve a factor of nine to about seventeen while losing little or no accuracy.[HPTD15][LWL17]
J. Frankle and M. Carbin reported pruning over 98,5% of weights in one of their networks while maintaining network capabilities, which amounts to a compression of over 50 times.[FC18]

In their recent paper "Rethinking the value of network pruning", Z. Liu et al. observed that pruned networks display the same capabilities, whether they retrained them with randomly reinitialized weights or retain and fine-tune the previous weights. They concluded that said weights can not be essential to a pruned network's quality, contrary to prior common belief. Thus Z. Liu et al. claimed that the architecture of pruned networks is responsible for its capabilities and that pruning can be interpreted as a kind of network architecture search.[LSZ$^+$18]
Based on the conclusion that trained weights only inform a search on the original architecture, it seems natural to question the amount of training necessary for them to establish their information content.
In a paper of Y. Li et el. from early 2019, they described a method named "Incremental pruning based on less training" for pruning of common convolutional network architectures at the filter level and especially before convergence. They compressed the networks by a factor of ten but also sped up training by a similar margin.[LZS19]

3.4 Additions to the Lottery Ticket Hypothesis

Even though J. Frankle and M. Carbin only proposed the Lottery-Ticket-Hypothesis early in 2019, additional papers on the topic exist. In a follow-up-paper from June 2019, and with the help of a few additional authors, they expanded their method to find winning tickets on deep convolutional network architectures that proved difficult before. They attributed this achievement to the decision of not returning to the very first state of the network but to one a few iterations into training.[FDRC19]
H. Zhou et al. recently performed an ablation study on the phenomenon of lottery tickets, which they documented in their paper "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask". They reaffirmed magnitude-based pruning and described super masks that improve accuracy when applied to the initial network even without additional training. Subsequently, they concluded that the masking of weights acts as training, moving weights in a direction that actual training would have taken them. Additionally, H. Zhou et al. found that a replacement of all weights in the pruned network by a constant with the same sign does not significantly influence the network's capabilities. They concluded that the sign of weights is the essential property for such neural networks.[ZLLY19]

4 Design

The following section describes our work from a high-level point of view. First, we define the components that describe an experiment. The subsequent sections specify these components for each experiment, respectively.

4.1 Components

With the following component structure, we intend to concisely communicate all parameters necessary to understand and reproduce the presented experiments. For each single experiment, we record the following components:

- **Aim of the Experiment**
  The performed experiments pursued different goals. This component records the motivation of the given experiment and thus prepares the evaluation found in chapter 7.

- **Dataset and Preprocessing**
  We deployed three different datasets. While chapter 6 provides a more detailed description, this component concisely specifies the type and shape of data points the experiment handles. Additionally, we note any preprocessing applied to the data before our experiment.

- **Task**
  A dataset embodies the experience of a given network, but the task researchers expect their architecture to solve may still differ. A collection of text, for example, could either be classified by one network or compressed by another. The task primarily formulates expectancies on the structure of a network's output, but it also has a significant influence on its design. Different tasks might also vary in difficulty.

- **Architecture**
  An accurate description of a network's architecture is vital to the reproducibility of any experiment. This component contains all parameters necessary to implement the network structure at hand in our framework and presents them in the shape of chapter 2. Any parameters that we inferred because we could not discern them from the referenced papers are mentioned here. J. Frankle and M. Carbin recorded the number of weights present in their architectures, which we utilized to check our implementation. In one case, where our number of weights partially disagree, we suggest possible explanations.

- **Training**
  During training, a network repeatedly classifies small collections of data-points, called **batches**. After each batch, the framework calculates the momentary classification error and updates the weights, thus concluding a **training iteration**. This kind of training forms a numerical optimizations algorithm and, as such, might not converge after utilizing each data point just once. A training pass over all data points is called an **epoch**, and networks usually absolve it more than once. Our framework defines training in terms of epochs in contrast to J. Frankle and M. Carbin, who visualized their results in terms of iterations. For this reason, this component supplies the conversion rate between the two alongside remaining training parameters.

- **Pruning**
  In their paper, J. Frankle and M. Carbin used different pruning percentages for different kinds of layers. Additionally, their results show that the quality of different architectures degrades at different speeds.[FC18] For this reason, we employ different numbers of pruning iterations for the experiments and record them in this component alongside any remaining information on the training the architectures absolve.

4.2 Reproduction: Dense Network | MNIST-Lenet-FCN

Aim of the experiment

The fully-connected Lenet network embodies the most basic architecture in the Lottery Ticket Hypothesis paper. Its reconstruction serves as a minimal working prototype for the codebase.

Dataset and Preprocessing

For this experiment, we utilized the same data as J. Frankle and M. Carbin; the image set MNIST. It contains gray-scale images of hand-written digits with a size of 28x28 pixels. We applied no preprocessing and utilized no data augmentation on the MNIST dataset.

Task and Architecture

The task J. Frankle and M. Carbin. chose for their research is topic classification. The network is expected to return a ten-dimensional one-hot-encoding recording the digit displayed on the image in question.

MNIST-Lenet-FCN Architecture

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 and Pruning

MNIST-Lenet-FCN Schedule

Training	epochs	50
	iterations per epoch	1000
	batch size	60
	loss	categorical crossentropy
	optimizer	Adam
	learning rate	$1.2 \cdot 10^{-4}$
Pruning	layers	Dense
	amount	20%
	iterations	25
	initial weights	266.610
	remaining weights	$\sim$1007

4.3 Reproduction: Convolutional Network | CIFAR10-Conv6

The Conv-6 architecture J. Frankle and M.Carbin developed for their paper, utilizes an additional popular kind of trainable layer, the convolutional layer, and has an order of magnitude more weights than the fully-connected Lenet architecture. Furthermore, it operates on an arguably more difficult dataset, CIFAR10.

Aim of the Experiment

The reproduction of the result they present for Conv-6 is a strong argument for the validity of our network concerning convolutional layers and more challenging datasets.

Dataset and Preprocessing

For this task, we utilized the image dataset CIFAR10, as did J. Frankle and M. Carbin. In contrast to MNIST, CIFAR10 contains colored images with a size of 32x32 pixels. Three gray-scale images encode an image's share of red, blue, and green, respectively. The result is the final size of 3x32x32 pixels. The CIFAR10 dataset was neither preprocessed nor enhanced.

Task and Architecture

As for the previous experiment, the task is image classification. While the number of classes remains the same as for MNIST, the complexity of possible objects arguably increases, as they are composed of real-world objects, such as horses and cars.

J. Frankle and M. Carbin developed Conv-6 based on the VGG architectures and only note the parameters necessary to infer the remaining parts of the infrastructure.[FC18] We based our implementation on those parameters and the referenced paper of K. Simonyan and A. Zisserman.[SZ14]
After Inference, the number of weights in our 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 our framework employs the canonical way to prepare a multidimensional input for a dense layer, a flattening layer, we assume that J. Frankle and M. Carbin either reported the wrong layer parameters or the wrong number of weights in their description.

CIFAR10-Conv6 Architecture

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]
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 and Pruning

CIFAR10-Conv6 Schedule

Training	epochs	36
	iterations per epoch	~833
	batch size	60
	loss	categorical cross entropy
	optimizer	Adam
	learning rate	$3 \cdot 10^{-4}$
Pruning	layers	Dense

CIFAR10-Conv6 Schedule

	2D Convolution
iterations	25
rate per iteration	20 %
	15 %
weighted rate	17,46 %
initial weights	1.117.194
	1.145.408
remaining weights	~4220
	~19698

4.4 Transfer: Newsgroups-End2End

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.[PVD18]
The architecture already performed quite well without the specific cost function they developed. To avoid problems with unique features and to increase the applicability of our findings, we omitted the special cost function. While the accuracy of our implementation dropped about ten percentage points, it still comes closer to the respective state-of-the-art than the Conv-6 architecture previously discussed.

Dataset and Preprocessing

The natural language dataset used for this experiment is called 20Newsgroups. 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 20Newsgroups website.[1] [PVD18] Otherwise, they only mention that they used the canonical split of training and test data, which is not sufficient 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. Furthermore, the provided vocabulary does not contain all tokens. For this experiment, we implicitly interpreted all tokens not referenced in the vocabulary as stopwords and omitted them.
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. Simple zero-padding would overexert computer memory and overparametrize the architecture. Figures 4.1 and 4.2 show the document length histograms of test and training split for 20Newsgroups, respectively. Our computation devices were able to support a few hundred tokens per document, dependant on the number of pruning iterations. Ten pruning iterations are necessary to achieve a competitive pruning rate and left us with about 200 hundred tokens per document. The red line visualizes the cut-off. It leaves about 70 percent of the documents untruncated.

Task and Architecture

Each document in the 20Newsgroups dataset was collected in a Usenet repository with an associated topic. For each document, the architecture returns its topic in a 20-dimensional one-hot-encoding.

The Newsgroups-End2End ensemble combines 16 very similar embedding architectures. First, each network employs a trainable word-embedding layer to reduce the dimensionality of the one-hot-encoded word tokens. Subsequently, a convolutional layer and a few processing layers embed the whole document into a three-dimensional space. The overarching ensemble then collects the three remaining features of each architecture and classifies them with a dropout-regularized dense layer. Figure 4.3 and 4.4 display the single architectures and the ensemble, respectively.

[1] The vocabulary length they documented does not agree with the length of the vocabulary provide at the website, which might be due to an update from the website's author.

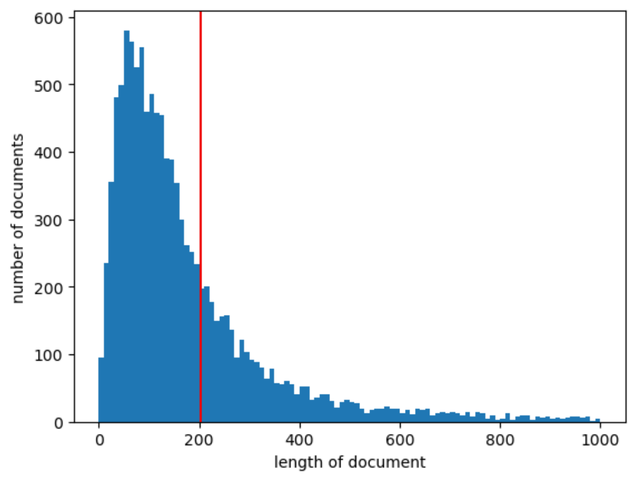


Figure 4.1.: Histogram of 20Newsgroups training data with cut-off

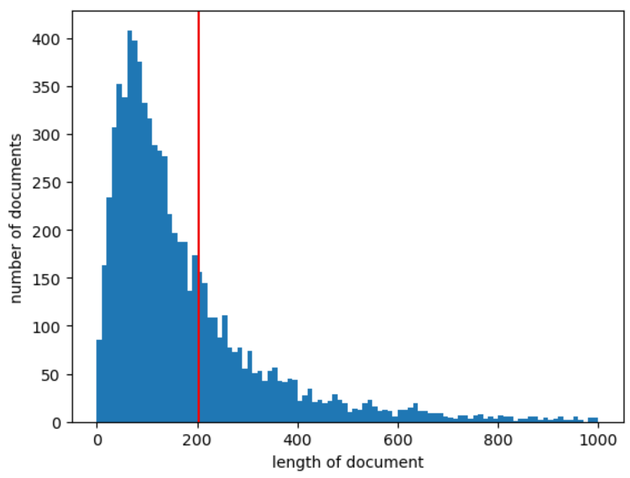


Figure 4.2.: Histogram of 20Newsgroups test data with cut-off

Source:
These figure were produced by the author.

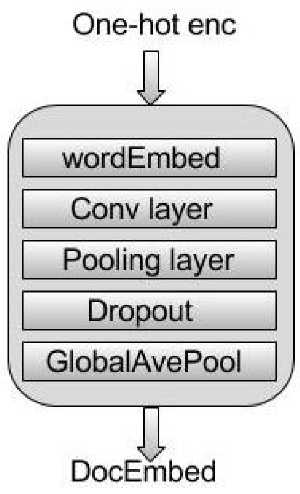


Figure 4.3.: Layers of one embedding network

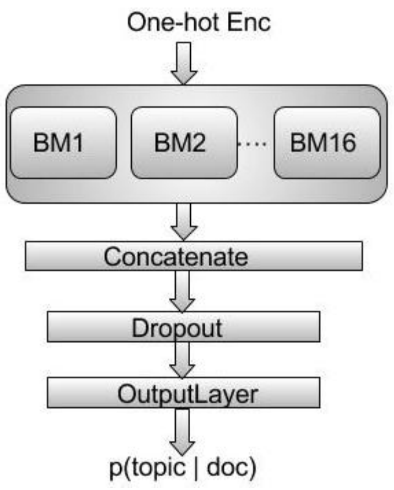


Figure 4.4.: Architecture of the whole network ensemble

Source:
"Joint Verification-Identification in end-to-end Multi-Scale CNN Framework for Topic Identification" [PVD18]

Training and Pruning

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

20Newsgroups-End2End Schedule

Training	epochs	10
	iterations per epoch	~188
	batch size	60
	loss	categorical cross entropy
	optimizer	Adam
Pruning	layers	Embedding
		1D Convolution
		Dense
	iterations	10
	rate per iteration	20%
		15%
		20%
	weighted rate	~20 %
	initial weights	293.702.400
		165.648
		980
	remaining weights	~31.536.055
		~32.611
		~105

4.5 Early Ticket: MNIST-Lenet-FCN

As this experiment shares an architecture with the reproduction discussed earlier, we omit the redundant subsections about the dataset, preprocessing, task, and architecture.

Aim of the Experiment

In the introduction of this thesis, we remarked that the search for a lottery ticket might be possible after less training. Essentially, J. Frankle and M. Carbin perform network architecture search on the initialized network. Their algorithm uses the trained weights solely to inform this search. In principle, searching for a performant architecture could be done without any training. This experiment aims to study the behavior of lottery tickets dependent on the amount of training provided before the network is pruned and reinitialized.

Pruning

The original network converges no later than 15 epochs into training. Thus we performed 15 experiments, each set to prune at a different epoch. The remaining pruning parameters remained as noted in the reproduction section.
All 15 networks share the same initialization to ensure the comparability of the result data. To collect data for the evaluation, all networks received the full training 50 epochs of the original experiment.

5 Implementation

Up to this point, the presented information is on a conceptual level. In contrast, this chapter examines the actual implementation in our framework.[1]

5.1 Representation of the components

Chapter 4 defines an experiment by its architecture, training parameters, and pruning setup. Figure 5.1 clarifies where those components can be found in the framework. Additionally, figure 5.2 describes which datasets are available. If we preprocessed a dataset, it notes the location of the responsible code.
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 search of a topic for 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 implemented, models are trained, 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 particular 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. It also distributes training to devices other than regulars CPU cores.[AAB+15] Of particular interest to this work is the speed-up achieved through the usage of GPUs. Because Tensorflow did not correctly handle the utilized multiclass data during the calculation of network accuracy, we also adopted, **scikit-learn**, an additional machine learning package, for the on-line evaluation. Scikit-learn is an open-source python framework built on *NumPy*, *SciPy*, and *matplotlib* that supplies a plethora of data analysis tools.[PVG+11]

5.3.2 Datasets and Preprocessing

The framework retrieves the datasets MNIST and CIFAR10 from a Tensorflow interface. The corresponding modules read out these interfaces, bring the data into the expected shape, and return the canonical split of training and test data points. For the 20Newsgroups dataset, the framework utilizes a scikit-interface.
We employ the Word Tokenizer of **NLTK**, an open-source natural language framework that also contains the Reuters-21578 dataset.[BK09]

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 meant **for internal use only**. On their website, the packages authors explicitly warn of untrusted data that was saved with pickle.

[1] We make our source code openly available in a public GitHub repository.
https://github.com/TimUnverzagt/Thesis

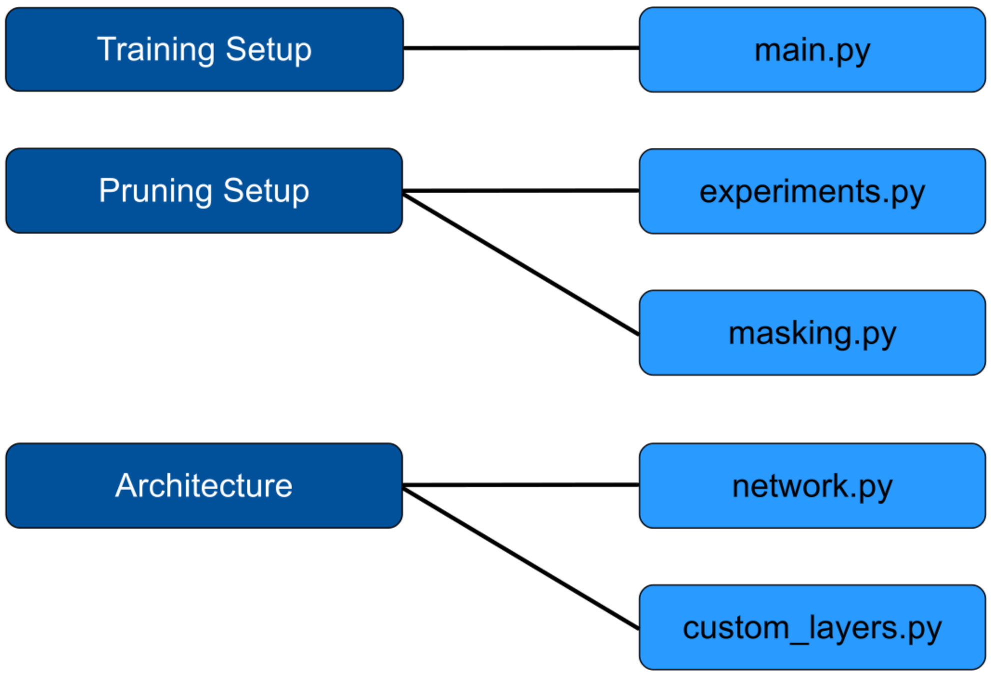


Figure 5.1.: Representation of the main components in the framework

Source:
This figure was produced by the author.

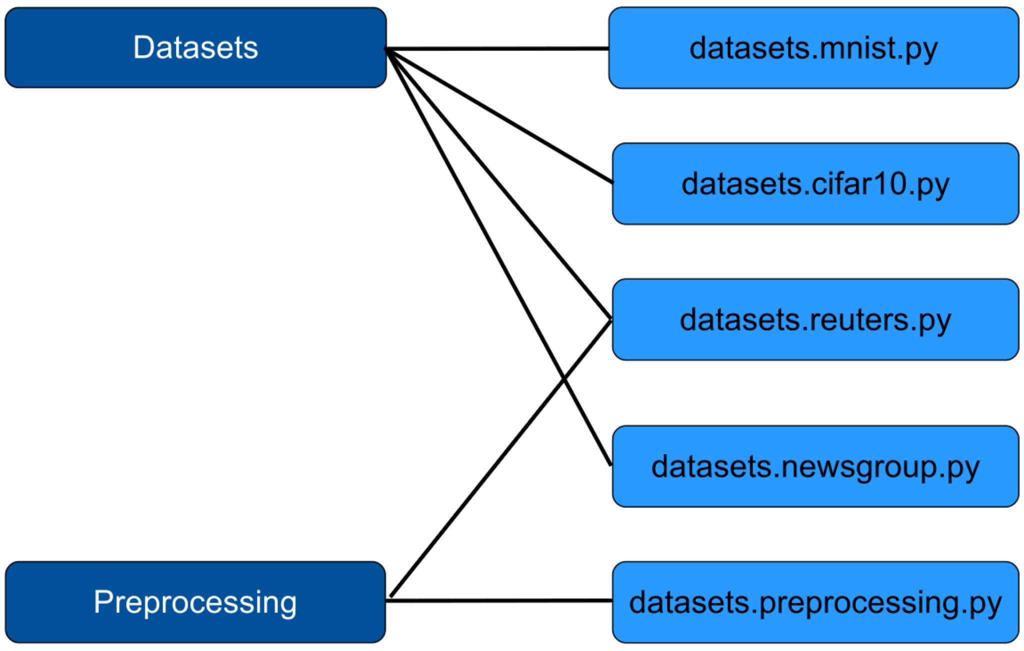


Figure 5.2.: Representation of the datasets and their preprocessing in the framework

Source:
This figure was produced by the author.

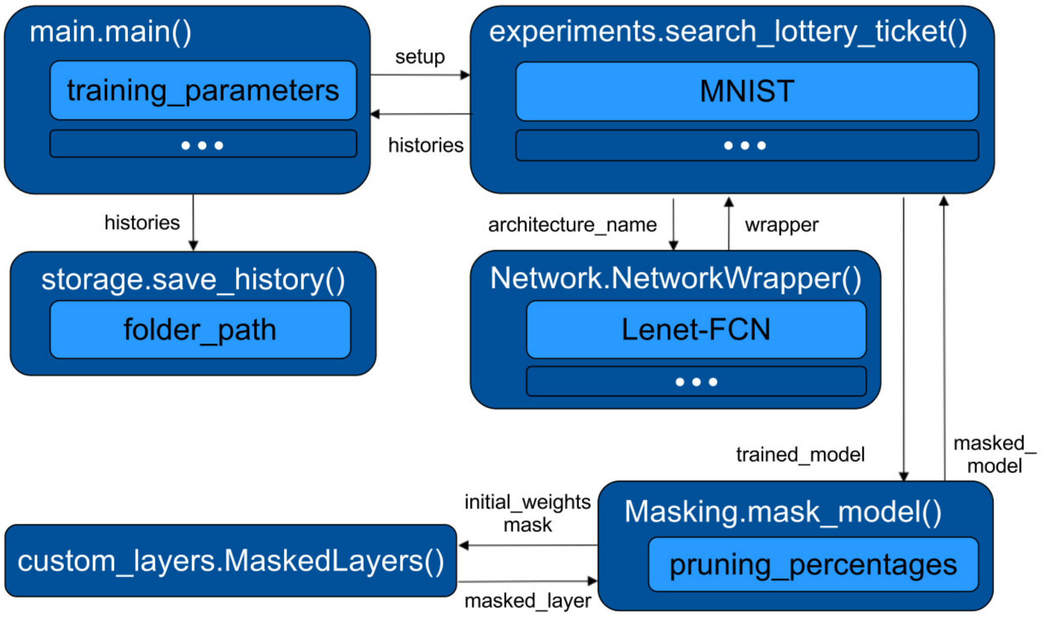


Figure 5.3.: Scheme of the framework during an example experiment

Source:
This figure was produced by the author.

5.4 Limitations

While our codebase is generally capable of unraveling and reproducing any architecture described in the functional Tensorflow API, at the moment it only supports the masking of the layers described in chapter 4. Additionally, Tensorflow itself does not support the pruning of single neural connections because it bundles them into tensors to model layers. As it can only flag whole tensors as non-trainable, a different solution is necessary. Our framework utilizes the same workaround recorded by J. Frankle and M. Carbin and represents pruned weights through masks on the layers tensors.

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 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, who 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, including the Lenet-FCN. 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 et alia.[CJG+14]

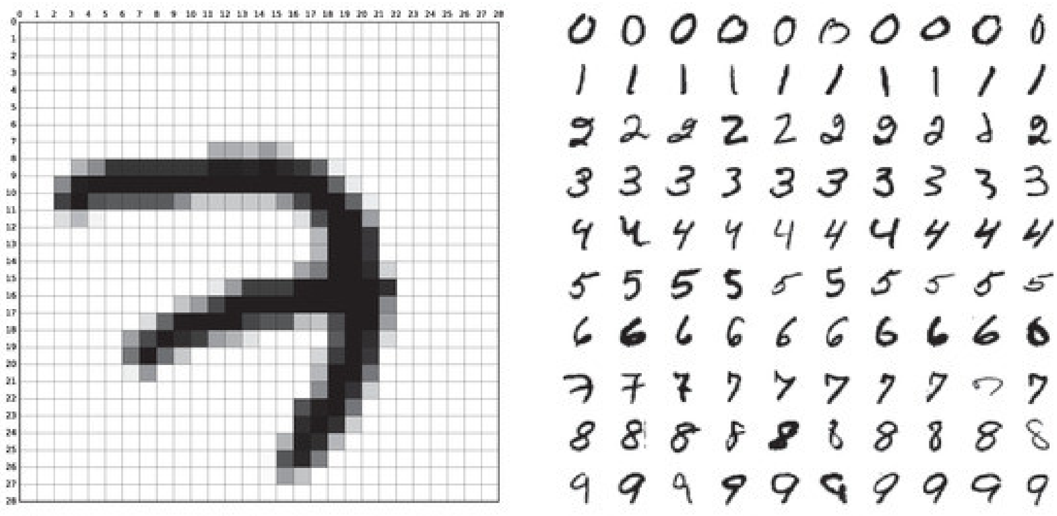


Figure 6.1.: Example images from the MNIST dataset

Source:
https://www.mdpi.com/applsci/applsci-09-03169/article_deploy/html/images/applsci-09-03169-g001-550.jpg

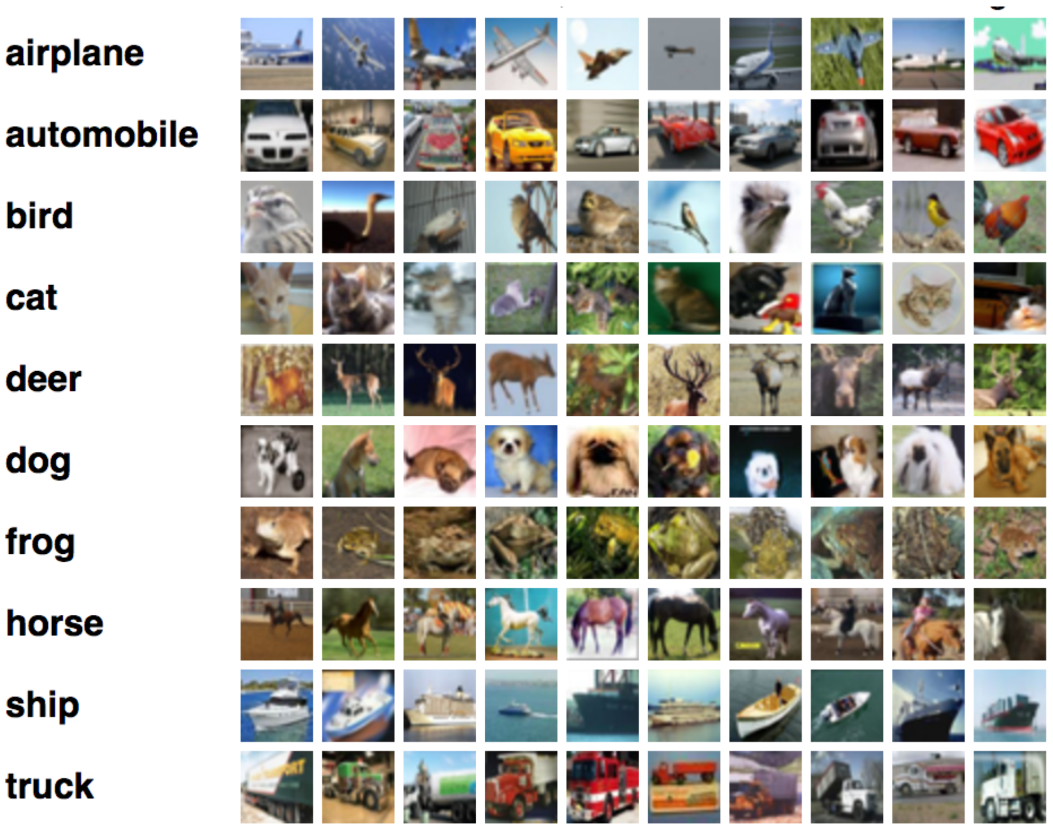


Figure 6.2.: Example images from the CIFAR10 dataset

Source:
https://www.cs.toronto.edu/∼kriz/cifar.html

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 20Newsgroups 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

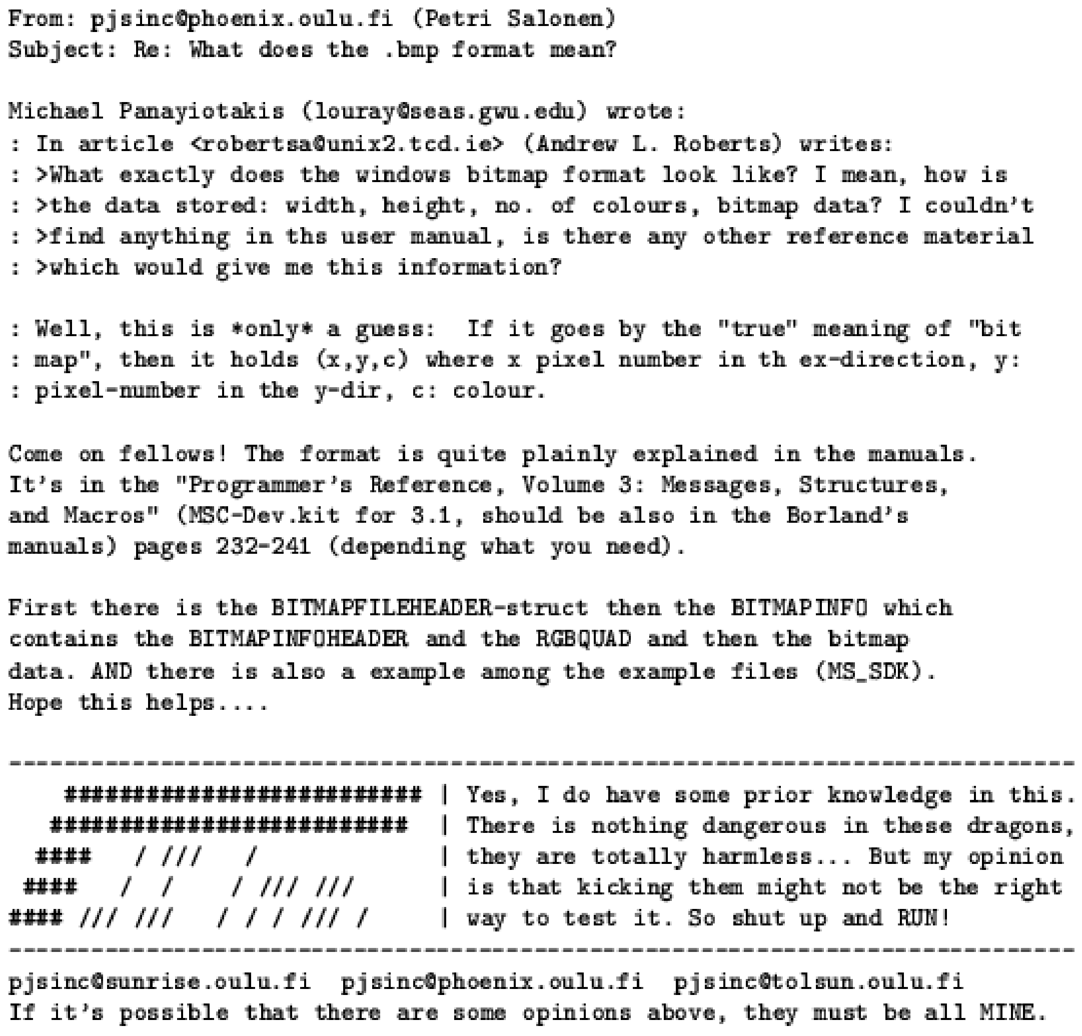
The Reuters-21578 dataset contains news articles published by the Reuters News Agency in 1987. Reuters-21578 differs from the previous datasets in the sense that it lacks a few fundamental properties. In particular Reuters-21578 is not only multi-class but rather multi-label meaning that any one data 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 describe those subsets, 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][DS05] 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.

[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

[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

- **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.

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 previously described mismatch of weights 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. [FC18] Figure 7.1 and figure 7.3 present these measures for the MNIST-FCN and 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.[1] 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 our 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 as additional points of comparison.[2]

Evaluation Results

As seen in figures 7.1 and 7.2, 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 between figure 7.3 and figure 7.4 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.

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 produced 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.

[1] The original figure is available in the paper of J. Frankle and M. Carbin. [FC18]

[2] For the Conv6 architecture, the effective pruning rate per iteration amalgamates its dense and convolutional pruning rate. 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. Furthermore J. Frankle and M. Carbin arrive at slightly different pruning rates after rounding.
While the labels nominally differ, we visualized our data with a grid corresponding to the same amount of training or pruning. It reduces legibility slightly in the case of Conv6 but increases ease of comparison across the board.

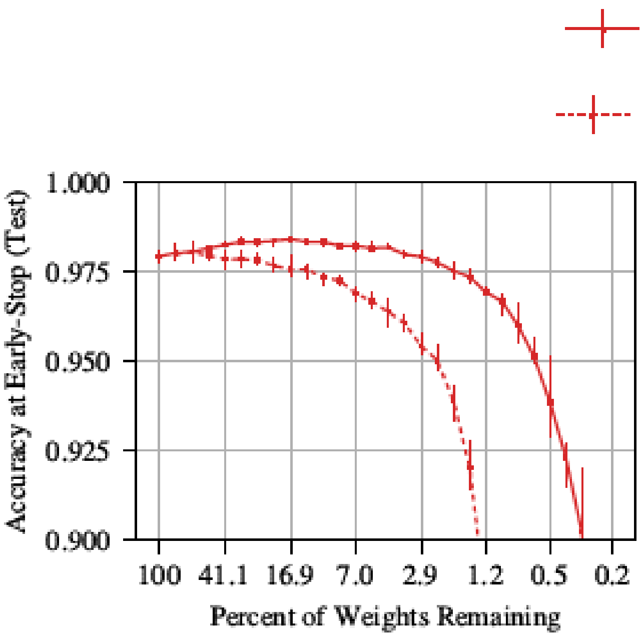


Figure 7.1.: LTH: MNIST-Lenet-FCN

Source:
"The Lottery Ticket Hypothesis" [FC18]

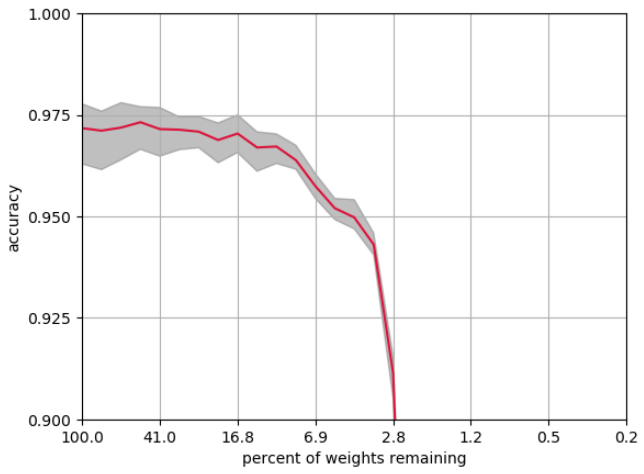


Figure 7.2.: Thes: MNIST-Lenet-FCN

Source:
This graph was produced by the author.

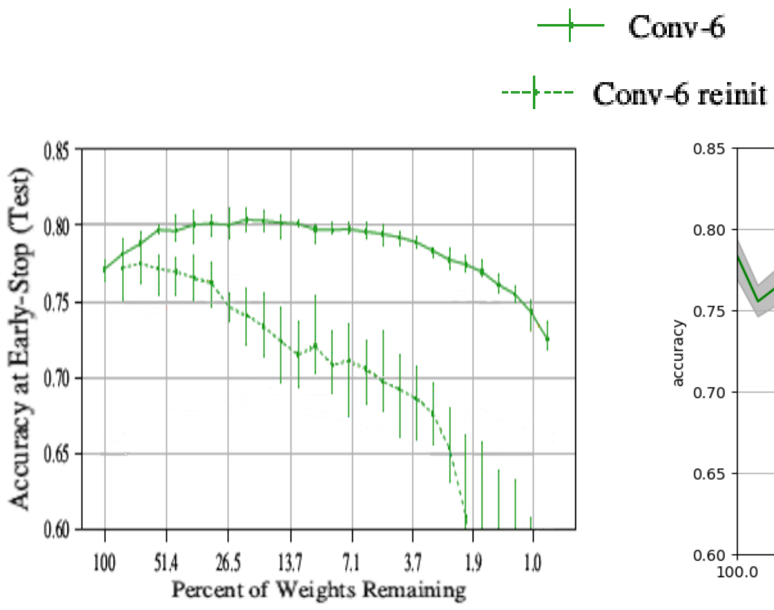


Figure 7.3.: LTH: CIFAR10-Conv6

Source:
"The Lottery Ticket Hypothesis" [FC18]

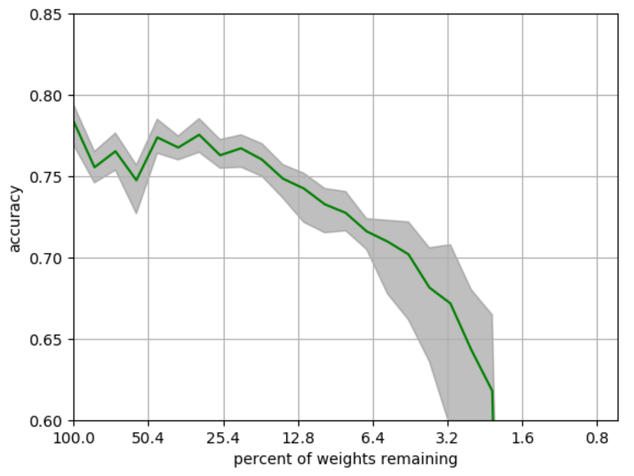


Figure 7.4.: Thesis: CIFAR10-Conv6

Source:
This graph was produced by the author.

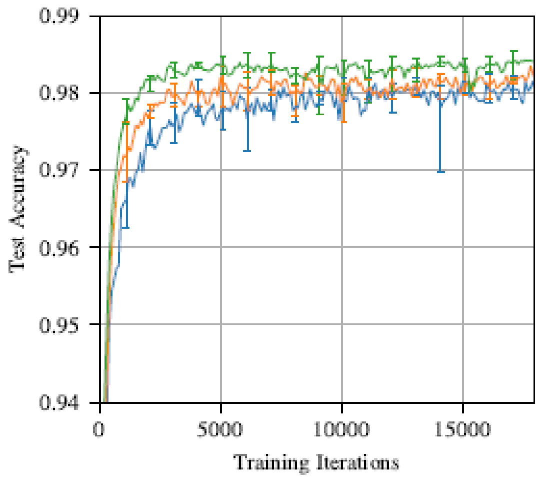


Figure 7.5.: LTH: Slightly pruned Lenet-FCN

Source:
"The Lottery Ticket Hypothesis" [FC18]

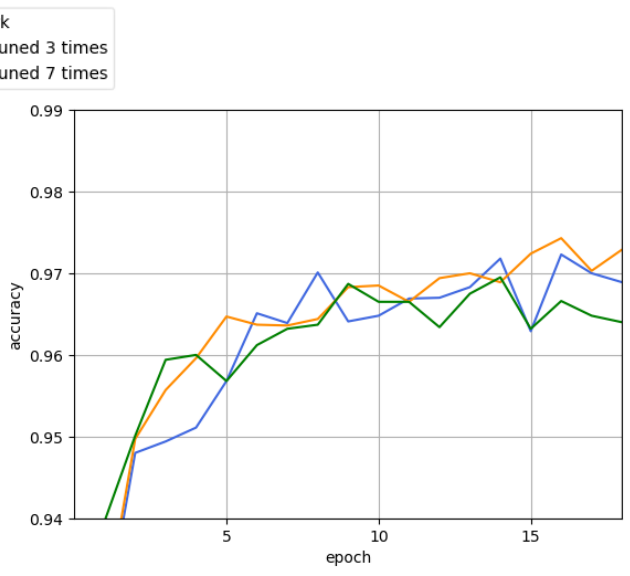


Figure 7.6.: Thesis: Slightly pruned Lenet-FCN

Source:
This graph was produced by the author.

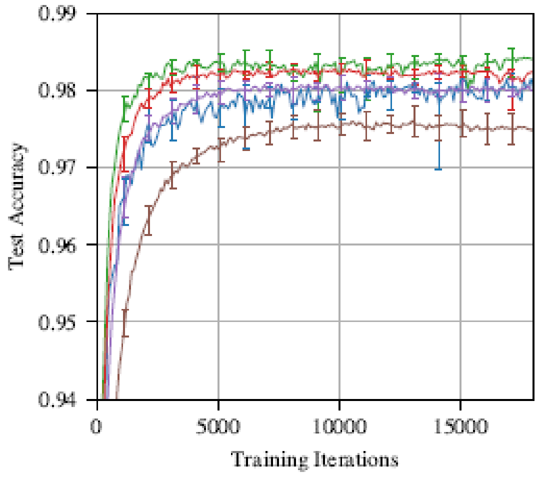


Figure 7.7.: LTH: Heavily pruned Lenet-FCN

Source:
"The Lottery Ticket Hypothesis" [FC18]

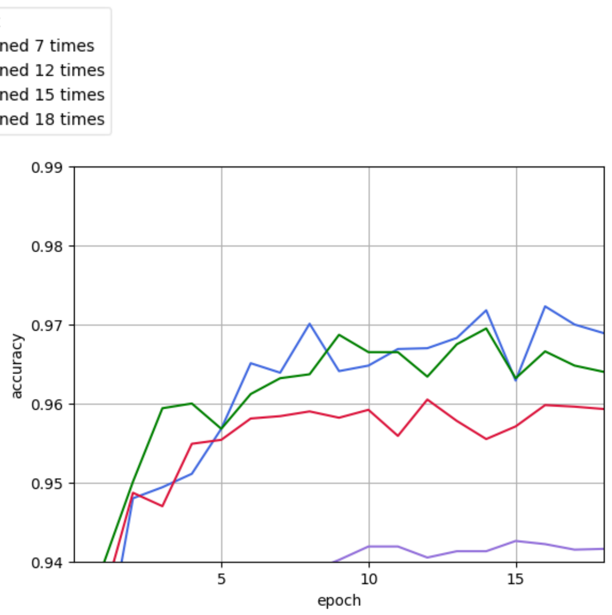


Figure 7.8.: Thesis: Heavily pruned Lenet-FCN

Source:
This graph was produced by the author.

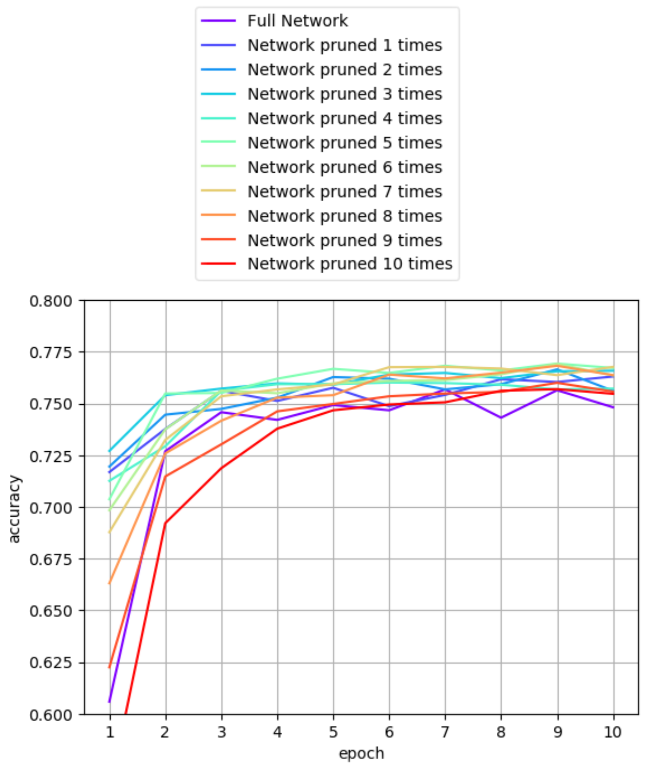


Figure 7.9.: Training history on 20Newsgroups

Source:
This graph was produced by the author.

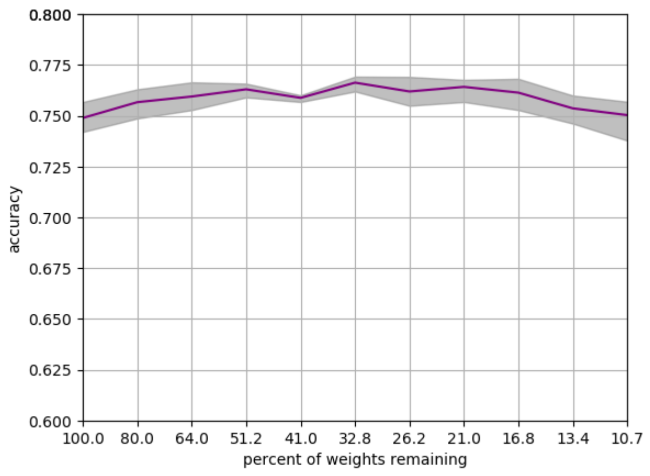


Figure 7.10.: Accuracy after convergence

Source:
This graph was produced by the author.

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 and it did so through the masking of networks with frozen initialization. As such, the transfer to another field still might yield a new pruning tool for additional tasks and applications. If the framework prunes about 90 percent of weights without the sacrifice of prediction quality, we consider the transfer successful.

Evaluation Setup

Figure 7.10 presents benchmarks collected in the same manner as for the other architectures, but fewer pruning iterations are displayed. Additionally, figure 7.9 provides the training accuracy we used to determine the amount of training necessary for convergence. The sheer size of the 20Newsgroups-End2End architecture 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 perform experiments with ten pruning iterations, the minimal number to prune a network to a competitive degree.

Evaluation Results

The right side of figure 7.4 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.[ZLLY19] Information on the development of the quality of said weights is still of interest.

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

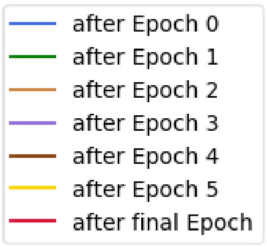
Figures 7.11 through 7.14 plot the mean accuracy achieved by implementations set to prune at the n-th epoch of training. To avoid visual clutter, the intervals representing minimal and maximal values are omitted.

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



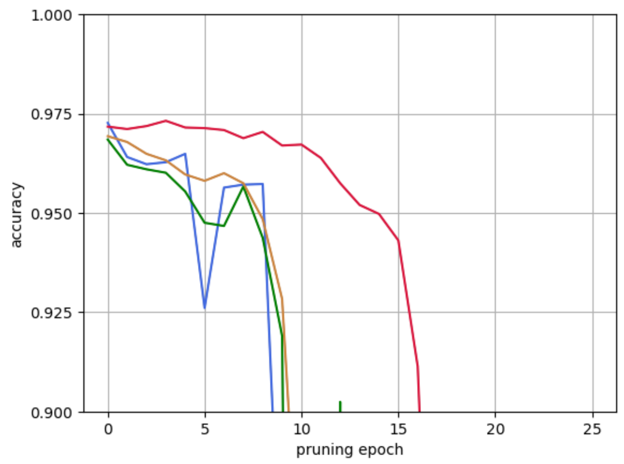


Figure 7.11.: Networks pruned at epochs 0|1|2

Source:
This graph was produced by the author.

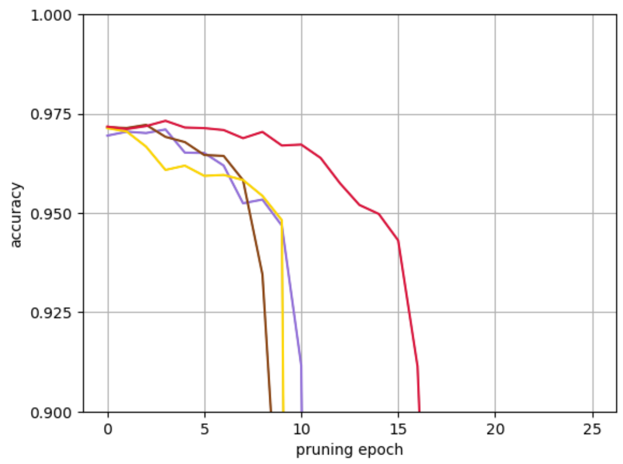
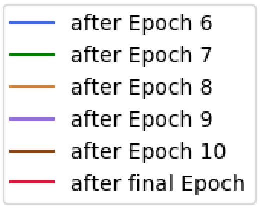


Figure 7.12.: Networks pruned at epochs 3|4|5

Source:
This graph was produced by the author.



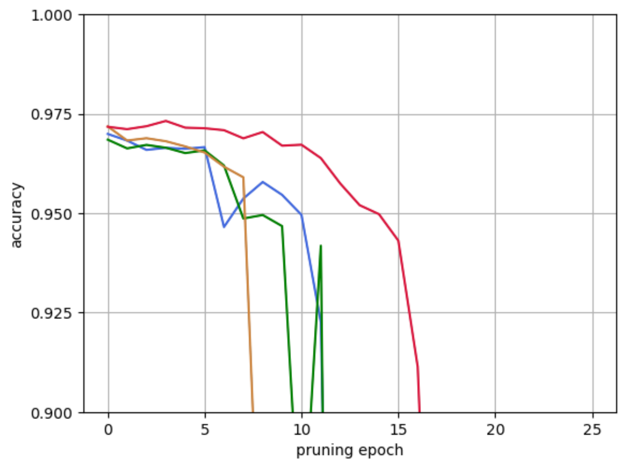


Figure 7.13.: Networks pruned at epochs 6|7|8

Source:
This graph was produced by the author.

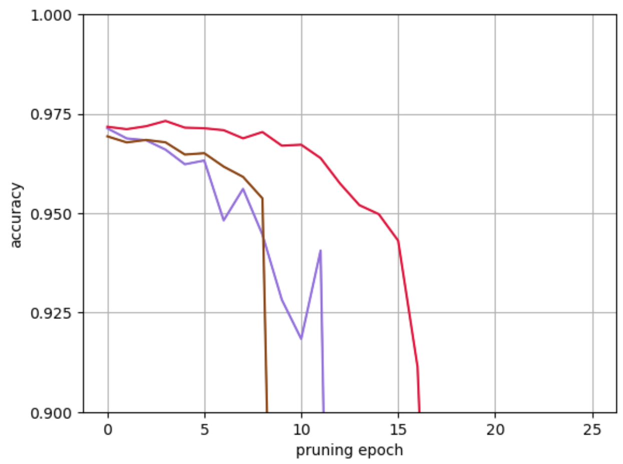


Figure 7.14.: Networks pruned at epochs 9 and 10

Source:
This graph was produced by the author.

8 Conclusions

To conclude this thesis, this last section summarizes the accomplished work, assures results and records 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, 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 in accuracy. 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 its reproduction experiments failed, the framework 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 backend. During the restoration of weights, 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 or the 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

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[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.

[BK09] Edward Loper Bird, Steven and Ewan Klein. *Natural Languag Processing with Python*. O'Reilly Media Inc., 2009.

[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.

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

[HHYX19] Lu Haonan, Seth H. Huang, Tian Ye, and Guo Xiuyan. Graph star net for generalized multi-task learning, 2019.

[Ho-18] Tien Ho-Phuoc. CIFAR10 to compare visual recognition performance between deep neural networks and humans. *CoRR*, abs/1811.07270, 2018.

[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.

[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.

[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.

[LJB$^+$95] Yann LeCun, LD Jackel, Leon Bottou, A Brunot, Corinna Cortes, JS Denker, Harris Drucker, I Guyon, UA Muller, Eduard Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia, 1995.

[LKK$^+$19] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *CoRR*, abs/1905.00397, 2019.

[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.

[PVG$^+$11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[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.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[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/.

[YS19] Ikuya Yamada and Hiroyuki Shindo. Neural attentive bag-of-entities model for text classification, 2019.

[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 the 20Newsgroups-End2End architecture

Newsgroup-End2End

Sequential Layer Defaults	**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
	output dimension	3
Sequential A8	input from	Input

Newsgroup-End2End

	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
	1D Average Pooling	pool size = 7
	output dimension	3
Concatenate	input from	Sequential A1
		Sequential A2
		Sequential A3
		Sequential A4
		Sequential A5

Newsgroup-End2End

		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