Zaim ul-Abrar Imran
Max Torre Schau

Performance Predictors for Neural Architecture Search used with Graph Convolutional Networks

Specialization project in Computer Science

Supervisor: Heri Ramampiaro
Co-supervisor: Espen A. F. Ihlen & Felix Tempel
December 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Abstract

Neural architecture search (NAS) is a field within automated machine learning which seeks to automate the process of finding suitable architectures for a given problem. NAS does, however, suffer from requiring high computational power to validate the accuracy of a proposed architecture. Therefore, different performance predictors are proposed to speed up this process.

The study extensively researches neural architecture search and why it is useful. In addition, the study shows what zero-cost proxies are and how they can improve neural architecture search. Lastly, we explored the use of zero-cost proxies in the context of graph convolutional network (GCN) for neural architecture search using the InMotion dataset for cerebral palsy (CP) prediction in infants.

The study found that zero-cost proxies can improve the performance of NAS and give promising results. The thesis gives us a good overview of the subject and experience which will be helpful for our master thesis. s

Sammendrag

NAS er et felt innen automatisert maskinlæring hvor man ønsker å automatisere prosessen av å finne egnede arkitekturer for et gitt problem. NAS lider imidlertid av å kreve høy beregningskraft for å validere hvor god en arkitektur er. Som et resultat av dette har det blitt foreslått en rekke ytelsespredikatorer for å effektivisere denne prosessen.

Studien undersøker grundig hva neural architecture search er, og hvorfor dette er nyttig. I tillegg viser studien hva zero-cost proxies er og hvordan de kan forbedre neural architecture search. Til slutt uforsket vi bruken av zero-cost proxies med GCN for neural architecture search ved bruk av InMotion-datasettet for prediksjon av cerebral parese (CP) hos spedbarn.

Studien fant at zero-cost proxies kan forbedre ytelsen til NAS, og at det generelt gir lovende resultater. Oppgaven gir oss en god oversikt over emnet, og erfaring som vil være nyttig for vår masteroppgave.

Contents

Figures

Tables

Acronyms

AGNN automated graph neural networks. 14

AUC area under the receiver operating characteristic curve. 20

AutoML automated Machine Learning. 3

BO bayesian optimisation. 7

CNN convolutional neural network. 1

CP cerebral palsy. 1

DAG directed acyclic graphs. 5

EC evolutionary computations. 7

EcoNAS economical evolutionary-based NAS. 13

GCN graph convolutional network. iii, v, 1

GNN graph neural networks. 13

GPU graphics processing unit. 4

GRASP Gradient Signal Preservation. 15

NAS neural architecture search. iii, v, 1

SNIP Single-shot Network Pruning. 15

synflow Iterative Synaptic Flow Pruning. 15

Chapter 1

Introduction

1.1 Background and Motivation

DeepInMotion is a collaboration project between NTNU, St Olavs Hospital and the hospital of Ålesund, aiming to develop artificial intelligence technology capable of detecting cerebral palsy (CP) at an early stage. At the very basic, the pipeline uses a convolutional neural network (CNN) to extract the movement with high precision from 2D images or videos (Groos et al. 2021). The result will then be processed by a GCN to predict CP for high-risk infants at three months of age (Groos 2022).

DeepInMotion used NAS to find an appropriate architecture for the given problem. NAS aims to find a suitable architecture automatically, replacing the manual process used for the last couple of years. This process can be exceedingly time-consuming and energy-intensive because it often requires training the candidate architecture fully to validate its performance. Zero-cost proxies predict the performance of a neural architecture without having to train it fully.

(Abdelfattah et al. 2021) did more prominent research on zero-cost proxies on regular CNNs and showed why such proxies are improving the efficiency of NAS algorithms. However, there has been no research on using zero-cost proxies with GCN.

We hypothesise that zero-cost proxies will significantly influence the efficiency of NAS algorithms with GCN and will investigate this further in this thesis.

1.2 Goals and Research Questions

The main goal of the thesis is to discover a performance predictor technique that may yield improvement in neural architecture search for graph convolutional network used within the InMotion dataset for CP-prediction for infants.

Main research question How can we obtain a computer-efficient neural architecture search for graph convolutional network to be used in infant movement analysis?

Research question 1 What is neural architecture search, and why is it useful?

Understanding the subject and why it is useful allows us to dive deeper into more specific subject domains.

Research question 2 How can zero-cost proxies improve neural architecture search, and what techniques are leading in the field today?

By deep diving into zero-cost proxies and how they work, we can implement them for the dataset, as well as get a good understanding of why they might yield to good results.

Research question 3 How will zero-cost proxies perform when used with the InMotion dataset for infant CP prediction?

We will investigate how different zero-cost proxies perform when used on the InMotion dataset to see if they yield improvements for neural architecture search with graph convolutional networks. By improvements, we mean that it will be far more efficient regarding runtime performance and how the zero-cost proxies correlate with the validation accuracy.

1.3 Structure

The thesis is structured as follows:

Chapter 1 - Introduction describes the background and motivation for writing the thesis. This chapter will describe the thesis's main goal and research questions.

Chapter 2 - Theory will present the essential literature for the thesis.

Chapter 3 - Related work will outline relevant research, such as using zero-cost proxies on CNN.

Chapter 4 - Experiment utilises the theory from chapter 2 and related work from chapter 3 to present an experimental plan for the thesis.

Chapter 5 - Results presents and evaluates the results from the experiments in Chapter 4.

Chapter 6 - Discussion will evaluate the results and discuss them in the context of the thesis' goal.

Chapter 7 - Conclusion presents our conclusion, as well as proposed future work.

Chapter 2

Theory

The theory chapter of this thesis provides a detailed overview of the fundamental concepts and principles underlying the research presented in the rest of the thesis. This includes an in-depth discussion of the relevant literature on neural architecture search, zero-cost proxies, and graph convolutional networks.

2.1 Deep Learning

Deep learning is a technique within machine learning which aims at teaching computers to learn by examples, similar to how humans learn. By processing multiple layers, computers can understand how data is represented with various levels of abstraction (LeCun et al. 2015).

2.2 AutoML

Machine learning has significantly boosted research and applications in the last decade. Fields such as computer vision, speech recognition and many others have seen significant advances due to the development of deep learning. Automated Machine Learning (AutoML) aims at automating the process of designing great architectures, which can be very tedious and error-prone for data scientists. AutoML constantly changes and includes hyperparameter optimisation, meta-learning and NAS (Hutter et al. 2019).

2.3 Neural Architecture Search

Deep learning has become increasingly popular in recent years due to its ability to solve advanced problems like speech recognition, visual object recognition, object detection and many others (LeCun et al. 2015). Most architectures are created by specialists, which is labour-intensive and susceptible to weaknesses or errors. Subsequently, a way of automatically designing and developing such algorithms has been a research field for a couple of years. Thus, NAS aims to automate the

previously manual process of designing architectures (Elsken et al. 2019). Consequently, NAS is a sub-field of AutoML (2.2).

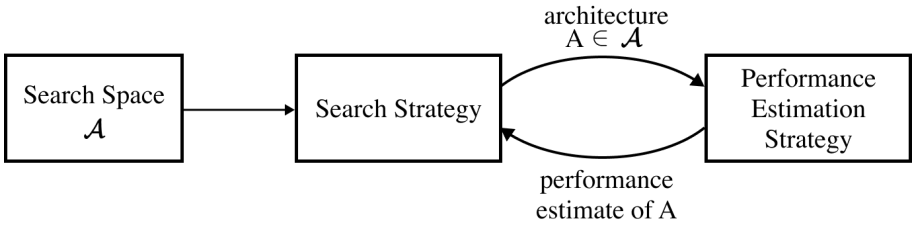


Figure 2.1: An overview of the different methods in NAS (Elsken et al. 2019)

Figure 2.1 shows how NAS works. The search space gives the algorithm a constraint regarding how it can be developed by defining a set of architectural choices the model might use. The constraint might be different operations such as convolution, fully connected and pooling. One might argue that this is the vital part as selecting the search space can reduce the search's complexity, which is vital to produce an acceptable model (Kyriakides and Margaritis 2020).

After defining a search space for the given problem, a NAS search algorithm will specify how to analyse the search space and propose a set of candidate architectures. This introduces the exploration-exploitation trade-off, which indicates that selecting an appropriate optimisation technique is vital because we want to find a global optimum and ensure that the search space is sufficiently investigated (Kyriakides and Margaritis 2020).

The framework must perform performance estimation for each candidate architecture to adjust the search strategy. The simplest solution is to train and validate the model, which is too computationally expensive because it usually involves days of training for graphics processing unit (GPU). Many hours of training require a lot of energy, which has a high environmental cost. Another downside of training the architectures is that it will limit the number of architectures that the search algorithm might discover. As a result, methods simplifying this phase have been undergoing research heavily (Elsken et al. 2019).

2.3.1 Challenges

Computational power

The most straightforward approach to determine the performance of a neural network is to train it until the validation accuracy has converged against a value or has been run for a fixed amount of epochs. However, training thousands of architectures may require hundreds or more GPU days (Ren et al. 2021). The computational power required may be available for larger companies with plentiful resources. However, for most users, this is computationally infeasible. As a result, the necessary computational power is considered a significant challenge for NAS.

Black-box optimization and lack of interpretability

NAS algorithms often treat the architecture search space as a black box, meaning they do not have access to the internal workings of the model architectures being searched. This can make it difficult to incorporate domain knowledge or bias the search towards certain types of architectures. This results in difficult-to-interpret architectures, making it challenging to understand how they make predictions or identify potential problems with the architecture. (H. Liu et al. 2018)

2.3.2 Search Space

When searching for a high-performing architecture, there are infinite variations that one might investigate. As a result, we define a search space that gives the search algorithm constraints regarding what kind of combinations it might examine. Prior knowledge of what kind of search space is effective on specific tasks may reduce the size, but it has the advantage of introducing human bias in the search space (Elsken et al. 2019).

Global search space is a way in which it tries to combine all possible operations to create chain-structured (sequential) networks. Then, the search space has the following parameters:

- the number of layers
- the type of each operation
- the hyperparameters of each operation, namely kernel size, number of filters etc.

Such a network can be described as a series of n layers, where layer L_i is taking L_{i-1} as input. However, this sort of search space is immense and very expensive.

Cell-based representations were inspired by successful architectures using repeated modules (Inception, ResNet). NASNet paper (Zoph et al. 2017), is one of the most popular cell- or block-based approaches. Cell-based representations differ from global search space because they search for cells or blocks instead of whole architectures (Elsken et al. 2019). Zoph et al. 2017 learns two sorts of cells; a normal cell which performs feature extraction (preserving dimensionality), and a reduction cell which reduces the dimensionality. By stacking such cells, we get the final architecture, as shown in figure 2.2.

Directed Acyclic Graphs

Directed acyclic graphs (DAG) is often used in NAS to represent the structure of a neural network. In this context, the graph's vertices represent different operations or layers in the network, and the graph's edges represent the data flow between these operations. This allows researchers to efficiently represent and manipulate the structure of a neural network during the search process.

One of the main advantages of using DAGs in NAS is that they provide a convenient way to encode the constraints on the network's architecture. For example,

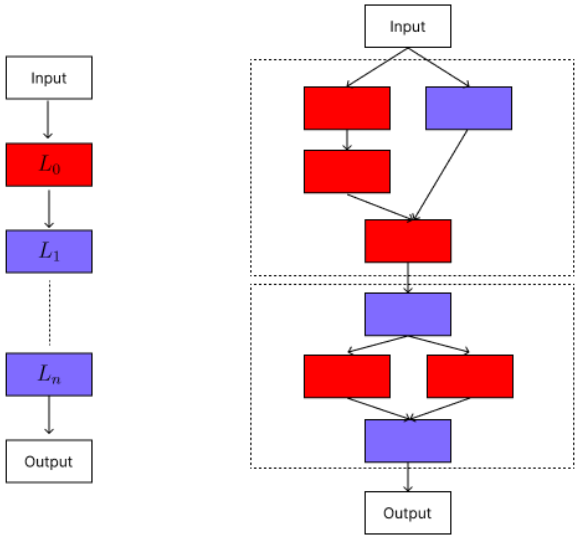


Figure 2.2: Left: chain-structured network, right: cells combined into an architecture

a DAG can enforce the requirement that a neural network must have a certain number of layers or that individual layers must be connected in a specific way. This can help to ensure that the search process only considers valid network architectures, which can speed up the search and improve its accuracy (H. Liu et al. 2018).

In addition to representing the structure of a neural network, DAGs can also be used to represent the search space over which the architecture search algorithm operates. This allows the algorithm to explore different network architectures efficiently and evaluate their performance, ultimately leading to the discovery of novel and effective network architectures. (Dong and Y. Yang 2019)

2.3.3 Search Strategies

Random Search

Random search is the most naive search strategy, and it will simply randomly pick a valid architecture based on the search space. The method does not require any learning model and is, therefore, quite fast. A random search may be effective if a search space is well constructed.

Reinforcement Learning

On the very basic, reinforcement learning is an area within machine learning in which an agent learns behaviour by some trial-and-error interaction with a dynamic environment (Kaelbling et al. 1996). We can consider NAS a reinforcement problem by looking at the creation of the architecture as the agent's action, in

which the action space is the problem's search space. The agent gets rewarded depending on the performance of the trained architecture. There are numerous approaches to representing an agent's policy, such as a recurrent neural network, proximal policy optimisation and q-learning (Elsken et al. 2019).

Evolutionary algorithms

Evolutionary algorithms are techniques used in optimisation and search methods. It is a subset of evolutionary computations (EC) and is an effective way of problem-solving for often encountered global optimisation problems because of its adaptable character (Vikhar 2016). Evolutionary algorithms in NAS work by selecting N initialised models randomly and then assessing performance by the given evaluation strategy. The best models are chosen as parents, and new models are mutated clones of the parents, which are re-evaluated again. The worst N models are removed from the population to make room for new offspring (Real et al. 2017).

Bayesian optimisation

Bayesian optimisation (BO) has been a popular approach for hyperparameter optimisation (Elsken et al. 2019). In general, BO optimises expensive functions to evaluate, such as the performance of architectures. This is a global optimisation problem that BO tries to solve. Given a costly to evaluate function f, BO aims at finding its optimal score within some domain x (Kandasamy et al. 2018). In other words, BO pursues to calculate $a^* = argmin_{a \in A} f(a)$, where A is the given search space, and $f(a)$ is the performance function of the neural network after training the architecture a for a fixed number of iterations (White, Neiswanger et al. 2021).

2.3.4 Performance Estimation

Performance Predictors

As mentioned in 2.3, NAS aims to automate the process of designing high-performing neural networks. However, this often requires training all the candidate networks, either partly or wholly, to get the accuracy of the neural networks. In most cases, this is an infeasible approach when the search space is ample, which is why performance predictors are introduced (Akhauri et al. 2022).

Any function that forecasts the eventual accuracy or ranking of architectures without fully training the architecture is referred to as a performance predictor (f). The function of the performance predictor should therefore take significantly less time than training and validating the neural network fully and have a high correlation or rank correlation with the validation error (White, Zela et al. 2021).

A performance predictor is defined by two main routines - initialisation and query. The initialisation routine does some general pre-computation, often before

the NAS algorithm. For model-based methods, the initialisation routine consists of fully training a set of architectures to get data points. The query routine will output the predicted accuracy with the architecture details as input (White, Zela et al. 2021).

There exist multiple categories of performance predictors:

Table 2.1: List of performance predictors

Model-based (trainable) methods
Learning curve-based methods
Hybrid methods
Zero-cost proxies
Weight sharing methods

Zero-Cost Proxies

Zero-cost proxies are a class of performance predictors. The name zero-cost comes from analysing a neural network at initialisation, indicating that it costs 'zero' to generate the score. At the same time, the predicted accuracy is closely correlated with the actual accuracy.

By performing a single forward/backward propagation pass using a single minibatch of data, zero-cost proxies methods can score a neural network (Akhauri et al. 2022). The intuition is that we can measure the 'trainability' of a neural network by looking at the initial gradient flow.

The paper *Zero-Cost Proxies for Lightweigt NAS*, (Abdelfattah et al. 2021) showed the usefulness of a range of zero-cost proxies, inspired by the pruning-at-initialisation literature. Each of the methods can be divided into one of two categories - data-independent or data-dependent.

Data-dependent zero-cost proxies use data to generate the score, whereas data-independent will not use the dataset in principle. Sometimes, it is used to set dimensions (White, Khodak et al. 2022).

Table 2.2: Different zero-cost proxies within the two categories data-independent and data-dependent

Data-independent	**Data-dependent**
Synflow	EPE-NAS
Zen-score	Fisher
GenNAS	Grad-norm
number of parameters in network	Grasp
...	...

2.4 Graph Convolutional Network

CNNs have shown excellent performance and result within various domains in the last couple of years, such as object detection and speech recognition (Albawi et al. 2017). CNNs operate on euclidean data structures, such as images. This is not the case for graphs, which are considered non-euclidean data because of their non-fixed structure. As a result, standard CNN operations do not necessarily effortlessly operate similarly on graphs. To solve this problem, GCNs were introduced. They utilize the graph's structure in such a way that a model can learn the different features by investigating the neighbourhood of the nodes (S. Zhang et al. 2019)

A graph can be defined as $G = (V, A)$ in which V represents the vertices/nodes of the graph $\{v_1, ..., v_n\}$. $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix representing the edges between the nodes (Wu et al. 2019). The cell's value depends on whether an edge exists between the corresponding node. For a node a_{ji}, if $(V_i, V_j) \in E$ (E is the set of all edges), the value is 1, otherwise 0. Figure 2.3 and 2.4 shows an example of this.

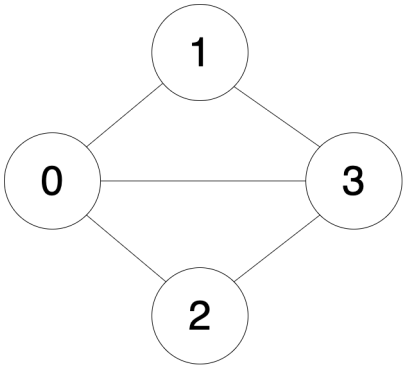


Figure 2.3: Graph G

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Figure 2.4: Adjacency matrix A

Like convolution in CNN, GCN produces a feature map based on the adjacency matrix to capture the relationship between the graph nodes. The convolution operation can be summarised in equation 2.1 (Yan et al. 2018) (Kipf and Welling 2016):

$$f_{out} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{f}_{in}\mathbf{W} \tag{2.1}$$

where $\tilde{D}$ is the degree matrix obtained by $\sum_j(A_{ij}+I_{ij})$, W are the weights, I the identity matrix, and f_{in} is the output of the previous layer. For instance, the vertex feature matrix (position, velocity, acceleration etc.) at the start of the layer. The result has to go through an activation layer such as ReLU(x) $= max(0,x)$ before being fed to the next layer.

For distance partitioning and spatial partitioning, equation 2.1 is used. However, the adjacency matrix is divided into numerous matrices, A_j where $A+I = \sum_j A_j$. With distance partitioning, K defines how many hops away one will go from the node. This is shown in figure 2.5.

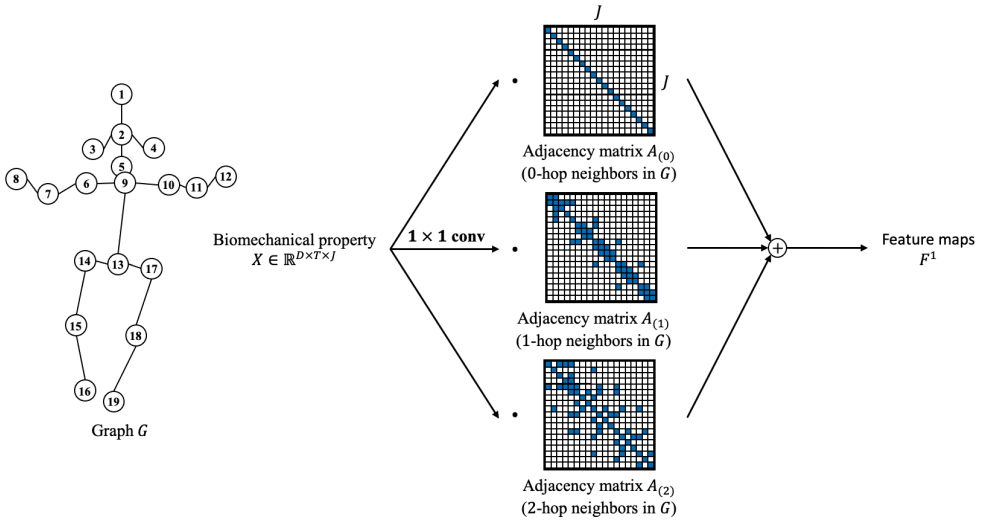


Figure 2.5: Graph convolution displaying 0-hop, 1-hop and 2-hop (Groos 2022)

Chapter 3

Related work

3.1 EcoNAS

In this paper, the authors emphasize that using proxies within a NAS-search strategy is inevitable due to the high computational cost of training and validating architectures fully. They look at how different reduction factors behave within CIFAR-10 (Krizhevsky 2009) and ImageNet (Russakovsky et al. 2014). Four common factors are mentioned:

- *(c)* the number of channels for CNNs
- *(r)* the resolution of input images
- *(e)* the number of training epochs
- *(s)* the sample ratio of the full training set

After the experiments, the authors make two main observations of the proxies (D. Zhou et al. 2020):

1) As long as the iteration numbers stay the same, using more training samples with fewer epochs is more reliable than using more epochs and training samples.

2) It can be reasonable to reduce the resolution of the input images. At the same time, reducing the network channels is more reliable than reducing the resolution.

Economical evolutionary-based NAS (EcoNAS) was developed based on the mentioned observations. EcoNAS is a faster approach for evolutionary-based NAS algorithms (D. Zhou et al. 2020). A new proxy was designed to take advantage of the observations made earlier and to reduce the search cost significantly. They also suggested a hierarchical proxy technique for training neural networks with various proxies based on their individual accuracy. We refer to the article for a more detailed explanation.

3.2 GraphNAS

As graph neural networks (GNN) are emerging as a tool to analyze non-euclidean data, the design of GNNs requires heavy manual work and domain knowledge.

Therefore it is crucial to automate this process with reinforcement learning to find the best design for any given problem. GraphNas uses a recurrent network, references as a controller in the paper, to generate architecture descriptions based on GNN layers. (Gao, H. Yang et al. 2020)

The author reused the search space in CNN NAS and added some operations used in state-of-the-art GNN networks. The operators that the author has decided to add to the search space are:

- Neighbor sampling operator; selects the receptive field $N(v)$ for a target node v.
- Message computation operator; computes the feature information for each node u in the receptive field $N(v)$.
- Message aggregation operator; aggregates information from the receptive field $N(v)$, which is similar to the pooling operators in CNNs.
- Multi-head and readout operator; used to stabilize the learning process of message computation operators

The author experimented with real-world datasets and showed that GraphNAS could generate a network that challenges the best networks created by humans in terms of validation set accuracy (Gao, H. Yang et al. 2020).

3.3 Auto-GNN

The authors propose the automated graph neural networks (AGNN) framework in this paper. AGNN uses a developed searching algorithm called reinforced conservative NAS. The algorithm consists of three components: A conservative explorer that filters out the best architectures yet discovered. An architecture modifier which barely modifies some operations in the retrained best architecture. An architecture modification causality trainer who uses reinforcement learning. In addition, AGNN uses a novel weight-sharing strategy in which similar architectures can share the weight. Thus, training each architecture from scratch will not be necessary.

The discovered architectures from the algorithm achieved better performance compared to the existing human-crafted models and traditional search methods (K. Zhou et al. 2019).

3.4 Zero-Cost Proxies for Lightweight NAS

Zero-cost proxies have been developed and used with CNNs. Abdelfattah et al. 2021 did an extensive survey on various zero-cost-proxies inspired by the pruning-at-initialization literature. The study experimented with zero-cost proxies on five benchmarks; NAS-Bench-101/201/NLP/ASR and PyTorchCV (Abdelfattah et al. 2021).

3.4.1 Grad Norm

Grad Norm is a simple zero-cost proxy that sums the euclidean norm of the gradients after performing a single forward- and backward-pass of a minibatch of training data (Abdelfattah et al. 2021). Given a vector of gradients $G = [g_1, g_2, ..., g_n]$, the grad norm is calculated as displayed in 3.1

$$grad\_norm = \sum_{i}^{G} \sqrt{g_i^2} \tag{3.1}$$

3.4.2 SNIP

Single-shot Network Pruning (SNIP) was introduced in (Lee et al. 2018) as a pruning-at-initialisation technique, aiming at reducing the number of parameters within a neural network without affecting the accuracy at inference-time (Frankle et al. 2020). In the original paper - the authors introduced a saliency criterion that, in a data-dependent manner, identifies connections in the network that are significant to the current task before training. Thus, redundant connections could be pruned, making the neural network smaller in terms of parameters. The connections are identified based on their influence on the loss function.
The technique was later used as a zero-cost proxy in the paper Zero-Cost Proxies for Lightweight NAS (Abdelfattah et al. 2021). In this paper, the score of the neural network was obtained by summing all parameters N in the model (3.2),

$$S_n = \sum_{i}^{N} S_p(\theta)_i \tag{3.2}$$

$$S_p(\theta) = |\frac{\delta L}{\delta \theta} \odot \theta| \tag{3.3}$$

where L is the loss function, θ is the parameters of the neural network and $\odot$ is the Hadamard product.

3.4.3 GRASP

Like SNIP, Gradient Signal Preservation (GRASP) is a pruning-at-initialisation technique. The authors argue that efficient training depends on preserving the gradient flow, from which the name gradient signal preservation comes. GRASP aims at finding the sub-networks as initialisation rather than after training. This is done by pruning the weights whose removal will cause a minor fall in the gradient norm after pruning (Wang et al. 2020).

3.4.4 Synflow

Iterative Synaptic Flow Pruning (synflow) was originally proposed in (Tanaka et al. 2020) for parameter pruning. By pruning more deficient important parameters, we get highly sparse networks with reduced network size and still close to the

same accuracy. Synflow is built on three main principles; avoiding layer collapse, conservation of synaptic saliency and magnitude pruning.

Layer collapse occurs when an algorithm prunes all parameters in a single weight layer even when prunable parameters remain elsewhere in the network. This renders the network untrainable, evident by sudden drops in the achievable accuracy for the network (Tanaka et al. 2020). Therefore, it is important to avoid layer collapse because the network will be unusable if a layer is removed.

Synaptic saliency is a class of score metrics that can be expressed as the Hadamard product

$$S(\theta) = \frac{\delta R}{\delta \theta} \cdot \theta$$

where R is a scalar loss function of the output y of a feed-forward network parameterized by θ (Tanaka et al. 2020). Further, the conservation of synaptic saliency theorem shows that the sum of synaptic saliency for all incoming parameters to a hidden neuron is equal to the sum of the synaptic saliency for the outgoing parameters from the hidden neuron (Tanaka et al. 2020). This, in turn, also means that the sum of synaptic saliency for all incoming parameters to a hidden layer is equal to the sum of the synaptic saliency for the outgoing parameters from the hidden layer. This can result in many problems when performing one-shot pruning, where a larger layer would have parameters with lower synaptic saliency. In comparison, the parameters for a smaller layer have higher synaptic saliency. Pruning one of the parameters for a smaller layer would result in pruning almost all the parameters in a larger layer to compensate for the conservation of synaptic saliency, resulting in layer collapse.
To prevent this from happening, Synflow introduces iterative pruning (magnitude pruning). By recalculating the synaptic saliency for all parameters for each iteration. This would re-evaluate all the parameters that survived the pruning and give a new score to the parameters that might have had a lower score in previous iterations. This would eliminate the imbalance between smaller and larger layers and reduce the possibility of layer collapse.
Finally, the Synflow algorithm, which is described in equation 3.4

$$R_{SF} = \mathbb{1}^T \left(\prod_{l=1}^{L} |\theta^{[l]}| \right) \mathbb{1} \tag{3.4}$$

The algorithm takes an input of ones (ex., an image where all the pixels have the value 1) and performs a forward passes them through the network. Then by calculating the product of the output, we get the loss R. This loss is then back-propagated through the network back to the layers and the equation 3.4.4 where we get the synaptic saliency score for each parameter θ

3.4.5 Fisher

Fisher was originally a pruning method introduced in (Theis et al. 2018). The objective of the pruning algorithm was to eliminate feature maps or parameters that don't significantly improve the model's overall performance. This is accomplished by removing the activational channels and their corresponding parameters, which will have a negligible effect on the loss by some estimation technique (Abdelfattah et al. 2021).

(Theis et al. 2018) used this as a base to calculate the network score:

$$S_z(z) = (\frac{\delta L}{\delta z} z)^2, S_n = \sum_{i=1}^{M} S_z(z_i) \tag{3.5}$$

where M is the length of the vectorised feature map, and S_z is the saliency per activation z.

By mapping all the different layers of the network, we can capture the activations and use them to score the network.

3.4.6 NASWOT / Jacobian Covariance

(Mellor et al. 2020) derived a metric by investigating linear maps of binary activation codes from the overlapping ReLU present at initialisation. These linear maps provide insight into how the network divides the input space.

The intuition is that the more similar the binary codes associated with two inputs are, the more challenging it is for the network to learn to separate them. When two inputs have the same binary code, they lie within the same linear region of the network and are particularly difficult to disentangle (Mellor et al. 2020).

Let's consider a mini-batch of data $X = \{x_i\}_{i=1}^{N}$ mapped through a neural network as $f(x_i i)$. We can define an indicator variable c_i that forms a binary code that defines a linear region. Using the binary codes, we can use Hamming distance $d_H(c_i, c_j)$ to measure how different the two inputs are (Mellor et al. 2020).

The correspondence between binary codes in the mini-batch can be computed with the kernel matrix

$$K_H = \begin{bmatrix} N_A - d_{H(c_1,c_1)} & \dots & N_A - d_{H(c_1,c_N)} \\ \vdots & \ddots & \vdots \\ N_A - d_{H(c_N,c_1)} & \dots & N_A - d_{H(c_N,c_N)} \end{bmatrix} \tag{3.6}$$

where N_A is the number of ReLU activations in the network. Then the final score metric s is derived from the logarithm of the kernel norm at initialisation.

$$s = log||K_H|| \tag{3.7}$$

3.4.7 Vote

The "Vote" zero-cost proxy is a technique that can be used to combine the predictions of multiple zero-cost proxies (such as Synflow, Jacob-Conv, and Fisher)

with improving the accuracy and efficiency of NAS(Abdelfattah et al. 2021). This is done by training each of the zero-cost proxies on a subset of the data using the appropriate training procedure for each proxy. Then use each of the zero-cost proxies to predict the entire dataset, estimating the performance of different model architectures without actually training and evaluating the architectures on the task. Finally, combine the predictions from the other proxies using a voting scheme, where the final performance estimate for each architecture is the average of the predictions from the different proxies.

$$Vote = \frac{Synflow + JacobConv + Fisher}{3} \tag{3.8}$$

Finally, use the final performance estimates from the "Vote" proxy to guide the NAS process, selecting the architectures with the highest performance estimates as the best ones.

Chapter 4

Experiments

From the research questions, one of the goals is to examine how zero-cost proxies can be applied to a GCN and how it might perform. As this thesis is in collaboration with In-Motion, we want to see how zero-cost proxies can be implemented in the already existing MovementOutcome framework (available at https://github.com/DeepInMotion/MovementOutcome).

4.1 Dataset

For this study, we analysed the InMotion dataset for infants' CP-detection, referred to in Groos 2022. The dataset contains the tracker coordinates with each infant's corresponding binary CP outcome. We utilised this particular dataset because we decided to use the MovementOutcome framework developed to be operated on the dataset, which would help us experiment rapidly.

We also considered using the NTU RGB+D 120 dataset, which contains 120 various activity classes collected from over 114 thousand video samples (J. Liu et al. 2019). However, due to the lack of time and available frameworks, we opted not to pursue this as we thought it would obstruct the progress of the experiments.

4.2 Creating benchmark

In other similar studies, such as Abdelfattah et al. 2021, already existing NAS benchmarks (i.e. NAS-Bench 101) were used to test different zero-cost techniques. For this study, no such benchmark existed such that it was necessary to create one.

The candidates were generated through a simple random search algorithm which used the search space to constraint the options. Then, we could verify that the randomly generated architecture was not previously trained by keeping track of existing candidates.

After obtaining a set of candidate architectures given the already existing search space, we fully trained each model to get the validation accuracy with the given dataset.

We stored the results in a hash map, using the architecture as the key to efficiently retrieve the different metrics of a given architecture. The MovementOutcome framework provided the validation accuracy for each model as the area under the receiver operating characteristic curve (AUC). Using AUC instead of the validation accuracy is an advantage because it does not require the definition of a threshold, which was appropriate for the imbalanced nature of the InMotion dataset.

4.3 Implement zero-cost proxies

Next, we implemented zero-cost proxies in the MovementOutcome framework, inspired by the work of Abdelfattah et al. 2021. The advantage of zero-cost proxies is that they only require a single forward- and backward pass with a minibatch. After performing the forward- and backward-pass, we could utilise the parameters of the network to calculate the score. For instance, the *grad_norm* is calculated as follows:

```
output, _ = model(data)
loss = loss_function(output, labels)
loss.backward()
score = get_score(model,
                  lambda l:
                  l.weight.grad.norm()
                  if l.weight.grad is not None
                  else torch.zeros_like(l.weight),
                  mode='param')
```

Figure 4.1: Calculate zero-cost score

The remaining zero-cost proxies were implemented using the theory presented in section 3.4. The different implementations can be found in the GitHub-repository.

4.4 Compare zero-cost proxies

We can compare the methods' performance when we have a benchmark and implementation for the different zero-cost proxies. We used an established technique, namely Spearman's rank correlation coefficient, to find the correlation between the validation accuracy and each zero-cost proxy score. The framework

developed by Groos 2022 used the AUC to calculate the performance of each architecture. The AUC metric can be used with the score of each zero-cost proxy to obtain the correlation given by the formula:

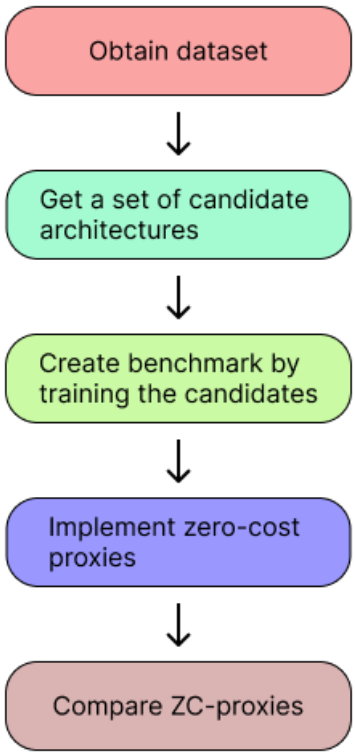
$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.1}$$

, where ρ is Spearman's rank correlation coefficient, d_i is the difference between the observations, and n is the number of observations.

4.5 Experimental plan

Figure 4.2 shows how we plan, on a very high level, to perform the experiment in this thesis.

Figure 4.2: Experimental plan



Chapter 5

Results

In this chapter, the results from the experiments will be presented. All experiments were performed on the Darth node at NTNU, which has 2 NVIDIA Tesla P100 GPUs with 16 GB of memory.

5.1 Search Space

The search space used to constrain the searching algorithm is the same as used in Groos 2022.

Table 5.1: Search space used in experiment

Name	Values
Attention	null, channel, frame, joint
Block type	Basic, bottleneck, mbconv
Bottleneck factor	2,4
Graph	Spatial, dis2, dis4, dis42
Initial block type	Basic, bottleneck, mbconv
Initial residual	Null, block, module, dense
Input temporal scales	1,2,3, linear
Input width	6,8,10,12
Main temporal scales	1,2,3,linear
Nonlinearity	ReLU, swish
Number of input modules	1,2,2003
Number of main level modules	1,2,2003
Number of main levels	1,2
Pool	Global, spatial
Residual	Null, block, module, dense
SE	Null, inner, outer, both
SE ratio	2,4
SE type	Relative, absolute
Temporal kernel size	3,5,7,9

5.2 Hyperparameters

As explained in chapter 4, each architecture had to be fully trained to obtain the architecture's relative performance (AUC). The authors set the following hyperparameters before each training:

Table 5.2: Hyperparameters before training

Batch size	32
Learning rate	0.005
Momentum	0.9
Weight decay	0.0

5.3 Experimental Results

We calculated the Spearman Rank Correlation using each zero cost proxy score and the AUC measure for each architecture. We used seven zero-cost proxies in the experiments; synflow, snip, grad_norm, grasp, fisher, jacobian, params and flops.

Table 5.3: Spearman rank correlation and average runtime for every zero-cost proxy and training

	ρ	**time in seconds**
fisher	-.106	18.171
flops	.171	21.312
grasp	-.000	21.583
grad_norm	-.095	18.185
jacobian	-.120	14.331
params	.191	19.575
snip	-.049	18.212
synflow	-.415	18.503
vote	.502	-
training	-	7107.121

Chapter 6

Discussion

In this chapter, we will discuss the key findings and implications of the research presented in this paper. We will reflect on the findings and limitations in the context of the research questions given in section 1.2.

6.1 Lack in literature

Chapter 3 gave an insight into relevant literature for this thesis. We found multiple papers regarding using NAS for GCN during our literature search, and all methods found used computationally heavy evaluation methods.

For instance, fully training each architecture to obtain the performance. However, more prominent research was done on different performance predictors for CNNs, such as Abdelfattah et al. 2021.

Our literature research found no research on performance predictors used with NAS-GCN. However, there were papers regarding NAS-GCN, but there was a lack of methods using performance predictors to make the algorithm more efficient (Nunes and Pappa 2020; Gao, H. Yang et al. 2020; Gao, P. Zhang et al. 2022).

Overall, our literature review showed that there is a need for more research on performance predictors for NAS-GCN, particularly on the use of zero-cost proxies. By exploring these and other performance predictors, we can develop more efficient and scalable methods for NAS on GCN and improve machine learning algorithms' performance in tasks involving graph data.

6.2 Limitations

6.2.1 Number of architectures

In our study, we managed to train a total of 29 architectures which we could use for calculating the Spearman Rank Correlation. We argue that the study would benefit from being conducted on a more extensive and diverse dataset that includes a broader range of human movements and actions.

Our research discovered that similar experiments did comparative research with over 15 000 trained architectures, which is far more than we have accomplished. Therefore, training more models would be preferable.

Additionally, a more general dataset would allow the researchers to generalize their findings to a broader range of applications and contexts, making the study results more widely applicable.

6.2.2 Dataset

Our experiment used the InMotion dataset for infant CP prediction, which contains data on human movements collected from infants. While this dataset provided valuable insights into the effect of zero-cost proxies in performance evaluation, we believe the study would be more insightful if it were conducted on a more general and extensive dataset for human action recognition.

6.3 Zero-cost proxies

There are several strengths and challenges associated with using zero-cost proxies with NAS. After reviewing relevant literature and our experiments, this is what we found:

6.3.1 Strength

Speed

Looking at Table 5.3 we can see that the speed of all of the zero-cost proxies is way faster than thoroughly training the model. Because the zero-cost proxies, at most, perform a single forward and backwards pass on a single batch.

Flexibility

Zero-cost proxies can be easily adapted to different tasks and can be used to evaluate a wide range of neural network architectures Table 5.1. This allows for greater flexibility in the search process and makes it possible to find the best architecture for a particular task.

Exploited potential

Zero-cost proxies are known as cheap, weak learners. However, experiments (White, Khodak et al. 2022) have shown that combining multiple zero-cost proxies and other optimization techniques will yield more robust performance. This is further confirmed in our results, showing that the combined vote score gives a better result.

6.3.2 Challenges

Limited information

Zero-cost proxies provide only a limited amount of information about the performance of a network architecture, as they are only surrogates for the true performance measure. This can make it difficult to accurately evaluate the performance of different architectures using only the proxy measure and can lead to suboptimal architectures being discovered by the search.

Difficulty in choosing evaluation measures

Choosing the appropriate evaluation measures for a given NAS task can be challenging, as different measures may be more or less relevant for different tasks or datasets. This can make it difficult to effectively use zero-cost proxies, as the choice of evaluation measures can significantly impact the performance of the architectures discovered by the search.

While zero-cost proxies can be a valuable tool for optimizing the search process in NAS, they also present several challenges that can make it difficult to use them effectively in this context.

6.4 Answers to research questions

Research question 1

In this research question, the authors wanted to investigate the field of NAS to acquire a good overview of the subject before the experiments. We extensively presented the research regarding NAS in section 2.3 by discussing the three main components; search space, search strategy and performance estimation. NAS proves beneficial because of its ability to discover high-performing architectures without introducing humans. At the same time, by providing a sufficient search space, a NAS algorithm may be able to discover architectures yet discovered.

There are, however, several challenges with NAS. First and foremost, the computational power required to train candidate architectures fully is generally infeasible for most, and it can require hundreds or more GPU days to complete. The use of thousands of GPU days can potentially be damaging to the environment due to the amount of energy that is required to run them. This energy consumption can contribute to greenhouse gas emissions, significantly contributing to climate change.

Research question 2

In section 2.3.4 and section 3.4, zero-cost proxies were discussed. From our research, we found that zero-cost proxies are very fast, providing a score from just a single forward/backward pass of a minibatch of data. Prominent research on zero-cost proxies with convolutional neural network showed that many methods had a moderate to strong correlation with the validation accuracy through

different datasets. They argued that if zero-cost proxies are proven to be consistent in their correlation with performance, they could significantly improve the efficiency and accuracy of NAS algorithms. Zero-cost proxies can eliminate bad architectures from a set of candidates. Further, they may be combined with other performance predictors to provide efficiency and accuracy.

From our research, the following zero-cost proxies have been used in the literature:

- Grad norm
- Synflow
- Snap
- Grasp
- Fisher
- Jacobian covariance
- Flops
- Params

Research question 3

Table 5.3 shows the spearman rank correlation for each method we used in the experiment. According to statstutor 2022, the strength of the correlation is described as follows:

- .00 - .19 "very weak"
- .20 - .39 "weak"
- .40 - .59 "moderate"
- .60 - .79 "strong"
- .80 - 1.0 "very strong"

The results vary a lot. We can observe that the zero-cost proxy vote performs the best with a spearman rank correlation of .502, meaning that the correlation is moderate according to statstutor 2022. However, the other methods have a very weak correlation.

The fact that vote has the most significant correlation is similar to the study conducted by Abdelfattah et al. 2021, in which vote performs quite well on different datasets. Next is synflow with a correlation of $-.415$, which is also considered a moderate correlation.

Regarding the time it took to score an architecture, the zero-cost proxies are, as expected, outperforming training each architecture significantly. Training each architecture took an average of 7107 seconds, almost two hours. On the other hand, the average zero-cost proxies used between 14 and 21 seconds to compute, a difference of almost 200 %.

Chapter 7

Conclusion and Future Work

In this chapter, we will discuss the conclusions that can be drawn from the work presented in this paper. Our aim is to use these conclusions as a starting point for further exploration in our master's thesis. We will also discuss future work that could build upon the research presented here. This chapter summarises the key findings of this paper and offers insights into how this work can be extended and built upon in the future.

7.1 Conclusion

This project aimed to discover a performance predictor technique that may yield improvement in NAS for GCN when used with the InMotion dataset for CP prediction of infants. By conducting a literature study, we learned what NAS is and why it is handy for different tasks regarding machine learning. We also looked at the various components of NAS: the search space, search strategy and performance estimation.

Further, we researched zero-cost proxies by looking at how they can improve NAS. In addition, we also looked at the state-of-the-art methods used within the literature as of today.

Lastly, we utilised our knowledge of zero-cost proxies to experiment with how they would perform on the InMotion-dataset for infant CP prediction. By calculating a score of each method, we could compute the Spearman Rank Correlation to calculate the correlation between the AUC of the fully trained architecture and each zero-cost proxy. The experiments did show that the vote method had the most significant correlation, with a spearman ρ of .502. The zero-cost proxies did, as expected, outperform the approach of fully training each architecture by a significant margin when considering how long it took.

In chapter 6, we discussed different aspects of our thesis. Even though the study may yield some directions regarding using zero-cost proxies within GCN-NAS, we argue that the study needs more depth. The low number of trained architectures and the dataset are reasons why we think the field of study needs more extensive research.

7.2 Future work

This thesis is written before our master thesis, which we will write next semester (spring 2023).

Based on what we found out in this project, we will discuss how the research may proceed. The proposed future work is rooted in the discussions and results presented in the thesis and aims to advance and extend the research in this area.

7.2.1 More general framework

The authors used the InMotion-dataset for infant CP detection and a framework created for that particular purpose for the experiments in this thesis. However, we would like to investigate a more general framework for different domains of human tasks. Mainly how zero-cost proxies will perform within a different setting than we did for this experiment. In section 6.1, we discussed what is lacking in the literature today, which is also a reason to proceed with the research of zero-cost proxies with GCN.

7.2.2 Different dataset

In addition to using a more general framework, we want to run experiments using at least one other dataset to test out zero-cost proxies on more diverse datasets. Running similar experiments on multiple datasets would be beneficial to study zero-cost proxies further. This could include datasets like NTU RGB+D 120 (J. Liu et al. 2019). Other researchers have used at least three different datasets in related work to ensure that the methods have been tested on various data. Therefore, we argue that the research would benefit from using other datasets.

Furthermore, using multiple datasets would also help validate our findings' generalizability. Previous research in this field has used at least three different datasets in their experiments, and we believe our study would benefit from this approach. By testing our methods on various datasets, we can ensure that our results are not specific to a particular dataset or domain but reflect the broader effectiveness of zero-cost proxies in predicting performance.

7.2.3 Investigate other performance predictors

One potential avenue for future work is to investigate other performance predictors that could be used in conjunction with the zero-cost proxies presented in this thesis. As our study showed, the vote method had the most significant correlation out of the zero-cost proxies with the AUC of the fully trained architectures, but there may be other predictors that could improve the accuracy of our predictions even further.

7.2.4 Combine zero-cost proxies with other performance predictors

One potential path for future research is combining zero-cost proxies with other performance predictors to improve estimation accuracy. Our preliminary findings using the "vote" metric show promising results, and we believe that there is potential in this approach.

Bibliography

Abdelfattah, Mohamed S, Abhinav Mehrotra, Łukasz Dudziak and Nicholas D Lane (2021). 'Zero-cost proxies for lightweight NAS'. In: *arXiv preprint arXiv:2101.08134*.

Akhauri, Yash, J Pablo Munoz, Nilesh Jain and Ravi Iyer (2022). 'Evolving Zero Cost Proxies For Neural Architecture Scoring'. In: *arXiv preprint arXiv:2209.07413*.

Albawi, Saad, Tareq Abed Mohammed and Saad Al-Zawi (2017). 'Understanding of a convolutional neural network'. In: *2017 international conference on engineering and technology (ICET)*. Ieee, pp. 1–6.

Dong, Xuanyi and Yi Yang (July 2019). 'Searching for a Robust Neural Architecture in Four GPU Hours'. In: DOI: 10.1109/CVPR.2019.00186.

Elsken, Thomas, Jan Hendrik Metzen and Frank Hutter (2019). 'Neural architecture search: A survey'. In: *The Journal of Machine Learning Research* 20.1, pp. 1997–2017.

Frankle, Jonathan, Gintare Karolina Dziugaite, Daniel M Roy and Michael Carbin (2020). 'Pruning neural networks at initialization: Why are we missing the mark?' In: *arXiv preprint arXiv:2009.08576*.

Gao, Yang, Hong Yang, Peng Zhang, Chuan Zhou and Yue Hu (July 2020). 'Graph Neural Architecture Search'. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, pp. 1403–1409. DOI: 10.24963/ijcai.2020/195. URL: https://doi.org/10.24963/ijcai.2020/195.

Gao, Yang, Peng Zhang, Hong Yang, Chuan Zhou, Zhihong Tian, Yue Hu, Zhao Li and Jingren Zhou (2022). 'GraphNAS++: Distributed Architecture Search for Graph Neural Networks'. In: *IEEE Transactions on Knowledge and Data Engineering*.

Groos, Daniel (2022). 'Convolutional networks for video-based infant movement analysis. Towards objective prognosis of cerebral palsy from infant spontaneous movements'. In.

Groos, Daniel, Heri Ramampiaro and Espen AF Ihlen (2021). 'EfficientPose: Scalable single-person pose estimation'. In: *Applied Intelligence* 51.4, pp. 2518–2533.

Hutter, Frank, Lars Kotthoff and Joaquin Vanschoren (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Kaelbling, Leslie Pack, Michael L Littman and Andrew W Moore (1996). 'Reinforcement learning: A survey'. In: *Journal of artificial intelligence research* 4, pp. 237–285.

Kandasamy, Kirthevasan, Willie Neiswanger, Jeff Schneider, Barnabas Poczos and Eric P Xing (2018). 'Neural architecture search with bayesian optimisation and optimal transport'. In: *Advances in neural information processing systems* 31.

Kipf, Thomas N. and Max Welling (2016). 'Semi-Supervised Classification with Graph Convolutional Networks'. In: *CoRR* abs/1609.02907. arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

Krizhevsky, Alex (2009). 'Learning Multiple Layers of Features from Tiny Images'. In.

Kyriakides, George and Konstantinos Margaritis (2020). 'An introduction to neural architecture search for convolutional networks'. In: *arXiv preprint arXiv:2005.11074*.

LeCun, Yann, Yoshua Bengio and Geoffrey Hinton (2015). 'Deep learning'. In: *nature* 521.7553, pp. 436–444.

Lee, Namhoon, Thalaiyasingam Ajanthan and Philip HS Torr (2018). 'Snip: Single-shot network pruning based on connection sensitivity'. In: *arXiv preprint arXiv:1810.02340*.

Liu, Hanxiao, Karen Simonyan and Yiming Yang (2018). *DARTS: Differentiable Architecture Search*. DOI: 10.48550/ARXIV.1806.09055. URL: https://arxiv.org/abs/1806.09055.

Liu, Jun, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan and Alex C. Kot (2019). 'NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding'. In: *CoRR* abs/1905.04757. arXiv: 1905.04757. URL: http://arxiv.org/abs/1905.04757.

Mellor, Joseph, Jack Turner, Amos Storkey and Elliot J. Crowley (2020). *Neural Architecture Search without Training*. DOI: 10.48550/ARXIV.2006.04647. URL: https://arxiv.org/abs/2006.04647.

Nunes, Matheus and Gisele L Pappa (2020). 'Neural architecture search in graph neural networks'. In: *Brazilian Conference on Intelligent Systems*. Springer, pp. 302–317.

Real, Esteban, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le and Alex Kurakin (2017). *Large-Scale Evolution of Image Classifiers*. DOI: 10.48550/ARXIV.1703.01041. URL: https://arxiv.org/abs/1703.01041.

Ren, Pengzhen, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen and Xin Wang (2021). 'A comprehensive survey of neural architecture search: Challenges and solutions'. In: *ACM Computing Surveys (CSUR)* 54.4, pp. 1–34.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg and Li Fei-Fei (2014). 'ImageNet Large Scale Visual Recognition Challenge'. In: *CoRR* abs/1409.0575. arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

statstutor (2022). *Spearman's correlation*. URL: https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf (visited on 29/11/2022).

Tanaka, Hidenori, Daniel Kunin, Daniel L. K. Yamins and Surya Ganguli (2020). *Pruning neural networks without any data by iteratively conserving synaptic flow*. arXiv: 2006.05467 [cs.LG].

Theis, Lucas, Iryna Korshunova, Alykhan Tejani and Ferenc Huszár (2018). 'Faster gaze prediction with dense networks and Fisher pruning'. In: *CoRR* abs/1801.05787. arXiv: 1801.05787. URL: http://arxiv.org/abs/1801.05787.

Vikhar, Pradnya A. (2016). 'Evolutionary algorithms: A critical review and its future prospects'. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 261–265. DOI: 10.1109/ICGTSPICC.2016.7955308.

Wang, Chaoqi, Guodong Zhang and Roger Grosse (2020). 'Picking winning tickets before training by preserving gradient flow'. In: *arXiv preprint arXiv:2002.07376*.

White, Colin, Mikhail Khodak, Renbo Tu, Shital Shah, Sébastien Bubeck and Debadeepta Dey (2022). 'A Deeper Look at Zero-Cost Proxies for Lightweight NAS'. In: *ICLR Blog Track*. https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/. URL: https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/.

White, Colin, Willie Neiswanger and Yash Savani (2021). 'Bananas: Bayesian optimization with neural architectures for neural architecture search'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12, pp. 10293–10301.

White, Colin, Arber Zela, Robin Ru, Yang Liu and Frank Hutter (2021). 'How powerful are performance predictors in neural architecture search?' In: *Advances in Neural Information Processing Systems* 34, pp. 28454–28469.

Wu, Felix, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu and Kilian Weinberger (Sept. 2019). 'Simplifying Graph Convolutional Networks'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6861–6871. URL: https://proceedings.mlr.press/v97/wu19e.html.

Yan, Sijie, Yuanjun Xiong and Dahua Lin (2018). 'Spatial temporal graph convolutional networks for skeleton-based action recognition'. In: *Thirty-second AAAI conference on artificial intelligence*.

Zhang, Si, Hanghang Tong, Jiejun Xu and Ross Maciejewski (2019). 'Graph convolutional networks: a comprehensive review'. In: *Computational Social Networks* 6.1, pp. 1–23.

Zhou, Dongzhan, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang and Wanli Ouyang (2020). 'EcoNAS: Finding Proxies for Economical Neural Architecture Search'. In: *CoRR* abs/2001.01233. arXiv: 2001.01233. URL: http://arxiv.org/abs/2001.01233.

Zhou, Kaixiong, Qingquan Song, Xiao Huang and Xia Hu (2019). 'Auto-gnn: Neural architecture search of graph neural networks'. In: *arXiv preprint arXiv:1909.03184*.

Zoph, Barret, Vijay Vasudevan, Jonathon Shlens and Quoc V. Le (2017). 'Learning Transferable Architectures for Scalable Image Recognition'. In: *CoRR* abs/1707.07012. arXiv: 1707.07012. URL: http://arxiv.org/abs/1707.07012.