Master's thesis

Zaim ul-Abrar Imran
Max Torre Schau

Exploring the Efficiency of Zero-Cost Proxies in NAS for Human Action Recognition

Master's thesis in Computer Science
Supervisor: Heri Ramampiaro
Co-supervisor: Espen A. F. Ihlen & Felix Tempel
June 2023

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

Zaim ul-Abrar Imran
Max Torre Schau

Exploring the Efficiency of Zero-Cost Proxies in NAS for Human Action Recognition

Master's thesis in Computer Science
Supervisor: Heri Ramampiaro
Co-supervisor: Espen A. F. Ihlen & Felix Tempel
June 2023

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

Abstract

Neural Architecture Search (NAS) and Graph Convolutional Networks (GCNs) are two fields within machine learning that have undergone major development in recent years. Finding a GCN architecture which yields optimal results can be time-consuming and resource-intensive. Zero-cost proxies, which are designed only to require a single minibatch of training data to score a neural network, have been introduced to make this process more efficient. The core focus of this thesis is to investigate the application and performance of zero-cost proxies for evaluating GCNs within the context of Human Action Recognition (HAR) tasks as a first step towards using them in a NAS algorithm. Furthermore, given the need for further research, the study aims to bridge this gap by evaluating different zero-cost proxies on GCN architectures. To the best of our knowledge, the study is the first to explore how zero-cost proxies perform on GCNs.

Through a series of analyses and experiments, the study demonstrates that integrating zero-cost proxies can significantly enhance the efficiency of NAS algorithms. The results reveal that the top-performing zero-cost proxies display a Spearman Rank Correlation (ρ) of approximately 0.8, indicating a strong to very strong correlation. However, no substantial improvement in correlation is detected when analysing architectures as they are trained for several epochs, implying that the zero-cost proxies might be most efficient at the initialisation of the neural network. Attempts to combine zero-cost proxies through vote and weighted arithmetic mean are showing promise, but they are not resulting in significant improvement compared to the individual application of each zero-cost proxy.

Sammendrag

Både Neural Architecture Search (NAS) og Graph Convolutional Networks (GCNs) er to felter innen maskinlæring som har gjennomgått en stor utvikling de seneste årene. Det å finne en GCN-arkitektur som gir gode resultater kan være svært tidskrevende og ressurskrevende. Zero-cost proxyer, som er utformet for å kun kreve en enkelt minibatch med treningsdata for å score et neural nettverk, har blitt introdusert for å gjøre denne prosessen mer effektiv. Hovedfokuset med denne avhandlingen er å undersøke bruken og ytelsen av zero-cost proxies for å evaluere GCN innenfor oppgaver relatert til gjenkjenning av menneskelig aktivitet (HAR) som et første steg mot å bruke dem i en NAS-algoritme. Basert på behovet for videre forskning i feltet, tar studien sikte på å bygge bro over dette gapet ved å evaluere forskjellige zero-cost proxies på GCN-arkitekturer. Så vidt vi vet, er studien den første til å utforske hvordan zero-cost proxies presterer på GCN.

Gjennom en serie analyser og eksperimenter, viser studien at integrering av zero-cost proxies kan betydelig forbedre effektiviteten til NAS-algoritmer. Resultatene viser at de best presterende zero-cost proxyene viser en Spearman Rank Correlation (ρ) på omtrent 0.8, noe som indikerer en sterk til veldig sterk korrelasjon. Imidlertid blir ingen betydelig forbedring i korrelasjon oppdaget når arkitekturene blir analysert etter å ha blitt trent i flere epoker, noe som antyder at zero-cost proxies er mest effektive ved initialiseringen av det nevrale nettverket. Forsøk på å kombinere zero-cost proxier ved hjelp av stemmegiving og vektet aritmetisk gjennomsnitt viser potensial, men gir ikke noen betydelig forbedring sammenlignet med å bruke zero-cost proxies individuelt.

Preface

This is the final project for our Master's in Computer Science at the Norwegian University of Science and Technology (NTNU). It's written by us, Zaim ul-Abrar Imran and Max Torre Schau, and is the last step in our studies.

The project is part of DeepInMotion, a collaboration project between NTNU and St. Olavs Hospital in Trondheim. Head of Department and Professor at the Department of Computer Science, Heri Ramampiaro has been the main supervisor of this Master's thesis, with Associate Professor at the Department of Neuromedicine and Movement Science, Espen Alexander F. Ihlen and PhD Candidate at the Department of Computer Science, Felix Tempel as co-supervisors.

Acknowledgement

We would first like to thank our supervisor, Heri Ramampiaro, for allowing us to work on the project. His guidance, support, and encouraging feedback have been invaluable. He also gave us many chances to showcase and discuss our work, which we found both inspiring and educational.

We would also like to thank our co-supervisors, Associate Professor Espen Alexander F. Ihlen and PhD Candidate Felix Tempel. The guidance meetings have been instrumental in our project's success, alongside their expert advice and continuous support. We are deeply grateful for their invaluable contribution to our learning experience.

Lastly, we want to thank our friends. Your friendship, helpful advice, and all the fun times we have had together truly helped us through our time at NTNU. We are so grateful for your support during this part of our lives. We also want to express our gratitude to our families for all your support and motivating words.

To Zaim: I want to thank you for all our memories; it's been a blast! Thank you for being a fellow student, colleague, collaborator, problem-solver, and, most importantly, a best friend through my five years at NTNU.
- Max

Max, the past five years have been an incredible journey filled with both friendship and partnership. Our shared experiences, from late-night study sessions to exhilarating concerts, have only served to deepen and strengthen our bond. Your support and friendship have been invaluable; I am deeply grateful. Thank you.
- Zaim

Contents

Figures

Tables

Algorithms

Acronyms

ANN	Artificial Neural Network.
Auto-GNN	Automated Graph Neural Network.
AutoML	Automated Machine Learning.
BO	Bayesian Optimisation.
CNN	Convolutional Neural Network.
CP	Cerebral Palsy.
DAG	Directed Acyclic Graphs.
EC	Evolutionary Computations.
EcoNAS	Economical Evolutionary-based NAS.
Flops	Floating Point Operations per second.
GCN	Graph Convolutional Network.
GNN	Graph Neural Network.
GPU	Graphics Processing Unit.
Grasp	Gradient Signal Preservation.
HAR	Human Action Recognition.
NAS	Neural Architecture Search.
NTNU	Norwegian University of Science and Technology.
NTU RGB+D	Nanyang Technological University's Red Blue Green and Depth-information.
Snip	Single-shot Network Pruning.
Synflow	Iterative Synaptic Flow Pruning.
VCNN	Vanilla Convolutional Neural Network.

Chapter 1

Introduction

This chapter introduces the background and motivation behind the research, as well as the problem statement and scope of the study. It also includes the main goal and research questions, which aim to investigate the potential of using zero-cost proxies to enhance the efficiency in Neural Architecture Search (NAS) algorithms, specifically for Graph Convolutional Network (GCN) applied to Human Action Recognition (HAR) tasks. Finally, the chapter concludes with a section on the research method and contributions of the study.

1.1 Background and Motivation

This thesis is written as part of a project named DeepInMotion ('DeepInMotion', 2023). Researchers from the Norwegian University of Science and Technology (NTNU) and St. Olavs Hospital cooperate in a cross-disciplinary collaboration involving child physiotherapists, paediatricians, neonatologists, movement scientists and computer engineers. The project has developed a pipeline for detecting Cerebral Palsy (CP) in infants by providing a video of the infant's movement. The pipeline employs a Convolutional Neural Network (CNN) to accurately extract movement from 2D images or videos (Groos et al., 2021). A GCN then processes the output to predict CP in high-risk infants at three months of age (Groos, 2022a).

The CP-prediction pipeline used NAS to find a suitable architecture automatically. However, the search could be faster and more efficient while still finding optimal architectures. Performance prediction offers a way to predict an architecture's relative performance for a specific issue. Compared to training the architecture until convergence, this approach may be considerably more effective. Recent studies (Abdelfattah et al., 2021; White et al., 2022) show that zero-cost proxies yield great promise regarding using it to rank different architectures on image classification tasks. However, to our knowledge, research is yet to be done on how zero-cost proxies perform on GCN architectures. Consequently, additional studies in this field may provide important findings which can be used to improve the CP-prediction pipeline. Also, additional knowledge about how zero-cost proxies can be used with graph-based learning could be gained.

1.2 Problem Statement

In recent years, NAS has emerged as a promising technique for automating designing neural networks with excellent performance on specific tasks (Zoph & Le, 2016). Several studies have shown that NAS can effectively identify suitable architectures for GCN problems. Particularly in the field of HAR, GCN have demonstrated their utility due to the representation of the human body as a graph.

Despite the recent advancements in NAS for GCN, evaluating architectures in existing studies remains a significant challenge. It is often infeasible to thoroughly train each candidate's architecture to obtain its ground truth accuracy (Zoph & Le, 2016). This issue is particularly pressing in resource-constrained environments, where training large numbers of models is infeasible.

Moreover, a significant limitation of the existing literature on GCN-NAS is the absence of performance predictors, further complicating the optimisation process. The lack of such predictors can make it challenging to efficiently identify the most promising architectures for a given task. Novel approaches, such as zero-cost proxies, have been proposed to predict candidate architectures' performance without requiring full training.

1.3 Scope of the Thesis

This thesis investigates the potential of utilising zero-cost proxies to enhance the efficiency of architecture search in NAS algorithms, specifically targeting GCN applied to HAR tasks. The research questions formulated for this study are designed to explore various aspects of zero-cost proxies and their effectiveness in ranking GCN architectures.

Nonetheless, it is crucial to outline the scope and limitations of this thesis to ensure that readers understand which topics will and will not be discussed. Although the overall goal is to improve and optimise the efficiency of NAS with GCN for HAR, the primary focus of this study is to analyse and evaluate different zero-cost proxies. As a result, a full implementation of the research findings within an NAS algorithm will not be provided.

By clarifying the scope of this thesis, the intention is to offer readers a more comprehensive understanding of the research focus and the study's limitations. While recognising that incorporating the findings into a practical NAS algorithm is an essential and valuable subsequent step, the primary objective of this thesis is to lay the foundation for future research by exploring the potential of zero-cost proxies in the context of GCN for HAR tasks.

1.4 Goal and Research Questions

This section highlights the overall goal and the research questions of this thesis. The goal outlines what the research ultimately seeks to achieve, while the research question lays out the central issues the study will address to accomplish this goal. Together, these guide the research design, methodology, and analysis.

Goal *Improve and optimise the efficiency of neural architecture search with graph convolutional networks for human action recognition.*

NAS has been used with GCN in different studies (Groos, 2022b; Peng et al., 2020; Zhou et al., 2019), but finding other, more efficient methods is still possible. By finding methods that are more effective than what exists today, more architectures can be researched, which may result in detecting other well-performing architectures. Also, as training and searching for neural networks may impact the environment, effective methods will significantly reduce the carbon footprint.

Research question 1 *How well can different zero-cost proxies rank GCN architectures compare to their validation accuracy?*

The motivation behind the research question is that evaluating the correlation between zero-cost proxies and ground truth validation accuracy can determine if they accurately indicate the ground truth. A potential enhancement to these results could make NAS algorithms more efficient by eliminating the necessity for exhaustive training.

Research Question 2 *How early can we identify the correlation between* *zero-cost proxies and validation accuracy during the warm-up phase of GCN training to potentially halt the training process sooner?*

Research Question 2 aims to investigate the potential for early identification of the correlation between zero-cost proxies and validation accuracy during the warm-up phase of training for GCN. The aim is to determine how early this correlation can be identified because it can halt the training process sooner, thus saving computational resources and time.

Research question 3 *How can we effectively combine zero-cost proxies using various techniques to enhance the efficiency and accuracy of architecture search in NAS algorithms?*

Through investigating Research Question 3, the study aims to identify effective techniques for combining zero-cost proxies. By leveraging the strengths of multiple zero-cost proxies, future NAS algorithms may become more efficient and accurate in discovering high-performing architectures. The outcomes of this research question can provide insights into how to optimise the use of zero-cost proxies in NAS algorithms and improve the architecture search process in the future.

1.5 Research Method

We conducted a comprehensive literature review in the fall of 2022 to understand current state-of-the-art approaches in the NAS field. Additionally, a thorough analysis of existing papers concentrating on performance predictors within the NAS domain was completed. The project's conclusion presents several recommendations for advancing research in the NAS field based on the insights gained from the review.

The research plan involves conducting multiple quantitative investigations to examine the performance and behaviour of zero-cost proxies on a HAR dataset. Subsequently, the study aims to explore the potential of these proxies in improving NAS algorithms.

1.6 Contributions

This thesis presents several critical contributions to the NAS field with GCNs for HAR. The following are the primary contributions made by this study:

Investigation of Zero-Cost Proxies: In this thesis, a thorough analysis of the relationship between zero-cost proxies and the performance of GCN models in HAR tasks is conducted. The study sheds light on the usefulness of using zero-cost proxies to estimate the performance of GCN architectures without costly training.

Combining Zero-Cost Proxies: The thesis presents methods, such as the majority vote method and the weighted arithmetic mean method, for combining different zero-cost proxies to improve the efficiency of the NAS algorithms.

Environmental Considerations: The study highlights the importance of considering the environmental implications of NAS and artificial intelligence research. Furthermore, the study contributes to developing more sustainable practices by reducing the NAS process's computational demands and training time.

In view of these contributions, the work in this thesis advances the understanding of NAS with GCNs in the context of HAR and provide a foundation for further exploration of zero-cost proxies and their potential applications.

1.7 Thesis outline

The thesis consists of the following seven chapters:

Chapter 1 - Introduction: The introductory chapter presents the study's background, motivation, problem statement, scope, goal, and research questions.

Chapter 2 - Theory: Chapter 2 introduces necessary background theory, including deep learning, NAS, GCN and HAR.

Chapter 3 - Related Work: A comprehensive review of the literature on NAS and GCN-NAS is provided, encompassing zero-cost proxies and recent research on GCN-NAS. In addition, a discussion on the gap and limitations in the literature is presented.

Chapter 4 - Method: This chapter covers the use of zero-cost proxies in NAS, including various proxies and their implementation, the use of warmup, and the combination of proxies using a custom vote measure and weighted arithmetic mean.

Chapter 5 - Results: The results, namely the correlation analysis, vote measure and weighted arithmetic mean, are presented in this chapter.

Chapter 6 - Discussion: This chapter discusses the study's findings and analyses and interprets the results and their implications. In addition, it discusses the study's limitations as well as the environmental implications.

Chapter 7 - Conclusion and Future Work: The conclusion of the thesis summarises the findings and discusses future work.

Chapter 2

Theory

This chapter begins by discussing the fundamentals of deep learning and neural networks, setting the groundwork for the subsequent exploration of challenges with using machine learning on graphs. Further, the chapter describes the core of GCNs, the graph convolution operation, and how it aggregates local information from neighbouring nodes to generate a new representation for each node. The chapter also introduces Automated Machine Learning (AutoML), an emerging area of machine learning that seeks to automate designing optimal machine learning architectures. In addition, the chapter provides an overview of NAS, a sub-field of AutoML that aims to automate the creation of high-performing neural networks. Finally, the field of HAR is presented.

This chapter builds upon the unpublished specialisation project from the fall of 2022; portions of the work presented here originate from that project.

2.1 Deep Learning

Deep learning, a subfield of machine learning, has gained significant attention in recent years due to its ability to learn hierarchical representations from raw data, especially in domains such as computer vision, natural language processing, and speech recognition. This contrasts conventional machine-learning techniques, which are limited in processing the same (LeCun et al., 2015).

The core building blocks of deep learning are Artificial Neural Network (ANN) inspired by the biological neural networks found in the human brain. ANNs consist of interconnected layers of artificial neurons called nodes, each receiving input from previous layers, processing the information, and propagating the output to the subsequent layers. Deep learning architectures typically involve multiple layers of these interconnected nodes, hence the term "deep" (Goodfellow et al., 2016).

One key advantage of deep learning over traditional machine learning techniques is its ability to automatically learn and extract features from raw data without relying on manual feature engineering. This process, called representation learning, enables deep learning models to understand hierarchical representations of the input data, with each layer capturing increasingly abstract and complex features (Bengio et al., 2013). This capability has led to breakthrough performance improvements in various applications, including image classification, natural language understanding, and speech recognition (Krizhevsky et al., 2017).

2.1.1 Neural Networks

Neural Networks are a part of neurocomputing, which aims to develop computational systems inspired by the human brain's structure and function. For example, pattern recognition, motor control, vision, flexible inference, intuition, and accurate guessing are all skills the brain is particularly good at, which is what neural networks aim to emulate (Anderson, 1995).

At a high level, a neural network is a collection of related nodes (called "neurons") arranged in layers. Data is sent to the input layer, where one or more hidden levels process it before being output by the final layer.

The basic unit of a neural network is a neuron, which receives input from other neurons or the input layer. The neuron then uses a mathematical function to this input, producing an output which is sent to other neurons in the next layer. The most commonly used function is the sigmoid function, given in equation 2.1.

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.1}$$

where z is the input to the neuron. The sigmoid function always outputs a decimal between 0 and 1.

The output of a neuron is determined by the weights and biases associated with its inputs. Each input is multiplied by a weight, and these weighted inputs are summed together with a bias to produce the neuron's input, as shown in equation 2.2.

$$z = \sum_{i=1}^{n} w_i x_i + b, \tag{2.2}$$

where w_i is the weight, x_i is the value of the ith input, n is the number of inputs, and b is the bias.

Equation 2.3 exhibits the neuron output when applying the sigmoid function to the input.

$$y = \sigma(z) \tag{2.3}$$

During the training process, which includes feeding training data through the network (illustrated in figure 2.1) and modifying the weights and biases based on the difference between the predicted output and the actual output (target), the weights and biases of a neural network are adjusted. Usually, a gradient descent optimisation technique is used to perform this operation.

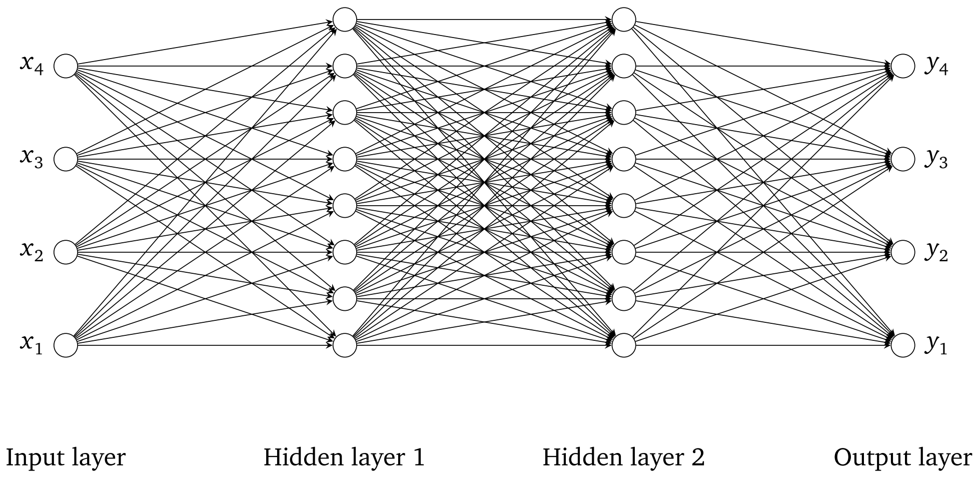


Figure 2.1: Illustration of Neural Network

2.2 Graph Convolutional Network (GCN)

Graphs are commonly used structures representing complex systems, including social networks, biological networks, and transportation systems (Scarselli et al., 2008). They consist of nodes and edges, representing the entities and relationships between them. Despite their widespread use, graphs pose unique challenges in machine learning due to their irregular structure, which makes it difficult to apply conventional learning techniques that rely on fixed-size inputs and Euclidean data representations (Battaglia et al., 2018).

Graph Neural Network (GNN) has emerged as a solution to these challenges, generalising deep learning techniques to graph-structured data (Gori et al., 2005). Among GNNs, Graph Convolutional Networks (GCNs) have become particularly popular owing to their ability to perform localised and efficient convolutions on graph data (Kipf & Welling, 2016). GCNs extend the concept of convolution from grid-like structures, such as images, to irregular graph structures, enabling the extraction of meaningful features from graph data while preserving their spatial relationships (Bronstein et al., 2017).

2.2.1 Graph Convolutions

The core of GCNs lies in the graph convolution operation, which aims to aggregate local information from neighbouring nodes to generate a new representation for each node. The main idea is to perform a weighted sum of the feature vectors of a node and its neighbours, where the weights are determined by the edge weights or some measure of node similarity. For instance, the feature vector of a node (a person) in a social network graph could include elements such as age, number of friends, job category, etc., that offer unique details about that node. This aggregation scheme allows GCNs to learn node representations that effectively capture the local graph structure. The equation 2.4, found in (Kipf & Welling, 2016), shows how the graph convolution operation is mathematically described.

$$\boldsymbol{H}^{l+1} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{A}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{l}\boldsymbol{W}^{l}) \tag{2.4}$$

The components in equation 2.4 are defined as follows (Kipf & Welling, 2016):

$H^{(l+1)}$ stands for the updated node representations (features) at layer $(l+1)$, while $H^{(l)}$ indicates the node features at layer l. $H^{(0)}$ is the input feature matrix for the first layer. The graph's adjacency matrix, represented by A, is modified to incorporate self-connections. This is achieved by adding a diagonal matrix with ones to the original adjacency matrix A, ensuring that the node's features are considered during the aggregation process.

The degree matrix, denoted by $\hat{D}$, corresponds to the modified adjacency matrix $\hat{A}$. It is a diagonal matrix wherein each diagonal entry signifies the degree (number of connections) of the corresponding node. The learnable weight matrix at layer l is represented by $W^{(l)}$ and is employed to transform the node features at each layer linearly.

Lastly, the activation function σ introduces non-linearity to the model. Typical choices for activation functions are ReLU, sigmoid, and tanh.

In the graph shown in figure 2.2, the nodes represent entities, and the lines between nodes represent the relationships between these entities. Here, we have four nodes: A, B, C, and D, and the connection between them.

The adjacency matrices (figure 2.3 and figure 2.4) are mathematical representations of the graph. In the matrices, the value at the intersection of a row and a column represents the presence (1) or absence (0) between the nodes.

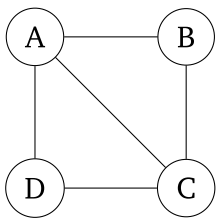


Figure 2.2: Graph with 4 nodes

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

Figure 2.3: Adjacency matrix corresponding to the graph without self-connections

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

Figure 2.4: Adjacency matrix corresponding to the graph with self-connections

2.3 Automated Machine Learning (AutoML)

Fields such as computer vision, speech recognition, and natural language processing have seen significant progress, primarily due to the development and application of deep learning techniques in recent years. However, despite these advancements, designing optimal machine learning architectures remains complex and time-consuming for data scientists. Automated Machine Learning (AutoML) is an area that seeks to automate this process.

2.3.1 Hyperparameter Optimisation

Hyperparameter optimisation is a critical component of the AutoML pipeline. Machine learning algorithms often contain hyperparameters that control their behaviour, and their optimal values are not learned directly from the training data. Instead, these parameters need to be set manually or searched for systematically. Traditional techniques for hyperparameter tuning include grid search, random search, and manual tuning. However, these approaches can be computationally expensive and inefficient. AutoML aims to streamline this process by employing advanced techniques such as Bayesian optimisation, evolutionary algorithms, and gradient-based methods. These methods can efficiently explore the hyperparameter space and identify suitable configurations, ultimately leading to improved model performance and reduced computational cost (Bergstra et al., 2011; Snoek et al., 2012).

2.3.2 Meta-learning

Meta-learning, also called "learning to learn", is another crucial component of AutoML. The central idea is to utilise prior knowledge gained from solving multiple related tasks to improve the learning efficiency and generalisation ability on new tasks. This is achieved by learning a model or an algorithm that can adapt quickly to novel tasks with limited data. In the context of AutoML, meta-learning can be employed in various ways. One popular approach is to use meta-learning for transfer learning, wherein a pre-trained model is fine-tuned on a new task with limited available data (Pan & Yang, 2010). Another application of meta-learning is hyperparameter optimisation, where a meta-model can be trained to predict the performance of different configurations across various tasks. This knowledge can guide the search for optimal hyperparameters more efficiently (Swersky et al., 2014).

2.4 Neural Architecture Search (NAS)

Most neural architectures are created by specialists, which is labour-intensive and susceptible to other weaknesses, including the need for an extensive expertise and the potential to introduce human bias. Subsequently, a way of automatically designing and developing such algorithms has been a research field for a couple of years. Neural Architecture Search (NAS) aims to automate the previously manual process of designing architectures (Elsken et al., 2019). Consequently, NAS is a sub-field of AutoML.

Given a search space F, a training set D_{train}, validation set D_{valid} and an evaluation metric M, a NAS algorithm aims at finding an optimal architecture $f^* \in F$ with the best metric M^* (such as validation accuracy) on the validation set D_{valid}. This can be written mathematically as shown in equation 2.5

$$\begin{aligned} f^* &= \text{argmax}_{f \in F} M(f(\theta^*), D_{\text{valid}}), \\ \theta^* &= \text{argmin}_{\theta} L(f(\theta), D_{\text{train}}), \end{aligned} \tag{2.5}$$

where θ^* is the learned parameters for the architecture f and L is the loss function (Zhou et al., 2019).

Figure 2.5 illustrates the overall concept of NAS. The search space gives the algorithm constraints regarding its development by defining a set of architectural choices the model might use. For example, a constraint might be different operations such as convolution, fully connected and pooling. One might argue that this is vital as selecting the search space can reduce the search's complexity, which is essential to produce an acceptable model (Kyriakides & Margaritis, 2020).

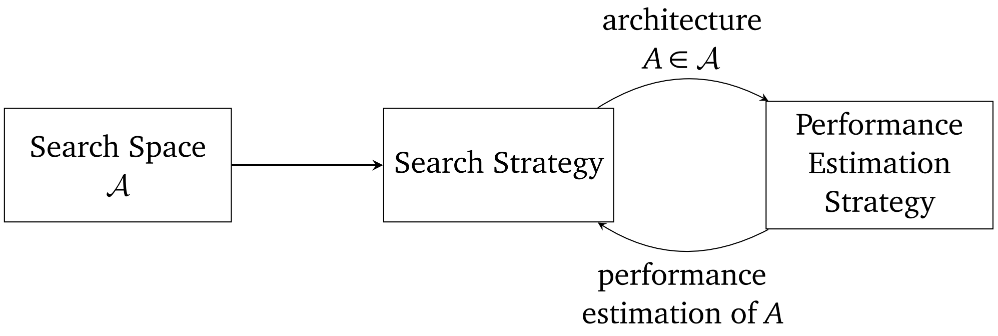


Figure 2.5: An overview of the different methods in NAS (Elsken et al., 2019)

After defining a search space for the given problem, the search strategy 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 & 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. However, this might require many hours of training, which requires a lot of energy and has a high environmental cost. Another downside of training the architectures is that it will limit the number of architectures the search algorithm might discover. As a result, methods simplifying this phase have been undergoing heavy research (Elsken et al., 2019).

Ultimately, the goal of NAS is to automatically discover the most optimal architecture regarding performance and processing power.

2.4.1 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 Graphics Processing Unit (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 optimisation and lack of interpretability

NAS algorithms often treat the architecture search space as a black box, meaning they cannot access the internal workings of the searched model architectures. 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 (Liu et al., 2018).

2.4.2 Search Space

When searching for a high-performing architecture, there are infinite variations that one might investigate. As a result, one defines 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 disadvantage of introducing human bias in the search space (Elsken et al., 2019).

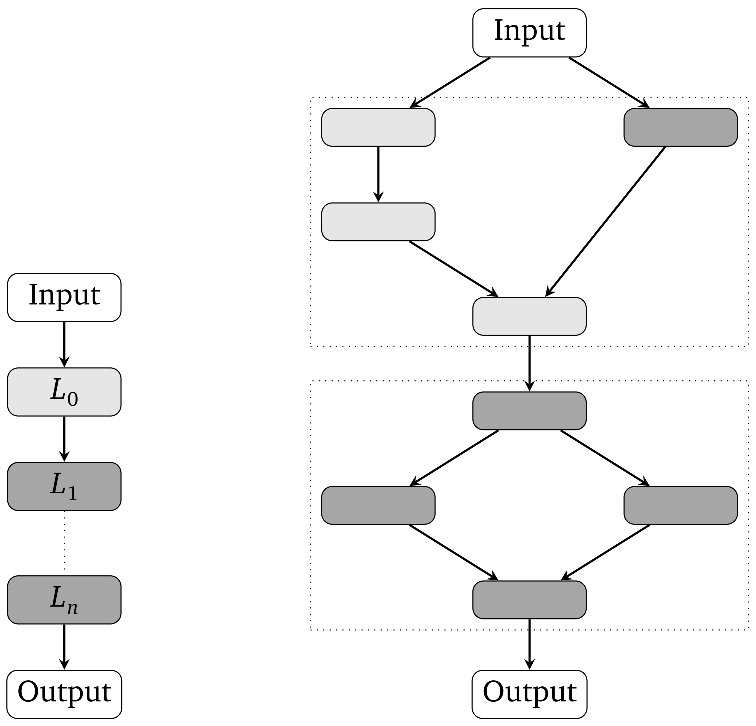
Global search space is a way in which it tries to combine all possible operations to create chain-structured (sequential) networks. Then, the search space has the parameters given in table 2.1.

Such a network can be described as a sequence of n layers, where layer L_i takes L_{i-1} as input, as given in figure 2.6a. However, this sort of search space is enormous and very expensive.

Table 2.1: Search Space Parameters

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

Cell-based representations were inspired by successful architectures using repeated modules (Inception, ResNet). The 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 explains 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 2.6b.



(a) Chain-structured network **(b)** Cells combined into an architecture

Figure 2.6: Search Space variations

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 architecture. For example, 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 ensure that the search process only considers valid network architectures, which can speed up the search and improve its accuracy (Liu et al., 2018).

In addition to representing the structure of a neural network, DAGs can also 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 discovering novel and effective network architectures (Dong & Yang, 2019).

2.4.3 Search Strategies

Random Search

Random search is the most naive search strategy, and it will simply randomly pick a good architecture based on the search space. Therefore, the method is relatively fast and does not require any learning model. 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). One 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 (Vikhar, 2016). Evolutionary algorithms in NAS randomly select N initialised models and then evaluate performance by the given evaluation strategy. The best models are chosen as parents, and new models have mutated clones of the parents, which are re-evaluated. Finally, the worst N models are removed from the population to make room for new children (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 $a^* = \arg\min_{a \in A} f(a)$, 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).

2.4.4 Performance Estimation

Performance Predictors

As mentioned in section 2.4, NAS aims to automate the designing of 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 we have a large search space dimension, which is why performance predictors are introduced (Akhauri et al., 2022).

Any function that predicts the eventual accuracy or ranking of architectures without fully training the architecture is referred to as a performance predictor f. Therefore, the performance predictor's function should 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 performs 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. Then, the query routine will output the predicted accuracy with the architecture details as input (White, Zela et al., 2021).

Multiple categories of performance predictors exist, as given in table 2.2.

Table 2.2: 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.

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 one can measure the ‘trainability’ of a neural network by looking at the initial gradient flow.

The paper *Zero-Cost Proxies for Lightweight NAS*, (Abdelfattah et al., 2021) showed the usefulness of a range of zero-cost proxies inspired by the pruning-at-initialisation literature. Each method can be divided into 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 et al., 2022). Table 2.3 lists data-independent and data-dependent zero-cost proxies. However, this collection is not exhaustive, and the other zero-cost proxies are not included.

Table 2.3: 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

2.5 Human Action Recognition (HAR)

Human Action Recognition (HAR) is a method that interprets the human body's gestures or motions via sensors, accelerometers or videos and uses these to predict human action (Jobanputra et al., 2019). Like many other computer vision tasks, HAR can be supervised and unsupervised, where supervised training requires a large amount of data (Ann & Theng, 2014). We can classify HAR into two problems; the localisation problem and the recognition problem. The localisation problem concerns where something is located in the video, whereas the recognition problem concerns the type of action we see (Vrigkas et al., 2015).

Due to problems like background clutter, partial occlusion, and changes in scale and frame resolution, capturing the specific action of a human within a video is a challenging task (Vrigkas et al., 2015). However, modelling the human body as three-dimensional data is another approach that has emerged in recent years. The human body can be divided into connecting joints forming a three-dimensional structure.

Chapter 3

Related work

This chapter reviews the literature on NAS and GCN-NAS. It covers the concept of zero-cost proxies in NAS and the key findings from research on the topic. Additionally, the chapter presents recent research on GCN-NAS, including automated graph neural networks and one-shot GNN-NAS with dynamic search space. Also, the chapter highlights the gaps and limitations of the literature today.

As there is no concrete literature on zero-cost proxies for GCN, we have chosen to include related work on other tasks (i.e. image classification) than HAR.

This chapter builds upon the unpublished specialisation project from the fall of 2022; portions of the work presented here originate from that project.

3.1 Performance Predictors

3.1.1 How Powerful are Performance Predictors in Neural Architecture Search

White, Zela et al., 2021 conducted a extensive study of performance predictors for NAS. 31 different predictors were utilised within the experiments across four search spaces and four datasets (NAS-Bench-201 with CIFAR-10, Cifar-100 and ImageNet16-120, NAS-Bench-101 and DARTS with CIFAR-10, and NAS-Bench-NLP with Penn TreeBank). For each predictor, the evaluation consisted of the initialisation time, query time and performance.

The study's motivation was that there were no existing approaches in the literature comparing the different performance predictors to each other. For each performance predictor, one had to go to the original paper proposing the method to find the evaluation. Consequently, the authors wanted to determine how zero-cost, model-based, learning curve extrapolation and weight-sharing methods compared.

Further, the paper proposed a new predictor, OMNI, which combines complementary information from three different families (learning curve, zero-cost and model-based) of performance predictors. OMNI showed great promise with substantially improved performance.

3.1.2 Neural Architecture Search without Training

Mellor et al., 2021 argued that NAS algorithms tend to be slow due to their extensive need for training architectures to guide the search algorithm. As a result, a proposed method for estimating a neural network's trained performance without needing to train it was developed. So, the authors show that it is, in fact, possible to perform NAS without training any of the architectures. The paper shows that by utilising the method, a network which achieves 92.81% accuracy was obtained within 30 seconds within the NAS-Bench-201 search space - remarkably faster than standard NAS algorithms. The results sparked the interest in scoring neural networks at initialisation and inspired other authors to create similar metrics.

3.2 Zero-Cost Proxies

3.2.1 Zero-Cost Proxies for Lightweight NAS

Abdelfattah et al., 2021 published a paper which did prominent research on the effect zero-cost proxies had on NAS. The authors proposed seven different zero-cost proxies and investigated them in the context of NAS on three different datasets; CIFAR-10 (Krizhevsky, 2009), CIFAR-100 (Krizhevsky, 2009) and ImageNet16-120 (Deng et al., 2009). Spearman Rank Correlation was used to calculate the correlation between the validation accuracy and the provided score of each proxy. The validation accuracies were obtained using the NAS-Bench-201 benchmark (Dong & Yang, 2020). NAS-Bench-201 contains 15,625 models for the three different image classification datasets. The study demonstrated that the zero-cost proxies presented by the authors not only matched but also surpassed conventional methods in terms of Spearman Rank Correlation. The paper reported that Iterative Synaptic Flow Pruning (Synflow) achieved a Spearman rank correlation of 0.82 on NAS-Bench-201, whereas Economical Evolutionary-based NAS (EcoNAS) attained a correlation of 0.61. Later, the study was extended with three new benchmarks; NAS-Bench-101 (Ying et al., 2019), NAS-Bench-NLP (Klyuchnikov et al., 2020) and NAS-Bench-ASR (Mehrotra et al., 2021). The results showed that the Synflow metric was the most consistent across the different benchmarks.

Further, the paper showed how zero-cost proxies could be applied as a zero-cost warm-up, which means that the proxies are used at the start of the search to initialise the algorithm. Thus, the expensive training and evaluation process is removed. The crucial factor in the zero-cost warm-up is the number of models for which we calculate and utilise the zero-cost metric (N). The advantage lies in that this value can often be much greater than the number of models we have the resources to train because $T << N$, where T is the number of models one can afford to train. The warm-up is performed on aging evolution, reinforcement learning and a binary predictor. The results showed that even the moderated correlated zero-cost proxies significantly speed up the different search algorithms across the datasets. In addition, the algorithms can find good-performing architectures.

3.2.2 A Deeper Look at Zero-Cost Proxies for Lightweight NAS

Building upon the research of Abdelfattah et al., 2021, White et al., 2022 delved deeper into the domain of zero-cost proxies for NAS. The team extensively analysed existing work regarding zero-cost proxies and performed novel experiments on the NAS-Bench-360 (Tu et al., 2021) and TransNAS-Bench-101 (Duan et al., 2021) benchmarks to increase the range of datasets and tasks involved.

In addition to the six zero-cost proxies in Abdelfattah et al., 2021, the paper added two baseline zero-cost proxies, Floating Point Operations per second (Flops) and params (number of parameters in the network). Their research shows that the zero-cost proxies perform unstable across different tasks and that no current zero-cost proxy consistently outperforms the others.

They argue that zero-cost proxies should be considered "weak learners", which can rapidly enhance the performance and effectiveness of other techniques within NAS.

3.2.3 NAS Bench Suite Zero

Krishnakumar et al., 2022 evaluated 13 zero-cost proxies on 28 different tasks and is thus the most extensive dataset for zero-cost proxies in the literature. This dataset may be vital in conducting faster experiments of zero-cost proxies as it offers precomputed zero-cost proxies scores on all tasks. To demonstrate the usefulness of the dataset, the authors conducted significant analyses of the different proxies, such as a bias analysis.

The article demonstrates that a technique is available to enhance the efficiency of a zero proxy by reducing biases. In this situation, biases may include a tendency to prefer more extensive architectures or those with more convolutions. In the paper, the authors consider the biases given in table 3.1.

Table 3.1: List of biases

conv:pool
cell size
num. skip connections
num. parameters

Through their research, they find that although many zero-cost proxies demonstrate different forms of biases to varying degrees, it is possible to reduce these biases and thus enhance their performance.

3.3 NAS for GCN

3.3.1 Auto-GNN

Automated Graph Neural Network (Auto-GNN) is a proposed method for automatically finding an optimal architecture of graph neural networks for a given graph-based task presented by Zhou et al., 2019. It is one of the earlier papers within GCN-NAS. The technique uses a reinforcement learning-based approach to search for an architecture that maximises task performance. Auto-GNN uses an RL agent to learn how to select and combine various GCN layers to construct an optimal GNN architecture.

The RL agent compiles the candidate architecture before the architecture can be trained on a graph classification task. During the training, the agent uses the model's accuracy on the validation set as the reward signal. As a result, the process is highly time-consuming as it requires training the network for multiple iterations of training and evaluations.

3.3.2 One-shot Graph Neural Architecture Search with Dynamic Search Space

There are no apparent applications of traditional NAS methods for GNNs because GCNs naturally differ from CNNs. The main reason is that the search space of GNNs is much more significant because of the variety of GNNs' message-passing components. Li et al., 2021 propose a dynamic search space that maintains a subset of the significant search space and a set of importance weights for operation candidates in the subset as the architecture parameters. After each iteration, the subset is pruned by removing candidates with low-importance weights and expanding with new operations. This dynamic subset of operation candidates is tailored for each edge in the computation graph of the neural architecture, ensuring the diversity of operations in the final architecture.

The paper demonstrates the effectiveness of this method through experiments on semi-supervised and supervised node classification tasks using citation networks, such as Cora, Citeseer, and Pubmed. The results show that the proposed method outperforms current state-of-the-art manually designed architectures and achieves competitive performance compared to existing GNN-NAS approaches, with up to 10 times speedup.

3.4 Summary and Implications

Section 3.1 presents the existing literature on performance predictors in NAS algorithms and the variety of types available. Further, the chapter shows that prior work has been performed on zero-cost proxies, but there is a lack of research on zero-cost proxies on GCNs. However, some work concerning NAS for GCN has been done, as shown in section 3.3. Still, the lack of performance predictors in the papers is a significant limitation, leaving a critical gap in evaluating and comparing proposed architectures effectively.

Applying zero-cost proxies in NAS for GCNs could potentially offer valuable insights into the performance of different network architectures without the need for expensive and time-consuming training processes.

Moreover, investigating zero-cost proxies for GCNs could also improve the efficiency of NAS algorithms. This would allow for a more rapid and accurate selection of optimal architectures and the discovery of novel GCN architectures more suited to specific tasks or datasets.

The limitations and gaps highlight the necessity of this thesis. Thus, the next logical step in this area of research is to conduct empirical testing and evaluation of zero-cost proxies within the context of GCNs.

Chapter 4

Method

This chapter discusses the different methods utilised in the experiments and the rationale behind the choices. The chapter begins with an overview of the planned research plan and explains how the benchmark is developed. After that, various zero-cost proxies, such as Flops, Params, EPE-NAS, L2-norm, Plain, Zen-score, and GradSign, are mathematically explained. Also, the implementation of the zero-cost proxies is provided. The chapter then looks into the combination of zero-cost proxies, discussing how to use the weighted arithmetic mean and vote to combine proxies to rank architectures.

4.1 Research Plan

In this section, the research methods employed in this study are presented. As figure 4.1 illustrates, the process comprises several steps. First, the process starts with generating random architectures, followed by comprehensive training of these architectures for benchmarking purposes. Simultaneously, research on zero-cost proxies is conducted, followed by developing a zero-cost framework. When the framework is created and all the architectures are fully trained, data regarding the performance of zero-cost proxies is collected. Lastly, a quantitative analysis is conducted using the acquired data to conclude the performance and effectiveness of the zero-cost proxies.

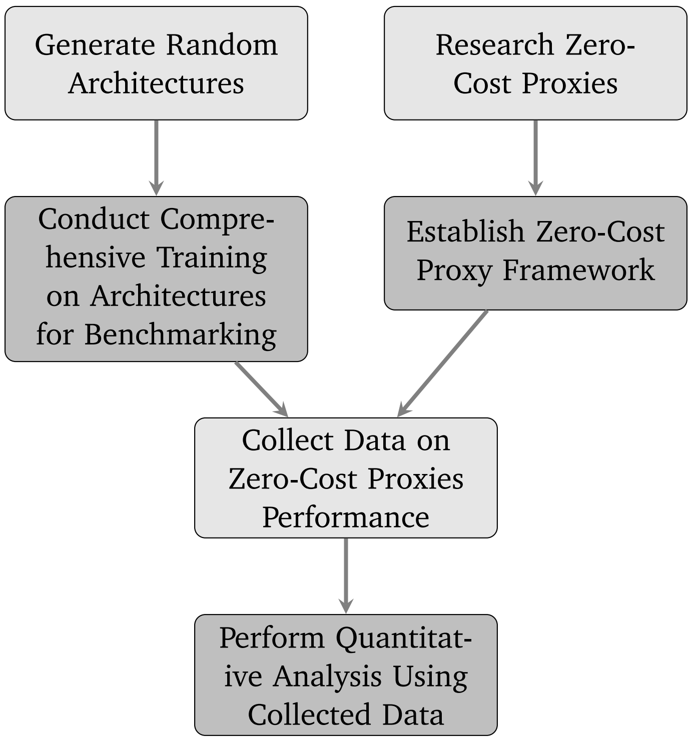

Generate Random Architectures
Research Zero-Cost Proxies
Conduct Comprehensive Training on Architectures for Benchmarking
Establish Zero-Cost Proxy Framework
Collect Data on Zero-Cost Proxies Performance
Perform Quantitative Analysis Using Collected Data

Figure 4.1: Flowchart illustrating the research process.

4.2 Dataset

Nanyang Technological University's Red Blue Green and Depth-information (NTU RGB+D) is a large-scale dataset for HAR. It contains over 56 thousand video samples and 4 million frames. The NTU RGB+D dataset contains 60 action classes such as drinking, eating, staggering, punching and kicking (Shahroudy et al., 2016).

The dataset is collected using Microsoft Kinect v2 sensors, in which they collected four modalities: depth maps, 3D joint information, RGB frames and IR sequences. In total, 25 body joints were captured in the dataset, in which each body joint is represented by x-coordinates, y-coordinates and depth (Shahroudy et al., 2016). The body joints are illustrated in figure 4.2.

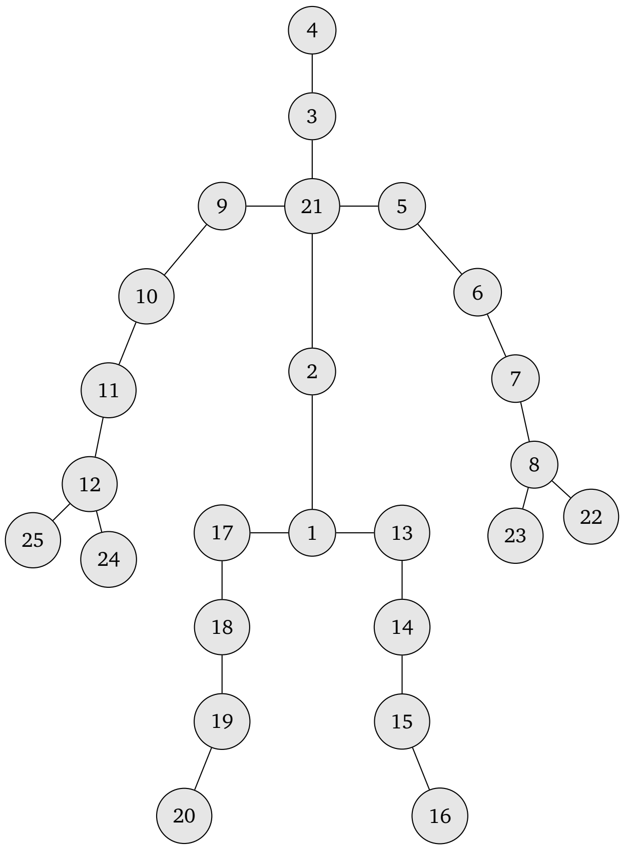


Figure 4.2: 1-base of the spine, 2-middle of the spine; 3-neck, 4-head, 5-left shoulder, 6-left elbow, 7-left wrist, 8-left hand, 9-right shoulder, 10-right elbow, 11-right wrist, 12-right hand, 13-left hip, 14-left knee, 15-left ankle, 16-left foot, 17-right hip, 18-right knee, 19-right ankle, 20-right foot, 21-spine, 22-tip of the left hand, 23-left thumb, 24-tip of the right hand, 25-right thumb

The NTU RGB+D dataset was later expanded into NTU RGB+D 120. The new dataset contains 114 480 video samples of 120 action classes, all from 106 separate human subjects.

The NTU RGB+D dataset has been employed in numerous studies, making it a well-established choice for research in the field of HAR (Cheng et al., 2020; Si et al., 2019; Yan et al., 2018). The dataset's large scale and diverse range of activities facilitate the training of models with high generalisation capabilities, which aligns with the goals of this study in achieving optimal final validation accuracy. Given the dataset's widespread application and success in HAR-related research, it is an appropriate foundation for this study.

4.3 Benchmark

Within NAS, a benchmark is a collection of already trained and evaluated models that can be queried to obtain their validation accuracy. Such benchmarks are crucial for research within NAS, as they spare the researchers thousands of GPU training time. Popular benchmarks within the literature are NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2020) and NAS-Bench-360 (Tu et al., 2021). These benchmarks contain thousand of trained and evaluated models on different datasets. However, to our knowledge, no similar benchmarks exist for GCN within HAR. Consequently, a benchmark had to be created for later experiments in the thesis.

4.3.1 GCN-NAS

The framework from the paper Learning Graph Convolutional Network for Skeleton-Based Human Action Recognition by Neural Searching (Peng et al., 2020) was used to create the benchmark. This approach offers ways to train individual models and run the provided NAS algorithm to find the best architectures for a given problem. In addition, the framework is developed to be used on the NTU RGB+D dataset.

The search space consists of eight function modules that can be applied in each network layer. To extract features U from a given node in a graph, the filter g_θ is approximated using Chebyshev polynomials with R-th order. The more significant R, the bigger the local receptive field of the GCN layer will be. Chebyshev polynomial is recursive and is given in equation 4.1.

$$\boldsymbol{Y} = \sum_{r=0}^{R} \theta_r' \boldsymbol{T}_r(\hat{\boldsymbol{L}})\boldsymbol{X}, \tag{4.1}$$

in which θ_r' is the Chebyshev coefficient. Recursively, the Chebyshev polynomial $T_r(\hat{L})$ is defined by equation 4.2.

$$\boldsymbol{T}_r(\hat{\boldsymbol{L}}) = 2\hat{\boldsymbol{L}}\boldsymbol{T}_{r-1}(\hat{\boldsymbol{L}}) - \boldsymbol{T}_{r-2}(\hat{\boldsymbol{L}}), \tag{4.2}$$

where $T_0 = 1$ and $t_1 = \hat{L}$. In the framework, $\hat{L}$ is normalised to the range $[-1, 1]$, where $\hat{L} = \frac{2L}{\lambda_{max}} - I_n$. Based on the work of (Kipf & Welling, 2016), $R = 1$ and $\lambda_{max} = 2$. This results in a first-order approximation of spectral graph convolutions, as shown in equation 4.3.

$$\begin{aligned} \boldsymbol{Y} &= \theta_0' \boldsymbol{X} + \theta_1' (\boldsymbol{L} + \boldsymbol{I}_n) \boldsymbol{X} \\ &= \theta_0' \boldsymbol{X} - \theta_1' (\boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}) \boldsymbol{X} \end{aligned} \tag{4.3}$$

Similarly, $\theta^{'}$ can be approximated with a unified parameter θ; this means that instead of using a separate parameter for $\theta^{'}$, it can be estimated using a single parameter, θ, which will be a simplified representation of $\theta^{'}$. Then, Y is given by equation 4.4.

$$\boldsymbol{Y} = \theta(\boldsymbol{I}_n + \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}) \boldsymbol{X} \tag{4.4}$$

Given the available search space, the networks may have different combinations of R-th order Chebyshev-polynomials. Intuitively, using higher R-th order, the filters of each layer will be able to aggregate more information from the neighbours as the hops increase.

In addition, the search space consists of three dynamic graph modules; spatial, temporal and spatial-temporal. The composition of the joints in a single frame is referred to as a spatial feature, and it is commonly represented as a set of 3D coordinates. These spatial features capture the posture and pose of the human body at a given time point, but they do not capture the motion over time. On the other hand, temporal features capture how spatial features change over time. Through a series of frames, these features capture movement and action information. Finally, spatial-temporal features combine the spatial and temporal aspects to capture both the posture and dynamics of motion over time.

To illustrate, table 4.1 shows the best-performing architecture found in Peng et al., 2020. The table illustrates that the different layers of the architecture prefer different mechanisms. For instance, the lower layers prefer all the dynamic modules, whereas the deeper layers prefer the temporal representation correlations (Peng et al., 2020).

Table 4.1: The searched architecture in the paper

M	L	L_n^4	L^4	L^3	L^2	$M(S)$	$M(T)$	$M(ST)$
k_1					✓	✓	✓	✓
k_2						✓	✓	✓
k_3						✓	✓	✓
k_4						✓	✓	✓
k_5					✓	✓	✓	
k_6					✓		✓	
k_7		✓			✓	✓	✓	✓
k_8					✓		✓	
k_9					✓		✓	
k_{10}							✓	

4.3.2 Definition of Fully Trained Models

Defining what constitutes a fully trained model is essential in the benchmark development context. Due to time and hardware resource limitations, a balance must be struck between training the models for optimal duration and ensuring the process remains feasible within the given constraints.

A threshold was determined based on the trade-offs between computational cost, training time and model performance. By setting this upper limit, it was possible to maintain a reasonable training duration while still allowing the models to reach a satisfactory level of performance. In the paper that introduced the framework, the authors (Peng et al., 2020) decided to stop the training process at 70 epochs, a practice adopted in this thesis.

Although the imposed threshold may not guarantee that every model reaches its absolute peak performance, it ensures that each model is trained sufficiently to compare relative performances. Furthermore, this definition of "fully trained" (up to 70 epochs) facilitates the efficient generation and evaluation of many models, which is crucial for successful benchmark development.

4.3.3 Experimental Setup and Benchmarking Methodology

The algorithm for generating and evaluating random architectures is described in algorithm 1. The benchmark consists of a large-scale dataset comprising a diverse set of fully trained architectures to facilitate a comprehensive investigation into the performance of zero-cost proxies.

To generate random architectures, the framework described in section 4.3.1 was utilised to develop an algorithm that avoids generating already generated architectures. The algorithm was constrained to create models consisting of four, six, eight, or ten layers to ensure a various range of architectures and explore a more comprehensive search space. In addition, other constraints were imposed, such as requiring at least one spatial, temporal, or spatial-temporal function module for effective feature extraction from the data.

Algorithm 1 Random Architecture Generation and Evaluation

```
Input: Number of architectures $N$
 1: Define constraints for layer count and function modules
 2: Initialize $i \leftarrow 0$
 3: while $i < N$ do
 4:     Generate a random architecture $A_i$ within constraints
 5:     while exists($A_i$) do
 6:         Generate a new random architecture $A_i$ within constraints
 7:     end while
 8:     Train $A_i$ for up to 70 epochs
 9:     Compute validation accuracy $V_i$ of $A_i$
10:     Store $A_i$ and $V_i$ in the benchmark dataset
11:     $i \leftarrow i + 1$
12: end while
```

The following hyperparameters were used for the training process:

Table 4.2: Hyperparameters for the training

Name	**Value**
weight decay	0.006
base learning rate	0.1
step	$[30, 45, 60]$
batch size	40
num. epochs	70

It should be noted that while previous work in this field has utilised significantly larger benchmark datasets, time and hardware constraints necessitated limiting the scope of this benchmark to produce baseline results.

To provide insight into the characteristics of the benchmark dataset, table 4.3 displays the minimum, average, and maximum values for training time and validation accuracy.

Table 4.3: Benchmark Dataset Statistics

Statistic	Minimum	Average	Maximum
Training time (hours)	4.35	13.16	53.30
Validation accuracy	0.92	0.94	0.95

4.4 Zero-Cost Proxies

This section aims to provide a comprehensive overview of the different zero-cost proxies implemented in the project and clarify their respective methodologies.

4.4.1 EPE-NAS

The main idea behind EPE-NAS is to assess how the network's gradients behave concerning the neural network's input. Thus, the need for training an entire network is eliminated (Lopes et al., 2021).

A linear map is defined as $w_i = f(x_i)$, in which x_i is a sample from a batch X. The linear map can be computed by equation 4.5.

$$J_i = \frac{\delta f(x_i)}{\delta x_i} \tag{4.5}$$

However, it is necessary to see how the network will behave through different data points from the dataset. Therefore, the Jacobian matrix w_i for different data points, $f(x_i)$, is calculated:

$$J = \begin{pmatrix} \frac{\delta f(x_1)}{\delta x_1} & \frac{\delta f(x_2)}{\delta x_2} & \cdots & \frac{\delta f(x_N)}{\delta x_N} \end{pmatrix}^T \tag{4.6}$$

The Jacobian matrix shows how the output of the network changes based on the input. After that, the correlation between data points of the same class is calculated to say how the untrained network can model complex functions (Lopes et al., 2021). The correlations are then used to calculate the covariance matrix for each class in the dataset:

$$C_{J_c} = (J - M_{J_c})(J - M_{J_c})^t \tag{4.7}$$

, where M_J is:

$$(M_{J_c})_{i,j} = \frac{1}{N} \sum_{n \in 1,\dots,N} J_{i,n} \tag{4.8}$$

Then, the correlation matrix is calculated for each class:

$$C_{J_c} = \frac{(C_{J_c})_{i,j}}{\sqrt{(C_{J_c})_{i,j} * (C_{J_c})_{j,j}}} \tag{4.9}$$

Each class is individually evaluated by summing the logarithm of the absolute values of the correlation matrix elements plus a small value $k = 1 \times 10^{-5}$.

$$E_c = \begin{cases} \sum_{i=1}^{N} \sum_{j=1}^{N} \; log(|(\sum_{J_c})_{i,j}| + K), & \text{if } C \leq \tau \\ \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \; log(|(\sum_{J_c})_{i,j}| + K)}{||\sum_{J_c}||}, & \text{otherwise} \end{cases} \tag{4.10}$$

The network's score can be calculated using the evaluations of the correlation matrices above:

$$s = \begin{cases} \sum_{t=1}^{C} |e_t|, & \text{if } C \leq \tau \\ \frac{\sum_{t=e}^{C} \sum_{j=i+1}^{C} |e_i - e_j|}{||e||}, & \text{otherwise} \end{cases} \tag{4.11}$$

e is the vector with the scores of the correlation matrices. For example, in the original paper (Lopes et al., 2021), they found empirically that $\tau = 100$.

4.4.2 Fisher

Initially introduced as a pruning method in (Theis et al., 2018), Fisher aims to remove feature maps or parameters that do not significantly contribute to the model's overall performance. This is achieved by eliminating activation channels and their corresponding parameters, estimated to have a negligible impact on the loss (Abdelfattah et al., 2021).

Theis et al., 2018 employed equation 4.12 to calculate the network score.

$$S_z(z) = \left(\frac{\partial L}{\partial z} z\right)^2, S_n = \sum_{i=1}^{M} S_z(z_i) \tag{4.12}$$

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, the activations can be captured and used to score the network.

4.4.3 Flops

As a zero-cost proxy, Flops is one of the simpler baselines as it simply goes through the model and calculates the number of floating point operations performed per second required to pass the input through the network. Therefore it could be considered a measure of the architecture's complexity (Ning et al., 2021).

4.4.4 Grad Norm

The Gradient Norm is a straightforward zero-cost proxy that computes the sum of the Euclidean norms of gradients after conducting a single forward and backward pass using a minibatch of training data (Abdelfattah et al., 2021). Given a vector of gradients $G = [g_1, g_2, ..., g_n]$, the gradient norm is calculated according to the formula presented in equation 4.13.

$$s = \sum_{i=1}^{n} \sqrt{g_i^2} \tag{4.13}$$

4.4.5 GradSign

GradSign uses the network's initial gradients like many other zero-cost proxies do. However, the proxy's central concept uses Ψ(psi) to examine the optimisation landscape of various networks at the level of specific training samples. Thus, GradSign differs from most other gradient-based methods because of its theoretical insights. Under plausible assumptions, these theoretical results show that a network with denser sample-wise local optima has lower training and generalisation losses.

The calculation of Ψ is often considered computationally infeasible for modern networks. To tackle this issue, the authors propose GradSign, which approximates Ψ. This method takes as input a mini-batch of sample-wise gradients that are evaluated at a randomly initialised point. Using this input, the method produces statistical evidence strongly associated with the well-trained predictive performance of the network, as measured by its accuracy on the entire dataset (Zhang & Jia, 2021).

4.4.6 Grasp

Gradient Signal Preservation (Grasp) is one of the pruning-at-initialisation techniques. Wang et al., 2020 argue that efficient training depends on preserving the gradient flow, from which the name gradient signal preservation comes. Therefore, 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).

4.4.7 Jacov

The Jacov-score is a zero-cost proxy that estimates architecture performance by analysing the gradient correlation structure in a neural network's output space (Mellor et al., 2020). The Jacov-score is computed using the eigenvalues of the correlation matrix of the batch Jacobian, as shown in equation 4.14.

$$s = -\sum_i \left(\log(v_i + k) + \frac{1}{v_i + k} \right) \tag{4.14}$$

where k is a small constant. The Jacov-score offers an efficient performance evaluation in NAS without extensive computation.

4.4.8 L2-norm

L2-norm is calculated by summing over the norm of all weights of each layer in a neural network.

4.4.9 NAS-WOT

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, forming a binary code defining 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, as given in equation 4.15.

$$K_H = \begin{bmatrix} N_A - d_H(c_1, c_1) & \dots & N_A - d_H(c_1, c_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(c_N, c_1) & \dots & N_A - d_H(c_N, c_N) \end{bmatrix}, \tag{4.15}$$

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.

$$s = |log|K_H|| \tag{4.16}$$

Because of matrix instability, a small epsilon was added to the diagonal to make it possible to calculate a determinant.

4.4.10 Params

Within a neural network, there exist several trainable parameters, which, similarly to Flops, are considered a measure of the architecture's complexity (Ning et al., 2021).

4.4.11 Plain

The Plain-score is simply looping through all layers of the network. If the layer's weight contains a gradient, the gradient is multiplied by the layer's weights. Ultimately, the method is summing the scores to obtain the final score.

4.4.12 Snip

The Single-shot Network Pruning (Snip) method was introduced by (Lee et al., 2018) as a pruning-at-initialisation technique, aiming to reduce the number of parameters within a neural network without affecting the accuracy during inference (Frankle et al., 2020). Frankle et al., 2020 proposed a data-dependent saliency criterion that identifies significant connections in the network related to the current task before training. This approach enables the pruning of redundant connections, thus reducing the neural network's parameter count. The connections are identified based on their influence on the loss function.

In the paper Zero-Cost Proxies for Lightweight NAS (Abdelfattah et al., 2021), Snip was repurposed as a zero-cost proxy. The score of the neural network was obtained by summing all parameters N in the model according to equation 4.17,

$$S_n = \sum_{i=1}^{N} S_p(\theta)_i \tag{4.17}$$

$$S_p(\theta) = \left| \frac{\partial L}{\partial \theta} \odot \theta \right| \tag{4.18}$$

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

4.4.13 Synflow

Iterative Synaptic Flow Pruning (Synflow), originally proposed in (Tanaka et al., 2020), is a method for parameter pruning. By pruning less significant parameters, highly sparse networks can be obtained with reduced network size while maintaining similar 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 makes the network untrainable, as evidenced by sudden drops in achievable accuracy (Tanaka et al., 2020). Therefore, it is crucial to avoid layer collapse since the network will become unusable if a layer is removed.

Synaptic saliency is a class of score metrics that can be expressed as the Hadamard product (equation 4.19).

$$S(\theta) = \frac{\partial R}{\partial \theta} \odot \theta \tag{4.19}$$

where R is a scalar loss function of the output y of a feed-forward network parameterized by θ (Tanaka et al., 2020). The conservation of synaptic saliency theorem demonstrates that the sum of synaptic saliency for all incoming parameters to a hidden neuron equals the sum of the synaptic saliency for the outgoing parameters from the hidden neuron (Tanaka et al., 2020). Consequently, the sum of synaptic saliency for all incoming parameters to a hidden layer equals the sum of the synaptic saliency for the outgoing parameters from the hidden layer. This can lead to issues when performing one-shot pruning, where a larger layer would have parameters with lower synaptic saliency. In comparison, a smaller layer would have parameters with 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, leading to layer collapse.

To prevent this, Synflow introduces iterative pruning (magnitude pruning) by recalculating the synaptic saliency for all parameters at each iteration. This re-evaluation process gives new scores to parameters with lower scores in previous iterations, eliminating the imbalance between smaller and larger layers and reducing the possibility of layer collapse. The Synflow algorithm is described in equation 4.20:

$$R_{SF} = \mathbb{1}^T \left(\prod_{l=1}^{L} |\theta^{[l]}| \right) \mathbb{1} \tag{4.20}$$

The algorithm takes an input of ones (e.g., an image where all the pixels have the value 1) and performs a forward pass through the network. The loss R is obtained by calculating the product of the output. This loss is then backpropagated through the network to the layers, and equation 4.19 is used to compute the synaptic saliency score for each parameter θ.

4.4.14 Zen-score

Zen-score is a measure of the expressivity of the network, which is computationally efficient and correlates positively with the model accuracy (Lin et al., 2021). Zen-score builds upon the expected Gaussian complexity (Φ-score), which can define the expressivity of a Vanilla Convolutional Neural Network (VCNN). In theoretical investigations, VCNN is a frequently used prototype in which several convolutional layers are stacked to form the main body of the network. Each layer has a convolutional operator followed by a ReLU activation function, where residual links and Batch Normalisation are excluded. A global average pool layer (GAP) brings the feature map resolution down to 1 x 1 before a fully-connected layer (Lin et al., 2021).

However, numerical overflow may occur for deep networks when directly calculating the Φ-score because of gradient explosion without batch normalisation layers (Lin et al., 2021). Zen-Score addresses this problem by adding batch normalisation layers and re-scaling the Φ-score by some constant.

4.4.15 Summary

Table 4.4 displays an overall view of the discussed zero-cost proxies, with the characteristics of each method's data dependence, data independence and type outlined.

Table 4.4: Summary of the implemented zero-cost proxies in the project

Zero-Cost Proxy	Data-dependent	Data-independent	Type
Epe-NAS	✓		Jacobian
Fisher	✓		Pruning-at-init
Flops	✓		Baseline
GradNorm	✓		Pruning-at-init
GradSign	✓		Gradient Sign
Grasp	✓		Pruning-at-init
Jacov	✓		Jacobian
L2-norm		✓	Baseline
Nwot	✓		Jacobian
Params		✓	Baseline
Plain	✓		Baseline
Snip	✓		Pruning-at-init
Synflow		✓	Pruning-at-init
Zen		✓	Piece. Lin.

4.5 Zero-Cost Framework

A novel Zero-Cost framework was created to calculate the scores of each zero-cost proxy. The framework, influenced by section 3.2.1, was developed as a versatile plug-and-play solution to be easily integrated into any NAS project. The primary goal of this framework is to provide an efficient and effective method of using zero-cost proxies for performance prediction.

Algorithm 2 outlines calculating zero-cost proxies for a given model. The algorithm takes four inputs: the model, a data loader, a loss function, and an optional override parameter. The model represents the neural network architecture under evaluation, while the data loader and the loss function are used for calculating the proxy metrics. The optional override parameter allows users to calculate only specified proxies selectively.

Algorithm 2 Calcuate Zero-Cost Proxies

```
Input: model, data_loader, loss_function, override
 1: score_store ← dict()
 2: proxies ← getProxies()
 3:
 4: for proxy in proxies do
 5:     if override is not empty then
 6:         if proxy not in override then
 7:             skip
 8:         end if
 9:     end if
10:
11:     start_time ← now()
12:     score ← proxy.calculateProxy(model, data_loader, loss_function)
13:     scor_store.proxy.time ← now() − start_time
14:     score_store.proxy.score ← score
15: end for
16:
17: return score_store
```

An empty *score_store* is initialised to keep track of the calculated scores and the time taken for each proxy. The algorithm then retrieves all the available proxy implementations using the *getProxies()* function. Next, a loop iterates through each proxy, and if the optional override parameter is not empty, it checks whether the current proxy is included in the override list. If the proxy is not in the list, it is skipped, and the loop continues to the next proxy.

For each selected proxy, the algorithm records the start time. Then, it calculates the proxy score using the *calculateProxy()* function with the model, data loader, and loss function as inputs. After calculating the score, the algorithm computes the time taken by subtracting the start time from the current time. Each proxy's calculated score and time are stored in the *score_store*.

4.6 Correlation

Calculating the correlation between the score of each zero-cost proxy and the validation accuracy is a logical approach to understanding the impact of the zero-cost proxies. Spearman Rank Correlation is a non-parametric statistical test that measures the strength and direction of the association between two ranked variables (Hauke & Kossowski, 2011). The Spearman Rank does not assume a linear relationship between the two variables. It is suitable for cases where the connection between the validation accuracy and the zero-cost proxy may not be linear. Additionally, the metric is less vulnerable to outliers because it focuses on the rank of the variables rather than their raw values.

Pearson correlation coefficient is the most commonly used correlation coefficient. The coefficient provides insights into the strength and direction of a linear relationship (Turney, 2022).

However, its utility for our project is limited due to several constraints. First, it assumes the variables in the dataset are normally distributed (Turney, 2022). The Shapiro-Wilk test is a statistical test used to check whether a sample of numbers has been drawn from a normally distributed population. The starting assumption (or null hypothesis) is that the population is normally distributed (Shaphiro & Wilk, 1965). The null hypothesis is disregarded if the test's calculated p-value is less than the predetermined alpha level, typically 0.05. Discarding the null hypothesis means we have enough evidence to conclude that the data set under investigation does not exhibit a normal distribution. The Shapiro-Wilk test result for each variable (zero-cost proxy) is shown in table 4.5.

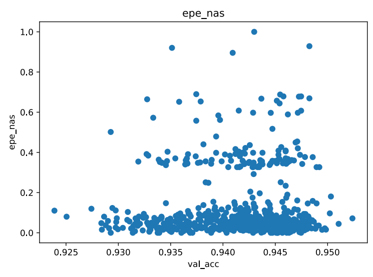
Table 4.5: P-values for each variable using the Shapiro-Wilk test

ZC-proxy	P-value	ZC-proxy	P-value
EPE-NAS	$< 1e-6$	NAS-WOT	0.003
Fisher	$< 1e-6$	Params	$< 1e-6$
Flops	$< 1e-6$	Plain	$< 1e-6$
Grad Norm	$< 1e-6$	Snip	$< 1e-6$
GradSign	$< 1e-6$	Synflow	$< 1e-6$
Grasp	$< 1e-6$	Val. acc	$< 1e-6$
Jacov	$< 1e-6$	Zen	$< 1e-6$
L2 norm	$< 1e-6$		

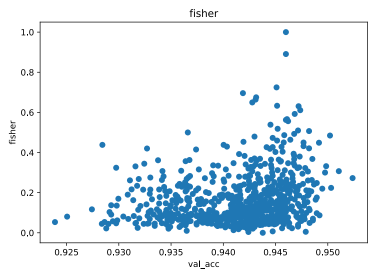
The p-values for all variables are smaller than the significance level ($\alpha = 0.05$), as seen in the table 4.5. Therefore, we can disprove the Shapiro-

Wilk test’s null hypothesis for each variable, showing that the distributions of those variables considerably differ from the normal distribution.

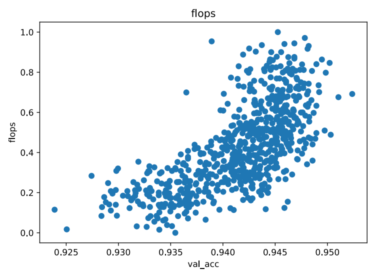
Further, Pearson’s correlation is highly sensitive to outliers. A few extreme observations can greatly distort the correlation coefficient, potentially leading to incorrect interpretations. This sensitivity is unwanted given that the zero-cost proxies may contain such outliers, as illustrated in figure 4.3, in which one can observe that the data contains outliers (i.e. Synflow in the bottom left corner).



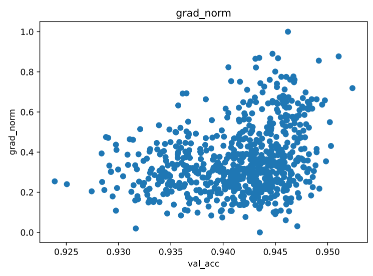
(a) EPE-NAS



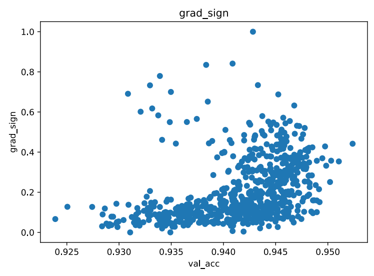
(b) Fisher



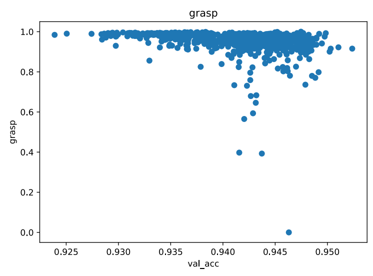
(c) Flops



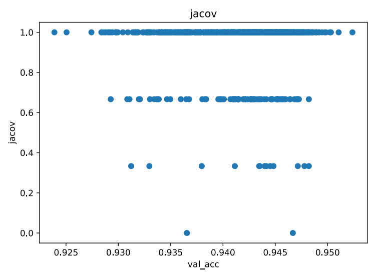
(d) Grad Norm



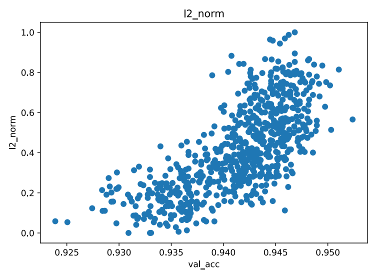
(e) GradSign



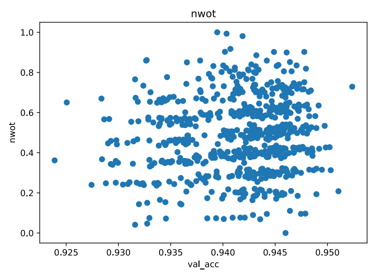
(f) Grasp



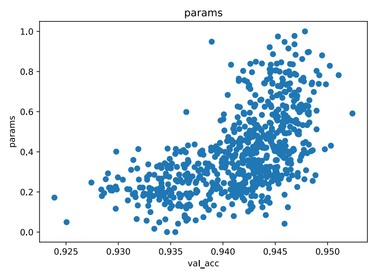
(g) Jacov



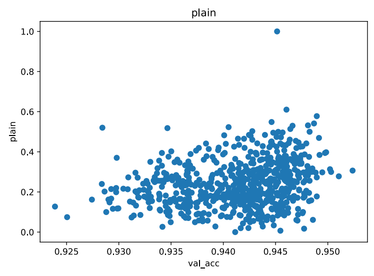
(h) L2 Norm



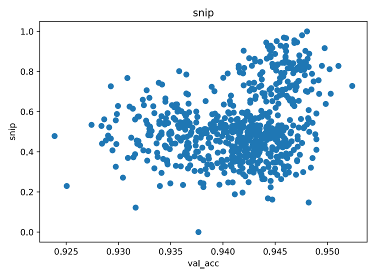
(i) NAS-WOT



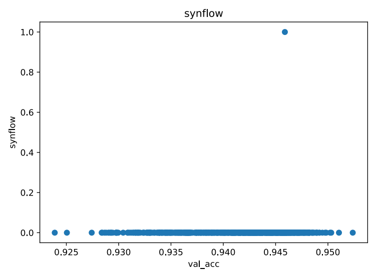
(j) Params



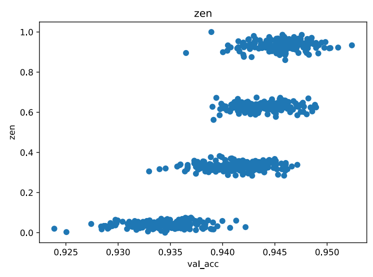
(k) Plain



(l) SNIP



(m) Synflow



(n) Zen

Figure 4.3: Each zero-cost proxy normalised with the validation accuracy

These findings are all pointing in favour of using the Spearman Rank. Lastly, the Pearson correlation coefficient relies on the actual values of the variables, whereas our focus is more on their rankings, which makes the Spearman rank correlation a more suitable measure for our project.

The Spearman Rank Correlation is denoted by the symbol ρ (rho) and ranges from -1 to 1, where -1 indicates a perfect negative relationship, 1 indicates a perfect positive relationship, and 0 shows no connection, as illustrated in figure 4.4 (Ltd, 2013).

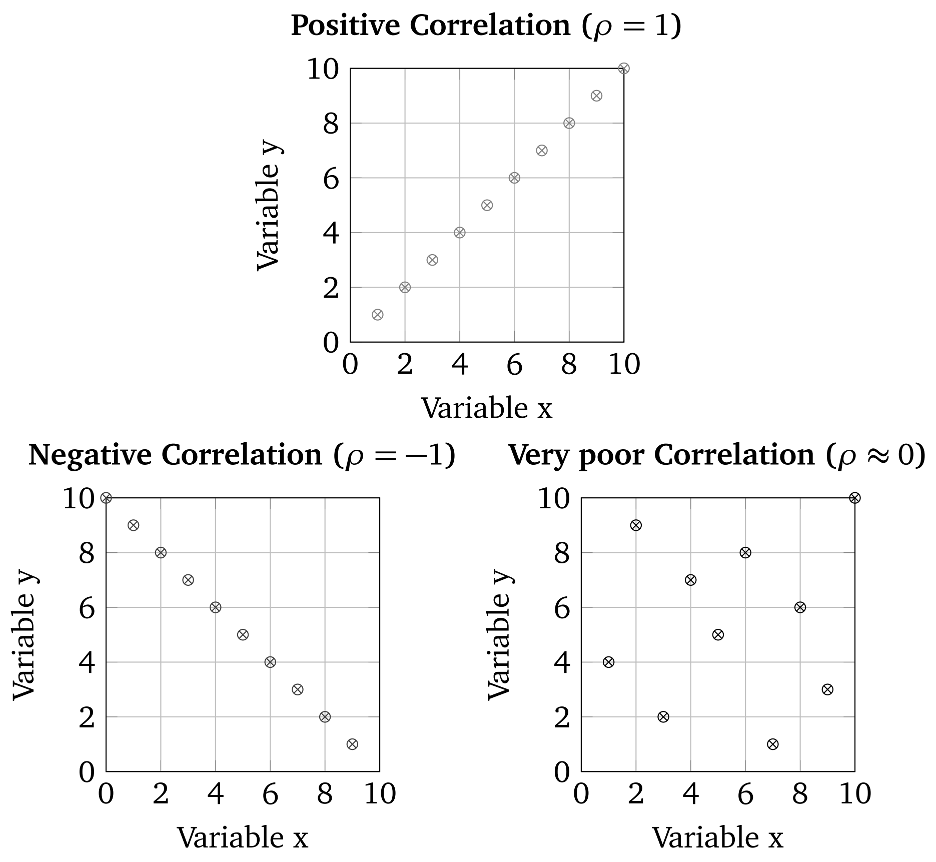


Figure 4.4: Scatter plots illustrating perfect positive, perfect negative correlations and very poor correlation

Spearman Rank Correlation works by calculating the difference in ranks of the two variables for each observation, then squaring these differences and summing them up, as given in equation 4.21.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}, \tag{4.21}$$

where d_i is the difference in ranks for each observation, and n is the number of observations.

According to statstutor, 2011, Spearman's correlation can generally be described using guidelines from table 4.6.

Table 4.6: Guidelines for interpreting Spearman's correlation

Correlation coefficient	Strength of correlation
.00-.19	Very weak
.20-.39	Weak
.40-.59	Moderate
.60-.79	Strong
.80-1.0	Very strong

In the context of zero-cost proxies and validation accuracy, Spearman Rank Correlation can be used to determine whether there is a monotonic relationship between the two. A positive correlation would indicate that as the zero-cost proxy improves, so does the validation accuracy, while a negative correlation would imply an inverse relationship. A correlation near zero would suggest that the two variables have little or no association.

4.7 Exploration of Zero-Cost Proxies via Warmup Strategy

4.7.1 Theoretical and Practical Considerations

In this section, we introduce the concept of a warmup strategy, which aims to determine an optimal epoch threshold that can provide a reliable estimation of the relative performance of different architectures. This contrast with the naive NAS approach of training numerous architectures, which can be computationally prohibitive.

The underlying principle for this strategy is based on the assumption that training an architecture for x epochs offers a more efficient evaluation than training the same architecture for y epochs if $x < y$. By identifying an appropriate warmup threshold, researchers can effectively balance the trade-off between computational expense and the accuracy of architecture performance estimation.

To achieve this, each architecture is trained for a predetermined number of warmup epochs, and the zero-cost proxies are calculated at each epoch.

Then, for every epoch, the Spearman Rank correlation coefficient between the zero-cost proxies and the final validation accuracy for all architectures is calculated. The epoch with the highest correlation is considered the optimal warmup point.

4.8 Combining Zero-Cost Proxies

Upon analysing the outcomes from addressing research question 1 and research question 2, it became necessary to reconsider the experimental plan. The insights gained from the initial results prompted further exploration into utilising zero-cost proxies. Consequently, a decision was made to undertake additional ad hoc experiments to delve deeper into this area of investigation.

4.8.1 Majority Vote Method

The majority vote method, introduced by Abdelfattah et al., 2021, is an approach for ranking candidate architectures based on the combined results of multiple zero-cost proxy metrics. This technique consolidates the rankings generated by different zero-cost proxy metrics, and the architectures are ranked according to the majority vote derived from these metrics. The algorithm of the voting method is outlined in algorithms 3 and 4.

Algorithm 3 Voting algorithm

```
function VOTE($mets$, $gt$)
    $numpos \leftarrow 0$
    for each element $m$ in $mets$ do
        if $m > 0$ then
            Increment $numpos$ by 1
        end if
    end for
    if majority of elements in $mets$ are positive then
        $sign \leftarrow +1$
    else
        $sign \leftarrow -1$
    end if
    return $sign * gt$
end function
```

Algorithm 4 Voting Accuracy for Metric Combinations

```
1: function CALC($acc$, $metrics$, $comb$)
2:     $tot \leftarrow 0$, $right \leftarrow 0$
3:     for each pair of distinct indices $i$ and $j$ do
4:         $diff \leftarrow acc[i] - acc[j]$
5:         if $diff \neq 0$ then
6:             $diffsyn$                              /* Initialize an empty list */
7:             for each metric $m$ in $comb$ do
8:                 $diffsyn \leftarrow metrics[m][i] - metrics[m][j]$
9:             end for
10:            /* Check if $diffsyn$ and $diff$ have same sign */
11:            $same\_sign \leftarrow$ VOTE($diffsyn$, $diff$)
12:            if $same\_sign > 0$ then
13:                Increment $right$
14:            end if
15:            Increment $tot$
16:        end if
17:    end for
18:    $votes \leftarrow \frac{right}{tot}$                  /* Calculate the voting accuracy */
19:    return ($comb, votes$)
20: end function
```

Given that it is uncertain which combination of zero-cost proxies would yield the best results, we developed a function to generate all possible subsets of the 14 zero-cost proxy metrics. Subsequently, the majority vote for each subset was calculated and compared to the ground truth provided by the validation accuracy of every trained architecture in the benchmark. For example, table 4.7 displays two architectures with given values for three zero-cost proxies (Synflow, Snip and Grad Sign) and their validation accuracy.

Table 4.7: Example of two architectures with validation accuracy and zero-cost proxy metrics.

Architecture	Metrics
1	Validation Accuracy: 0.85
	Synflow: 0.62
	Snip: 0.75
	Grad Sign: 0.28
2	Validation Accuracy: 0.89
	Synflow: 0.48
	Snip: 0.82
	Grad Sign: 0.36

By examining the validation accuracy, one can deduce that Architecture 2 is superior to Architecture 1. Upon analysing the computed zero-cost proxy metrics, the following observations can be made:

- **Synflow**: Architecture 1 (0.62) > Architecture 2 (0.48)
- **Snip**: Architecture 1 (0.75) < Architecture 2 (0.82)
- **GradSign**: Architecture 1 (0.28) < Architecture 2 (0.36)

In this case, one positive difference (Synflow) and two negative differences (Snip and Grad Sign). Consequently, the majority vote favours Architecture 2, consistent with the validation accuracy.

4.8.2 Weighted Arithmetic Mean

The weighted arithmetic mean, as described in (*Weighted Arithmetic Mean*, 2008), is a widely used technique in statistics and data analysis. This study applied the weighted arithmetic mean method to the zero-cost proxies using Spearman's rank correlation as weights for ranking various architectures. The algorithm is outlined in algorithm 5.

Algorithm 5 Weighted Arithmetic Mean for datapoint d

```
Input: Zero-cost proxies $P_d = \{p_1, p_2, \ldots, p_n\}$
Input: Validation accuracy $A_d$
 1: Normalise zero-cost proxies: $P_{d_{norm}} = \{p_{1_{norm}}, p_{2_{norm}}, \ldots, p_{n_{norm}}\}$
 2: for $i = 1$ to $n$ do
 3:     Calculate Spearman's rank correlation $r_i$ between $p_{i_{norm}}$ and $A_d$
 4:     Assign weight $w_i = r_i$
 5: end for
 6: Initialize: $\text{Score}_d \leftarrow 0$, $total\_weight \leftarrow 0$
 7: for $i = 1$ to $n$ do
 8:     $\text{Score}_d \leftarrow \text{Score}_d + (p_{i_{norm}} * w_i)$
 9:     $total\_weight \leftarrow total\_weight + w_i$
10: end for
11: $\text{Score}_d \leftarrow \frac{\text{Score}_d}{total\_weight}$
12: Calculate correlation between $\text{Score}_d$ and $A_d$ using Spearman's rank
    correlation
```

Each zero-cost proxy value is first normalised with min-max normalisation, ensuring they are on a comparable scale for accurate comparison and combination. Next, weights were assigned to each zero-cost proxy based on their performance. Finally, Spearman's rank correlation evaluated the correlation between each proxy and the validation accuracy. Higher correlation values indicated better performance and proxies with stronger correlations received larger weights.

For the weighted arithmetic mean calculation, the normalised value of each zero-cost proxy was multiplied by its respective weight for every data point, and the products were summed. The combined score was then calculated by dividing the sum of these products by the total sum of the weights using equation 4.22.

$$\text{Score} = \frac{\sum x_i * w_i}{\sum w_i}, \tag{4.22}$$

where w_i is the weight assigned to the i-th zero-cost proxy, x_i represents the normalised value of the i-th zero-cost proxy. The sums are calculated using all zero-cost proxies.

After obtaining the combined scores, the architectures were ranked based on these scores, with higher scores indicating better-performing architectures. Finally, the effectiveness of the weighted arithmetic mean approach was evaluated by calculating the correlation between the weighted arithmetic mean score and the validation accuracy using Spearman's rank correlation coefficient.

Chapter 5

Results

This chapter outlines the study's findings, responding to the defined research questions. The first section evaluates the capability of zero-cost proxies in ranking GCN architectures concerning their validation accuracy. Subsequently, the correlation analysis during the warm-up phase of GCN training is identified. Lastly, sections 5.1.1 and 5.2 examines techniques for combining zero-cost proxies to improve efficiency and accuracy in NAS algorithms.

5.1 Correlation Analysis

Establishing a benchmark provided the foundation for investigating the correlation between various zero-cost proxies and the ground truth represented by the validation accuracy. The Spearman Rank Correlation was employed to measure the correlation, as discussed in section 4.6.

Table 5.1 displays the obtained results, utilising the 693 architectures from the created benchmark.

Table 5.1: Correlation coefficients between proxy scores and model performance before training

ZC-proxy	ρ
EPE-NAS	0.0090
Fisher	0.2405
Flops	0.7241
Grad Norm	0.2653
GradSign	0.4979
Grasp	−0.4244
Jacov	−0.0465
L2 norm	0.7325
NAS-WOT	0.1214
Params	0.6164
Plain	0.2490
Snip	0.2468
Synflow	0.7599
Zen	**0.7827**

The results in table 5.1 reveal that specific proxies correlate strongly with the validation accuracy, while others demonstrate weaker correlations. Table 5.1 exhibits that Zen has the highest correlation with model performance, with a coefficient of 0.7827. Synflow is also performing well, with a coefficient of 0.7599. This suggests that Zen and Synflow are the most effective zero-cost proxies among the ones considered in this study.

Conversely, the EPE-NAS and Jacov proxies exhibit a weak correlation with the model performance, with coefficients of 0.0090 and −0.0465, respectively.

It is also worth noting that the L2-norm, Flops, and params proxies show relatively strong correlations with coefficients of 0.7325, 0.7241, and 0.6164, respectively. Although these proxies might be less effective than Zen and Synflow, they still demonstrate considerable potential for predicting model performance.

The findings of this correlation analysis provide valuable insights into the predictive capacity of each proxy in relation to the ground truth. This understanding can inform the development of more efficient and effective NAS approaches, thereby reducing computational demands and enabling more rapid progress in the field.

5.1.1 Vote

The results will only display the top-5 combinations, as showing all possible variations would not be feasible. From the results shown in table 5.2, the top-performing combinations of metrics exhibit almost the same custom correlation measure (majority vote score) of > 0.788. This suggests that these combinations of metrics perform similarly in terms of the agreement between the majority of metrics and accuracy differences.

Table 5.2: Vote scores

ZC-proxies	**Value**
Synflow, Zen, Fisher	**0.7899**
Synflow, Zen, NAS-WOT	0.7888
Synflow, Zen, Flops	0.7885
Synflow, Zen, Plain	0.7883
Synflow, Zen, Grad Norm	0.7883

It is important to note that these results are based on the custom correlation measure (see section 4.8.1), which differs from standard correlation measures such as Spearman's rank correlation. The custom correlation measure captures the extent to which the majority of metric differences agree with the differences in dataset accuracies. A higher correlation value indicates a stronger agreement, while a lower correlation value suggests a weaker agreement.

5.1.2 Warmup

Figures 5.1 and 5.2 presents the variation in correlation during the initial ten epochs, with epoch -1 representing the starting correlations (same as table 5.1). As shown in the figures, Synflow and Zen exhibit the highest performance, maintaining a stable correlation at ≈ 0.8 throughout the observed epochs. A minor increment in correlation can be observed, although its impact is relatively insignificant. Furthermore, Params, L2-norm, and Flops exhibit consistently strong performance with a correlation coefficient (ρ) exceeding 0.6 across all epochs under consideration.

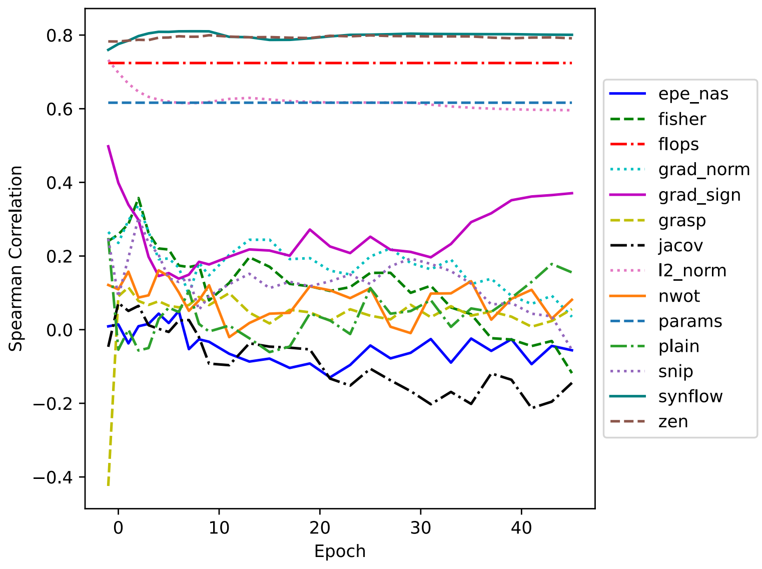


Figure 5.1: Correlations for all zero-cost proxies over multiple epochs

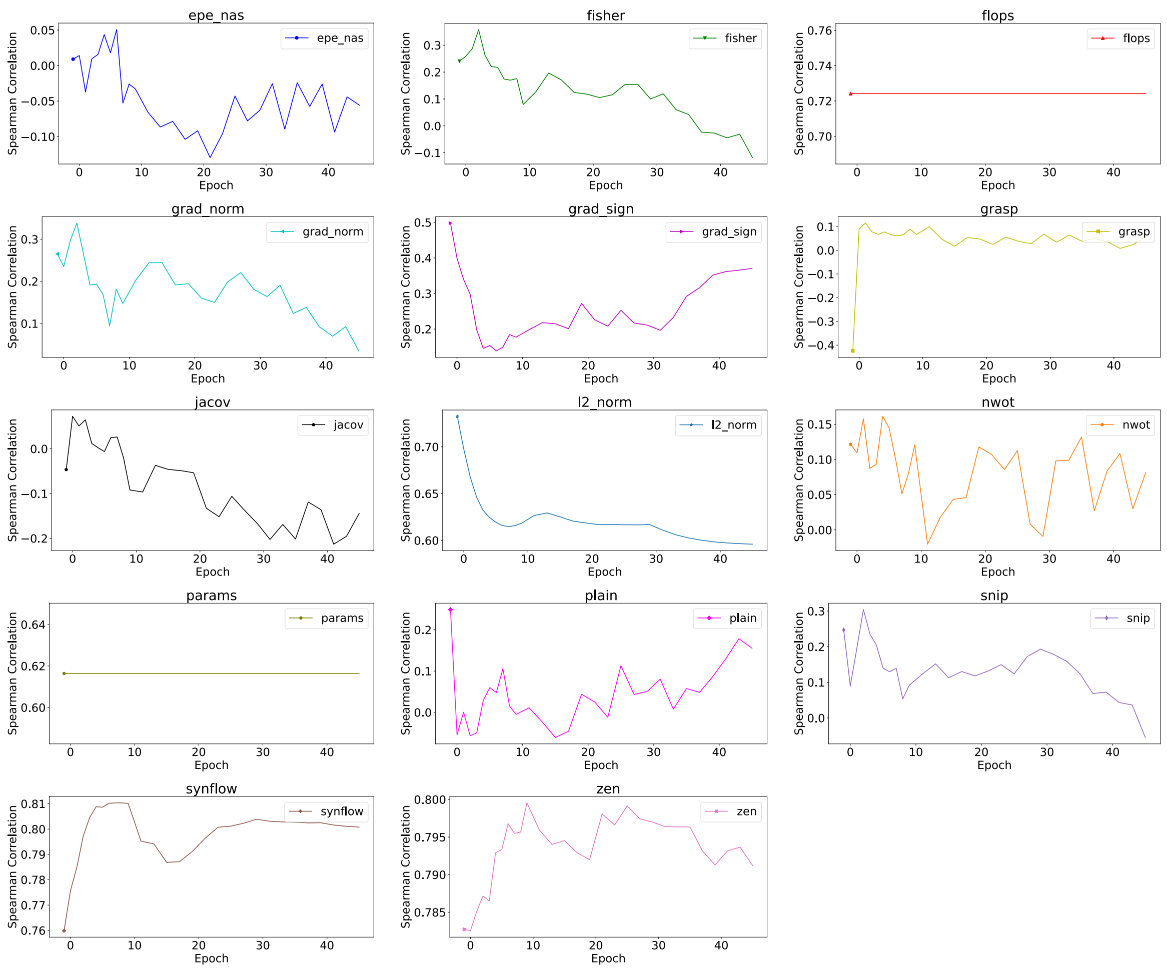


Figure 5.2: Each zero-cost proxy with the correlation over multiple epochs

5.2 Weighted Arithmetic Mean

Figure 5.3 displays a scatter plot illustrating the relationship between validation accuracy and the weighted arithmetic mean. Every data point displayed within the graph corresponds to a unique architecture evaluated during the experiment. A dotted red line represents the best-fit linear regression line through the data points. Additionally, the graph includes Spearman's rank correlation coefficient in the title (0.766), which measures the monotonic relationship between the validation accuracy and the weighted arithmetic mean.

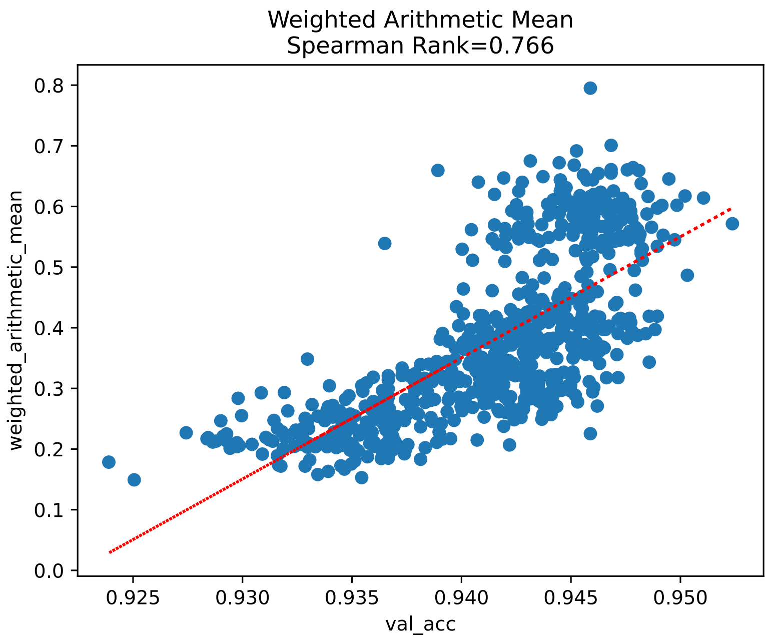


Figure 5.3: Weighted Arithmetic Mean

Chapter 6

Discussion

This chapter discusses the study's findings on improving and optimising the efficiency of NAS with GCN for HAR. The chapter aims to analyse and interpret the results of each research question and their implications in the context of the overall goal. We also discuss the significance and potential impact of the findings on the field of NAS with GCN for HAR and acknowledge the study's limitations. In addition, a discussion on the overall environmental implication is included.

6.1 Interpretation and Significance of Results

6.1.1 RQ1 How well can different zero-cost proxies rank GCN architectures compare to their validation accuracy?

Section 5.1 highlights the results for the given research question in which the correlation between the different zero-cost proxies and the validation accuracy is presented. Positive correlation values indicate that as the proxy metric increases, the validation accuracy also increases, whereas negative values indicate the opposite. The magnitude and sign of the Spearman rank correlation coefficient, also known as Spearman's rho (ρ), can be used to interpret the strength and direction of the association between two ranked variables (Pallant, 2016).

The results showed that Zen and Synflow performed the best with a spearman ρ of 0.7827 and 0.7599, respectively. This is a strong, close to very strong, correlation, as indicated by statstutor, 2011, demonstrating that the zero-cost proxies can confidently predict model performance without incurring significant computational overhead. The strength of these correlations suggests that both Zen and Synflow effectively capture properties critical for achieving high validation accuracy in neural networks in this context. Furthermore, given the variations in the validation accuracy in the benchmark $[0.92, 0.95]$, and the provided correlations, both Synflow and Zen are sensitive to slight differences in performance among the models, effectively ranking them based on their validation accuracy.

For comparison, Abdelfattah et al., 2021 reported the results on zero-cost proxies on NAS-Bench 201 given in table 6.1. Note that this is a benchmark for a different task (image classification).

Table 6.1: Comparison of various methods on different datasets

Dataset	Grad Norm	Snip	Grasp	Fisher	Synflow	Jacov
CIFAR-10	0.58	0.58	0.48	0.36	**0.74**	0.73
CIFAR-100	0.64	0.63	0.54	0.39	**0.76**	0.71
ImageNet16-120	0.58	0.58	0.56	0.33	**0.75**	0.71

The results in table 6.1 from Abdelfattah et al., 2021 show that the performance of the zero-cost proxies varies across different datasets. It is important to note that NAS-Bench 201 is a different benchmark and has other properties than the dataset used in this study. Nonetheless, a comparison can provide valuable insights into the generalisability and robustness of these methods across various tasks.

From table 6.1, it is evident that Synflow performs consistently well across all three datasets, with correlation coefficients of 0.74, 0.76, and 0.75 for CIFAR-10, CIFAR-100, and ImageNet16-120, respectively. The results support the thesis' finding that Synflow is a reliable and robust predictor of model performance. Similarly, Grad Norm and Snip exhibit relatively high correlations across the datasets, although not as strong as Synflow. This suggests that these methods may also provide valuable information when predicting model performance but with varying effectiveness.

The practical implications of this thesis's findings imply that Synflow and Zen could be valuable components of a NAS algorithm in the context of GCN for HAR tasks, given their strong correlations with model performance. However, it is essential to note that the current study utilised a specific framework designed for GCN HAR tasks with the NTU RGB+D dataset. This means that the demonstrated efficiency of using zero-cost proxies, such as Synflow and Zen, is effective for this case.

However, the generalisability of these results to other tasks, datasets, or frameworks remains to be established. Further research is needed to assess the effectiveness and applicability of Synflow, Zen, and other zero-cost proxies across a broader range of neural network architectures and tasks. By doing so, the academic community can better understand the potential benefits and limitations of incorporating zero-cost proxies into NAS algorithms for various problem domains.

The significance of these findings lies in their potential to improve the efficiency and effectiveness of NAS methods by identifying zero-cost proxies with solid predictive capacities. Furthermore, by understanding which proxies are more reliable in ranking GCN architectures to their validation accuracy, researchers can prioritise their use in NAS algorithms and reduce the overall computational demands of the architecture search process.

This is particularly important given neural networks' growing scale and complexity, often requiring significant computational resources to explore and evaluate. Focusing on zero-cost proxies with strong correlations can significantly reduce the time required to identify optimal architectures.

Furthermore, these findings stimulate additional research into developing and refining zero-cost proxies that exhibit stronger correlations with validation accuracy. This could lead to the discovery of new proxies that further improve the efficiency and accuracy of NAS methods, thus accelerating progress in the field.

Finally, the reduced computational demands also result in lessening the environmental impact. One can significantly lower the process's energy consumption and associated carbon footprint by decreasing the training time and computational resources required for NAS. This aligns with the growing global concern for sustainable practices in artificial intelligence research and development.

6.1.2 RQ2 How early can we identify the correlation between zero-cost proxies and validation accuracy during the warm-up phase of GCN training to potentially halt the training process sooner?

The results presented in chapter 5 provide insights into how the relationship between zero-cost proxies and validation accuracy evolves during the warmup phase of GCN training. By analysing the Spearman Rank correlation coefficients between the 14 proxies and the model performance for each epoch (see figure 5.1), one can determine the optimal warmup point for each architecture.

The correlation coefficients reveal that some zero-cost proxies, such as Synflow, Flops, Params, and L2 norm, consistently show strong positive correlations with the validation accuracy throughout the warmup phase. On the other hand, some proxies, like Grasp, GradSign, and Jacov, display weak correlations. This indicates that specific zero-cost proxies are more reliable indicators of an architecture's potential performance.

Moreover, the optimal warmup points vary across the different zero-cost proxies, as evident from the highest correlation coefficients achieved at different epochs. For example, Synflow achieves its highest correlation at epoch 8, while L2 norm peaks at initialisation. This indicates that there is no optimal warmup point for all proxies, and the choice of a warmup point depends on the specific proxy used for performance estimation.

The significance of the results is that there is no improvement in using warmup regarding the correlation and that the zero-cost proxies are most effective when the network is initialised. From an academic perspective, it is crucial to note this study's implications on NAS algorithms. The findings suggest that the necessity of training architectures within a NAS algorithm may be an oversimplified assumption. Given that zero-cost proxies were found to be most effective at the initialisation stage, one can assume that the requirement of training architectures may be less vital to the successful implementation of a NAS algorithm than previously believed.

6.1.3 RQ3 How can we effectively combine zero-cost proxies using various techniques to enhance the efficiency and accuracy of architecture search in NAS algorithms?

White et al., 2022 argued that zero-cost proxies have untapped potential and referred to preliminary research, which showed zero-cost proxies shine when combined rather than individually. How to combine them is another question that one should raise, as there are many different opportunities to discover. This study investigated the effectiveness of combining zero-cost proxies to improve the efficiency and accuracy of architecture search in NAS algorithms, as posed in Research Question 3.

Considering the high Spearman's rank correlation for Zen and Synflow, combining methods has little room for improvement, as the correlation is already strong. However, it is important to note that by using methods which combines zero-cost proxies, one can utilise properties of different proxies, which can maximise the different metric's variety to create a better correlation.

The presented methods in this study are relatively simple yet efficient methods discovered. However, various other methods can be explored for combining zero-cost proxies. One approach is supervised learning, which involves training a model on labelled input-output pairs to learn the underlying relationship between input features and target outputs. An approach in the context of NAS is developing a machine learning model specifically designed to learn how to rank architectures, utilising zero-cost proxies as input features. This method aims to leverage the information the zero-cost proxies provide to establish a relative ranking of architectures, guiding the search process towards the most promising candidates effectively and efficiently.

RankNet is a pairwise ranking model based on a neural network architecture proposed by Burges et al., 2005 in the paper *Learning to rank using gradient descent*. The model is designed to learn how to rank entities by minimising a pairwise loss function, typically using cross-entropy loss or a variant thereof. The core idea behind RankNet is to represent entities using input features and predict their relative ranking based on pairwise comparisons. During training, the model receives pairs of entities represented by their respective feature vectors and learns the underlying relationship between these features and the desired ranking. The training data includes binary labels indicating which entity in a pair is superior based on their ranking.

Applying RankNet to NAS algorithms offers several advantages. First, by leveraging the pairwise ranking approach, RankNet allows for a more nuanced understanding of the relative performance of different architectures based on their zero-cost proxies. This is particularly beneficial in the context of NAS, where the search space is vast and evaluating each architecture based on its actual performance is computationally expensive.

Furthermore, by training a neural network to learn the relationships between zero-cost proxies and architecture performance, RankNet has the potential to uncover hidden patterns and interactions among these proxies that may not be immediately evident. This can lead to a more accurate and efficient ranking of candidate architectures, reducing the search time and computational resources required for NAS.

6.2 Limitations

6.2.1 Dataset size

One of the main limitations of this study is the relatively small dataset size. Comparable studies like (Abdelfattah et al., 2021; White et al., 2022) used state-of-the-art benchmarks consisting of thousands of fully trained architectures. The developed dataset in this thesis consists of 693 trained architectures, each with their respective validation accuracy. For each architecture, the score of 14 different zero-cost proxies is calculated and then used for various analyses, such as determining the correlation between the proxies and the validation accuracy.

Although this dataset provides insights into the relationship between zero-cost proxies and the performance of GCN models within HAR tasks, the small number of trained architectures may limit the generalisability of the findings. A larger dataset with more architectures could reveal more subtle relationships between the proxies and the validation accuracy, leading to more accurate conclusions.

The relatively small dataset size may also affect the statistical significance of our results. Additionally, a larger dataset would allow for a more robust exploration of potential relationships between different architectures and proxy scores.

Future work should expand the dataset to address these limitations by incorporating more trained architectures with varying characteristics. This would increase the generalisability of the findings and enable a more comprehensive understanding of the relationships between zero-cost proxies and the performance of GCN models within HAR tasks.

6.2.2 Limitations of Relying Solely on the GCN-NAS Framework

Using the GCN-NAS framework for exploring the effectiveness of zero-cost proxies offers a valuable starting point. However, focusing exclusively on this framework introduces limitations that may affect the approach's generalisability, robustness, and thoroughness. This section will discuss these limitations and their potential impact on the approach's applicability to other GCN-NAS frameworks.

By concentrating on the GCN-NAS framework, the approach may become tailored to the specific characteristics of this framework, thereby restricting its generalisability to other GCN-NAS frameworks. This limitation could compromise the method's effectiveness when applied to alternative GCN-NAS frameworks with different search spaces, function modules, or optimisation techniques.

The unavailability of multiple GCN-NAS frameworks for evaluation prevents a comprehensive comparative analysis, making it challenging to identify the relative strengths and weaknesses of the proposed NAS acceleration technique. However, with the ability to compare performance across frameworks, pinpointing areas of improvement or potential pitfalls in the acceleration method becomes more effortless.

Evaluating the zero-cost proxies on a single GCN-NAS framework restricts the ability to assess the technique's robustness. The proxies should be tested across multiple frameworks to determine their adaptability to various search spaces, hyperparameter configurations, and problem domains. This lack of validation can lead to overestimating the technique's performance, potentially concealing its weaknesses.

It should be noted that an attempt to implement the Zero-Cost Framework (section 4.5) into a new GCN-NAS framework developed by the DeepInMotion team at NTNU and St. Olavs Hospital was conducted. However, as this is outside this thesis's scope, the results are not included. The initial experiments exhibit a Spearman Rank $\rho \gtrapprox 0.8$ for multiple proxies. The GCN-NAS framework is still under development and can be explored further when completed. The initial results can be found in appendix A.

Future research could consider utilising multiple GCN-NAS frameworks to address these limitations, exploring diverse search strategies and validating the technique across different problem domains.

6.3 Environmental Implications

6.3.1 Energy consumption / Creating benchmark

As elaborated in section 4.3.3, 693 neural network architectures were trained and evaluated to obtain their validation accuracy for later use in experiments. The training process required a significant investment of computational resources. A script was utilised to capture the total training time in seconds to measure the training time for each architecture accurately.

The total training time, represented by T, was captured to be 32833726 seconds. To provide a more comprehensible measure, the seconds were converted into GPU days, where one GPU day represents the continous use of a single GPU for 24 hours. By dividing T by the number of seconds in an hour (3600), and then again by the number of hours in a day (24), we could obtain the number of GPU days:

$$\text{GPU days} = \frac{T}{3600 \times 24} = \frac{32833726}{3600 \times 24} \approx 380$$

So in total, 380 GPU days were used to obtain the benchmark for the experiments.

For contextual comparison, NASBench-101 is a benchmark for NAS introduced by (Ying et al., 2019). The benchmark contains many CNN architectures trained and evaluated on the CIFAR-10 dataset using over 100 TPU [1] years of computation time.

6.3.2 Reduced Search Time and Future Benefits

Creating a benchmark involves substantial time and energy investments, as demonstrated in this study and other benchmarks (Dong & Yang, 2020; Tu et al., 2021; Ying et al., 2019). In addition, training and evaluating diverse neural network architectures requires significant computational resources, leading to increased energy consumption and longer training durations. Therefore, although the research required a considerable investment in GPU days, the long-term benefits of this investment should be considered.

The most naive approach for any general NAS algorithm is to generate a set of candidate architectures, train them until convergence, and find the best-performing architecture. However, this approach is computationally infeasible with a large search space dimension. In addition, this specific approach exhibits a substantial carbon footprint because of its extensive usage of GPU days. This study and similar studies (Abdelfattah et al., 2021; White et al., 2022) have discussed how zero-cost proxies might be used within a NAS algorithm to speed up the search process significantly. Consequently, creating a benchmark for exploring the possibilities of reducing the computationally heavy search process should be considered as a small investment for a more significant impact. The investment can be perceived as a stepping stone towards developing more efficient and sustainable approaches in the long run.

[1]Tensor Processing Unit introduced by Google purposely designed for machine learning workloads

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has explored the application of zero-cost proxies in NAS with GCN for HAR tasks. It has been demonstrated through comprehensive analysis and experimentation that using zero-cost proxies can enhance the efficiency of NAS algorithms. The experiments show that the best-performing zero-cost proxies exhibit a strong to very strong correlation of a Spearman ρ of ≈ 0.8, indicating that some of the zero-cost proxies can confidently rank architectures. Since the benchmark consists of high-performing architectures, the results imply that the best zero-cost proxies correlate strongly with high-performing architectures. In addition, the thesis showed no improvement regarding warm-up, as no significant correlation was discovered after training the architectures compared to the initialisation. The results showed that some of the proxies did improve, but considering it requires training and the improvement was not significant, it can be concluded that further research and optimisation are necessary to enhance their performance. Finally, vote and weighted arithmetic mean were implemented to combine zero-cost proxies, and the result showed that the combination yields potential but not any substantial improvement compared to using each zero-cost proxy individually.

The limitations of this study have been acknowledged, including the relatively small dataset size and the dependence on the GCN-NAS framework for NAS acceleration. These limitations emphasise the need for continued research and validation to ensure the generalisability and robustness of the findings. Furthermore, the significance of considering the environmental implications of this work has been emphasised, as the advancement of more efficient and sustainable algorithms and techniques is vital for addressing the escalating concerns surrounding the carbon footprint and energy consumption in research.

With regards to the overall goal of the thesis, namely to *improve and optimise the efficiency of neural architecture search with graph convolutional networks for human action recognition*, the thesis has shown that there is great potential in using zero-cost proxies within a NAS algorithm. Especially as the experiments show that the best zero-cost proxies have a spearman ρ of ≈ 0.8, there is no doubt that utilising zero-cost proxies in a NAS algorithm will be far more efficient than today. As discussed in the thesis, earlier work showed that utilising zero-cost proxies in different NAS algorithms (Random Search, Reinforcement Learning, Aging evolution and Binary Predictor) exhibits great improvement in efficiency, which backs our statement that the study's findings will have a positive impact on NAS in GCN for HAR.

7.2 Future Work

Considering the findings and limitations of this thesis, various directions for additional future research are suggested.

Expand dataset size Increasing the number of trained architectures in the dataset would reinforce the generalisability and statistical significance of the results. Furthermore, by incorporating more architectures with diverse characteristics, future research could examine more nuanced relationships between zero-cost proxies and the performance of GCN models within HAR tasks.

Investigate other zero-cost proxy combination techniques Exploring alternative methods for combining zero-cost proxies, such as supervised learning models (neural networks or decision trees) or unsupervised learning approaches (clustering), could improve the efficiency and accuracy of architecture search in NAS algorithms.

Explore multiple GCN NAS frameworks Utilising various GCN NAS frameworks would allow for a more comprehensive comparative analysis, evaluation of the robustness of the acceleration technique, and investigation of different search strategies, optimisation methods, and search space configurations.

Validate zero-cost proxies across different problem domains Testing the zero-cost proxies across diverse problem domains would ensure their adaptability to various search spaces and hyperparameter configurations and comprehensively evaluate their performance.

By following these research directions, the field of NAS can continue progressing, particularly in the context of GCN for HAR, and contribute to developing more efficient, accurate, and sustainable algorithms and techniques.

Incorporate zero-cost proxies in a NAS algorithm Incorporating zero-cost proxies into NAS algorithms has great potential for improving their performance and efficiency by reducing search time with no cost of accuracy. Zero-cost proxies are good at predicting validation accuracy at fully trained, which means they can be used to estimate the performance of GCN architectures without costly training. This approach can be used to optimise different aspects of NAS algorithms, such as accuracy and efficiency.

Bibliography

Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł., & Lane, N. D. (2021). Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*.

Akhauri, Y., Munoz, J. P., Jain, N., & Iyer, R. (2022). Evolving zero cost proxies for neural architecture scoring. *arXiv preprint arXiv:2209.07413*.

Anderson, J. A. (1995). *An introduction to neural networks*. MIT press.

Ann, O. C., & Theng, L. B. (2014). Human activity recognition: A review. *2014 IEEE international conference on control system, computing and engineering (ICCSCE 2014)*, 389–393.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, *24*.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, *34*(4), 18–42.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Proceedings of the 22nd international conference on Machine learning*, 89–96.

Cheng, K., Wei, Y., & Mu, C. (2020). Skeleton-based action recognition with synchronous local and non-local spatio-temporal learning and frequency attention. *Sensors*, *20*(4), 1194.

Deepinmotion. (2023). https://www.ntnu.edu/inb/deepinmotion

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Dong, X., & Yang, Y. (2019). Searching for a robust neural architecture in four gpu hours. https://doi.org/10.1109/CVPR.2019.00186

Dong, X., & Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Duan, Y., Chen, X., Xu, H., Chen, Z., Liang, X., Zhang, T., & Li, Z. (2021). Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5251–5260.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, *20*(1), 1997–2017.

Frankle, J., Dziugaite, G. K., Roy, D. M., & Carbin, M. (2020). Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *2*, 729–734.

Groos, D. (2022a). Convolutional networks for video-based infant movement analysis. towards objective prognosis of cerebral palsy from infant spontaneous movements.

Groos, D. (2022b). Convolutional networks for video-based infant movement analysis. towards objective prognosis of cerebral palsy from infant spontaneous movements.

Groos, D., Ramampiaro, H., & Ihlen, E. A. (2021). Efficientpose: Scalable single-person pose estimation. *Applied Intelligence*, *51*(4), 2518–2533.

Hauke, J., & Kossowski, T. (2011). Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, *30*(2), 87–93.

Jobanputra, C., Bavishi, J., & Doshi, N. (2019). Human activity recognition: A survey. *Procedia Computer Science*, *155*, 698–703.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., & Xing, E. P. (2018). Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, *31*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, *abs/1609.02907*. http://arxiv.org/abs/1609.02907

Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., & Burnaev, E. (2020). Nas-bench-nlp: Neural architecture search benchmark for natural language processing. https://doi.org/10.48550/ARXIV.2006.07116

Krishnakumar, A., White, C., Zela, A., Tu, R., Safari, M., & Hutter, F. (2022). Nas-bench-suite-zero: Accelerating research on zero cost proxies. *arXiv preprint arXiv:2210.03230*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Kyriakides, G., & Margaritis, K. (2020). An introduction to neural architecture search for convolutional networks. *arXiv preprint arXiv:2005.11074*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, N., Ajanthan, T., & Torr, P. H. (2018). Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Li, Y., Wen, Z., Wang, Y., & Xu, C. (2021). One-shot graph neural architecture search with dynamic search space. *Proceedings of the AAAI conference on artificial intelligence*, *35*(10), 8510–8517.

Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., & Jin, R. (2021). Zen-nas: A zero-shot nas for high-performance image recognition. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 347–356.

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. https://doi.org/10.48550/ARXIV.1806.09055

Lopes, V., Alirezazadeh, S., & Alexandre, L. A. (2021). Epe-nas: Efficient performance estimation without training for neural architecture search. *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V*, 552–563.

Ltd, L. R. (2013). *Spearman's rank-order correlation: A guide to when and how to use the spearman's rank-order correlation test*. https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

Mehrotra, A., Ramos, A. G. C., Bhattacharya, S., Dudziak, Ł., Vipperla, R., Chau, T., Abdelfattah, M. S., Ishtiaq, S., & Lane, N. D. (2021). Nas-bench-asr: Reproducible neural architecture search for speech recognition. *International Conference on Learning Representations*.

Mellor, J., Turner, J., Storkey, A., & Crowley, E. J. (2021). Neural architecture search without training. *International Conference on Machine Learning*, 7588–7598.

Mellor, J., Turner, J., Storkey, A., & Crowley, E. J. (2020). Neural architecture search without training. https://doi.org/10.48550/ARXIV.2006.04647

Ning, X., Tang, C., Li, W., Zhou, Z., Liang, S., Yang, H., & Wang, Y. (2021). Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, *34*, 12265–12277.

Pallant, J. (2016). *Spss survival manual: A step by step guide to data analysis using ibm spss* (6th ed.). Open University Press/McGraw-Hill Education.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Peng, W., Hong, X., Chen, H., & Zhao, G. (2020). Learning graph convolutional network for skeleton-based human action recognition by neural searching. *Proceedings of the AAAI conference on artificial intelligence*, *34*(03), 2669–2676.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q., & Kurakin, A. (2017). Large-scale evolution of image classifiers. https://doi.org/10.48550/ARXIV.1703.01041

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., & Wang, X. (2021). A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, *54*(4), 1–34.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, *20*(1), 61–80.

Shahroudy, A., Liu, J., Ng, T.-T., & Wang, G. (2016). Ntu rgb+d: A large scale dataset for 3d human activity analysis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shaphiro, S., & Wilk, M. (1965). An analysis of variance test for normality. *Biometrika*, *52*(3), 591–611.

Si, C., Chen, W., Wang, W., Wang, L., & Tan, T. (2019). An attention enhanced graph convolutional lstm network for skeleton-based action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1227–1236.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

statstutor. (2011). *Spearman's correlation*. Retrieved November 29, 2022, from https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf

Swersky, K., Snoek, J., & Adams, R. P. (2014). Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*.

Tanaka, H., Kunin, D., Yamins, D. L. K., & Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow.

Theis, L., Korshunova, I., Tejani, A., & Huszár, F. (2018). Faster gaze prediction with dense networks and fisher pruning. *CoRR*, *abs/1801.05787*. http://arxiv.org/abs/1801.05787

Tu, R., Khodak, M., Roberts, N. C., & Talwalkar, A. (2021). Nas-bench-360: Benchmarking diverse tasks for neural architecture search.

Turney, S. (2022). Pearson correlation coefficient (r) | guide & examples.

Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its future prospects. *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGT-SPICC)*, 261–265. https://doi.org/10.1109/ICGTSPICC.2016.7955308

Vrigkas, M., Nikou, C., & Kakadiaris, I. A. (2015). A review of human activity recognition methods. *Frontiers in Robotics and AI*, *2*, 28.

Wang, C., Zhang, G., & Grosse, R. (2020). Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.

Weighted arithmetic mean. (2008). Springer New York. https://doi.org/10.1007/978-0-387-32833-1

White, C., Khodak, M., Tu, R., Shah, S., Bubeck, S., & Dey, D. (2022). A deeper look at zero-cost proxies for lightweight nas [https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/]. *ICLR Blog Track*. https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/

White, C., Neiswanger, W., & Savani, Y. (2021). Bananas: Bayesian optimization with neural architectures for neural architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 10293–10301.

White, C., Zela, A., Ru, B., Liu, Y., & Hutter, F. (2021). How powerful are performance predictors in neural architecture search?

Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *Thirty-second AAAI conference on artificial intelligence*.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., & Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. *International Conference on Machine Learning*, 7105–7114.

Zhang, Z., & Jia, Z. (2021). Gradsign: Model performance inference with theoretical insights. *arXiv preprint arXiv:2110.08616*.

Zhou, K., Song, Q., Huang, X., & Hu, X. (2019). Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184*.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). Learning transferable architectures for scalable image recognition. *CoRR*, *abs/1707.07012*. http://arxiv.org/abs/1707.07012

Appendix A

Use of the Zero-Cost Framework on GNN-NAS Project

The research group at DeepInMotion is currently engaged in a project that develops a GNN-NAS framework for HAR. By giving us access to the codebase and providing 38 trained architectures with the corresponding validation accuracy, we could perform a small experiment similar to what is conducted in this thesis. We calculated the Spearman Correlation for every zero-cost proxy using this thesis's developed Zero-Cost framework (section 4.5). The results are presented in table A.1.

	Spearman ρ	
	acc_top1	**acc_top5**
EPE-NAS	0.0712	0.0806
Fisher	-0.5170	-0.5589
Flops	0.2923	0.2740
Grad Norm	**0.8720**	**0.8989**
GradSign	-0.2323	-0.2790
Grasp	0.6172	0.6594
Jacov	-0.0951	-0.0561
L2 norm	-0.1661	-0.1403
NAS-WOT	0.1326	0.2132
Params	0.2658	0.2658
Plain	-0.0117	0.0028
Snip	0.7937	0.8097
Synflow	-0.1710	-0.2460
Zen	-0.1852	-0.1875

Table A.1: Spearman Correlation between zero-cost proxies and validation accuracy.

The results show that both Grad Norm and Snip exhibit very strong correlation with a Spearman Rank Correlation of $(0.8720, 0.8989)$ and $(0.7938, 0.8097)$, respectively. Note that with only 38 data points, the analysis might lack the statistical power needed to detect the correlation confidently. Nevertheless, the results show great potential and can be a good option to include in the framework in the future.