BACHELOR PAPER

Term paper submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Computer Science

On the State of ANN Computation Graphs and the potential of Conditional Neural Networks

By: Daniel Nepp

Student Number: 51832066

Supervisor: Markus Petz, MSc

Vienna, May 25, 2021

FH University of Applied Sciences
TECHNIKUM WIEN

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, May 25, 2021 Signature

Kurzfassung

In dieser Arbeit wird der momentane Stand der Technik im Bereich des *Deep Learning* thematisiert. Konkret werden künstliche neuronale Netzwerken in Bezug auf den Aufbau der Rechengraphen als auch ihrer Lernstrategien untersucht, um häufig vorkommende Architekturmuster zu identifizieren. Das Ziel ist hierbei diese Architektur Muster im Hinblick auf biologoische neuronale Netzwerke zu untersuchen um mögliche Herausforderungen im momentanen Stand der Technik aufdecken zu können. Darauf aufbauend werden *bedingte neuronale Netzwerke (conditional neural networks)* untersucht, eine Kategorie von neuronalen Netzwerken mit dem Potential die identifizierten Probleme in dem Bereich des Deep Learning zu lösen. Zudem werden aktuelle Probleme identifiziert mit denen bedingte neuronale Netze konfrontiert sind. Anhand der daraus gewonnenen Erkenntnisse stellt diese Arbeit eine eigene bedingte NN Architektur vor um diese in darauffolgenden Kapiteln auf die besprochenen Vor- und Nachteile zu testen.

Schlagworte: Maschinelles Lernen, Tiefes Lernen, Neuronale Netzwerke, Bedingte Neuronale Netzwerke, Künstliche Intelligenz, Rechengraphen, KI

Abstract

This work discusses the current state of the art in the area of *Deep Learning*. It examines artificial neural networks with respect to the structure of their computation graphs as well as commonly used learning strategies in order to identify common architectural design patterns. The goal is to distinctly identify those architectural patterns which represent discrepancies compared to biological neural networks to investigate and uncover possible challenges in current state of the art design choices. On this basis *conditional neural networks* will be discussed. They are a special category of neural networks with the potential to overcome these challenges in the field of deep learning. Additionally the thesis discusses the challenges which are faced by current state of the art conditional neural networks. Furthermore a custom conditional neural network architecture is introduced which was tested on the discussed advantages and disadvantages.

Keywords: Machine Learning, Deep Learning, Neural Networks, Conditional Neural Networks, Sparsity, Artificial Intelligence, Computation Graph, Online Learning, AI

Acknowledgements

I hereby want to thank my partner Emily Riemer for supporting me throughout the making of this thesis. I also want to express my sincere gratitude to Jenny Gebauer, Oliver Schweigler, MSc Markus Petz and DI Harald Demel for proofreading this thesis while also providing helpful advice and feedback. Lastly I want to acknowledge my cat Hailey for sharing her calm aura with me during stressful times.

Contents

1 Introduction

1.1 Motivation

The core motivation for this thesis has been the incredible technological advance in the field of deep learning, or to be more precise: advances in neural network architectures. Artificial Neural Networks (ANNs) are considered to be universal function approximators [29, chapter 4] [13, subsection 6.4.1]. This property has enjoyed a growing potential for concrete applications and fitness for deployment throughout recent years. A development which is also accelerated by the significant advances in parallel computing and growing accessibility of large amounts of data [13, section 1.2]. This extraordinary development has been a key motivation for this endeavour. The core inspiration for the ideas outlined in this paper has been a set of publications concerning novel ideas about so called Conditional Deep Learning Networks (CDLN s) [37][11][10] and their methods for hard layer routing, which seem to tackle relevant challenges in the field.

1.2 Problem

This thesis seeks to answer the following main question:

What are the potentials and challenges of conditional neural networks?

However, in order to assess this question thoroughly, it will be answered through the following 4 sub-questions, addressed throughout the chapters of this thesis:

1. How are neural network architectures commonly structured?
2. What are possible limitations arising from these structures?
3. What are conditional neural networks and how are they trying to tackle these limitations?
4. What challenges are conditional neural networks facing?

1.3 Methods

1.3.1 Research

Prominent books about the field of deep learning establish a basis and serve as a source for most of the contextual claims made in this thesis. [13] [29] [32] The more specific content related to concrete results is based on the large body of peer reviewed papers referenced throughout the chapters of this thesis.

1.3.2 Search Techniques

Research has been conducted based on querying search engines like Google Scholar, Microsoft Academic and Research Gate. In order to be able to quickly search through long blocks of texts across a large body of research (which mostly did not end up in this thesis) carefully crafted keywords were used. The technique was a useful tool for both finding relevant scientific literature as well as relevant information within said literature. It was extraordinarily useful when reading, analysing and understanding a given paper or article. Searching for particular words on any type of literature lead to an immediate understanding of the contents at hand. Difficult papers where information is condensed into complex paragraphs which often assume a certain context could be understood more easily by searching for unclear or unknown references in the given context.

1.3.3 Experimentation

Based on the theoretical groundwork layed by the discussed research in the previous chapters of this thesis, a custom neural network architecture is introduced which tries to explore a unique approach to conditional neural networks. The architecture was examined by testing it on a relatively small number of token sequences and analysing it with respect to convergence, validation loss, activity saturation and routing bias.

1.4 Related Work

The main focus of this thesis is to examine common Artificial Neural Network (ANN) architectures with respect to their computation graph structure and compare them with approaches using conditional branching. This thesis tries to explain the reasoning behind conditional computation in deep learning and highlights the current and future potential of Conditional Deep Learning Network (CDLN) architectures. Large parts of it have been based on the concepts introduced in “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer” [37] as well the conceptual predecessor “Learning Factored Representations in a Deep Mixture of Experts” [10] which introduced the fundamental idea of conditionally selected "*experts*". Furthermore, “Energy-Efficient and Improved Image Recognition with Conditional Deep Learning” [31] and “Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition” [30] provide interesting results with respect to energy consumption of conditional architectures. Another interesting approach and with an even higher degree of conditional branching is “PathNet: Evolution Channels Gradient Descent in Super Neural Networks” [11] which demonstrates potentials of stacked conditional layers for transfer learning with a combination of stochastic and evolutionary training.

2 State of the Art

2.1 Introduction

The field of deep learning is populated densely by a huge variety of ANNs whose use cases stretch across a wide spectrum of applications such as Computer Vision (CV), Natural Language Processing (NLP) or general-purpose pattern recognition. The underlying architectures used for such tasks are too numerous to cover in one paper, however, there seem to be certain reoccurring patterns shared by many designs which may be prevalent on further inspection. [5, p. 12-14] [25, p. 0-1]

Core principles of modern networks and training methods have been present since the eighties and nineties of the previous century. This is especially true for state-of-the-art training methods for the adjustment of weight variables in ANNs, which continue to use partial derivatives for reverse mode differentiation and error propagation, a technique famously termed backpropagation back in the 1960s and then popularized in 1986 (“Learning Representations by Back-propagating Errors” [34]). This almost exclusively used training method is now considered the workhorse of ANNs. [5, p. 14-15] [20, p. 33]

Although there have been interesting advances in training methods, like synthetic gradients [15], at their core, popular weight update methods seem to be mostly variations of backpropagation, relying heavily on partial derivatives and error propagation in some form. [13, section 6.5-6.6] Therefore, this paper will not attempt to examine the current state of a potentially already established technology. However, the following chapters will attempt to categorize and examine the architectures of popular ANNs in terms of their underlying computation graphs and structure, while also comparing them to properties of biological neural networks.

2.2 The Computation Graph

ANNs can be described via illustrations of their computation graphs, which is especially useful when trying to differentiate them for back-propagation. [13, subsection 6.5.1] However, these illustrations do not adhere to a common scheme describing all architectures. This poses a problem when trying to comprehend, analyse, and present differences and commonalities of various ANNs. To be able to properly examine and present multiple ANN architectures with respect to their structure, we will now proceed to define a simple illustration legend which will be used as a basis for comparing structural properties of popular ANNs in the subsequent parts of this thesis.

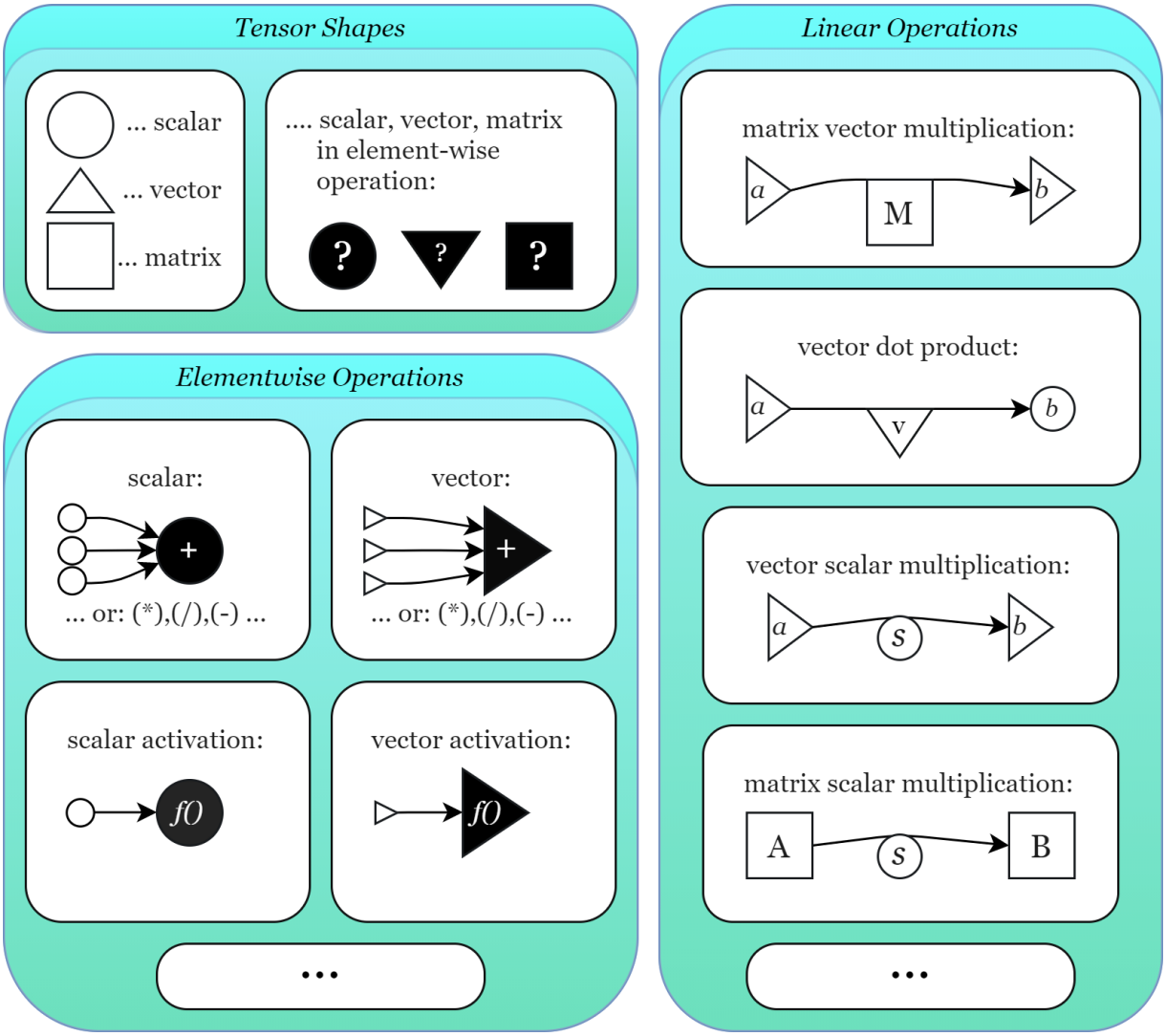

Figure 1: A Computation-Graph Legend

Similar as in other computation graphs, nodes are tensors and arrows mark their relationship as well as flow of information. There are three types of node shapes matching a corresponding tensor shape (scalar, vector, matrix) which can also be filled black to clearly identify an

element-wise operation. Linear operations on the other hand are represented by attaching the right operand (usually a weight tensor) of the operation to the edge of arrows marking state transitions. This is useful because it does not distract from the actual flow of information for a network.

2.3 Common Neural Networks

Extensive research about the different types of ANNs used in research and deployment, will yield a variety of architectures which often are compositions of various design types. Identifying these types and assessing their popularity is challenging. We therefore try to approximate a collection of popular designs by picking architectures discussed in the literature referenced by this theses. Namely: books [13][32][29], a survey [25] and widely shared and liked Medium articles about ANNs [3] [16], which may very well reflect the interests among the ANN research community. The following is the result of this approximated popularity as an unsorted list of architecture categories. It will be examined with respect to a certain set of properties defined in the subsequent section.

- Simple Feed-Forward Networks [13, chapter 6]
- Convolutional Neural Networks (CNNs) [13, chapter 9]
- Capsule Networks
- Recurrent Neural Networks (RNNs) [13, chapter 10]
    1. Gated Recurrent Unit Networks (GRUs)
    2. Long-Short Term Memory Networks (LSTMs)
- Residual Networks (ResNets) [14]
- Generative Adversarial Networks (GANs)
- Echo State Networks (ESN) [13, section 10.8]
- Deconvolutional Neural Networks (DNNs)
- Auto Encoder Networks (AEs) [13, chapter 14]
    1. Variational Auto Encoder Networks (VAEs)
- Hopfield Neural Networks
- Boltzmann Machines [13, chapter 20]

2.4 Common Properties

Studying these architectures gives some insight into reoccurring patterns that these networks usually have in common. This is especially true for the structure of the networks, more specifically the underlying structure of the computation graphs which we will discuss. Based on our research we will now proceed to define 3 categories / properties which are applicable to biological neural networks but only partially to popular artificial neural network architectures like those previously mentioned.

The following properties were identified and chosen based on aspects which could shine light on areas in ANN model design which potentially need improvement or further research focus.

2.4.1 Full Activity - Sparse Activity

Biological neural networks are known for their sparsity, namely the observation that for a given time frame only parts of all neurons are active at high frequency, whereas large portions of the total amount of neurons stay mostly silent or fire only very infrequently. This view is supported by data which examines the firing behavior of neurons within the neocortical regions of the human brain. The following excerpt is taken from the abstract of a thorough review on this subject matter, namely "Experimental evidence for sparse firing in the neocortex."

> *The advent of unbiased recording and imaging techniques to evaluate firing activity across neocortical neurons has revealed* ***substantial heterogeneity in response properties in vivo****, and that* ***a minority of neurons are responsible for the majority of spikes****. Despite the computational advantages to sparsely firing populations, experimental data defining the fraction of responsive neurons and the range of firing rates have not been synthesized. [...]*
>
> — [4] (Abstract)

It has been theorized that neurons activate selectively only for cases in which their use makes sense. This would explain away this observation of sparsity as a smart measure for energy saving [22] as well as selective and local memory access, where neurons propagate their encoded information exclusively in relevant situations [8]. Based on this sparsity, as can be observed in nature, an ANN architecture displaying "*partial*" or "*sparse*" activity will only use a subset of all containing parameters to make a single prediction.

If we consider for example networks using the rectified-linear unit as a hidden activation function, then the produced hidden states will partially be populated by zeros for negative or zero input scalars. In theory, any linear propagation following after these states can be omitted, as they will not influence the network any further. [9, p.1, p.8]

However, in practice, no such distinction will be made when propagating information through a traditional ANN. An exception to this would be so called "*Conditional Neural Networks (CLNN s)*" which either have stacked hidden layers which are being chosen for activation selectively or simply output the results at any layer (based on certain confidence conditions) to save computational costs. [30, p. 1]

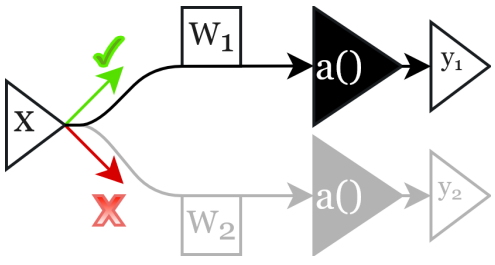

Figure 2: Sparse Activity

Conversely to biological neural networks, ANNs usually however display "*full activity*". This is the case if every parameter of the network is involved in forming an output given any input that is being fed into the neural network. The vast majority of current state of the art network architectures fall under the category of "*full activity*" networks. (CNNs / DNNs, FFNNs, RNNs, AEs, . . .)

2.4.2 Feed Forward - Computation Graph Branching

ANNs are at their core merely differentiable computation graphs consisting of tensors and operations. The usage of tensors is especially helpful because they are an abstraction of scalars, vectors and matrices [7] [13, section 2.1]. In practice, this is implemented as n-dimensional arrays which enable the storage of large amounts of weight and activation scalars compactly on a single memory layer (typically in RAM) for accelerated parallel execution ideally on a single compute device to avoid being bottle-necked by bandwidth.

When viewing a traditional feed forward neural network as a computation graph of tensors and operations then this graph structure does not contain branches [13, chapter 6]. During execution the entire model resides in RAM and all parameters are being utilized to form a given output.

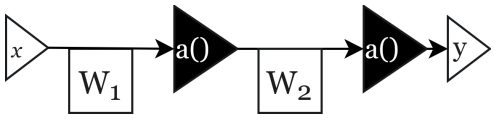

Figure 3: A CG of a FFNN

However theoretically introducing branches would allow for selective / sparse activity propagation (subsection 2.4.1) and therefore the ability to store inactive branch variables on slower memory with higher capacity until needed again.

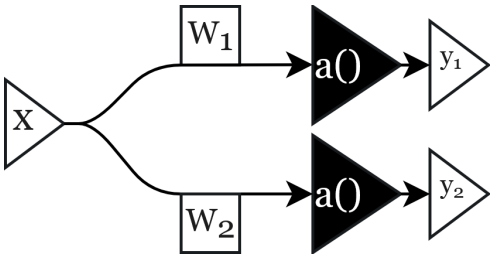

Figure 4: Branching CG

Based on that, a neural network architecture has so called “computation graph branches” if the flow of information goes through different (linear) tensor operations which eventually merge back together to ultimately form the final output. Branching for different elementwise operations does not count as computation graph branching because they can easily be performed within a single activation, however two weight tensors and different elementwise activations applied to one forward pass would fall under this definition.

A good example of this property can be found in ResNet, in which previous layer results are being copied back into layers further down the end of the network. Components performing this operation are being referred to as “residual layers”. [14]

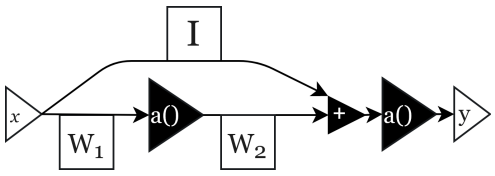

Figure 5: A residual layer

2.4.3 Directed A-Cyclic - Recurrent

As previously mentioned, the underlying architecture of many ANNs adheres to a feed forward pipeline-like structure. In such networks information flows solely in one direction, meaning that such a neural network cannot retain information of previous forward passes.

In essence such a design can be viewed as an acyclic directed computation graph [13, chapter 6]. Therefore, it is not only a pure method in a computational sense, but also a mathematical (total) function being both left-total as well as right-unique (for tensors). This is based on the fact, that traditionally ANNs consist of simple linear and non-linear operations forming a directed acyclic (computation) graph (DAG) of tensors and operations. [24, p. 8-10] The most obvious example of such a “pipeline” neural network architecture would be a standard feedforward neural network.

Conversely an ANN deviating from this design would be a Recurrent Neural Network (RNN) which would also allow for cyclic propagation of information through the computation graph. [32, p. 143] [24, p. 10-13] This flow of information is being illustrated by Figure 6. Recurrency is a key property found in biological neural networks which are notoriously cyclic [18] [19].

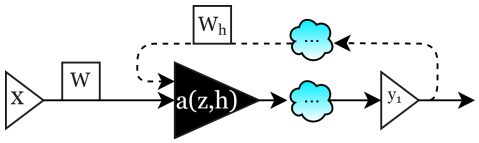

Figure 6: A computation graph demonstrating a recurrent flow of information.

Therefore future ANN models aiming towards achieving Artificial General Intelligence (AGI) surely would need to have some form of recurrent flow of information incorporated into them.

2.4.4 Batch Learner - Online Learner

State of the Art ANNs are most often trained on multiple samples at once. [36, p.1] These sets of samples, often referred to as batches [21, chapter 4], are being fed into a given network in the form of one or more large tensors storing these stacked samples densely in RAM, which is why memory is often a determining factor for the batch size [13, subsection 8.1.3]. The final gradients applied to the weight parameters of a given network will then be the average of the gradients produced by the set of samples. This approach is the reason why state of the art ANNs are generally being referred to as *batch learner* [36, p.1]. It has several advantages:

- As an average of many gradients the final gradient applied to the weights will have less noise, which makes it a more informed step towards the global.
- This reduced level of noise will ensure a more stable convergence during training.
- Using batches also enables the use of Single Instruction Multiple Data (SIMD) dedicated hardware like GPUs, which greatly accelerates the propagation of multiple samples through the network.

— [21, chapter 4]

However, there seem to be trade-offs associated with this approach. As observed by Nitish Shirish Keskar et al. [17] :

> *The stochastic gradient descent method and its variants are algorithms of choice for many Deep Learning tasks. These methods operate in a small-batch regime wherein a fraction of the training data, usually 32–512 data points, is sampled to compute an approximation to the gradient.* ***It has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize.*** *There have been some attempts to investigate the cause for this generalization drop in the large-batch regime, however the precise answer for this phenomenon is, hitherto unknown. In this paper, we present ample numerical evidence that supports the view that large-batch methods tend to converge to sharp minimizers of the training and testing functions – and that sharp minima lead to poorer generalization. In contrast, small-batch methods consistently converge to flat minimizers, and our experiments support a commonly held view that this is due to the inherent noise in the gradient estimation. We also discuss several empirical strategies that help large-batch methods eliminate the generalization gap and conclude with a set of future research ideas and open questions.*
>
> — [17] (Abstract)

This degradation in performance when using larger batches has lead to a more balanced approach, namely the use of so called *mini-batches*, which are simply smaller batches whose size might be adjusted during training. [21, chapter 4.1]

Not using a batch at all and merely calculating a gradient based on a single sample is being referred to as *stochastic learning*, or more generally *online learning* [21, chapter 4.1] [36] [13, subsection 8.1.3]. Although this approach is faster, because less data and computational resources are needed for the adjustment of weights, the calculated gradient will contain much more noise. This noise will translate to the weights of a given network when applying the gradient, which is widely believed to be the reason why this approach is susceptible to a phenomenon called *catastrophic inference*, which is simply the observation that ANNs tend to forget what has already been learned upon introducing new information (or noise) [12, p. 1].

Finally, batch learning is an artificial learning method. Evidently, Biological Neural Networks (BNNs), as can be experienced in day to day live, receive, process and learn newly introduced information sequentially and not in the form of batches.

2.5 Resulting Property Table

The properties defined under the previous section, namely *sparse/full activity* (2.4.1), *feed forward/branching* (2.4.2), *a-cyclic/recurrent* (2.4.3) and *batch/online learning* (2.4.4), either apply to a given network or they do not, meaning the properties are boolean in nature. Running these three properties through my list of common neural networks will yield the following truth table:

ANN Type	Recurrency	Branching	Sparse Activity	Online Learner
Feed-Forward Networks	False	False	False	False
ResNets	False	True	False	False
RNNs (LSTM, GRU)	True	True	False	False
CNNs	False	True	False	False
Capsule Nets	False	True	False	False
DNNs	False	False	False	False
GANs	False	True	False	False
AEs	False	True	False	False
VAEs	False	False	False	False
Hopfield NNs	True	False	False	False
Boltzmann Machine	True	False	False	False

Table 1: State of the Art ANN architecture property truth table

The ANN types evaluated in this table ought to represent versions as they are *typically* implemented and trained. In practise one can build variations of these types which deviate from this overview. Such variations for example will be mentioned in chapter 3. Any of these mentioned types could for example easily be trained without using batches, making them technically online learners. Therefore, *the table above is subject to change and may not be true in the future*.

3 Conditional Neural Networks

3.1 Introduction

As established previously, the four discussed properties apply to popular neural networks only partially or not all, despite the fact, that said properties seem to be prevalent in biological neural networks.

This chapter will discuss a relatively novel category of neural networks which were briefly mentioned in subsection 2.4.1, namely "CLNN s“, which are also being referred to as “CDLN s”. ANNs falling under this category tackle the previously discussed challenges.

Such ANNs have both branching as well as sparse activity. Therefore such architectures can be arbitrarily large without considerable increases in additional computational or memory costs [30, p. 2] and a potential increase in execution time for forward passes [6]. This approach has found promising use in Natural Language Processing (NLP) [37], Reinforcement Learning (RL) [6], image recognition / classification tasks where the prediction of a very large number of classes is required [31] [26] , transfer learning [11] and online learning [23].

3.2 Gating

To achieve both branching and sparsity of activity, conditional neural networks use sophisticated gating / routing mechanism based on which the decision, which part of the network ought to be activated next, or if any further parts ought to be activated at all, is made.

These mechanisms can come in different forms, such as continuous, binary, stochastic, random or sparse functions.

Depending on the type they can either be differentiable and therefore trainable via backpropagation [10] [6] [37] or they are based on other techniques like evolutionary algorithms [11].

3.3 Challenges

3.3.1 Training

As previously discussed in subsection 2.4.4, traditional ANNs are batch learners, meaning that during training, models are being fed with a set of samples at once. A method commonly referred to as *batching* (*full-batching*, *mini-batching*) [32, p. 32-33].

For conditional neural networks however, it is difficult to process batches containing more than one sample. This is because two distinct samples might be routed differently through the network, which means that one would have to create batches based on common activation paths for the data set.
This approach has been tested in [37, p. 5] and [10], where input samples were batched after running them through the gating mechanism, which in these cases is a single feed forward ANN at the beginning of the network. This gating network routes activity to a number of sub-networks referred to as *experts*. However, depending on the gating mechanism and the degree of conditional branching, this approach might not be feasible, in which case the training can only be done by applying the training samples sequentially.

This severely compromises training efficiency, because the computation becomes far less parallelizable, which in many cases renders available SIMD accelerator hardware effectively unusable. Besides graphical workloads, GPUs for example are highly optimized devices for parallel arithmetic types of workloads, however their performance drops drastically when attempting to introduce conditional branching as in conditional neural networks [37, p. 2].

The previously mentioned papers [37] and [10] could still batch samples because the network architecture consists of a single conditional layer only. The architecture described in “PathNet: Evolution Channels Gradient Descent in Super Neural Networks” however consists of stacked conditional layers with their own gating mechanism. Every sample sent into this architecture might be routed differently through the conditional layers having a unique activity path. Contrary to the approaches taken in [37] and [10], which used back-propagation as training method, the conditional paths in *path-net* were trained using an evolutionary algorithm [11, p. 1].

3.3.2 Routing Bias

Another challenge for CDLN s has been the emergence of routing bias during training, where the gating mechanism will converge towards a state in which the same few sub-networks will always be chosen. This problem has been observed in [10] as well as in more recent work [37].

This routing imbalance is self-reinforcing, because a choice favoured by the gating mechanism will lead to that conditional path being trained more thoroughly, which then will subsequently lead to it being chosen even more favorably, as it simply yields better results [37, p. 5][10, p. 2].

In order to mitigate this emergent bias the researchers in [10] and [37] introduced soft and hard constraints to the gating mechanism as well as a modified loss function which ensures that choices will adhere to a global balance. As reported in [37], this technique forces the gating network to converge towards a state in which individual *expert layers* are chosen evenly.

3.4 Potential

3.4.1 Efficiency

Due to the fact that conditional neural networks only require a subset of their total number of parameters to produce an output, they tend to also require less computational resources and therefore display higher performance results than implementations of traditional ANN architectures. The following results (P. Panda et al., 2015)[30, p. 12] show the performance of conditional networks in terms of energy efficiency (computational cost) as well as predictive performance (accuracy).

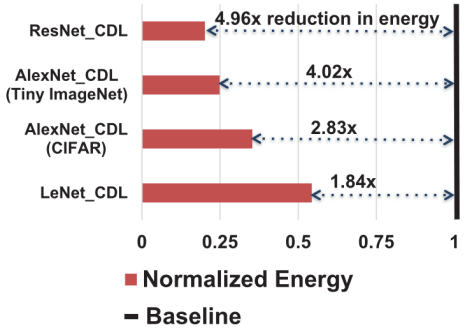

Figure 7: Proportional Energy Efficiency comparison between CDL and Baseline implementations.

The measurements displayed on the left are based on comparing popular neural network architectures to CDL versions of themselves. The researchers turned these existing architectures into conditional based versions and measured their proportional energy consumption compared to their default (baseline) implementations. The most impressive CDL implementation, namely a version of ResNet, displayed a reduction in energy consumption by a factor of almost 5, meaning that on average the baseline implementation consumed almost 5 times more computational cost and therefore also 5 times more energy when executed.

However energy efficiency alone is not relevant if the architecture does not perform well in terms of predictive performance. Therefore the researchers also measured model accuracy for both CDL as well as baseline implementations. These results were also promising as the accuracy of the tested models was not compromised.

Network	Baseline	CDLN
LeNet (MNIST)	97.55%	98.92%
AlexNet-8 (CIFAR)	78.38%	79.19%
AlexNet-8 (Tiny ImageNet)	65.24%	66.14%
ResNet-50 (Tiny ImageNet)	55.4%	56.52%

Figure 8: Accuracy of CDL Compared to Baseline.

3.4.2 Robust Online Learning

As previously mentioned in subsection 2.4.4, ANNs are known to be susceptible to a phenomenon called *Catastrophic Inference*, also referred to as *Catastrophic Forgetting*, which is especially true for *stochastic learning* / *online learning*. This phenomenon is simply the observation that ANNs tend to forget what has already been learned upon introducing new information. It arises due to the fact that when updating a given ANN the parameters will be adjusted so that the model accommodates the training data introduced in the most recent epoch [12, p. 1].

For this reason and those previously and more thoroughly discussed in subsection 2.4.4, state of the art ANNs are being trained using batches of samples instead of single samples.

As established in chapter 3, executing a conditional neural network requires only a specialized subset of the total number of parameters. Therefore, when back-propagating the error for a single training sample only this subset of parameters will be updated. Conversely, the unused parts of the network will simply stay constant.

This inherent property of CDLN s makes them more resilient to catastrophic inference [2, p. 1]. In theory this should make them more suitable for being *stochastic/online learner* than traditional ANNs. This theoretical reasoning is also supported by the results in [23] which demonstrate that the selectively used sub-networks referred to as *experts* can mitigate the emergence of catastrophic inference.

3.4.3 Transfer Learning

Transfer Learning (TL) is a method in which a trained model is being incorporated into another model either fully or only by using parts of it.

TL has been successfully applied to CDLN . Promising results for this can be found in “PathNet: Evolution Channels Gradient Descent in Super Neural Networks” [11].

The stacked layers of the architecture described in this paper consist of multiple sub-networks which can be activated conditionally. By fixing certain conditional paths and transferring them into a new model it has been shown that this new model would learn significantly faster [11, chapter 3].

4 Neural Activity Routing Unit

This chapter is dedicated to a potentially novel conditional neural network architecture which seeks to improve upon the gating and routing mechanism of current state of the art Conditional ANN architectures.

What follows in this chapter has been heavily inspired by

- *"The Sparsely-Gated Mixture-of-Experts Layer"* [37]
- *“Learning Factored Representations in a Deep Mixture of Experts”* [10]
- *“PathNet: Evolution Channels Gradient Descent in Super Neural Networks”* [11]
- *“Energy-Efficient and Improved Image Recognition with Conditional Deep Learning”* [31]
- *“Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition”* [30]
- *“Dynamic Routing in Artificial Neural Networks”* [26]
- *“Dynamic Routing Between Capsules”* [35]

The subsequent subsection will first explain the structure of this potentially novel unit architecture after which a custom model designed using this unit will be put to test.

A Neural Activity Routing Unit (NARU) consists of two fundamental components, namely:

1. **Bundles**
   Simple sub-nets similar to the so called *expert* networks described in [37] and [10] or the described *modules* in [11], which can either be active or dormant during a specific time step.

2. **Connections**
   Gated recurrent connections between bundles, which produce their inputs.

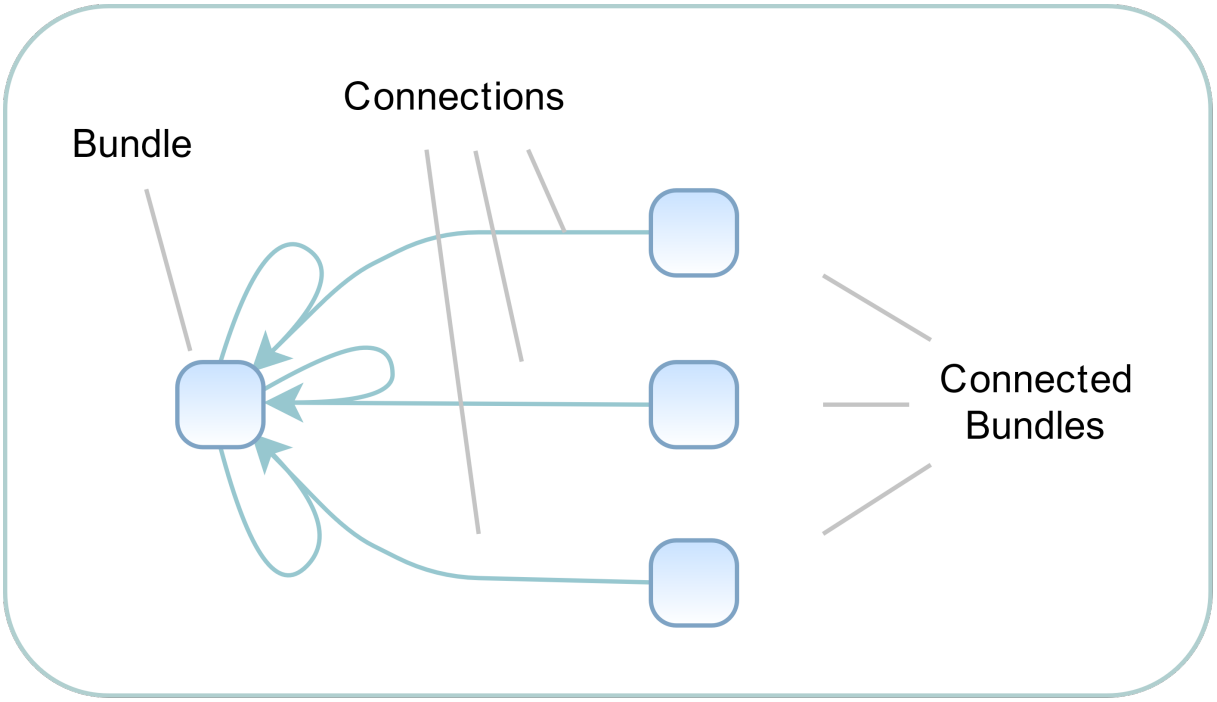

Figure 9: A Neural Activity Routing Unit (NARU)

A single neural activity routing unit consists of one bundle having an arbitrary number of connections whose outputs produce the input for said bundle.

4.1 Connections

The following illustration describes the forward pass for a single connection in terms of its computations graph.

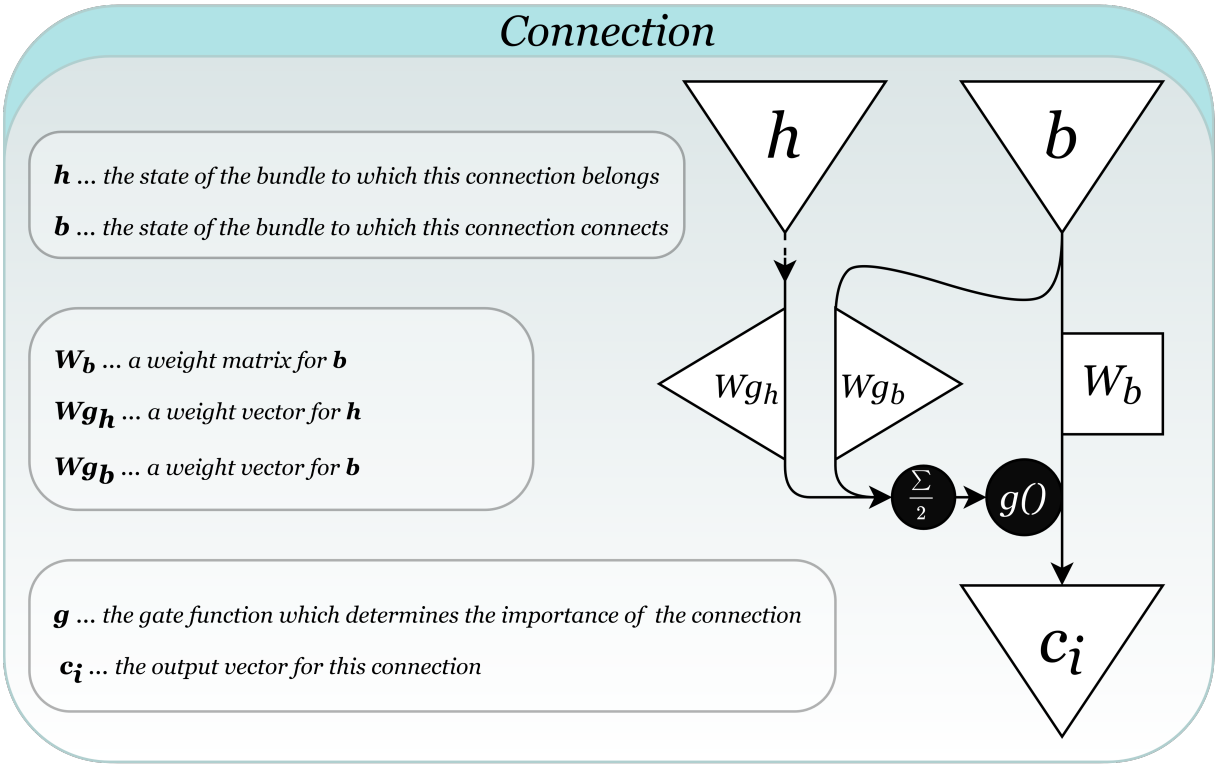

Figure 10: The Computation Graph of a Connection

A single connection receives two inputs, the state ***b*** of the bundle to which it connects as well as the state ***h*** of the bundle to which it belongs (time invariant). In essence a connection is a dense feed forward pass which is also gated. The vector ***z*** is the result of the weighted matrix multiplication with ***b***:

$$z = b \cdot W_b \tag{1}$$

This vector will then simply be multiplied by the scalar value ***g***, producing the final output of a connection, namely: ***ci***, where ***i*** is the index of the current connection among all connections belonging to the current bundle.

$$c_i = z * g \tag{2}$$

The calculation of the gate involves both ***b***, as well as ***h***. The dot product of both vectors with their corresponding weight vectors ***Wgb*** and ***Wgh*** produces two scalar values whose average will be sent through a gating function to form the final gating value.

$$g = sig(\frac{h \cdot W_{g_h} + b \cdot W_{g_b}}{2}) \tag{3}$$

The full forward pass for a connection is being described by the following expression:

$$c_i = (b \cdot W_b) * sig(\frac{h \cdot W_{g_h} + r \cdot W_{g_b}}{2}) \tag{4}$$

The results explored in this thesis are based on the implementation as described by equation 4. However for future variations one might consider increasing the complexity of connections to improve the performance of the network (especially with respect to the ability to routing as well as better attention). Producing the input vector ***z*** as well as its gating scalar ***g*** could involve the use of more complex feed forward passes having 2 or more layers.

4.2 Bundles

The connection vectors ***ci*** produced by the connections to other NARU bundles will serve as inputs to a current bundle. These vectors will be summed and divided by the total number of vectors to form a single input vector to the current bundle. A regular non-linear activation function will then be applied to this single input vector to form the final output of a bundle. Conceptually this bundle sub-network can be any feed-forward neural network, however for simplicity the reference implementation tested in this thesis does not involve more than one feed forward layer.

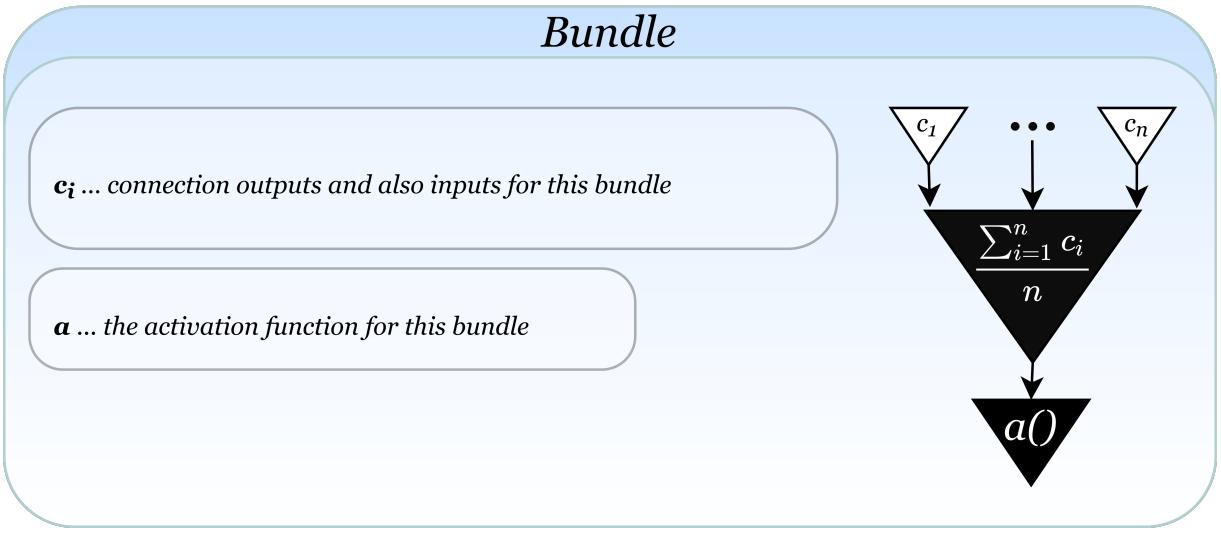

Figure 11: The Computation Graph of a Bundle.

The computation graph illustrated above can also be described by the following expression:

$$b = a(\frac{\sum_{i=1}^{n} c_i}{n}) \tag{5}$$

The activation function ***a*** will form the output vector ***b*** which is also the final state of a bundle for a given time step ***t***. As referenced in section 4.1, in the context of a given connection, this bundle state can be any of the two variables ***h*** and ***b***. However ***h*** represents the previous bundle state of the current Neural Activity Routing Unit (NARU) whereas ***b*** will allways be the bundle state of the unit to which the current connection connects.

4.3 Routing

The connections between bundles will be segregated into two distinct categories, which will be referred to as **routes** and **sources**. If a bundle **A** is connected to a bundle **B** and this connection is of type **route** then the connection of **B** to **A** will be of type **source**.

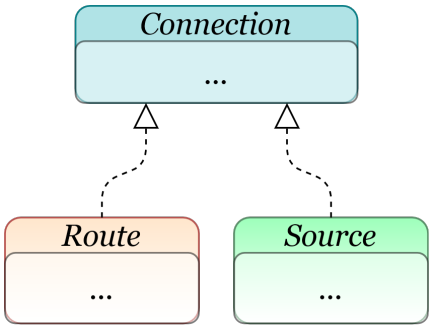

Figure 12: Connection Types

Both connection types are identical in terms of their computation graph, however they differ in terms of their purpose. Although information can flow from **A** to **B**, conceptually only the source connection will be treated as the intended direction for the flow of information.

This relationship between two bundles, as illustrated by the Figure 13 below, still lacks the core attribute which distinctly defines conditional neural networks, namely: **sparse activity**. To achieve this within a network of connected bundles we only want to activate a subset of bundles (and their connections) for a given time step. The responsibility of choosing which bundle ought to be activated next will be placed on a currently active bundle which can select a connected bundle based on a simple routing algorithm which utilizes the **route** type of connection between the two bundles.

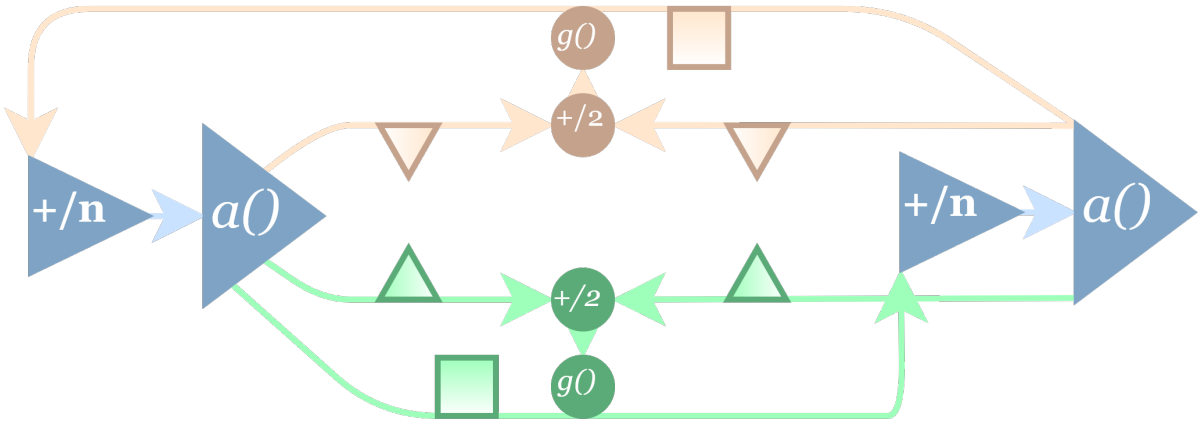

Figure 13: Two Interconnected Bundles

So for a given bundle **A**, having route connections to other bundles **B**, **C** and **D**, the routing algorithm will determine which one of **B**, **C** and **D** ought to be activated next based on the corresponding gating mechanism of the route connections to **B**, **C** and **D**.

As previously established, connections are in essence gated inputs for their respective bundles. Conceptually one could assume that the gate value produced by the gating function of a given connection serves as a score for the importance of the bundle state to which it connects. This role would simply emerge during training, because the gating mechanism would converge to a state in which it acts as an attention mechanism allowing bundles to selectively observe other bundles.

If a current bundle deems another one as important then this assumption also infers that the information encoded in this other bundle is useful, meaning that **both bundles might share very similar responsibilities**. In that case it is reasonable to assume that the other bundle could be activated next. This reasoning leads to the following conditional routing algorithm:

> ***For a given currently active bundle, the bundle which ought to be activated next will be chosen by the route whose gate value is the largest among all the route connections of the current bundle.***

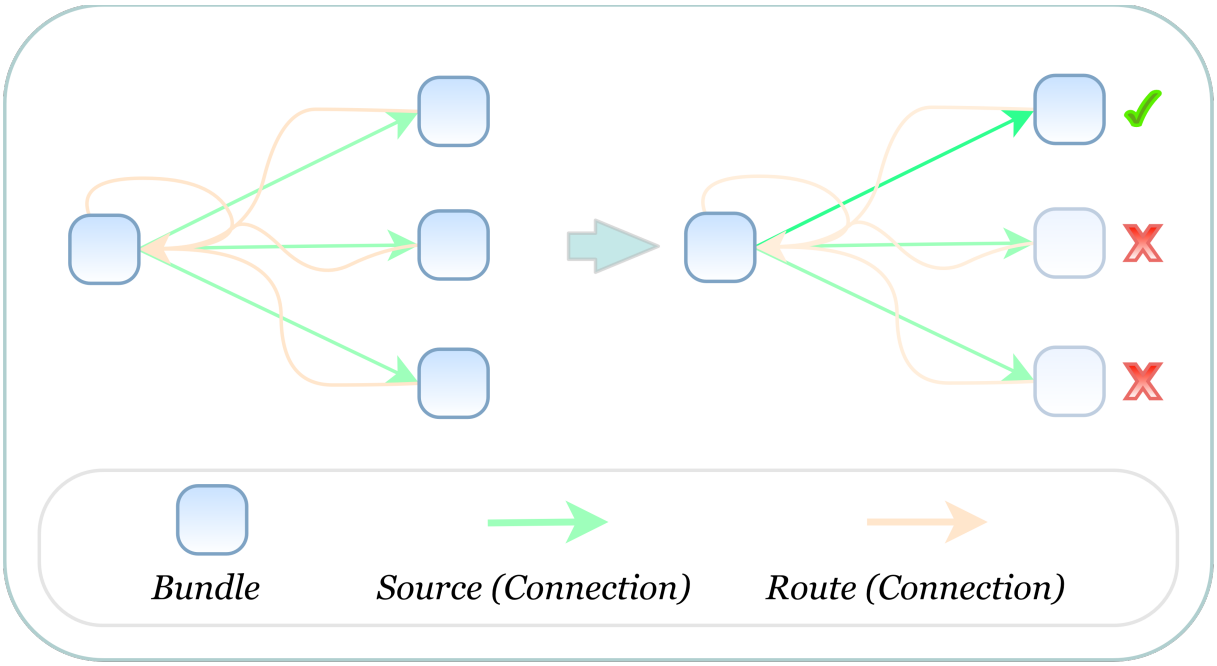

Figure 14: Directed Routing between interconnected Bundles

When constructing a network of NARUs, the previously defined distinction between routes and sources allows for a directed flow of information throughout the entire network. As illustrated by the NARU in Figure 14, the single bundle connected to 3 other bundles can choose which one of them ought to be active next, however these bundles cannot elect the left bundle.

4.4 Avoiding Routing Bias

As discussed in subsection 3.3.2, a major challenge of CDLN s is routing bias, the phenomenon that the routing mechanism does not choose the network sub-sets evenly. In order to prevent the emergence of this bias during training, NARU was purposefully designed without the use of a bias as additive parameter to the pre-image of non-linear activation functions.
As can be observed in equation 5, the activation for a given bundle does not involve the use of an additional vector of parameters.

$$a\left(\frac{\sum_{i=1}^{n} c_i}{n} \cancel{+b}\right) \tag{6}$$

This design choice is based on the reasoning that the input to a given bundle will consist of a set of connection output vectors which has previously been active and another set of vectors which has been inactive. Because among all connections of a given bundle only a single connection will be elected as route, the majority of bundles will be constant. Consequently the states of the majority of connection outputs will simply vary based on the gating scalar between 0 and 1, as is defined by equation 3, however the encoded information would only be lost to that degree.

$$a\left(\frac{\sum_{i=1}^{n} c_i}{n}\right) \longrightarrow a\left(\frac{\sum_{i=1}^{|A|} c_{a_i}}{|A|} + \frac{\sum_{i=1}^{|B|} c_{b_i}}{|B|}\right) \tag{7}$$

When segregating this conceptual group of active input vectors into set ***A*** and the set of relatively constant input vectors into set ***B***, then this resembles the usage of a bias. However, contrary to the traditional bias, this virtual bias is not constant because it is dependent on the states of a potentially larger number of dormant NARUs whose states can mutate over longer periods of time.

This makes the state of this virtual bias dependent on the inputs fed into the network.
If a given model receives information for which it has not been trained, then this would consequently lead to noise in the gating mechanism which also leads to a more random decision made by the routing algorithm as is defined in section 4.3. The routing should therefore be more unbiased when encountering unexpected information.

4.5 Directed NARU Networks

The concrete architecture tested in this thesis is a composition of NARUs forming a graph in which activity is being routed directly towards the end of the network. However the activation paths will always have the same length, which is equals to the depth of the network. Conversely, allowing activity to be routed in cycles would cause the execution time and cost of a single forward pass to be fluctuating and most likely also to be unpredictable.

Due to the gating mechanism of a single unit, this architecture is conceptually a Recurrent Neural Network in which information can flow in cycles. This is evident by equation 4 as well as the routing mechanism which consists of connections directed towards the flow of information within a network of NARUs.

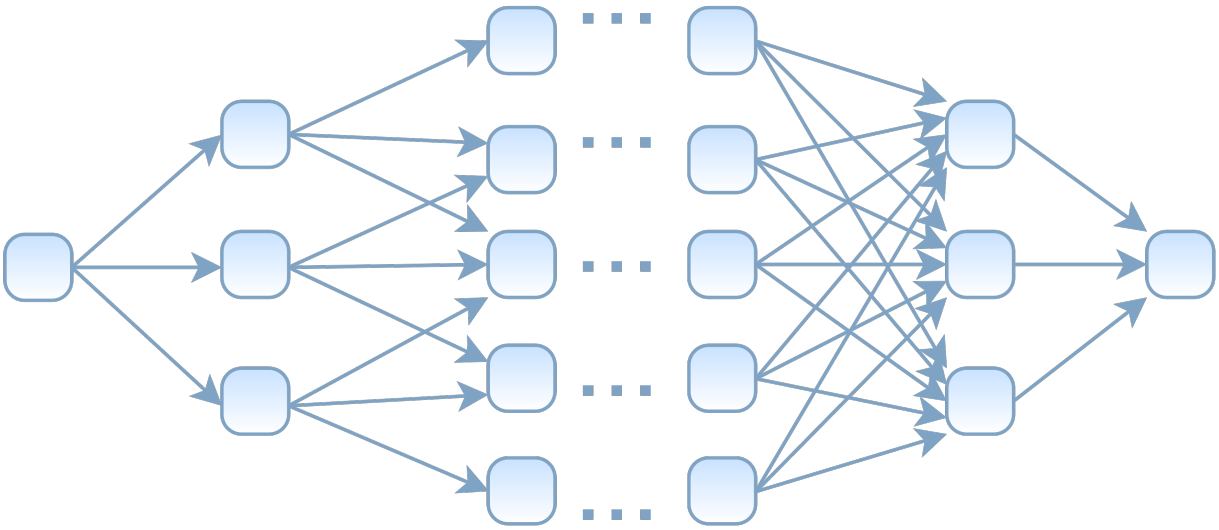

Figure 15: A Directed Network of NARUs

The vertically stacked bundles illustrated in Figure 15 will be referred to as *capsules* similar as in the stacked layers described in ""Dynamic Routing Between Capsules"" [35]. Every bundle within a capsule will be structurally identical, however bundles not within the same capsule might have different input and output dimensions. The degree of interconnectedness between capsules will be limited by a *maximum cone size*. A single bundle cannot have more connections than this threshold. The reason for this sparsity is to save computational cost. The cone size does not need to be larger than the size growth (number of bundles) between two consecutive capsules in order to make it possible for an input signal to be able to visit any bundle in the entire network.

4.5.1 Model Specification

The model consists of a total of *7 capsules*, each of which contains a number of bundles ranging *from 1 to 18*. These capsules are connected with a *maximum cone size of 6*.

Capsule	1	2	3	4	5	6	7
Number of bundles	1	6	12	18	12	6	1
Bundle dimensions	50	76	102	128	102	76	50
Routes per bundle	6	6	6	6	6	1	0
Sources per bundle	0	1	3	4	9	12	6

Table 2: Structural Summary of the Tested Model

This model receives inputs in the form of tokens encoded into vectors matching the input dimensionality of 50.

This embedding has been based on the technique outlined in [33]. The embedding used for this experiment was [1], a pre-trained mapping between a total of 400 thousand tokens alongside corresponding vectors of size 50.

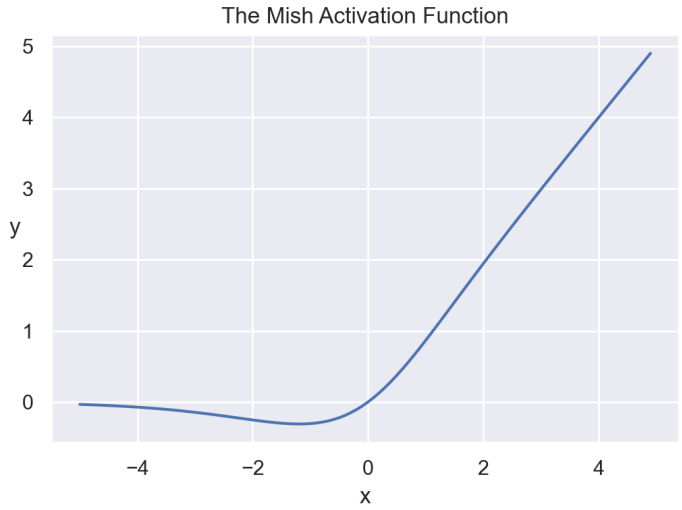

Figure 16: The used activation function ***a()***

The activation function used for the implementation of ***a()***, as is defined in equation 5, is the so called *Mish activation function*, which is defined as:

$$f(x) = x * tanh(softplus(x)) \tag{8}$$

The usage of this function is based on the findings reported in “Mish: A Self Regularized Non-Monotonic Activation Function” [27], which show improved convergence for very deep models with a large number layers. Although the model tested in this thesis does not have many layers, it still propagates information recurrently through time.

The source code for the implementation of this concrete model specification alongside the test setup can be viewed at [28].

4.6 Results

The model specified in subsection 4.5.1 was trained across *200 epochs* using a data set containing *1500 sentences with an average length of 25 tokens*. Due to the high degree of branching of this architecture, the use of batches in the form of single tensors containing multiple samples was not possible. Instead *small batches containing 10 sample* were fed into the network sequentially, for every epoch. After which the accumulated gradients were divided by the batch size to form an average gradient which was then applied to the weights.

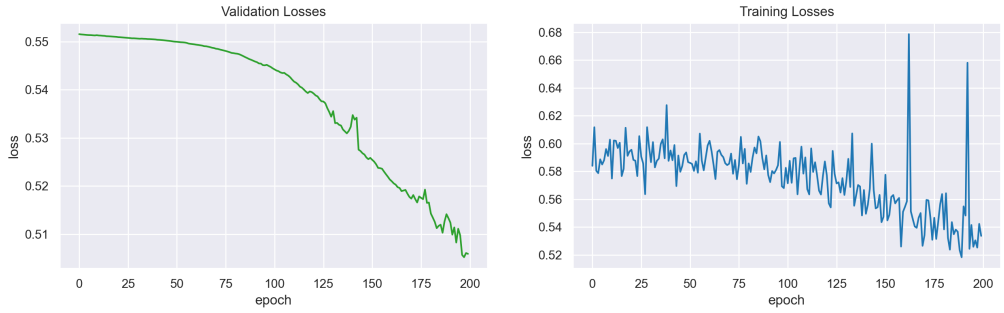

Figure 17: Validation and Training loss for a directed NARU network

As can be observed in Figure 17, the model converged reliably and the validation loss also decreased throughout the 200 epochs. However the rate of convergence was very slow and noisy. Due to the high degree of branching the use of accelerator hardware did not yield a speed-up in training time. Therefore longer training sessions were not feasible as the already performed training was incredibly time consuming.

The routing behavior of the network has been measured throughout the training session by recording the indices of the active bundles for every capsule within the network. This information has been used to raise a variety of metrics, namely *activity saturation*, *route reorganisation* and *routing bias*. These metrics will be displayed and discussed for further examination in the following paragraphs.

In order to validate the claim that the network would only use a subset of parameters to perform a prediction, the degree of *cumulative total network utilisation* was raised from the taken measurements for every sentence in every epoch.

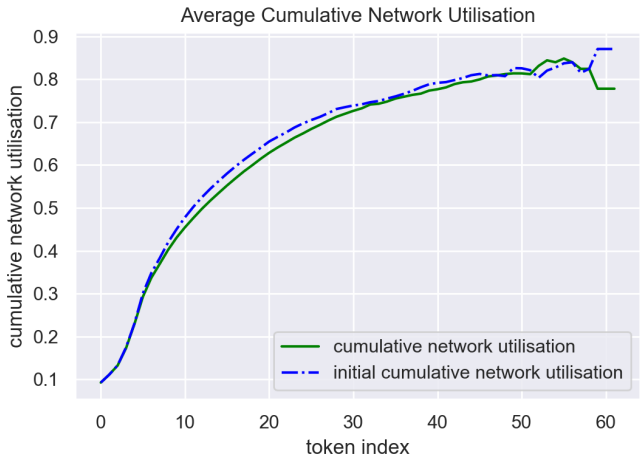

Figure 18: The Cumulative Network Utilisation

Figure 18 shows the average network utilisation across all epochs. The x axis represents both the token index of a given sentence as well as the time step.

Besides to total average across all epochs, Figure 18 also displays the initial cumulative activity saturation of the network in order to indicate how this metric has changed during training.

The rate of saturation slightly decreased over time which might indicate that the architecture has learned to use its bundles selectively to predict tokens within a given sentence.

Figure 19 shows the number of route changes across all sentences for every epoch. Route changes were measured for every sentence within the training data. A route change is always based on a recording of the previous route generated by a given sentence. Conceptually, this metric can also be understood as a degree of internal reconfiguration. If the weights are not updated then no changes will occur when a sentence is reintroduced to the network. This is simply because it will deterministically produce the exact same predictions as previously. This might change however upon updating the weights of the network. Because this metric contains high levels of noise an additional moving average was introduced which ought to enable the ability to identify a potential trend.

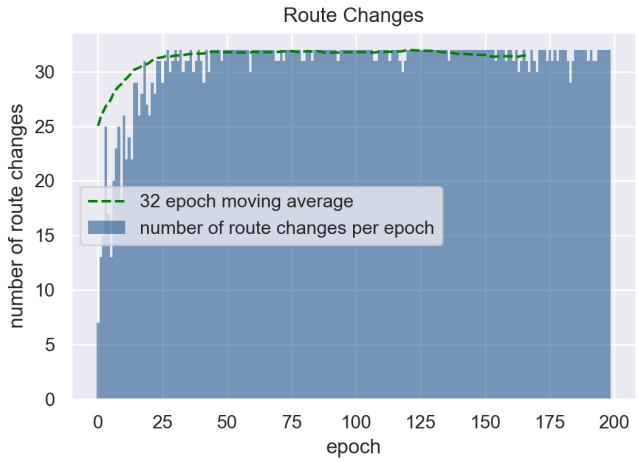

Figure 19: Frequency of route changes during training

As has already been discussed in subsection 3.3.2, a major challenge of CDLN has been the emergence of routing bias during training.

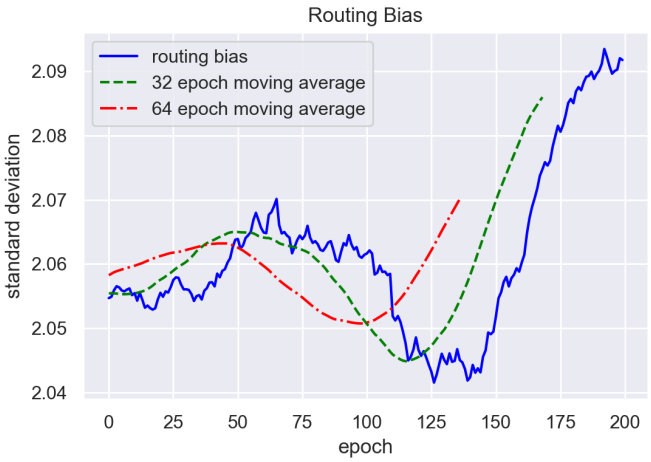

Figure 20: Change in Routing Bias during Training (accumulated deviation from expected average)

In order to detect the emergence of said bias, the standard deviation of the distribution of routes was calculated for all epochs during training. This metric can be observed in Figure 20, which shows how the standard deviation changes throughout the entire training session.

Two additional moving averages were introduced in order to help identifying potential trends. However there seems to be no statistically significant trend which might indicate that a routing bias exists or would ever emerge.

5 Conclusion

Based on the categorization of ANN architectures established in section 2.4 chapter 2, we suggest that many design approaches for state of the art ANN architectures are very dissimilar to BNNs in potentially unfavorable ways. The most relevant of which would be sparsity, which means that large parts of a neural network stay either inactive for a given period of time, or fire in a less frequent manner. This property observable in BNNs allows them to utilize neurons selectively. It has been theorized that this is due to reasons related to energy efficiency as well as memory locality, where neurons encode information that might not be needed for longer periods of time [8]. A novel type of architecture tries to overcome these challenges at least partially. So called “Conditional Neural Networks” or “Conditional Deep Learning Networks” have been successfully trained to route information based on various gating/routing mechanisms, improving both energy efficiency as well as accuracy compared to state of the art architectures [30][31][26]. These types of networks have also yielded promising results for transfer and online learning [23] as well as reinforcement learning [11]. However this approach also faces major challenges, namely the emergence of routing bias and the difficulty of applying batches during training.

Literature

Books

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org, ISBN: 978-0-262-03561-3.

[29] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2018, ISBN: 978-3-0300-6856-1. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[32] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. Beijing: O'Reilly, 2017, ISBN: 978-1-4919-1425-0. [Online]. Available: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/.

Journal Articles

[2] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task aware continual learning," p. 10, 2020. DOI: 10.1016/S0079-7421(08)60536-8. [Online]. Available: https://www.researchgate.net/publication/343461550_Conditional_Channel_Gated_Networks_for_Task-Aware_Continual_Learning.

[4] A. L. Barth and J. F. Poulet, "Experimental evidence for sparse firing in the neocortex," *Trends in Neurosciences*, p. 11, 2012. DOI: 10.1016/j.tins.2012.03.008. [Online]. Available: https://www.cell.com/trends/neurosciences/fulltext/S0166-2236(12)00051-3.

[5] I. Basheer and M. N. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application.," *Journal of Microbiological Methods*, p. 31, 2000. DOI: 10.1016/S0167-7012(00)00201-3. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/11084225/.

[6] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," 2016. arXiv: 1511.06297 [cs.LG].

[7] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," p. 30, 2014. arXiv: 1403.2048 [cs.ET]. [Online]. Available: https://arxiv.org/abs/1403.2048.

[9] A. Davis and I. Arel, "Low-rank approximations for conditional feedforward computation in deep neural networks," 2014. arXiv: 1312.4461 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1312.4461v4.

[10] D. Eigen, M. Ranzato, and I. Sutskever, “Learning factored representations in a deep mixture of experts,” 2014. arXiv: 1312.4314 [cs.LG].

[11] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, “Pathnet: Evolution channels gradient descent in super neural networks,” *CoRR*, vol. abs/1701.08734, 2017. arXiv: 1701.08734. [Online]. Available: http://arxiv.org/abs/1701.08734.

[12] R. M. French, “Catastrophic forgetting in connectionist networks,” *Psychology of Learning and Motivation*, p. 18, 1999. DOI: 10.1016/S0079-7421(08)60536-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[14] K. He, Xiangyu, S. Ren, and J. Sun, “Deep residual learning for image recognition,” p. 0, 2015. DOI: 1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[15] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, “Decoupled neural interfaces using synthetic gradients,” p. 0, 2017. DOI: 1608.05343. [Online]. Available: https://arxiv.org/abs/1608.0534.

[17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *CoRR*, vol. abs/1609.04836, 2016. arXiv: 1609.04836. [Online]. Available: http://arxiv.org/abs/1609.04836.

[18] T. C. Kietzmann, C. J. Spoerer, L. K. A. Sörensen, R. M. Cichy, O. Hauk, and N. Kriegeskorte, “Recurrence is required to capture the representational dynamics of the human visual system,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 43, 21854–21863, 2019, ISSN: 1091-6490. DOI: 10.1073/pnas.1905544116. [Online]. Available: http://dx.doi.org/10.1073/pnas.1905544116.

[19] K. Kohitij, K. Jonas, S. Kailyn, E. B. Issa, and J. J. DiCarlo, “Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior,” *bioRxiv*, 2018. DOI: 10.1101/354753. eprint: https://www.biorxiv.org/content/early/2018/06/26/354753.full.pdf. [Online]. Available: https://www.biorxiv.org/content/early/2018/06/26/354753.

[20] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, “An introduction to neural networks,” p. 0, 1993. DOI: 10.1.1.18.493. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.493.

[22] P. Lennie, “The cost of cortical computation,” *Current Biology*, p. 7, 2012. DOI: 10.1016/s0960-9822(03)00135-0. [Online]. Available: https://www.cell.com/current-biology/fulltext/S0960-9822(03)00135-0.

[23] H. Li, P. Barnaghi, S. Enshaeifar, and F. Ganz, “Continual learning using multi-view task conditional neural networks,” 2020. arXiv: 2005.05080 [cs.LG].

[24] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” p. 38, 2015. DOI: https://arxiv.org/abs/1506.00019. [Online]. Available: https://arxiv.org/abs/1506.00019.

[25] W. Liua, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications.,” *Elsevier*, p. 26, 2017.

[26] M. McGill and P. Perona, “Dynamic routing in artificial neural networks,” p. 10, 2017.

[27] D. Misra, “Mish: A self regularized non-monotonic activation function,” 2020. arXiv: 1908.08681 [cs.LG].

[30] P. Panda, A. Sengupta, K. R. S. of Electrical, and C. Engineering, “Conditional deep learning for energy-efficient and enhanced pattern recognition,” p. 0, 2015. DOI: 10.3850/9783981537079_0819. [Online]. Available: https://academic.microsoft.com/paper/2267244539/reference/search?q=Conditional%20Deep%20Learning%20for%20energy-efficient%20and%20enhanced%20pattern%20recognition.

[31] P. Panda, A. Sengupta, and K. Roy, “Energy-efficient and improved image recognition with conditional deep learning,” p. 0, 2017. DOI: 10.1145/3007192. [Online]. Available: https://www.researchgate.net/publication/313594599_Energy-Efficient_and_Improved_Image_Recognition_with_Conditional_Deep_Learning.

[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-propagating Errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0. [Online]. Available: http://www.nature.com/articles/323533a0.

[35] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” 2017. arXiv: 1710.09829 [cs.CV].

[36] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, “Online deep learning: Learning deep neural networks on the fly,” *CoRR*, vol. abs/1711.03705, 2017. arXiv: 1711.03705. [Online]. Available: http://arxiv.org/abs/1711.03705.

[37] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” p. 19, 2017. arXiv: 1701.06538 [cs.LG].

Incollections

[21] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop.,” in *Neural Networks: Tricks of the Trade (2nd ed.)* Ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., vol. 7700, Springer, 2012, pp. 9–48, ISBN: 978-3-642-35288-1. [Online]. Available: http://dblp.uni-trier.de/db/series/lncs/lncs7700.html#LeCunBOM12.

Web Sources

[1] [Online]. Available: https://nlp.stanford.edu/projects/glove/ (visited on 05/16/2021).

[3] Y. Andre, *11 essential neural network architectures, visualized and explained*, 2020. [Online]. Available: https://medium.com/analytics-vidhya/11-essential-neural-network-architectures-visualized-explained-7fc7da3486d8.

[8] K. Clancy and M. Muun, *Here's why your brain seems mostly dormant*, 2015. [Online]. Available: http://nautil.us/issue/27/dark-matter/heres-why-your-brain-seems-mostly-dormant.

[16] L. James, *The 10 neural network architectures machine learning researchers need to learn*, 2018. [Online]. Available: https://medium.com/cracking-the-data-science-interview/a-gentle-introduction-to-neural-networks-for-machine-learning-d5f3f8987786.

[28] D. Nepp, *Directed-naru model implementation and experiments source code*. [Online]. Available: https://github.com/Gleethos/NARU.

[33] M. Pellarolo, *How to use pre-trained word embeddings in pytorch*, 2018. [Online]. Available: https://medium.com/@martinpella/how-to-use-pre-trained-word-embeddings-in-pytorch-71ca59249f76 (visited on 05/16/2021).

List of Figures

List of Tables

Acronyms

ANN Artificial Neural Network

CDL Conditional Deep Learning

CLNN Conditional Neural Network

CDLN Conditional Deep Learning Network

NLP Natural Language Processing

CV Computer Vision

RNN Recurrent Neural Network

GRU Gated Recurrent Unit

LSTM Long-Short Term Memory

GAN Generative Adversarial Network

CNN Convolutional Neural Network

ESN Echo State Network

AE Auto Encoder Network

VAE Variational Auto Encoder Network

NARU Neural Activity Routing Unit

SIMD Single Instruction Multiple Data

AGI Artificial General Intelligence

RAM Random Access Memory

RL Reinforcement Learning

TL Transfer Learning

BNN Biological Neural Network