UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRIA E INFORMÁTICA

UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME MONTEIRO GADELHA

AN INVESTIGATION OF NEURAL ARCHITECTURE SEARCH IN THE CONTEXT OF DEEP MULTI-TASK LEARNING

CAMPINA GRANDE – PB

2024

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

An Investigation of Neural Architecture Search in the Context of Deep Multi-Task Learning

Guilherme Monteiro Gadelha

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Inteligência Artificial, Ciência de Dados e Arquiteturas Dedicadas

Dr. Herman Martins Gomes
Dr. Leonardo Vidal Batista

Campina Grande, Paraíba, Brasil

G124i Gadelha, Guilherme Monteiro.
An investigation of neural architecture search in the context of deep multi-task learning / Guilherme Monteiro Gadelha. – Campina Grande, 2024.
105 f. : il. color.

Tese (Doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr. Herman Martins Gomes, Prof. Dr. Leonardo Vidal Batista".
Referências.

1. Neural Architecture Search. 2. Multi-Task Learning. 3. Image Classification. I. Gomes, Herman Martins. II. Batista, Leonardo Vidal. III. Título.

CDU 004.8(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

GUILHERME MONTEIRO GADELHA

AN INVESTIGATION OF NEURAL ARCHITECTURE SEARCH IN THE CONTEXT OF DEEP MULTI-TASK LEARNING

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande, pertencente à linha de pesquisa de Inteligência Artificial, Ciência de Dados e Arquiteturas Dedicadas e área de concentração Ciência da Computação como requisito para a obtenção do Título de Doutor em Ciência da Computação.

TESE APROVADA EM 29/01/2024

Documento assinado digitalmente
gov.br
LEONARDO VIDAL BATISTA
Data: 02/04/2024 18:11:55-0300
Verifique em https://validar.iti.gov.br

DR. LEONARDO VIDAL BATISTA, UFPB
Orientador

Documento assinado digitalmente
gov.br
HERMAN MARTINS GOMES
Data: 02/04/2024 12:11:30-0300
Verifique em https://validar.iti.gov.br

DR. HERMAN MARTINS GOMES, UFCG
Orientador

Documento assinado digitalmente
gov.br
TIAGO MARITAN UGULINO DE ARAUJO
Data: 02/04/2024 13:53:32-0300
Verifique em https://validar.iti.gov.br

DR. TIAGO MARITAN UGULINO DE ARAÚJO, UFPB
Examinador

Documento assinado digitalmente
gov.br
ANSELMO CARDOSO DE PAIVA
Data: 05/04/2024 18:46:40-0300
Verifique em https://validar.iti.gov.br

DR. ANSELMO CARDOSO DE PAIVA, UFMA
Examinador

DR. LEANDRO BALBY MARINHO, UFCG
Examinador

Documento assinado digitalmente
gov.br
LEANDRO BALBY MARINHO
Data: 09/04/2024 20:58:35-0300
Verifique em https://validar.iti.gov.br

DR. EANES TORRES PEREIRA, UFCG
Examinador

Documento assinado digitalmente
gov.br
EANES TORRES PEREIRA
Data: 03/04/2024 16:27:45-0300
Verifique em https://validar.iti.gov.br

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secretaria-copin@computacao.ufcg.edu.br / copin@copin.ufcg.edu.br

REGISTRO DE PRESENÇA E ASSINATURAS

ATA Nº 001/2024 (TESE N° 134)

Aos vinte e nove (29) dias do mês de janeiro do ano de dois mil e vinte e nove (2029), às quatorze horas (14:00), de forma REMOTA, através do Google Meet, reuniu-se a Comissão Examinadora composta pelos Professores LEONARDO VIDAL BATISTA, Dr., UFPB, Orientador, funcionando neste ato como Presidente, HERMAN MARTINS GOMES, Dr., UFCG, Orientador, LEANDRO BALBY MARINHO, Dr., UFCG, EANES TORRES PEREIRA, Dr., UFCG, ANSELMO CARDOSO DE PAIVA, Dr., UFMA, TIAGO MARITAN UGULINO DE ARAÚJO, Dr., UFPB. Constituída a mencionada Comissão Examinadora pela Portaria Nº 021/2023 do Coordenador do Programa de Pós-Graduação em Ciência da Computação, tendo em vista a deliberação do Colegiado do Curso, tomada em reunião de 04 de Dezembro de 2023 e com fundamento no Regulamento Geral dos Cursos de Pós-Graduação da Universidade Federal de Campina Grande - UFCG, juntamente com o Sr(a) GUILHERME MONTEIRO GADELHA, candidato(a) ao grau de DOUTOR em Ciência da Computação, comigo Paloma Nascimento Porto, Secretária(o) dos trabalhos, presentes ainda professores e alunos do referido centro e demais presentes. Abertos os trabalhos, o(a) Senhor(a) Presidente da Comissão Examinadora anunciou que a reunião tinha por finalidade a apresentação e julgamento da tese "AN INVESTIGATION OF NEURAL ARCHITECTURE SEARCH IN THE CONTEXT OF DEEP MULTI-TASK LEARNING", elaborada pelo(a) candidato(a) acima designado, sob a orientação do(s) Professor(es) LEONARDO VIDAL BATISTA e HERMAN MARTINS GOMES, com o objetivo de atender as exigências do Regulamento Geral dos Cursos de Pós-Graduação da Universidade Federal de Campina Grande - UFCG. A seguir, concedeu a palavra ao candidato, o qual, após salientar a importância do assunto desenvolvido, defendeu o conteúdo da tese. Concluída a exposição e defesa do candidato , passou cada membro da Comissão Examinadora a arguir o doutorando sobre os vários aspectos que constituíram o campo de estudo tratado na referida tese. Terminados os trabalhos de arguição, o Senhor Presidente da Comissão Examinadora determinou a suspensão da sessão pelo tempo necessário ao julgamento da tese. Reunidos em caráter secreto no mesmo recinto, os membros da Comissão Examinadora passaram à apreciação da tese. Reaberta a sessão, o Presidente da Comissão Examinadora anunciou o resultado do julgamento, tendo assim, o candidato obtido o Conceito APROVADO. A seguir, foi encerrada a sessão e lavrada a presente ata, que vai assinada por mim, Paloma Nascimento Porto, pelos membros da Comissão Examinadora e pelo candidato. Campina Grande, 29 de janeiro de 2024.

Documento assinado eletronicamente por **Tiago Maritan Ugulino de Araújo**, **Usuário Externo**, em 30/01/2024, às 15:15, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Guilherme Monteiro Gadelha**, **Usuário Externo**, em 30/01/2024, às 15:51, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Anselmo Cardoso de Paiva**, **Usuário Externo**, em 30/01/2024, às 16:31, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **LEANDRO BALBY MARINHO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 31/01/2024, às 07:05, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Leonardo Vidal Batista**, **Usuário Externo**, em 31/01/2024, às 11:56, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **HERMAN MARTINS GOMES**, **PROFESSOR 3 GRAU**, em 31/01/2024, às 15:02, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **EANES TORRES PEREIRA**, **PROFESSOR 3 GRAU**, em 05/02/2024, às 08:39, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **PALOMA NASCIMENTO PORTO**, **ASSISTENTE EM ADMINISTRACAO**, em 05/02/2024, às 08:42, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **4148276** e o código CRC **6AC6DDE2**.

Referência: Processo nº 23096.002465/2024-64 SEI nº 4148276

Abstract

Multi-task learning (MTL) is a design paradigm for neural networks that aims to improve generalization while solving multiple tasks simultaneously in a single network. MTL has been successful in various fields such as Natural Language Processing, Speech Recognition, Computer Vision, and Drug Discovery. Neural Architecture Search (NAS) is a subfield of Deep Learning that proposes methods to automatically design neural networks by searching and arranging layers and blocks to maximize an objective function. Currently, there are few methods in the literature that explore the use of NAS to build MTL networks. In this context, this work investigates a sequence of comparative experiments between multi-task networks, single-task networks, and networks created with a neural architecture search strategy. These experiments aim to understand better the differences between these paradigms of neural network design and compare the results achieved by each. We investigated neural network architectures for different use cases, such as the ICAO-FVC dataset, MNIST, FASHION-MNIST, Celeb-A, and CIFAR-10 datasets. Additionally, we experimented with a well-established dataset of NAS to benchmark new proposed methods in the field. Our experiments have revealed that the NAS technique, developed through Reinforcement Learning, is capable of discovering optimal architectures in a shorter time than the current state-of-the-art technique based on Regularized Evolution. Furthermore, this technique has demonstrated competitive results across various datasets of multi-task learning, in terms of accuracy and equal error rate. While it may not be the top performer in the case of ICAO-FVC, it still delivers a competitive outcome and holds the potential to uncover even better architectures than the best handcrafted one.

Resumo

O aprendizado multitarefa (MTL) é um paradigma de design para redes neurais que visa melhorar a generalização enquanto resolve múltiplas tarefas simultaneamente em uma única rede. A MTL tem tido sucesso em vários campos, como Processamento de Linguagem Natural, Reconhecimento de Fala, Visão Computacional e Descoberta de Medicamentos. Neural Architecture Search (NAS) é um subcampo do Deep Learning que propõe métodos para projetar redes neurais automaticamente, pesquisando e organizando camadas e blocos para maximizar uma função objetivo. Atualmente, existem poucos métodos na literatura que exploram o uso de NAS para construção de redes MTL. Neste contexto, este trabalho investiga uma sequência de experimentos comparativos entre redes multitarefa, redes monotarefa e redes criadas com uma estratégia de busca de arquitetura neural. Esses experimentos visam compreender melhor as diferenças entre esses paradigmas de projeto de redes neurais e comparar os resultados alcançados por cada um. Investigamos arquiteturas de redes neurais para diferentes casos de uso, como o conjunto de dados ICAO-FVC, conjuntos de dados MNIST, FASHION-MNIST, Celeb-A e CIFAR-10. Além disso, testamos um conjunto de dados bem estabelecido de NAS para avaliar novos métodos propostos em campo. Nossos experimentos revelaram que a técnica NAS, desenvolvida através do Reinforcement Learning, é capaz de descobrir arquiteturas ótimas em um tempo menor do que a atual técnica de última geração baseada na Evolução Regularizada. Além disso, esta técnica demonstrou resultados competitivos em vários conjuntos de dados de aprendizagem multitarefa, em termos de acurácia e equal error rate. Embora possa não ter o melhor desempenho no caso do ICAO-FVC, ainda oferece um resultado competitivo e tem o potencial de descobrir arquiteturas ainda melhores do que a melhor feita à mão.

Agradecimentos

Esta tese não teria sido possível sem o apoio, a orientação e a ajuda de muitas pessoas. Estou genuinamente grato a todos eles.

Primeiramente, gostaria de agradecer aos orientadores desta tese, Herman Martins e Leonardo Batista, pelas ideias, orientações e insights alcançados durante o período da pesquisa e pela dedicação para produzir comigo um trabalho de alta qualidade.

Agradeço também a todos os amigos e colegas que me fazem crescer a cada dia profissional e pessoalmente.

Quero expressar também a minha gratidão à minha família pelo apoio e incentivo inabaláveis ao longo da minha jornada de vida. Gostaria também de agradecer à Thássia por estar sempre ao meu lado, independentemente da decisão que eu tome.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo financiamento parcial desta pesquisa. Além disso, ao Laboratório de Inteligência Artificial e Arquiteturas Dedicadas (LIAD) por fornecer parte da infraestrutura necessária para a execução de nossos experimentos. Por fim, um agradecimento especial à Vsoft Tecnologia Ltda. que incentivou minha pesquisa de doutorado e me permitiu ver a aplicação de empreendimentos acadêmicos em cenários práticos por meio de sua equipe de P&D.

Contents

List of Abbreviations

ICAO	International Civil Aviation Organization
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
MRTD	Machine Readable Travel Documents
DL	Deep Learning
STL	Single-Task Learning
MTL	Multi-Task Learning
NAS	Neural Architecture Search
RL	Reinforcement Learning
SSS	Size Search Space
TSS	Topology Search Space
DAG	Direct Acyclic Graph
RE	Regularized Evolution

List of Figures

List of Tables

Chapter 1

Introduction

The chapter introduces the context and background of this thesis, describes an existing problem in the field of study, the automatic design of multi-task neural networks, and lists expected contributions and the structure of this doctoral thesis.

1.1 Context and Background

The use of Machine Learning and Deep Learning is expanding, with numerous applications in fields such as Computer Vision (CV) and Natural Language Processing (NLP). Deep learning has made a significant impact on a variety of tasks, such as object detection [Redmon and Farhadi, 2018], image segmentation [Long et al., 2015; Ronneberger et al., 2015], face detection [Kumar et al., 2019; Yang et al., 2015; Sun et al., 2018], named-entity recognition [Li et al., 2022; Shen et al., 2017], and image captioning [Karpathy et al., 2014]. Its accuracy is consistently improving and is widely adopted in production scenarios.

In seminal works, a neural network's architecture was defined and tuned meticulously by highly qualified experts to solve a specific task following the Single-Task Learning (STL) paradigm [Caruana, 1997]. However, manually finding and tuning a neural network is time-consuming. Early research observed that the tasks' performance could increase if some of them are solved together, following the so-called Multi-Task Learning paradigm [Caruana, 1997; Torralba et al., 2007]. Besides, MTL networks present a lower memory footprint and higher inference speeds when compared to equivalent solutions based on STL [Vandenhende et al., 2020].

However, it remains an open challenge to design the network architectures for solving multiple tasks simultaneously. A literature review presented in this research and other existing literature reviews [Ruder, 2017; Zhang and Yang, 2017; Vandenhende et al., 2020] have identified, in recent years, different strategies for designing these networks in a partial or fully automatic way. Besides that, the literature review also identified an increasing interest in Neural Architecture Search (NAS) applied to Multi-Task Learning (MTL) - as presented in Chapter 3 - since it can reduce the amount of time dedicated to develop and tune new deep neural networks, especially for the case of multi-task networks.

Considering these literature reviews and previous works, we designed experiments to evaluate STL, MTL, and Neural Architecture Search (NAS) models for a specific scenario: the International Civil Aviation Organization (ICAO) compliance checking - which is described in Chapter 4. Ultimately, we compared the performances of each approach in terms of well-defined metrics: accuracy and Equal Error Rate (EER).

The aforementioned experiments guided the proposition of a new NAS technique for partial or fully automated design of network architectures in the context of MTL problems such as the ICAO photographic compliance checking. To evaluate the generalization of the proposed technique, we expanded the experiments to include other domain datasets (MNIST [LeCun et al., 1998], CIFAR-10 [Krizhevsky, 2009], CelebA [Liu et al., 2015], and Fashion-MNIST [Xiao et al., 2017]).

The NAS technique proposed in this study yields promising results in the ICAO compliance checking scenario when compared to the state-of-the-art (SOTA) works in terms of equal error rate (EER). It also demonstrates excellent generalizability by achieving SOTA results in traditional datasets. However, it is worth noting that the best results are still achieved by a handcrafted MTL neural network in the ICAO-FVC dataset.

1.2 Research Questions

We formulated the following research questions:

- RQ 1: How to apply NAS to design MTL architectures automatically for image classification tasks?

- RQ 2: How do the architectures discovered through NAS differ from handcrafted architectures designed using the STL and MTL paradigms, and what insights can be gained from this comparison?

The above two questions we address are related to NAS and MTL. Specifically, we aim to investigate the potential of using NAS techniques to optimize the performance of MTL networks for image classification tasks. By leveraging the power of NAS, we hope to discover cost-effective architecture designs to enhance the capabilities of MTL networks.

1.3 Contributions

- The development of a new technique of NAS in the context of MTL able to find new MTL architectures faster than the available techniques;
- Apply the created NAS technique to the FVC-ICAO dataset and achieve a competitive result;
- A comparative study of the proposed technique with existing related works on multiple datasets (MNIST, CIFAR-10, CelebA, Fashion-MNIST);

1.4 Thesis Structure

This thesis is structured as follows: Chapter 2 describes basic concepts to help understand the techniques and experiments studied. Chapter 3 evaluates related works on Multi-Task Learning and Neural Architecture Search. Subsequently, Chapter 4 presents the methodology and experimental designs to help with the proposition of the technique developed in this thesis. Chapter 5 discusses the experiment's results and performs an analysis of the errors. Finally, Chapter 6 summarizes this thesis, its main results, provides some conclusions and possible future research steps.

Chapter 2

Fundamental Concepts

This chapter presents some key concepts needed to better understand the research topics and the proposed thesis. The following specific concepts are included in this chapter: Transfer Learning, Interpretability, Multi-Task Learning, Neural Architecture Search, and Reinforcement Learning.

2.1 Transfer Learning

Transfer learning (TL) is a technique used in machine learning to transfer the knowledge learned from one task to another related task. The primary goal of transfer learning is to reuse the knowledge acquired in one domain to solve a problem in a different but related domain. The idea behind TL is that there are common features shared by different tasks, such as image recognition or natural language processing, and that these features can be leveraged to improve performance on a new task. By using the learned weights of a pre-trained neural network, the model can be fine-tuned on a new dataset with fewer examples, reducing the need for a large amount of training data. This makes it possible to achieve higher accuracy and faster convergence when training on a new task. In summary, TL allows us to apply the knowledge acquired from one task to another related task, thus accelerating the learning process and improving the performance of the model [Goodfellow et al., 2016].

2.2 Grad-CAM and Interpretability

Grad-CAM [Selvaraju et al., 2020] is a technique that enhances the transparency and explainability of convolutional neural networks. This innovative method uses gradients to evaluate an image and determine the most critical regions used by the network to classify it. By superimposing a heatmap on the original image, Grad-CAM allows users to easily identify the relevant regions that the neural network has considered to produce its classification.

This technique is especially useful in cases where the neural network's processes are not immediately apparent. Grad-CAM's visual explanations provide insights into how the model is weighting and processing data, thus making it easier to understand how it works.

For example, when identifying a "dog" or a "cat" in an image, Grad-CAM maps the crucial regions of the image considered by the network to produce the classification. These regions are then visually highlighted using a heatmap, making it easy to see which parts of the image were most influential in the network's decision-making process. Figure 2.1 demonstrates this example.

Overall, Grad-CAM is useful for ensuring that convolutional neural networks are transparent and trustworthy. By providing visual explanations for the decisions made by these models, Grad-CAM helps ensure that they are used responsibly and ethically.

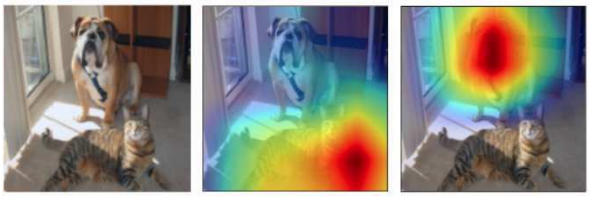


Figure 2.1: Grad-CAM outputs when the deep neural network is asked to identify a "dog" (on the right) or a "cat" (on the center) in an image. The original image is on the left. Note that the regions in red that the network is paying attention to make the classification. *Source: [Selvaraju et al., 2020]*

2.3 Multi-Task Learning (MTL)

Multi-task learning is a learning approach that aims to enhance the generalization performance of related tasks, as compared to solving them independently (STL) [Caruana, 1997].

In MTL, the tasks are learned in parallel and share common representations. These shared representations enable the network to solve tasks more efficiently.

There are several benefits of using the MTL paradigm. Firstly, there is a reduction in memory utilization as the network shares many layers between the solved tasks. Secondly, the inference speed is faster since many unnecessary calculations are avoided. Calculations occur once, and the results are shared among tasks during inference. Lastly, the different tasks may act as regularizers for each other, thereby improving overall performance [Vandenhende et al., 2020]. Additionally, MTL enables positive results to be achieved even when training data is limited [Kaiser et al., 2017].

2.4 Neural Architecture Search (NAS)

2.4.1 Definition

The common practice of searching and training deep learning neural networks involves manual probing of network designs and hyperparameters optimization, which is time-consuming and requires a lot of expertise [Dong et al., 2022]. In this scenario emerges the subfield of NAS.

Neural Architecture Search is a Deep Learning (DL) subfield dedicated to studying new automatic techniques for neural architecture design, where the networks layers or blocks are searched and arranged in different ways within the network architecture [Zoph and Le, 2017; Zoph et al., 2018]. Each new architecture is tested against a validation set until convergence, and the best architecture is selected. Figure 2.2 [Elsken et al., 2019] shows the NAS general schema.

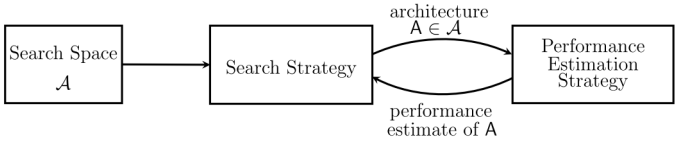


Figure 2.2: NAS general scheme. Initially, a *Search Space* ($\mathcal{A}$) of neural architectures is defined. Then, a *Search Strategy* selects a neural architecture (A) and its performance is measured. The search repeats these steps until the architecture with the best performance is found. *Source: [Elsken et al., 2019]*

2.4.2 Search space exploration strategies

Some strategies to explore the search space have been applied: Random search, Bayesian optimization, evolutionary methods, Reinforcement Learning (RL), and gradient-based methods [Elsken et al., 2019]. Random searching is usually applied as a baseline to evaluate the other strategies, as we also do in this research. It just randomly selects neural network architectures in a defined search space.

Bayesian optimization [Jin et al., 2018; Kandasamy et al., 2018] is commonly used for Hyperparameter Optimization, a closely related field of NAS, whose goal is to automatically find the best set of hyperparameters in a neural network [Hutter et al., 2020; Bergstra et al., 2011; Domhan et al., 2015].

Historically, evolutionary methods or algorithms have been applied to evolve neural networks architectures, with many examples in the literature [Angeline et al., 1994; Stanley and Miikkulainen, 2002; Floreano et al., 2008] but with limited success in achieving or surpassing human performance on datasets evaluations with hand-crafted neural networks architectures. However, in recent works [Liu et al., 2018b; Real et al., 2019], these methods achieved reasonable success in reducing search time and presenting competitive results.

Also, recently, techniques applying RL or Gradient-based methods were successful across multiple datasets and domains, e.g. image classification [Zoph and Le, 2017; Maziarz et al., 2019], object detection [Zoph and Le, 2017], semantic segmentation [Chen et al., 2018], speech recognition [Hu et al., 2021; Chen et al., 2020].

Gradient-based methods [Liu et al., 2018b; Xie et al., 2019] use a gradient-descent algorithm to improve the performance of the neural architecture search. These methods treat the search space as a continuous space and then can differentiate architecture representations [Liu et al., 2018b]. So, the problem of navigating the search space becomes similar to the loss function minimization occurring in deep neural network training. In this approach, the time spent on the neural architecture search is highly optimized compared to the other search strategies.

NAS methods based on RL [Zoph and Le, 2017; Zoph et al., 2018; Pham et al., 2018] were responsible for the popularization of the field in 2017, when Zoph *et al.* [Zoph and Le, 2017] obtained competitive results on CIFAR-10 and Penn Treebank datasets. Although this first result required nearly three to four weeks with 800 GPUs for processing and training the

models, the field has evolved, and better RL methods have emerged. Next, we discuss this strategy in detail.

2.5 Reinforcement Learning (RL)

This section discusses Deep Reinforcement Learning (RL), the REINFORCE algorithm, and the mathematical background of this framework. The presented description is based on the work of [Graesser and Keng, 2020]. Figure 2.3 depicts the general framework of a problem in RL. In this framework, an agent performs *actions* on an *environment*. Based on the outcomes of these actions, the agent's internal *state* is modified. The interactions between the agent and the environment are *interpreted* and generate a *reward*, which can either increase or decrease the agent's progress toward the final goal.

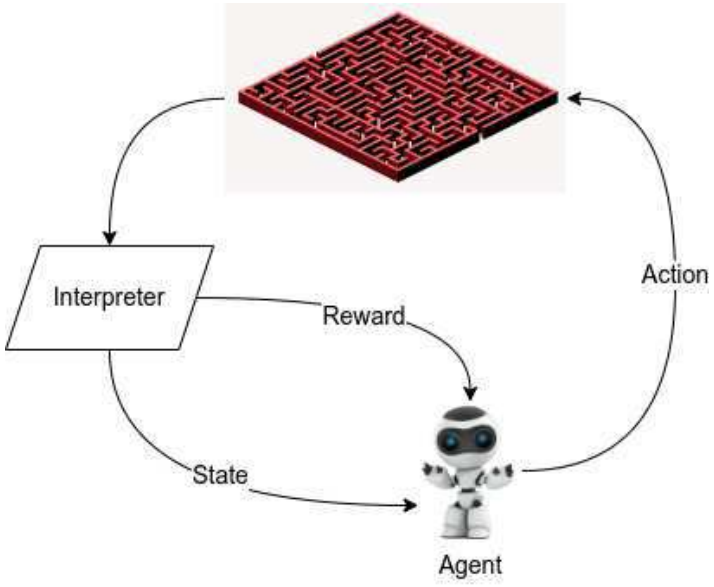


Figure 2.3: RL general framework. Source: own author.

2.5.1 Policy

In the context of Reinforcement Learning, a policy π is a function that takes a specific state and returns action probabilities. An action $a \sim \pi(s)$ is selected based on this function. A good policy is essential to maximize the cumulative discounted rewards, and function approximation techniques are necessary to achieve this. A deep neural network is a powerful

tool for representing a policy using learnable parameters θ, and this policy network is known as π_θ. In other words, the policy is defined by its θ parameters. As an agent operating in an environment, your objective is a specific aim you strive to achieve, such as winning a game or attaining the highest score possible. A trajectory is the sequence of your actions and the resulting states and is denoted as $\tau = s_0, a_0, s_1, a_1, ..., s_T, a_T$. The return of a trajectory $R_t(\tau)$ is the discounted sum of rewards from the present time step t to the end of the trajectory, expressed by Equation 2.1.

$$R_t(\tau) = \sum_{t'=t}^{T} \gamma^{t'-t} r'_t \tag{2.1}$$

It should be emphasized that while the total amount starts at a specific time step, the discount factor γ is raised to the power of $t' - t$ to calculate the return, where t' is the current step in the agent trajectory. Therefore, the power must be corrected by starting at the time step t [Graesser and Keng, 2020].

2.5.2 Objective Function

The objective function determines the anticipated return of an agent based on the complete trajectories. This is expressed in Equation 2.2.

$$J(\pi_\tau) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \gamma^t r_t] \tag{2.2}$$

The calculation in Equation 2.2 suggests that the expectation is determined by obtaining several trajectories from a particular policy, also known as $\tau \sim \pi_\theta$. To ensure accuracy, it is crucial to obtain a larger number of samples and use an appropriate policy, π_θ. The discount factor γ determines the significance given to future rewards while computing the reinforcing signal $R(\tau)$. A higher γ value indicates a greater emphasis on future rewards, and the optimal value of γ varies depending on the problem. When the agent needs to consider the impact of its actions on future rewards over a more extended period, it is advisable to use a higher γ value.

2.5.3 Policy Gradient

In the policy gradient algorithm, we introduced both the policy denoted by π_θ and the objective function $J(\pi_\theta)$. The policy permits an agent to execute an action, whereas the objective sets a goal for the agent to achieve. The policy gradient is the last part of the algorithm that aims to resolve the following problem:

$$\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] \tag{2.3}$$

Where $\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$ is the expected reward value of the entire trajectory of the agent. We utilize gradient ascent to maximize the objective function J by modifying the policy parameters θ. As per calculus, the gradient shows the direction of the steepest ascent. We compute the gradient and employ it to update the parameters to improve the objective, as depicted in Equation 2.4.

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\pi_\theta) \tag{2.4}$$

The parameter update, which is represented by the symbol α, is determined by the learning rate. The policy gradient is defined by $\nabla_\theta J(\pi_\theta)$, which represents the gradient of the policy. Equation 2.4 describes the method of updating the parameter θ with the policy gradient. The scalar value of the learning rate α impacts the magnitude of the update.

Equation 2.5 presents the formula for the policy gradient, where the expected value is taken over all possible trajectories τ generated by the policy π_θ. The policy gradient takes into account the reward at each time step t denoted by $R_t(\tau)$ and the log probability of the action executed by the agent at time step t denoted by $\nabla_\theta log\pi_\theta(a_t|s_t)$.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} R_t(\tau)\nabla_\theta log\pi_{\theta(a_t|s_t)}] \tag{2.5}$$

The steps to obtain Equation 2.5 are outlined in [Graesser and Keng, 2020]. $\pi_\theta(a_t|s_t)$ denotes the probability of the agent's action at time step t, which is sampled from the policy. The policy gradient updates the action probabilities based on the rewards received, increasing the probability of actions that result in positive returns and decreasing the probability of actions that result in negative returns. Over time, the policy learns to produce actions that yield higher rewards.

To estimate the policy gradient accurately, the REINFORCE algorithm relies on Monte Carlo sampling, which is a powerful technique that leverages random sampling to generate data and approximate a function. The method gained prominence thanks to Stanislaw Ulam, a renowned mathematician who worked at the Los Alamos research lab during the 1940s [Graesser and Keng, 2020].

By integrating Monte Carlo sampling, Equation 2.5 can be converted into Equation 2.6. The notation $\mathbb{E}_{\tau\sim\pi_\theta}$ means that as we collect more trajectories τ by applying the policy π_θ and calculate their mean value, we get closer to the actual policy gradient $\nabla_\theta J(\pi_\theta)$. This method remarkably enhances the capability to estimate the policy gradient and enhance overall performance.

$$\nabla_\theta J(\pi_\theta) \approx \sum_{t=0}^{T} R_t(\tau) \nabla_\theta log \pi_{\theta(a_t|s_t)} \tag{2.6}$$

We utilize the REINFORCE algorithm to compute the policy gradient by means of Monte Carlo sampling with a single trajectory. Although this technique provides an unbiased estimate of the policy gradient, it is affected by high variance. To reduce this variance, it is common practice to utilize a baseline value. In this thesis, we investigate the use of reward normalization as a solution to tackle the issue of reward scaling, as discussed in Chapter 4.

The variance of the policy gradient estimate can be high when employing Monte Carlo sampling, owing to various factors. Firstly, actions are randomly selected from a probability distribution, contributing to the variance. Secondly, the starting state may differ across episodes, leading to an increase in variance. Furthermore, the environment transition function may be stochastic, exacerbating the variance issue. To address this problem, a recommended approach is subtracting an appropriate action-independent baseline from the returns. This technique is depicted in Equation 2.7 [Graesser and Keng, 2020].

$$\nabla_\theta J(\pi_\theta) \approx \sum_{t=0}^{T} (R_t(\tau) - b(s_t)) \nabla_\theta log \pi_{\theta(a_t|s_t)} \tag{2.7}$$

2.5.4 REINFORCE Algorithm

A standard algorithm used to define the best policy is the REINFORCE algorithm [Williams, 1992]. The central intuition behind the algorithm is that good actions should become more

probable during the agent environment exploration because it feeds a *reinforce* mechanism with positive outcomes. At the same time, it punishes the agent with negative outcomes when the interaction with the environment is harmful. The agent's goal must be to maximize the objective by selecting good actions. This happens once the agent *learns* to interact with the environment and to select good actions, which are ***reinforced*** by positive rewards [Graesser and Keng, 2020].

The REINFORCE algorithm aims to instruct agents on acting in an environment by creating a policy that determines action probabilities based on the states. This is done by adjusting the action probabilities using the policy gradient, making REINFORCE a policy gradient algorithm. Algorithm 1 [Graesser and Keng, 2020; Williams, 1992] integrates the previously defined formulae. Note that a new trajectory is calculated for every new episode, and the old one is discarded after each parameter update in the algorithm.

Algorithm 1 REINFORCE Algorithm

Require: Initialize $MaxEpisode$

Require: Initialize learning rate α

Require: Initialize weights θ of a policy network π_θ

for $episode = 0, ..., MaxEpisode$ **do**
  Sample a trajectory $\tau = s_0 a_0, r_0, ..., s_T, a_T, r_T$
  Set $\nabla_\theta J(\pi_\theta) = 0$
  for $t = 0, ..., T$ **do**
    $R_t(\tau) = \sum_{t'=t}^{T} \gamma^{t'-t} r'_t$
    $\nabla_\theta J(\pi_\theta) = \nabla_\theta J(\pi_\theta) + R_t(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$
  end for
  $\theta = \theta + \alpha \nabla_\theta J(\pi_\theta)$
end for

When applied to NAS, it is possible to map the presented concepts as follows: the RL *environment* is the search space, the *actions* are the networks architectures found - also called child networks -, and the *rewards* are the validations accuracies scores achieved for each child network. During the training phase, while the validation increases, the reward is positively ***reinforced*** by choosing better child network architectures. Usually, a Recurrent Neural Network (RNN) - such as a Long-Short Term Memory (LSTM) - is the *agent* propos-

ing and selecting new neural architectures, as we see in works like [Zoph and Le, 2017; Zoph et al., 2018]. In the case of NAS, the LSTM learns to propose better architectures through the feedback received and the minimization of a proposed loss function (cost). One of the main advantages of deep RL over supervised learning, for example, is that it does not need to obtain optimal results in every learning iteration to achieve an optimal result at the end of the learning process. The drawback of using RL is that it usually takes longer searches to reach optimal results. Besides that, RL-based NAS methods may suffer from convergence problems, presenting unstable results during the search process.

Chapter 3

Related Works

This chapter details previous related works, specifically in Multi-Task Learning and Neural Architecture Search, that are close to this doctoral research.

3.1 MTL

Historically, deep multi-task techniques (MTL) have been divided into two categories: *Soft-Parameter Sharing* and *Hard-Parameter Sharing* [Vandenhende et al., 2020; Ruder, 2017]. In Hard-Parameter Sharing, a shared encoder branch is commonly subdivided into task-specific branches that specialize in solving a single task. In Soft-Parameter Sharing, there is an algorithm to automatically find where to share or branch within the network, and the task branches intersect at multiple points. Figure 3.1 [Vandenhende et al., 2020] shows the difference between (a) hard-parameter sharing and (b) soft-parameter sharing. Note that there is a common branch in the case of hard-parameter sharing. In contrast, on soft-parameter sharing, there is information exchange between the task branches, and during the training, the connections - which can happen at multiple points - are strengthened or weakened. This work focuses on hard-parameter sharing methods once they produce memory and computation efficient MTL networks [Zhang et al., 2022].

Numerous examples of hard-parameter sharing MTL techniques exist in the literature. Some works developed techniques to create branches simultaneously from a backbone model [Ruder, 2017; Nekrasov et al., 2019; Suteu and Guo, 2020; Leang et al., 2020] and from a single point in the network, similarly as we do in this research and described in the following

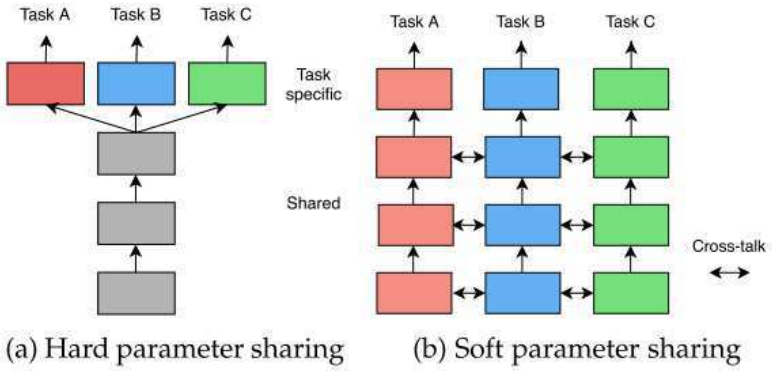


Figure 3.1: Soft and Hard-Parameter Sharing in MTL. *Source: [Vandenhende et al., 2020]*

paragraphs of this section.

Other techniques allow branches to emerge from different points in the network. In the Learning-to-Branch [Guo et al., 2020] technique, the authors derive a tree-like architecture using a differentiable neural architecture search technique. The developed technique learns where to branch in the network without previous calculations of task relatedness. BMTAS (Branched Multi-Task Architecture Search) [Bruggemann et al., 2020] also makes a differentiable neural architecture search on a tree-like architecture, extending [Guo et al., 2020] work by applying a computational budget to limit the neural architecture search. However, both works fail to guarantee an optimal multi-task architecture [Choromanska et al., 2015; Sun et al., 2020] and to control the user's budget during the search space exploration.

A more recent work, TS-MTL (Tree-Structured Multi-Task Learning) [Zhang et al., 2022], addresses these issues and proposes a tree-structured multi-task model recommender, which chooses neural network architectures in a designed search space and respects a user-defined computation budget. The authors evaluated their work with NYU-v2 and Tiny Taskonomy datasets and different backbone models: Deeplab-ResNet34 and MobileNetV2. In both cases, they report competitive results when compared with the state-of-the-art results. The basic schema of our work also uses a base model and derives multi-task architectures from them through the use of multiple task branches.

Some previous works are based on task-relatedness calculation. The Fully-Adaptive Feature Sharing (FAFS) [Lu et al., 2017] technique calculates the task relatedness level in order to group similar tasks in shared branches. Depending on the difficulty of solving a task with others, the tasks are grouped or ungrouped, and new network layers are stacked in a greedy

fashion to compose the final model. The experiments made in this work by the authors are in the CelebA dataset. A different aspect of the work of this thesis is that their technique does the task grouping automatically, which is one of our future research topics.

In the What-to-Share [Vandenhende et al., 2019] technique, the authors also calculate the layer-sharing possibility based on the task-relatedness score. The difference to [Lu et al., 2017] is that this score is calculated previously to training, which becomes a drawback in the method adoption and requires the training of single-task networks before the multi-task network training.

Other works, such as AdaShare [Sun et al., 2019] and AutoMTL [Zhang et al., 2022], propose techniques that learn a task-specific policy to select the layers that should be executed for a given task during the MTL network training. The policy also considers the user's computational budget, besides the tasks' accuracy scores. The authors used many dataset to validate their technique: NYUv2, Tiny-Taskonomy, CityScapes, and PASCAL VOC; each dataset has its own type of task, and the competitive results achieved in terms of accuracy demonstrates the method's effectiveness. The complexity of the method may hinder the interpretability and implementation of task-specific policies for layer execution in various contexts.

Deep Elastic Network (DEN) [Ahn et al., 2019] proposes an estimator network and a selection network trained together based on samples of inputs of tasks. The first network is based on a backbone model, from which the technique derives network architectures with a hierarchical structure. The second network selects candidate models based on an input instance of a task. The authors used some datasets to evaluate their approach: CIFAR-10, CIFAR-100, Tiny-ImageNet, STL-10, and ImageNet. Based on several experiments, they report competitive results in terms of accuracy in each dataset evaluated and resource efficiency once the model selection network leverages compactness as a positive characteristic when choosing a new model.

Shi *et al.* proposes in the work Multitask Learning with Progressive Parameter Sharing (MPPS) [Shi et al., 2023] a new method based on a soft-parameter sharing paradigm to dynamically decide the links between the network neurons based on their own proposed algorithm. This algorithm estimates the task relatedness inspired by curriculum learning during training. The author used the datasets NYUv2, Multi-CIFAR100, and Cityscapes to

evaluate their technique.

Arnaldo *et al.* [Gualberto et al., 2022] applies a semi-supervised learning approach to train a handcrafted MTL network based on autoencoders in the context of ICAO compliance checking. The method is the first to apply a MTL solution to solve all 23 ICAO requisites and achieves SOTA results in 9 out of 23 requisites - all results are discussed in Chapter 5 - when evaluated in a benchmark platform. However, the method leaks in generalization since the authors designed it for the resolution of this specific problem and did not test it in other datasets with other domains of image classification, as we did in this work.

Other recent works also can be cited, such as [Zhang et al., 2023; Xu et al., 2023]. In the first research, [Zhang et al., 2023] proposed a recommender of tree-based architectures that uses a given pre-trained base model and an architecture within a user-specified computation budget. The second work [Xu et al., 2023] used knowledge distillation between a teacher MTL model for a student MTL model. Although both works show improvements in the context of MTL, we can not compare directly with ours since they tested their methods in other datasets.

Table 3.1 summarizes the works found in our literature review, considering works until 2023 in MTL.

Method	Citation	Method Type	Evaluated Datasets
LearningToBranch	[Guo et al., 2020]	Hard-Parameter Sharing	CelebA, Taskonomy
BMTAS	[Bruggemann et al., 2020]	Hard-Parameter Sharing	PASCAL, NYUDv2
TS-MTL	[Zhang et al., 2022]	Hard-Parameter Sharing	NYUv2, TinyTaskonomy
FAFS	[Lu et al., 2017]	Hard-Parameter Sharing	CelebA, Deepfashion
AdaShare	[Sun et al., 2019]	Soft-Parameter Sharing	NYUv2, CityScapes, TinyTaskonomy, DomainNet, ...
AutoMTL	[Ahn et al., 2019]	Hard-Parameter Sharing	NYUv2, TinyTaskonomy, CityScapes, PASCAL VOC
MPPS	[Shi et al., 2023]	Soft-Parameter Sharing	Multi-CIFAR100, NYUv2, Cityscapes
TreeMTL Recommender	[Zhang et al., 2023]	Hard-Parameter Sharing	NYUv2, TinyTaskonomy, DomainNet
MTL-KDDP	[Xu et al., 2023]	Hard-Parameter Sharing	NYUv2, PASCAL-Context
IcaoNet	[Gualberto et al., 2022]	Hard-Parameter Sharing	ICAO-FVC

Table 3.1: Set of related works in MTL with the highlighted method types and datasets used for evaluation.

3.2 NAS

This section describes some relevant works from the literature that developed and applied NAS techniques or strategies and inspired our work.

Based on the successful results achieved by Zoph and Le [Zoph and Le, 2017], the Efficient Neural Architecture Search (ENAS) [Pham et al., 2018] work applies the one-shot model strategy [Elsken et al., 2019]: search for an optimal subgraph within a large computational graph through RL. The seminal work by Zoph and Le was evaluated on the CIFAR-10 and Penn Treebank datasets and reported a 96.35% accuracy score and 62.4 perplexity score in the respective datasets. ENAS authors also used the CIFAR-10 and Penn Treebank datasets to evaluate their solution and obtained a 97.11% accuracy score in the first dataset and reported a low perplexity score in the second dataset: 55.8 (lower perplexity scores indicate better predictive performance and that the model has a better understanding of the language context).

Stochastic Neural Architecture Search (SNAS) [Xie et al., 2019] uses RL, but differently from [Pham et al., 2018] and [Zoph and Le, 2017], the authors change the feedback mechanism from "constant rewards", such as validation accuracy, to a "generic loss" calculated during training. The proposed method is not based on Reinforcement Learning or Regularized Evolution and achieves a 97.15% accuracy score in the CIFAR-10 dataset. Besides that, the method does enforce a resource-efficient constraint, producing smaller architectures when possible. Despite the promising results, the generalizability of the method to other tasks and domains is not fully explored: the analysis is limited to only two datasets.

Other work in NAS, Progressive Neural Architecture Search (PNAS) [Liu et al., 2018a] used sequential model-based optimization (SMBO) to perform the neural architecture search. This strategy starts with a simple model and increases its complexity. It stacks new blocks in the final model architecture as the heuristic progresses in the search, eliminating unpromising model architectures simultaneously. To predict the architecture performance, the technique trains a surrogate function to optimize the search time so that there is no need to train the entire proposed model in every iteration.

Differentiable Architecture Search (DARTS) [Liu et al., 2018c] uses a one-shot model strategy but performs the architectural search using the gradient descent algorithm by contin-

uously relaxing the different neural architecture representations, making them differentiable. Differently from works with RL-based NAS or RE-based NAS, DARTS uses a gradient-based method to navigate into a continuous search space of architectural networks.

MTL-NAS [Gao et al., 2020] proposes a neural architecture search technique leveraged by a gradient-based search algorithm that explores a predefined NAS search space inspired in previous researches [Misra et al., 2016; Gao et al., 2018]. The search space design enables the algorithm to find inter-task relations inside the neural networks, discovering multiple fusion points between the task-specific branches. This approach suffers from low flexibility in relation to the architectures that can be generated if compared with other methods such as DARTS [Liu et al., 2018c] or the ones based on reinforcement learning, where more diverse search spaces may exist.

A recent work on NAS named LayerNAS [Fan et al., 2023] proposes a method to reduce the search space available for creating new architectures considering certain constraints. However, there is a problem with the method's stability: it can find efficient architectures at the beginning of the search, but the efficacy of new proposed architectures decreases as the search advances.

Other recent works, but not closely related to this work, also can be cited: [Qin et al., 2023] proposed a technique based on supernets in the context of graph neural architecture search (GraphNAS) and MTL. Graph neural networks (GNNs) are a class of artificial neural networks capable of processing data represented as graphs [Wu et al., 2022] with many applications, such as molecular biology, natural language processing, and social networks. [Wang et al., 2023] also proposed a technique based on supernets, but for the context of visual transformers with the neural architecture search space designed for producing visual transformers networks, differently from our work that focuses on convolutional neural networks. Lastly, [Shen et al., 2023] used mathematical programming for creating optimized convolutional neural networks, the main idea of their work is to demonstrate how convolutional neural network still can surpass visual transformers-based architecture if well designed. Table 3.2 summarizes the works found in our literature review, considering works until 2023 in NAS.

Technique	Citation	Method Type	Evaluated Datasets
SNAS	[Xie et al., 2019]	Gradient-based	CIFAR-10, ImageNet
PNAS	[Liu et al., 2018a]	Accuracy Prediction	CIFAR-10, ImageNet
ENAS	[Pham et al., 2018]	Gradient-based	PennTreebank, CIFAR-10
DARTS	[Liu et al., 2018c]	Gradient-based	CIFAR-10, PennTreebank, ImageNet, WikiText-2
NAS-RL	[Zoph and Le, 2017]	Reinforcement Learning	CIFAR-10, PennTreebank
NASNet	[Zoph et al., 2018]	Reinforcement Learning	CIFAR-10, PennTreebank, ImageNet, COCO
MTL-NAS	[Gao et al., 2020]	Gradient-based	NYUv2, Taskonomy
LayerNAS	[Fan et al., 2023]	Dynamic Programming	CIFAR-10, CIFAR-100, ImageNet16-120
MTGC	[Qin et al., 2023]	Supernet-based	Synthetic Datasets, OGB
MDL-NAS	[Wang et al., 2023]	Supernet-based	ImageNet-1K, COCO, ADE20K
DeepMAD	[Shen et al., 2023]	Mathematical Programming	ImageNet-1K, COCO, ADE20K, UCF101, Kinetics400

Table 3.2: Set of related works in NAS, highlighting the method type and evaluated datasets.

3.3 Final Considerations

As presented in this chapter, MTL and NAS continuously evolve, with important advancements in recent years. Each strategy has its advantages and disadvantages. MTL is easier to implement and evaluate and cheaper to train but does not generalize as well as NAS approaches, which, in turn, are harder to implement and more expensive to train. However, NAS approaches may generalize better between different domains [Elsken et al., 2019].

We have encountered a challenge in the field that we are focusing on, which is to develop a new technique of NAS for the context of MTL. Observe that the developed NAS technique is hardware-agnostic, which means it does not focus on creating new networks that are resource-efficient as some works do, such as BMTAS [Bruggemann et al., 2020] and ENAS [Pham et al., 2018]. We plan to explore other topics in the future, such as creating branches of tasks within an MTL neural network architecture automatically to achieve optimal results for the entire set of tasks. We believe that this approach could be beneficial for MTL networks, as stated in the literature, and intend to research methods to achieve this. For now, we are concentrating on creating the NAS method and following the appropriate methodology. In addition, we need to address the generalizability of any proposed method across various domains and tasks using multiple datasets. We have outlined our methodology in the next chapter.

Chapter 4

Methodology

This chapter presents the methodology adopted in this thesis, the datasets used for experimental evaluation, the evaluation metrics, and the architecture and underpinnings of each approach we studied.

4.1 Overview

The diagram presented in Figure 4.1 illustrates the methodology adopted in this study. Our research involved the exploration of five distinct network designs, each stemming from different paradigms: Single-Task Learning (STL), Multi-Task Learning (MTL), and Neural Architecture Search (NAS). Initially, we utilized the FVC-ICAO dataset to evaluate the Single-Task Learning (STL) approach. This initial phase allowed us to gain insights into the performance of single-task scenarios, where the context is relatively straightforward and networks can be evaluated more effectively through error analysis. These findings were a foundational benchmark for subsequent experiments in the domains of the Multi-Task Learning (MTL) and Neural Architecture Search (NAS).

Subsequent to the Single-Task Learning (STL) experiments, we conducted Multi-Task Learning (MTL) experiments with the aim of enhancing results by employing different configurations and network designs at both the data and model levels. These Multi-Task Learning (MTL) experiments yielded significant improvements. By comparing the outcomes of various architectural approaches and paradigms, such as Single-Task Learning (STL) and Multi-Task Learning (MTL), we were able to define the basis architecture and search space

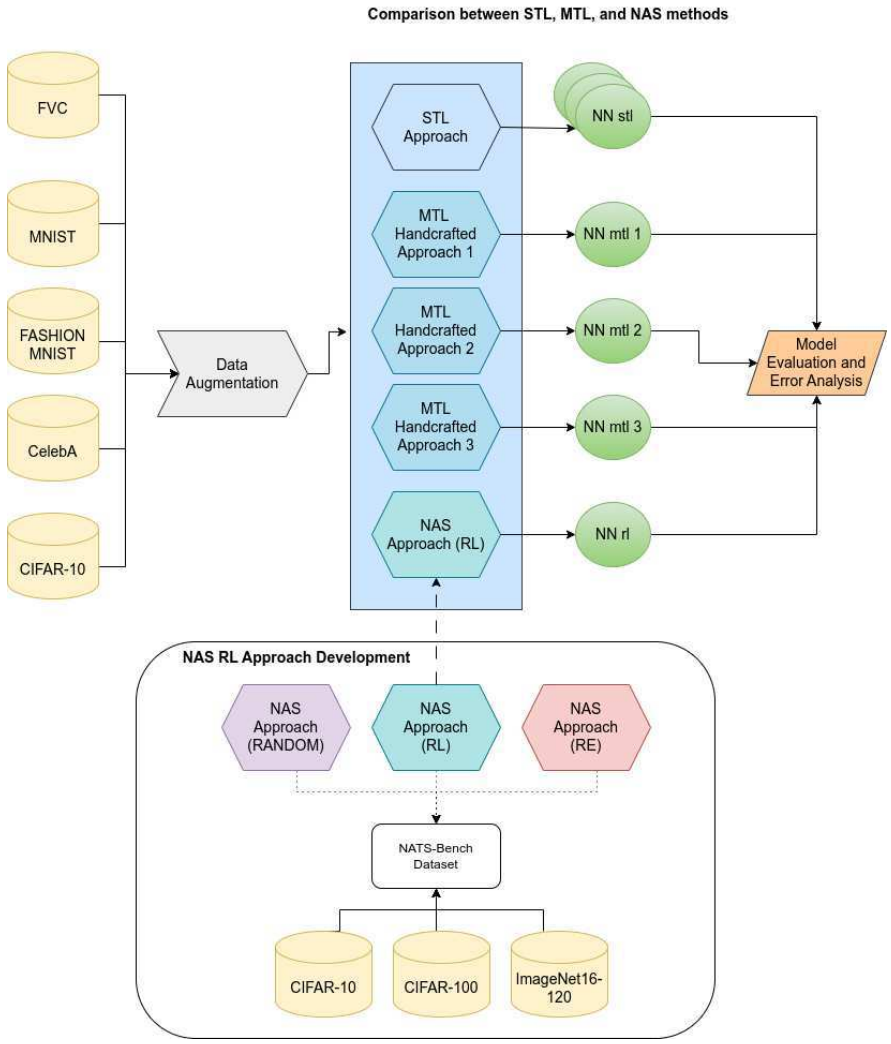


Figure 4.1: Scheme of applied methodology, specifying the used datasets and each approach experimented: STL, MTL, and NAS, which derive distinct neural networks (NN) and are evaluated with common metrics (Accuracy and EER). Note that previously to the comparison between STL, MTL, and NAS-based methods in the five selected datasets, we developed and tuned the proposed NAS RL-based method in the NATS-Bench dataset, which is based on CIFAR-10, CIFAR-100, and ImageNet16-120 image datasets.

for our Neural Architecture Search (NAS) approach.

In the course of our study, besides ICAO-FVC, we also introduced the use of traditional datasets, including CIFAR-10, Celeb-A, MNIST, and FASHION-MNIST. Prior to feeding these datasets into the networks, we applied a data augmentation step to expose the networks to a broader range of data variations, ultimately benefiting the training process. To evaluate

the trained networks, we employed standard metrics such as Accuracy and Equal Error Rate (EER) to facilitate a comparison of results and provide guidance for developing our Neural Architecture Search (NAS) technique.

Note that we treat the datasets as binary task datasets, regardless of whether it is a multi-class dataset or not, we let the EER threshold decide in the end whether a given network result will belong to class 0 or class 1. Therefore, the network must also incorporate into its internal knowledge whether or not the dataset is multi-class, in which the same image can have multiple simultaneous labels. Previous works also use some of these datasets as described in Chapter 3.

Subsequently, following a comprehensive literature review and an in-depth analysis of potential Neural Architecture Search (NAS) algorithm types and available datasets, we decided to adopt a Reinforcement Learning approach. Our technique was implemented and fine-tuned based on the NATS-Bench dataset, which is detailed in Section 4.8 of this chapter.

In the following sections, we delve into the specifics of the datasets used, the software and materials employed, and the design particulars of the networks under consideration.

4.2 Datasets Used

4.2.1 FVC dataset

The first dataset, called FVC, was used in the ICAO case study in STL, MTL, and NAS contexts. The FVC-Ongoing competition [Ferrara et al., 2022] built the FVC dataset as a reference dataset for ICAO requisites compliance checking [Ferrara et al., 2012].

Next, we briefly describe the ICAO compliance checking context and the specification ISO/IEC 19795-4 for facial images of passports. Before designing our experiments, we mapped the facial image requisites described in the ISO/IEC official documents into a set of tasks to be solved by the deep neural networks.

ICAO

International Civil Aviation Organization (ICAO) is a United Nations entity whose objectives include establishing good practices and patterns for aerial navigation and developing

security, defense, operational effectiveness, sustainable economic development, and environmental responsibility policies in the aviation sector [ICAO, 2022].

One of the ICAO specifications is related to travel documents that must be readable by machines (Machine Readable Travel Documents (MRTD))[ICAO, 2015]. This specification defines patterns for an expedited identification document of people in traffic, such as passports with electronic chips, during international travels between countries that are also ICAO State members. The defined standard specifies how biometric data, such as facial images, digital fingerprints, and iris images, should be captured and stored digitally. So, any system in a member country can read, understand, and use the same pattern, avoiding disseminating different patterns and system incompatibilities. To achieve this, ICAO have partnerships with internationally recognized standards development organizations, such as International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).

ISO/IEC 19795-4 Specification

The ICAO specification [ICAO, 2015] known as ISO/IEC 19795-4 defines required processes and the main components of a biometric identification system, as shown in Figure 4.2 for the case of facial identification and verification. Initially, the person's identity emitting the new document must be established and checked. Next, a proper device captures the raw data, such as a person's frontal picture. Then the face location is identified in the image, and the system creates a template. Lastly, a comparison between the two templates verifies the person's identity.

Figure 4.2: Facial Identification Process

ICAO defines facial identification as the best biometric identification method for various reasons:

- Facial pictures do not share information the person does not routinely disclose to the general public;

- Facial photographs are socially and culturally accepted internationally;
- Faces are already collected and verified routinely as part of applying for an identification document such as a passport or ID card;
- Human verification of facial biometrics is relatively simple and familiar to the border control authorities.

Despite being a widespread method, not every photograph conforms to use in a biometric identification system. The quality of the facial image obtained is of fundamental importance for successful identification. Because of this, the ISO/IEC 19794-5 [ISO, 2017] standard proposes, among other things, best practices for face photographs, as well as specifies properties, attributes, and restrictions related to face images, the process of obtaining photos of faces and their storage, transmission, and recording.

Following this ISO standard, ICAO defines 23 requirements that passport photos must meet and that a biometric identification system must consider in the process of capturing new images at the time of registration [Ferrara et al., 2022]. Some examples of these requirements are the following: the image cannot be blurred, the person must be looking directly at the camera, the hair cannot cover the eyes, and the person cannot wear a hat or cap.

The system must evaluate each ICAO requirement during the capturing process, and the facial image must achieve a minimum quality score to be accepted by the system. Otherwise, the document issuer receives feedback explaining the reason for the rejection and asking for a new image. For example, in Figure 4.3, we can see examples of facial images with problems (the three on the left) and one that complies (the last one on the right) with the ICAO requirements.

Figure 4.3: Examples of facial images with different qualities.

In our experiments, we map each of the 23 ICAO requisites as a binary (compliant or non-compliant) task to the deep neural network to solve. Table 4.1 lists the ICAO requisites [Ferrara et al., 2012] and the aliases used in our research.

The FVC dataset is composed of sub-datasets gathered from different origins and is provided as a training dataset by the FVC-Ongoing competition organization, as follows:

- AR dataset [Benavente, 1998] with 1741 images of size 768 x 576 pixels;
- FRGC dataset [Phillips et al., 2005] with 1935 images of size 2048 x 1536 pixels;
- PUT dataset [Kasiński et al., 2008] with 291 images of size 1704 x 2272 or 1200 x 1600 pixels;
- 804 artificially generated images by applying ink-marked/creased, pixelation, and washed-out effects to compliant images from the AR database;
- new manually acquired 817 images of size 1600 x 1200 pixels to cover missing requisites labels.

Also, we included other *ad-hoc* images for increasing the number of available samples by ICAO requisite and for having images with higher resolution. As it is well known in the field, increasing the number of images and their general quality tends to improve the deep neural network efficacy [Goodfellow et al., 2016]. Figure 4.4 presents some examples of images from the FVC dataset.

We selected random partitions of this dataset to perform our experiments with the following proportions: 75% are training images, 15% are validation images, and 10% are test images. These partitions are the same for all experiments involving the FVC dataset. The dataset has 5,865 images in total.

4.2.2 Traditional datasets

Additionally, we performed experiments with traditional datasets briefly described in this section to evaluate the generalizability of our proposed NAS approach.

#	Requisite Name	Alias
1	Blurred	BLURRED
2	Looking Away	L_AWAY
3	Ink Marked \Creased	INK_MARK
4	Unnatural Skin Tone	SKIN_TONE
5	Too Dark \Too Light	LIGHT
6	Washed Out	WASHED_OUT
7	Pixelation	PIXELATION
8	Hair Across Eyes	HAIR_EYES
9	Eyes Closed	EYES_CLOSED
10	Varied Background	BACKGROUND
11	Roll \Pitch \Yaw Greater than $8°$	ROTATION
12	Flash Reflection on Skin	REFLECTION
13	Red Eyes	RED_EYES
14	Shadows Behind the Head	SH_HEAD
15	Shadows Across Face	SH_FACE
16	Dark Tinted Lenses	DARK_GLASSES
17	Flash Reflection on Lenses	FLASH_LENSES
18	Frames too Heavy	FRAMES_HEAVY
19	Frame Covering Eyes	FRAME_EYES
20	Hat \Cap	HAT
21	Veil over Face	VEIL
22	Mouth Open	MOUTH
23	Presence of Other Faces or Toys too Close to Face	CLOSE

Table 4.1: ICAO requisites and respective aliases used in this thesis.

Figure 4.4: Examples of images in FVC dataset. *Source: own author.*

MNIST

The MNIST dataset [LeCun et al., 1998] has its roots in the NIST Special Database 1 and Special Database 3. Created to gauge the accuracy of algorithms in recognizing handwritten characters, these databases served as the foundation for MNIST, which is widely regarded as a standard for testing machine learning models, especially in deep learning and image classification. The dataset contains 60,000 training images and 10,000 test images of handwritten digits recorded in black and white with a shape of 28x28. We implemented the NAS architecture in the way the neural architecture had to learn ten tasks, one binary task for each digit between 0 and 9. Figure 4.5 shows examples extracted from the MNIST test dataset.

MNIST images consist of grayscale pixel values that vary from 0 (black) to 255 (white), indicating the intensity of each pixel. MNIST dataset is intentionally designed to be difficult as it includes a wide range of writing styles, sizes, and orientations of digits. The dataset also includes handwritten digits from different individuals, which adds to its diversity and complexity. This diversity makes it a realistic and challenging task for the digit recognition algorithms to classify and recognize the digits accurately.

FASHION-MNIST

FASHION-MNIST [Xiao et al., 2017] is a dataset by Zalando Research, which includes 70,000 grayscale images of clothing and fashion items. It provides a challenging dataset for image classification tasks, divided into a training set of 60,000 images and a testing set of 10,000 images. Each image has 28x28 pixels.

FASHION-MNIST is similar to MNIST, but instead of handwritten digits, it contains

Figure 4.5: MNIST dataset sample. *Source: [Steppan, 2022]*

clothing items. FASHION-MNIST contains images with complex textures, patterns, and shapes, making them challenging for classification. The appearance of clothing items varies greatly, making it difficult to recognize details and distinguish between similar categories. Figure 4.6 shows an example of the dataset.

Celeb-A

Researchers at the Chinese University of Hong Kong created the CelebA [Liu et al., 2015] dataset to aid facial analysis and recognition research. It contains over 200,000 celebrity images, making it ideal for real-world facial recognition tasks.

CelebA is a rich dataset with 40 binary attributes for each image and precise facial landmarks. It's ideal for evaluating facial recognition and landmark detection algorithms due to its varied lighting conditions, expressions, and poses. Celebrities of all types are included. See Figure 4.7 for a sample.

CIFAR-10

University of Toronto researchers developed the CIFAR-10 [Krizhevsky, 2009] dataset as part of the Canadian Institute for Advanced Research program. It contains 60,000 color images categorized into 10 classes, including "airplane", "automobile", "bird", "cat", "dog", "deer", "frog", "horse", "ship", and "truck". The dataset is used to train and evaluate image

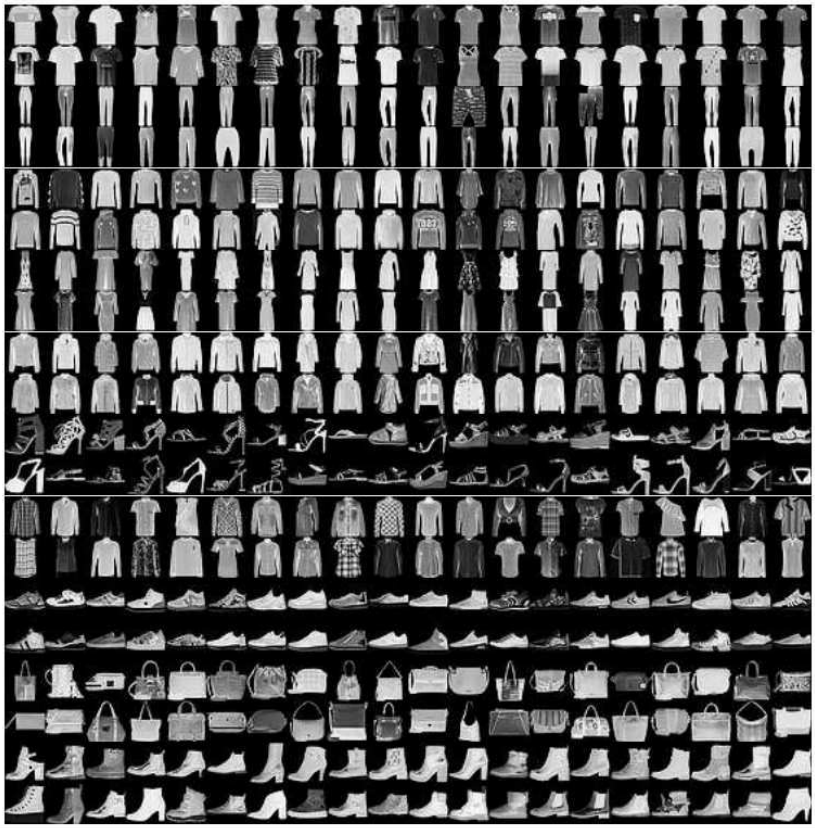
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure 4.6: FASHION-MNIST dataset sample. *Source: [Xiao et al., 2017]*

classification models and is split into a training set of 50,000 images and a test set of 10,000 images. All images are in RGB color format and have a fixed size of 32 by 32 pixels.

The CIFAR-10 dataset has images with labels assigned from 0 to 9 representing different categories. The dataset diversity presents a challenge for image classification algorithms, making it an excellent tool to evaluate their effectiveness. Figure 4.8 shows an extract of the dataset.

4.3 Data Augmentation

Our datasets can be considered undersized for the Deep Learning field, especially the FVC dataset: datasets commonly have hundreds of thousands of samples for successful deep network training. However, there are techniques to improve the model's generalization, such as data augmentation [Goodfellow et al., 2016].

Figure 4.7: Celeb-A dataset sample. *Source: [Liu et al., 2015]*

We performed data augmentation operations in the input images for the FVC dataset so that the network could identify the correct patterns in the images despite small variations, such as rotations in a predefined range of degrees. Table 4.2 shows the data augmentation operations applied and a brief explanation.

Operation	**Input Values**	**Description**
Horizontal Flip	True	Horizontally flip the image
Rotation Range	20	Rotates the image in a defined range of degrees
Zoom Range	0.15	Applies zoom of -15% to +15% in the image
Width Shift Range	0.1	Expands image randomly between 10% to the right or 10% to the left
Height Shift Range	0.1	Expands image randomly between 10% up and 10% down
Shear Range	0.15	Shear intensity in counter-clockwise direction in degrees
Fill Mode	Nearest	Repeats the nearest pixel in the eventually created empty spaces on the image borders.

Table 4.2: Data augmentation operations applied to FVC dataset

As further explained in Section 4.4, we used Tensorflow[1], Keras[2] and PyGlove [Peng et al., 2020] frameworks for network training, implementing data augmentation, and NAS

[1]Official site: https://www.tensorflow.org/. Date of access: April 4th, 2022

[2]Official site: https://www.tensorflow.org/guide/keras. Date of access: April 4th, 2022

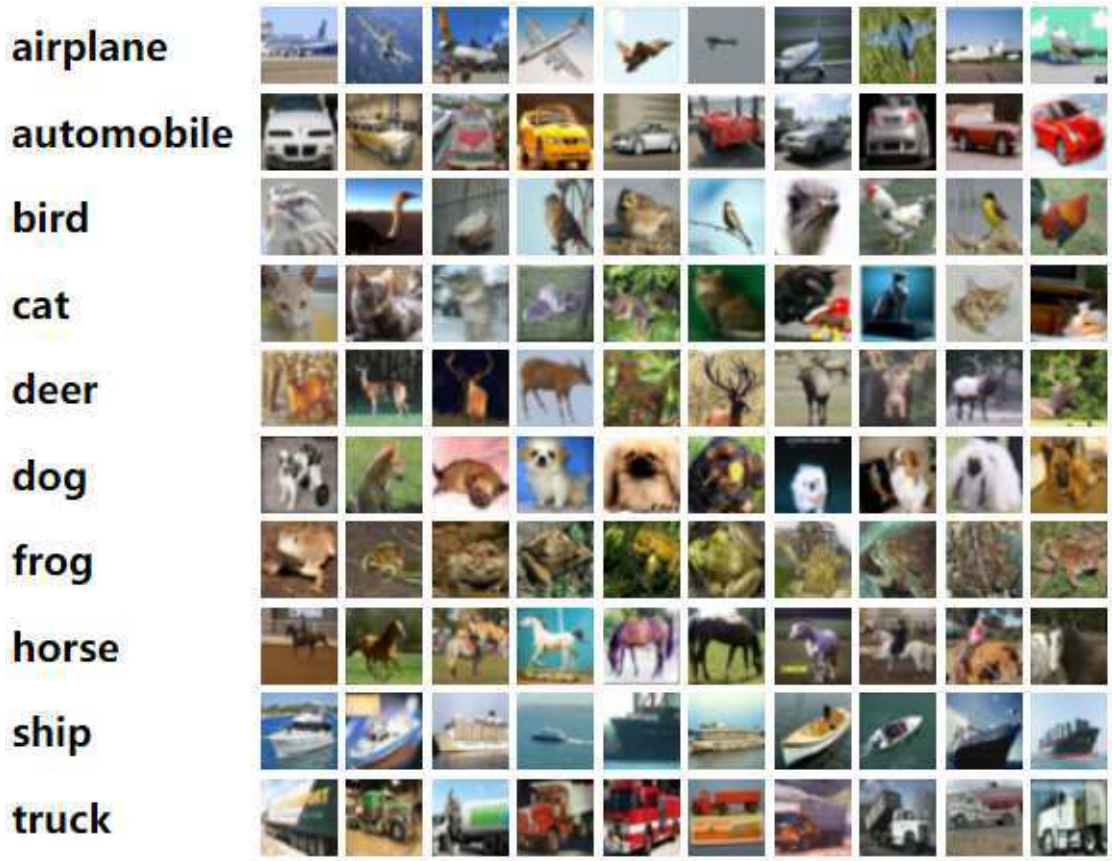


Figure 4.8: CIFAR-10 dataset sample. *Source: [Krizhevsky, 2009]*

approaches. We obtained the values listed in Table 4.2 experimentally, and, to the best of our efforts, these yielded the best results so far.

4.4 Softwares and Frameworks

All implemented scripts and auxiliary material is available digitally on a GitHub repository[3], so all experiments can be verified and reproduced.

We used Python programming language[4] and common libraries for data analysis, such as Pandas[5], NumPy[6], and Scikit-Learn[7]. For model training and evaluation, we used Tensorflow

[3]Official site: https://github.com/guilhermemg/icao_nets_training. Date of access: April 5th, 2022.
[4]Official site: https://www.python.org/. Date of access: April 5th, 2022
[5]Official site: https://pandas.pydata.org/. Date of access: April 5th, 2022
[6]Official site: https://numpy.org/. Date of access: April 5th 2022
[7]Official site: https://scikit-learn.org/. Date of access: April 5th, 2022

and Keras frameworks. The NAS experiments were greatly simplified using the PyGlove[8] framework[Peng et al., 2020] and the NATS-Bench API[9] [Dong and Yang, 2020; Dong et al., 2022].

We conducted the experiments and data analysis using Jupyter Notebooks[10] - also available in the GitHub repository - in a Linux Ubuntu 20.04 machine with a single Nvidia GPU 2080 RTX with 6GB of GDDR memory, 32GB of DDR4 RAM, and AMD Ryzen 5 3600 6-core processor with 12 threads.

4.5 Metrics

4.5.1 Accuracy

Accuracy if the ratio between the sum of true positives (TP) and true negatives (TN) and the total of analyzed samples (true positives, true negatives, false positives (FP), and false negatives (FN)). Equation 4.1 formalizes this metric [Guido, 2017].

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

When analyzing binary classifiers with the accuracy metric, it is important to consider an issue: the class unbalancing. When a class is much more present in the dataset, for example, 80% of the instances are of 1's in a distribution of 1's and 0's; if one classifier estimates uniquely 1's, it has already reached 80% of accuracy without learning to distinguish the first class from the second. So, the analysis must consider this problem when investigating the classifiers' outputs, which may hide bad results behind high accuracy scores.

Despite being sensitive to unbalanced datasets, which is our case for some requisites, we instead use accuracy together with a less sensitive metric: Equal Error Rate, as discussed in the following subsection. Depending on the evaluated dataset we use one metric or the other.

[8]Official site: https://github.com/D-X-Y/NATS-Bench. Date of access: November 25th, 2023.

[9]Official website: https://github.com/google/pyglove. Date of access: November 25th, 2023.

[10]Official site: https://jupyter.org/. Date of access: April 4th, 2022.

4.5.2 Equal Error Rate - EER

Examples of biometric systems are fingerprint recognition, face recognition, and iris recognition systems. The accuracy of such systems is usually measured based on the number of mistakes they perform in a test with multiple combinations of valid and invalid pairs of collected features, like fingerprints or facial pictures. For example, a valid pair of features is a pair of fingerprints captured from the same finger. An invalid one is a pair of fingerprints captured from different fingers or of different persons.

Equal Error Rate is a standard metric to check biometric systems' performance. It is the error rate at a specific threshold t in which False Non-Match Rate (FNMR) and False Match Rate (FMR) are equal. The threshold t is the acceptance threshold, i.e., if the matching score is equal to or above t, the user is admitted into the system. Otherwise, he/she is not allowed. FNMR - also called False Rejection Rate (FRR) - at threshold t is the rate of incorrect non-matches over all comparisons between genuines and impostors, i.e., this represents the difficulty that a real user may encounter in logging into the system. FMR - also called False Acceptance Rate (FAR) - at a threshold t is the rate of incorrect matches over all comparisons, i.e., how much lower this number is, the more difficult it is to penetrate the system. ZeroFNMR and ZeroFMR are the points where both FNMR is zero and FMR is zero [Maltoni et al., 2009]. Figure 4.9 illustrates this concept.

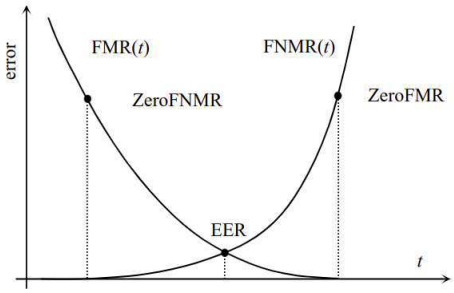


Figure 4.9: Example of FMR and FNMR curves with EER, ZeroFNMR, and ZeroFMR points. *Source: [Maltoni et al., 2009]*

4.6 Single-Task Learning

Initially, we made a series of experiments with Single-Task networks, where each task was an ICAO requisite. The main idea was to compare the available methods from the literature for the same classification tasks and the results obtained with task-specific neural network training.

Our STL method uses the same single-task architecture for all 23 tasks separately and is based on the concept of transfer learning. Different base models, e.g. previously trained models with other datasets and used as a starting point of the training, were tested: VGG16 [Simonyan and Zisserman, 2015], VGG19 [Simonyan and Zisserman, 2015], MobileNetV2 [Sandler et al., 2018], Inception-V3 [Szegedy et al., 2015] and ResNet50-V2 [He et al., 2016].

The VGG-16 and VGG-19 networks share similar architectures, utilizing 3x3 convolutional filters and max-pooling to downsample the input. Both networks consist of a series of convolutional layers followed by a fully connected network. The VGG-16 has 16 layers, while the VGG-19 has 19 layers.

The ResNet50-v2 utilizes residual blocks and skip-connections to overcome the vanishing gradient problem. Meanwhile, the MobileNet-v2 leverages depthwise convolutions to reduce computational cost while maintaining accuracy.

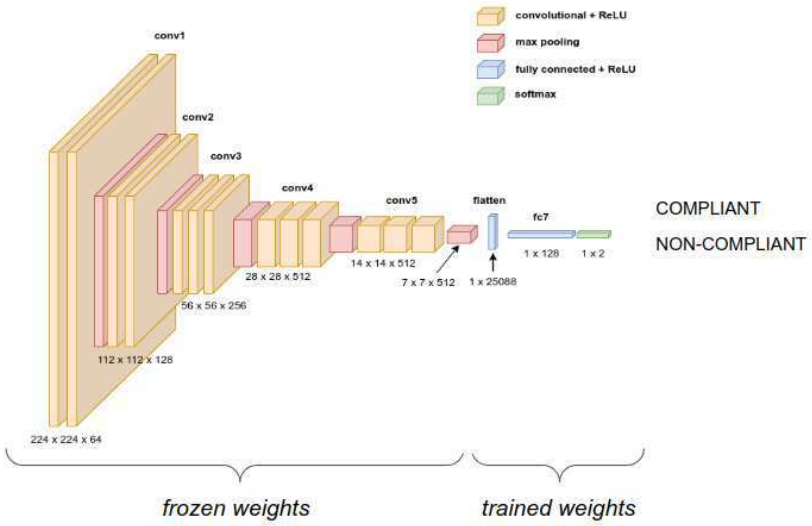


Figure 4.10: Single-Task Network architecture using Transfer Learning technique

Inception-v3 is a computer vision model that efficiently captures features at different scales using various-sized convolutional filters within its "Inception modules". It uses 1x1 convolutions to reduce dimensionality and auxiliary classifiers to improve training.

The best base model identified by our tests was VGG16. We have frozen the base model weights during the network training, as illustrated in Figure 4.10, and trained just the dense layers and the classification layer to learn the new tasks. Each single-task network determines if an input image is compliant or non-compliant with a specific ICAO requisite. Chapter 5 discusses the experiments and results.

4.7 Multi-Task Learning

After the single-task networks trainings and evaluations, we studied different architectural settings with all ICAO tasks being analyzed simultaneously in the same neural network. Thus, checking handcrafted architectures again, but through multi-task learning. In the following paragraphs, we discuss the proposed architecture of each MTL experiment.

4.7.1 Architecture HANDCRAFTED 1

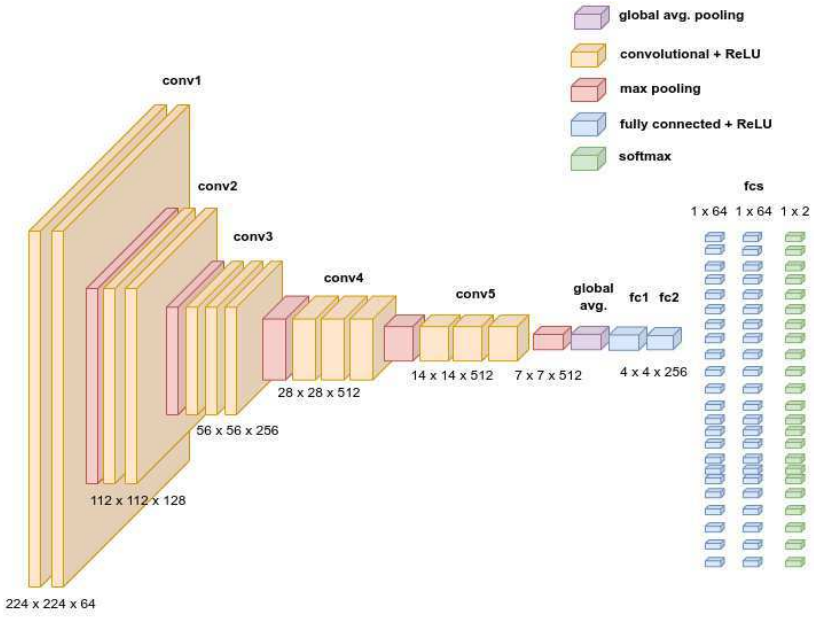


Figure 4.11: Multi-Task Learning *Handcrafted 1* architecture

The first architecture tested was similar to the one used in the single-task learning experiments. As in the STL case, we used VGG16 as the base model, removing the original network output layer and freezing the trained weights. Also, new layers and an output layer were added, with 23 branches corresponding to each ICAO task. Figure 4.11 shows the general schema of this network.

A fully connected (FC) layer with 64 neurons, followed by a fully connected layer with two neurons corresponding to the binary outputs of each task, compose each branch of the network. The activation function used in the output layer was softmax.

4.7.2 Architecture HANDCRAFTED 2

Figure 4.12 presents this network configuration. Note that we maintained the same general schema from architecture Handcrafted 1. However, we removed the shared branch and linked all task branches directly to the Global Average Pooling layer. Also, we kept just one FC layer (1 x 64) for each task branch to test the learning potential of each branch with the minimum of FC layers possible. The number of FC layers is key for our research and is explained in more detail in Section 4.8 and Chapter 5.

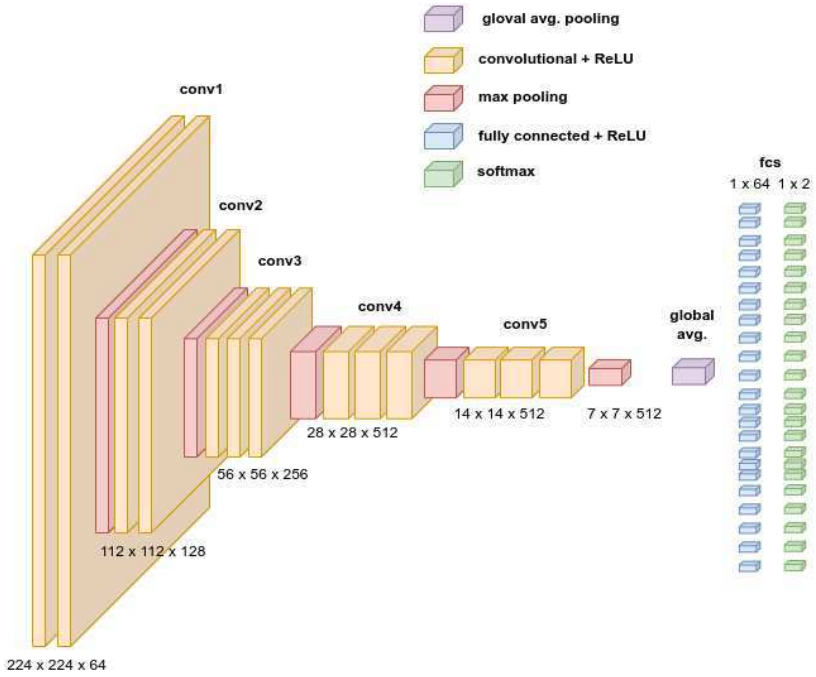


Figure 4.12: Multi-Task Learning *Handcrafted 2* architecture

4.7.3 Architecture HANDCRAFTED 3

Next, we grouped some ICAO requisites into common task branches with shared weights. So we could test the hypothesis that distinct tasks may benefit each other and increase the total gain while the network solves some tasks jointly, which means higher accuracy and lower EER.

The ICAO tasks were hand-picked and manually grouped by analyzing their general characteristics, assuming they may share common features that the network could use to solve them more effectively. For example, if the network should analyze the face's bottom half region for different tasks, they should be in the same group, e.g., the tasks of mouth and veil classification, because both tasks analyze this region of the face. Figure 4.13 shows this proposed architecture.

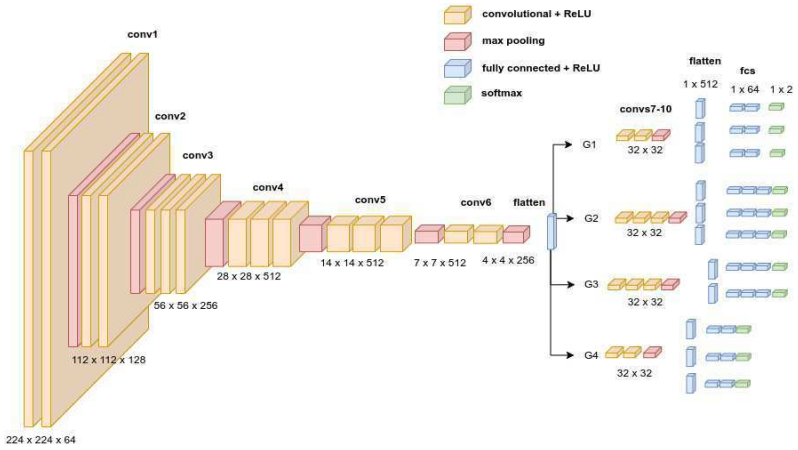


Figure 4.13: Multi-Task Learning *Handcrafted 3* architecture

We grouped the following tasks into common branches:

- Group 1: Background, Close, Ink Mark, Pixelation, Washed Out, Blurred, Shadow Head;
- Group 2: Mouth and Veil;
- Group 3: Red Eyes, Flash Lenses, Dark Glasses, Looking Away, Frame Eyes, Hair Eyes, Eyes Closed, and Frames Heavy;
- Group 4: Shadow Face, Skin Tone, Light, Hat, Rotation, and Reflection.

In this architecture, we also experimented with increasing the number of FC layers per task branch, e.g., the branches' length, so we could verify the hypothesis that increasing the number of FC layers of a task branch would benefit the network's efficacy in solving it. As a result, all 23 task's branches have more FC layers than in the first and second studied MTL architectures, but the number of FC layers is fixed for each branch yet, as shown in Table 4.3.

Task Group	**Number of dense layers**
Group 1	2
Group 2	3
Group 3	3
Group 4	2

Table 4.3: Number of FC layers for each task branch in Handcrafted 3 architecture

4.8 Neural Architecture Search

We implemented new approaches with automatic neural architecture search using as baseline the different handcrafted MTL architectures described in previous sections. Firstly, we had to define what kind of searching we would target: micro-searching or macro-searching. To reduce the search space and make the problem treatable, considering the number of resources available (just a single GPU), we decided to do a micro-searching and define the architectural search based on each branch's numbers of convolutional layers and FC layers.

Figure 4.14 shows the generic architecture our search is based on. We maintain the base model and the general aspect of Handcrafted 3 MTL architecture, where we have grouped branches of tasks. We assure that the algorithm may find equally powerful architectures as the Handcrafted 3. The neural architecture search consists of finding the values *m1*, *m2*, *m3*, *m4*, *n1*, *n2*, *n3*, and *n4*. These eight values, which we call a **config**, correspond respectively to the four grouped tasks branches' lengths in terms of the number of convolutional layers (m_i), where $i \in [1, 2, 3]$ and the number of dense layers (n_j), where $j \in [1, 2, 3, 4, 5]$.

Figure 4.15 depicts the neural architecture search process. We have a controller component (1) first selects a candidate architecture (2) and trains it in a validation set (3), then this

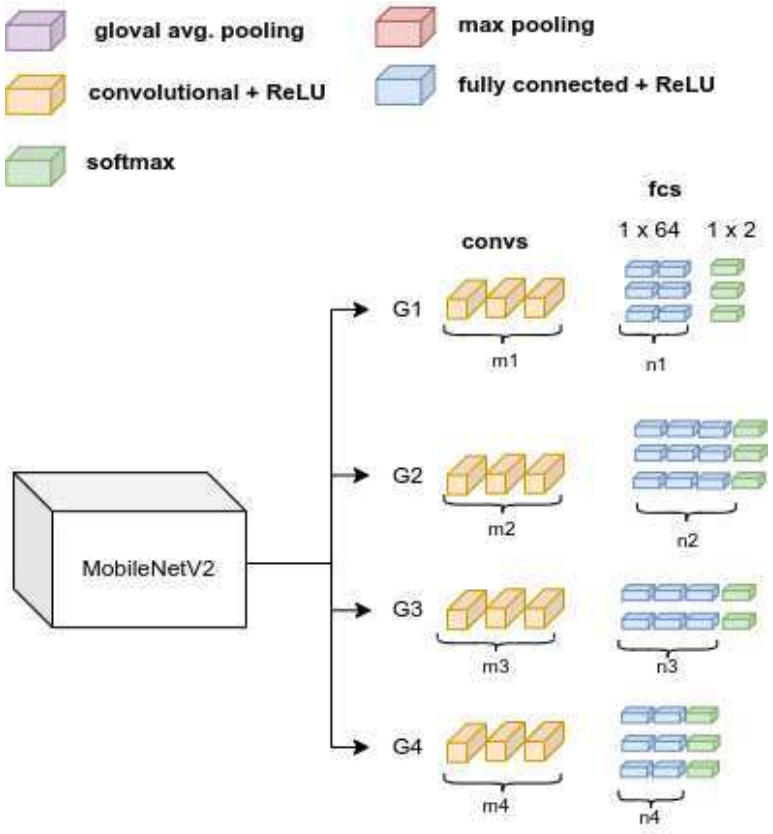


Figure 4.14: Neural Architecture Search generic architecture showing the searched parameters *m1,m2,m3,m4* relative to the number of convolutional layers, and *n1,n2,n3,n4*, relative to the number of dense layers. *Source: own author.*

candidate architecture is evaluated based on a chosen metric like accuracy or EER (4), and finally, the controller stores the result in its memory. This process occurs for many iterations (also called *trials*), and the architecture with the best result found at the end is chosen as the search result (5).

4.8.1 NAS Approaches

Random Search

We evaluated three different NAS approaches. The first was random search [Bergstra et al., 2012]. We randomly sorted from the predetermined search space (number of dense layers and number of convolutional layers) the *config* values and trained the random neural architecture. This interval was designed based on the number of available resources, so the proposed

architectures did not extrapolate the established training time limits. This first approach consists of a baseline for comparison with the other approaches.

Regularized Evolution

Regularized Evolution (RE) [Real et al., 2019] may be considered the state-of-the-art method. RE-based NAS involves evolutionary algorithms to create and select architectures based on their fitness scores. This method maintains a population of architectures and evolves them over successive generations. Compared to reinforcement learning-based methods, this approach is less complex, computationally efficient, and requires fewer resources. Additionally, it is faster at searching as multiple architectures can be evaluated concurrently, making it particularly effective in discovering architectures in large and discrete search spaces. We also have experimented with regularized evolution to compare the achieved results with the proposed method.

Reinforcement Learning Search

The process of neural architecture search fits very well with the deep reinforcement learning framework as discussed in Chapter 3. We based our work on previous research: we trained one LSTM network - our agent - as in [Zoph and Le, 2017; Zoph et al., 2018; Pham et al., 2018; Sable, 2019] and used Monte Carlo-based sampling as in [Yu et al., 2016] in the process of architectural composition.

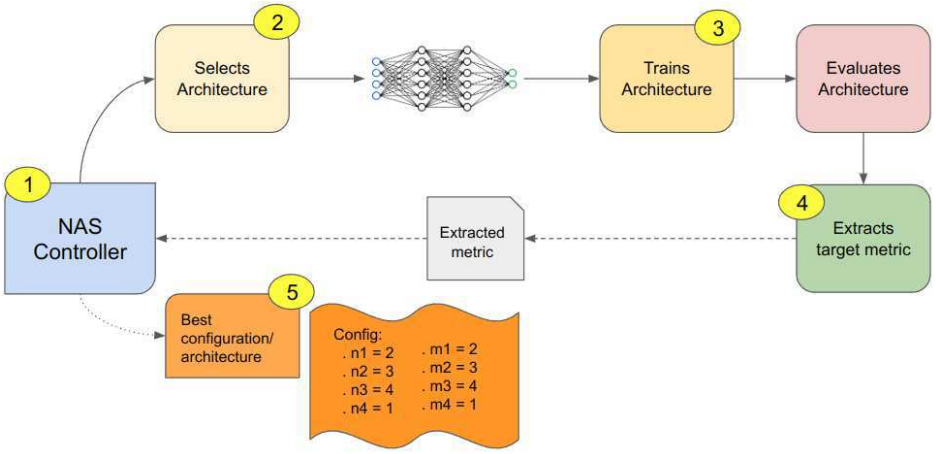


Figure 4.15: Neural Architecture Search basic process. *Source: own author.*

As the first trial, the NAS process starts with a random *config*, which represents a random proposed neural network for training and evaluation. Following the training and evaluation of each new architecture, we calculate the mean accuracy of all tasks and record it in the NAS Controller's history with the respective architecture configuration.

Figure 4.16 depicts the training schema of the LSTM model used as a learning agent in the Reinforcement Learning approach. The Search Space contains the list of candidates for the number of dense layers and the number of convolutional layers, independent of the task group. Based on the search space values and LSTM output, the controller selects the four numbers for each type of layer and each group of tasks using Monte Carlo sampling. So, the model is assembled and passed to the Model Trainer module, where it is trained, and the validation accuracy is measured.

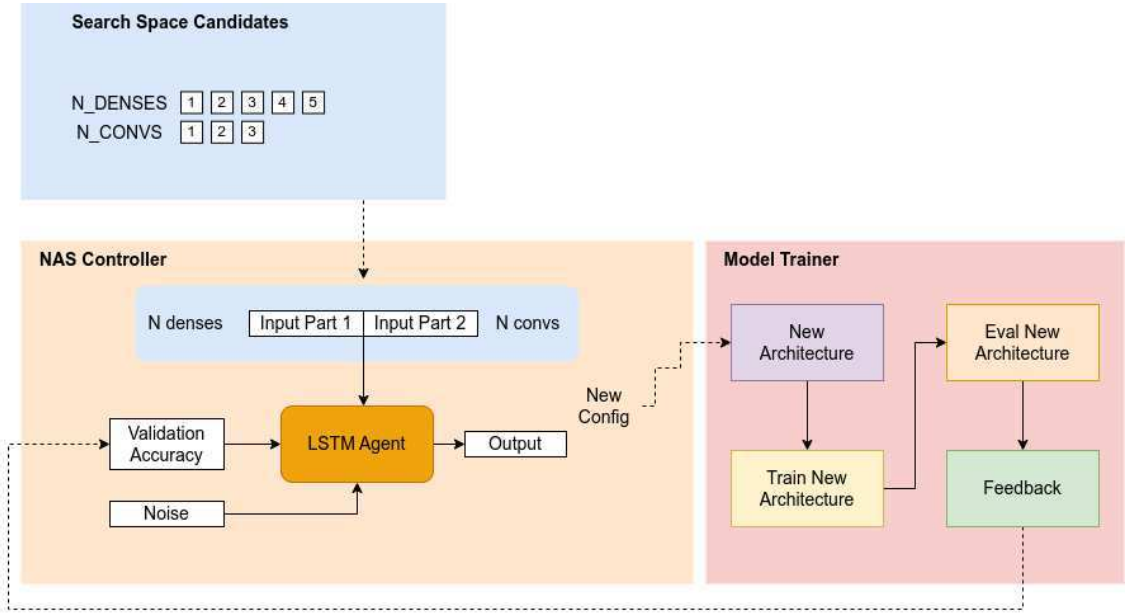


Figure 4.16: LSTM Training Schema. We have three main modules: the Search Space, the Model Trainer, and the NAS Controller. Note the LSTM agent receives the four numbers of dense layers and the four numbers of convolutional layers. When a batch of proposed architectures is trained and evaluated, the LSTM is retrained with this last batch of data. *Source: own author.*

After a batch of 20 architectures and respective accuracies are formed, the Controller model (LSTM) is retrained with this new data. The process repeats for a predefined number of trials. During the REINFORCE feedback loop, this validation accuracy is used as input

to calculate the discounted reward of the algorithm in the customized loss function of the LSTM. Note we also insert some noise in the LSTM input jointly with the previous models configurations. We checked experimentally that this method helps to improve the variability of proposed configurations (architectures), preventing the LSTM from repeatedly proposing the same architectures. When the searching process ends, we recreate the best model architecture found, train it for 50 epochs, and evaluate it in the test set to obtain the final results

Policy Gradient Formula

The process of applying reinforcement learning generally involves a series of steps. First, the agent executes an action based on a policy in a given environment. This leads to a new state in which the agent updates its policy based on a reward function. In the end, the process is repeated with the updated policy and new state.

In Chapter 2, we explained policy gradients as a set of methods used to update the policy, such as the REINFORCE algorithm. When implementing policy gradients in neural architecture search (NAS), the REINFORCE algorithm maximizes the log-likelihood per search step with the reward at each step. This technique aims to optimize the objective function that maximizes the agent's total rewards. Equation 4.2 presents the policy gradient formula we use in our Reinforcement Learning approach. In the context of NAS, the validation accuracy of each architecture created by the controller can serve as the reward function.

$$\nabla_\theta J(\pi_\theta) = \sum_{t=0}^{T} \nabla_\theta log \pi_{\theta(a_t|s_t)}(\sum_{t'=t+1}^{T} (\gamma^{t'-t-1}(r(s_{t'}) - b(s_{t'})))) \tag{4.2}$$

The equation above represents a reinforcement learning algorithm. The variable b is referred to as the baseline and it has a fixed value of 0.5. The purpose of the baseline is to balance the positive and negative rewards equally. This means that all values are normalized around this baseline. Additionally, γ is another variable that represents the discount factor. In this case, we have set γ to 0.3 using a more conservative approach. This approach gives more importance to the recently discovered networks over future ones, which helps to stabilize the search process.

Using the REINFORCE gradient for neural architecture search makes it possible to get

improved architectures. In this context, the feasible actions refer to the various layers or connections in the architecture, and the policy is the softmax distribution of these layers per controller step.

4.8.2 Transfer Learning and NAS

Our proposal for NAS (Neural Architecture Search) employs Transfer Learning from a base model, similar to our STL and MTL studies. Other studies have also demonstrated the effectiveness of Transfer Learning in the context of NAS [Wistuba et al., 2019; Xie et al., 2019; Wong et al., 2018; Istrate et al., 2018; Kokiopoulou et al., 2019]. We experimented with different base models, including MobileNetV2 and VGG16, which showed similar accuracy with the ICAO-FVC dataset. However, the inference time of MobileNetV2 was significantly lower than that of VGG16, which made the costly neural architecture search process quicker. Due to resource limitations, we did not explore deep VGG16 as a base model and instead used MobileNetV2 for all NAS experiments. The change on the base network used in the STL and MTL experiments in contrast with the NAS experiments may partially threaten the validity of our methodology. However, we achieved the desired results in a timely manner by conducting exploratory research, despite the unavailability of source code and NAS benchmarks during the experiments. We developed all necessary scripts and source code by studying various source repositories and existing studies that came to our knowledge during the research development. Moreover, we took the necessary precautions, such as verifying the accuracy of the base models before implementing any changes.

4.8.3 NAS Benchmarks

To do a more extensive analysis, we performed our experiments with Neural Architecture Search in a previously calculated dataset of NAS: the NATS-Bench [Dong et al., 2022], so that we could observe the method's effectiveness across different datasets (CIFAR-10, CIFAR-100 [Krizhevsky, 2009], and ImageNet16-120 [Chrabaszcz et al., 2017]). Building benchmarks such as NATS-Bench allows quicker advancements in the research field. In particular, evaluating the NAS algorithms investigated in this thesis did not require the training and evaluation of every proposed child architecture.

Other similar benchmarks are available for use, such as NAS-Bench-101 [Ying et al., 2019], NAS-Bench-201 [Dong and Yang, 2020], TransNAS-Bench-101 [Duan et al., 2021], and NAS-Bench-1Shot1 [Zela et al., 2020]. Each presents a dataset of architectures in a limited search space and the respective evaluation data like training time, training accuracy, validation accuracy, test accuracy, and architecture specifications. With these precomputed data, we can evaluate multiple NAS algorithms within seconds instead of hours or days, as would be necessary if we train and evaluate every proposed architecture. The main idea behind these benchmarks is to help create NAS algorithms capable of finding the best architecture faster than previously developed algorithms made available inside each NAS benchmark.

NATS-Bench Dataset

We preferred using the NATS-Bench dataset to develop and refine our NAS technique because it has a well-established API with good documentation and is already used in other related works [Fan et al., 2023; Li et al., 2023]. The NATS-Bench dataset has two different search spaces: a Size Search Space (SSS) and a Topology Search Space (TSS).

Figure 4.17 depicts the NATS-Bench search spaces and the basic architecture skeleton. The diagram contains three sets of cells linked by residual blocks. The base architecture is a ResNet-like neural network, with a sequence of five ($N = 5$) residual cells intercalated with two residual blocks and a global average pooling at the end. The figure top section exhibits the SSS search space, which has eight distinct options for the number of channels or filters of each cell, while the bottom part presents five diverse candidate operations that establish the topology of the Direct Acyclic Graph (DAG) forming the cells, defining the TSS search space. Considering this setup, a macro-search is done during the neural architecture search process (remind Chapter 2).

The datasets used for training and evaluating the proposed architectures are CIFAR-10 [Krizhevsky, 2009], CIFAR-100 [Krizhevsky, 2009], and ImageNet-16-120 [Chrabaszcz et al., 2017]. Every possible cell architecture is trained and evaluated within these two search spaces, and the results are stored in three different moments. The SSS has 32768 architectures, and the TSS has 15625 cell candidates, representing 48393 different architectures.

In the next chapter, we discuss the experiments performed with the different setups described in this chapter.

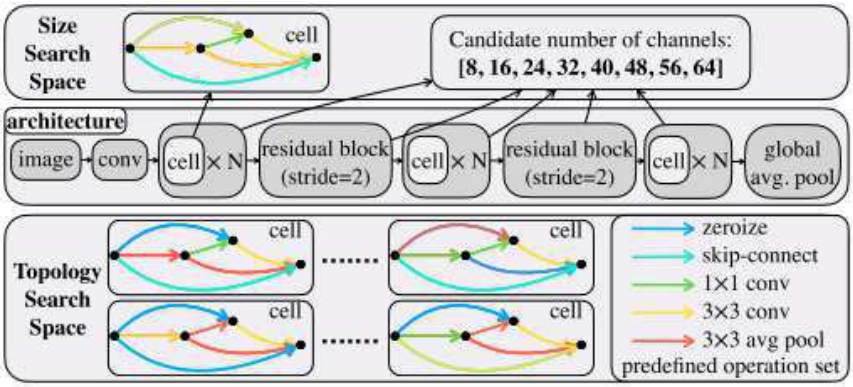


Figure 4.17: NATS-Bench dataset search spaces. *Source: [Dong and Yang, 2020]*

.

Chapter 5

Experimental Evaluation and Discussion

This chapter presents the experiments performed and the individual results in the test sets, followed by a discussion.

5.1 Single-Task Learning

Table 5.1 shows the results obtained for each STL network trained with ten epochs. We decided on ten epochs because the training converged with this number of epochs, we did not use early stopping in these experiments. Figure 5.1 depicts an example of this convergence for the *dark glasses* STL training. All networks have approximately 3.2M trainable parameters from 17.9M parameters, i.e., 14.7M were not modified during the training. In general, the results are not competitive: 15 out of 23 requisites presented an EER above 10%, and on average, the EER was 15.09%.

Note, that we did not evaluate the requisite Ink Mark once the random test set selected did not have instances of this requisite, so we could not calculate the metrics for it. We also did not evaluate this requisite throughout the other experiments for the same reason.

Error Analysis

Grad-Cam [Selvaraju et al., 2020] is a technique developed for aiding the explanation of the CNN-based models' decisions through visualizations produced on top of evaluated images. The technique facilitates the understanding of the model during analysis and gives insights into the model's failures.

ICAO Requisite	**EER**
Mouth	21.20%
Rotation	27.94%
Looking Away	21.04%
Eyes Closed	18.31%
Close	3.32%
Hat	1.40%
Dark Glasses	1.81%
Frames Heavy	50.00%
Frame Eyes	15.58%
Flash Lenses	11.76%
Veil	2.38%
Reflection	19.40%
Light	14.63%
Shadow Face	15.18%
Shadow Head	9.18%
Blurred	10.21%
Ink Mark	-
Skin Tone	21.13%
Washed Out	1.06%
Pixelation	31.79%
Hair Eyes	12.94%
Background	7.30%
Red Eyes	14.40%
Mean EER	15.09%
EER std dev.	11.54%
EER Median	14.52%

Table 5.1: Single-Task networks results for 10 epochs

The technique produced a heatmap plotted over the analyzed image in our implementation. The region that the network is paying attention to when deciding - compliant or

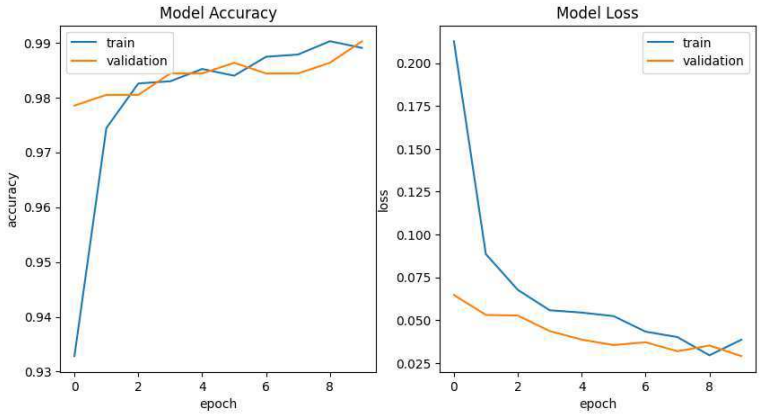


Figure 5.1: Training curves of accuracy and loss with STL with the dark glasses requisite.

not compliant with an ICAO requisite - is highlighted in green and yellow. In contrast, the regions highlighted in red are those the model is not paying attention to.

For example, we can observe in Figure 5.2 that the network is looking for the region right below the person's nose when the person is wearing a veil, meaning that it is not compliant with the *Veil* requisite. This pattern occurs in all true negative (the person truly is with the face occluded) images of Veil requisite and is coherent with human intuition. Similar results were observed in successful trainings - with low EERs - such as for *Dark Glasses*: 1.81% EER and *Hat*: 1.40% EER ICAO requisites. Observe that we evaluate each ICAO requisite in isolation, so there are no shared weights and, therefore, no learned features between the tasks, contributing to better results.

Figure 5.2: True negative examples of Veil requisite with Grad-CAM heatmaps

We also identified other patterns through Grad-CAM's heatmaps analysis. The network

pays attention to the person's shirt and headwear, such as hats and caps. Figure 5.3 exemplifies these cases. Despite the network's positive assertion (correctly classifies a compliant image) and high accuracy for this ICAO requisite, this can lead to failures in generalization.

Figure 5.3: True positives examples of Veil requisite with Grad-CAM technique applied.

Figure 5.4 shows the single case where the network failed to identify that the person was not wearing a veil. However, we can check that the case is dubious once the person has the face partially occluded by the shirt. Cases like this may even trick a human being into labeling this requisite, but certainly should not pass a system of ICAO requisites checking deployed in a real-world scenario.

Figure 5.4: False positive case of Veil requisite with Grad-CAM.

The Grad-CAM heatmap's analysis suggests a multi-task learning approach may be successful for the ICAO case: the fact the network makes hits in one task - such as the *Veil* requisite - while looking into regions of interest of other tasks - such as the *Hat* requisite -

reinforces the hypothesis that jointly learning the tasks may be beneficial and lead to better results. The Appendix A presents further STL results followed by Grad-CAM analysis of other ICAO requisites.

5.2 Multi-Task Learning

In this section, each multi-task learning approach result is analyzed in isolation and some conclusions are drawn.

5.2.1 Architecture HANDCRAFTED 1

We performed three experiments with the Handcrafted 1 approach. In the first one (**Exp. I**), we trained the network by ten epochs and observed the evolution of training curves: *accuracy vs. epoch* and *loss vs. epoch* while monitoring the training convergence. In the second one (**Exp. II**), we increased the number of trained epochs to 200 to see if the final accuracy would be higher with more training epochs and again observed the training curves. Lastly (**Exp. III**), we tried to fine-tune the base model for ten epochs, froze the base model weights, and trained the whole model for 200 epochs so that we could test the effect of the base model's fine-tuning for some epochs during the training. We selected the best model in all experiments based on the epoch with the highest validation accuracy. Table 5.2 details the obtained results in the test set.

It is possible to observe significant improvements through the experiments I to III for most ICAO requisites ending below 10% EER threshold: *Eyes Closed*, *Close*, *Flash Lenses*, *Light*, *Veil*, *Shadow Head* and *Hair Eyes*, and a special group of ICAO tasks that were even better with a mean EER of less than 2%: *Hat*, *Dark Glasses*, *Washed Out*, *Red Eyes*. Comparatively to the STL approach, most requisites also had a better result in this MTL approach.

It is important to notice that the great change in the EER result for the Frames Heavy case - from 0.88% to 50% - was mainly due to the class unbalancing, i.e., we had just two samples of NOT COMPLAINT images (0.69%) for this requisite in a total of 288 test images. So, in cases like this, a small change in the output may cause great variations over the final EER. Consequently, we decided to use the *Median EER* in addition to the *Mean EER* once this statistic is less vulnerable to outliers and better represents the distribution middle value if

ICAO Requisite	10 epochs	200 epochs	Fine-Tuned + 200 epochs
Mouth	30.65%	19.12%	**6.0%**
Rotation	28.8%	25.24%	27.06%
Looking Away	28.8%	25.58%	11.46%
Eyes Closed	21.86%	18.31%	**2.04%**
Close	7.17%	2.45%	**7.69%**
Hat	20.15%	1.82%	**0.98%**
Dark Glasses	6.53%	1.62%	**1.62%**
Frames Heavy	0.88%	50.0%	50.0%
Frame Eyes	19.58%	15.58%	**3.74%**
Flash Lenses	27.15%	8.55%	**4.08%**
Veil	12.25%	2.38%	**2.38%**
Reflection	37.47%	18.39%	15.32%
Light	39.94%	12.37%	**9.63%**
Shadow Face	22.5 %	13.89%	17.96%
Shadow Head	19.44%	7.94%	**8.21%**
Blurred	17.86%	10.26%	12.39%
Ink Mark	-	-	-
Skin Tone	45.74%	19.82%	19.39%
Washed Out	24.82%	0.7 %	**0.35%**
Pixelation	47.12%	24.32%	34.86%
Hair Eyes	20.72%	13.94%	**2.68%**
Background	30.56%	7.3 %	21.54%
Red Eyes	21.42%	18.96%	**1.77%**
Mean EER	24.16%	14.48%	**11.87%**
EER std dv.	11.88%	11.25%	12.61%
Median EER	22.18%	13.92%	**7.95%**

Table 5.2: Mean and Median EER on the test set of Handcrafted 1 MTL trainings. The best results (below 10%) and the final mean and median results are highlighted in bold.

outliers are present [Boslaugh, 2008]. We observed this fact in the results of Exp. III where *Median EER* is 7.95%, while *Mean EER* is 11.87%.

Finally, we observed no tendency of overfitting in the training of any ICAO task. Figure 5.5 shows train and validation accuracy curves presented few differences for each dataset split (train and validation).

5.2.2 Architecture HANDCRAFTED 2

We executed five experiments for the Handcrafted 2 approach: the first and second were trainings with 10 and 200 epochs, respectively, similar to Experiments I and II of the Handcrafted 1 approach.

The last three experiments tested modifications in data augmentation algorithm parameters. In Exp. III, we did no rotations in the images and changed the width and height shift range from 0.2 to 0.1 simultaneously. In Exp. IV and V, we performed no rotations in the images and trained the network for 50 and 200 epochs, respectively.

The hypothesis tested in these five experiments are (i) the negative effect of the rotation operation during data augmentation, which could be harming the validity of samples for the Rotation (Roll, Pitch, Yaw) requisite, mischaracterizing it; (ii) we also evaluated the effect of the shifting (translation) operation, whose value could be inadequate and then harming the training general efficacy; (iii) the effect of the number of training epochs, for this we checked the results for each requisite and for the general set of requisites in this proposed handcrafted architecture. Table 5.3 shows the achieved results.

We can observe that Exp. IV had the best results in terms of Median EER (5.07%) compared to the others. The Rotation requisite had slight improvement and showed the best result so far, even when compared to the STL training, confirming the hypothesis that image rotations for data augmentation were prejudicial to the network training because it harmed the characterization of this specific requisite. Other requisites also presented the lower EERs so far: *Eyes Closed*, *Light*, *Background*, and *Red Eyes*.

Following the successful empirical results achieved in the Handcrafted 2 model in Experiment IV, we also decided not to make any rotations and use the same width and height shifts (0.1) during data augmentation for the subsequent experiments.

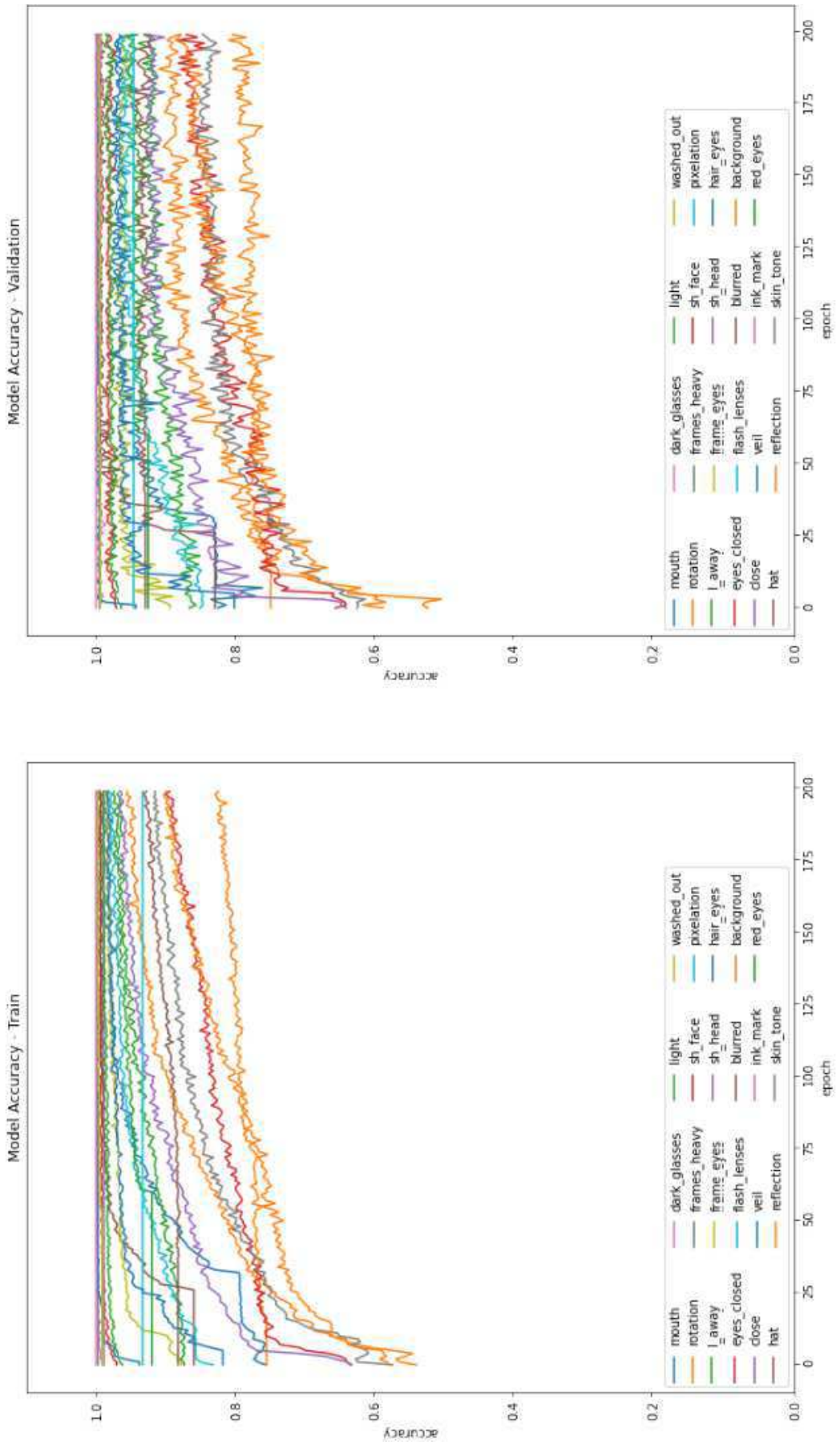


Figure 5.5: Training curves of MTL training - Experiment III - 200 epochs after 50 epochs of fine-tuning.

ICAO Requisite	Exp. I	Exp. II	Exp. III	Exp. IV	Exp. V
Mouth	4.11%	6.92%	6.22%	5.06%	3.45%
Rotation	24.52%	22.79%	20.10%	18.27%	18.27%
Looking Away	8.96%	9.58%	6.86%	8.83%	12.79%
Eyes Closed	2.58%	3.54%	1.98%	**1.98%**	2.90%
Close	10.84%	2.62%	2.62%	4.02%	4.02%
Hat	3.92%	4.07%	4.07%	2.88%	1.19%
Dark Glasses	0.00%	0.98%	0.39%	0.20%	0.98%
Frames Heavy	0.00%	50.00%	50.00%	50.00%	50.00%
Frame Eyes	5.95%	7.24%	4.34%	5.08%	3.26%
Flash Lenses	5.00%	6.18%	3.42%	4.70%	2.54%
Veil	0.00%	1.12%	2.38%	2.94%	50.00%
Reflection	10.90%	14.19%	12.15%	13.87%	13.87%
Light	13.04%	9.63%	11.62%	**8.03%**	12.94%
Shadow Face	8.33%	11.11%	6.76%	11.39%	13.42%
Shadow Head	7.03%	6.38%	8.21%	9.72%	8.97%
Blurred	11.43%	12.63%	9.62%	11.25%	11.05%
Ink Mark	-	-	-	-	-
Skin Tone	16.52%	17.94%	14.90%	16.64%	14.18%
Washed Out	0.00%	0.18%	50.00%	0.18%	50.00%
Pixelation	20.61%	21.66%	18.20%	22.99%	28.15%
Hair Eyes	8.00%	5.31%	3.05%	4.73%	5.10%
Background	5.84%	4.89%	4.51%	**3.87%**	6.23%
Red Eyes	3.00%	2.25%	1.98%	**1.98%**	3.54%
Mean EER	7.75%	10.06%	11.06%	9.48%	14.40%
EER std dv.	6.61%	10.98%	13.69%	10.91%	15.89%
Median EER	6.49%	6.65%	6.49%	**5.07%**	10.01%

Table 5.3: Handcrafted 2 MTL experiments results. The best results are highlighted in bold.

5.2.3 Architecture HANDCRAFTED 3

In this last MTL approach, we did three experiments (I, II, and III) varying solely the number of training epochs (10, 50, and 200, respectively), evaluating the training convergence and final Mean EER and Median EER. So, we could evaluate the architecture efficacy with three different numbers of epochs and observe the eventual improvement or decrease in terms of EER for each ICAO requisite and for the entire set of requisites. The results are available in Table 5.4.

Through the results evaluation, we are able to see very low EER for the following requisites: *Close*, *Dark Glasses*, *Frames Heavy*, *Skin Tone*, and *Pixelation*, which all had 0% EER in the test set. Skin Tone and Pixelation were two difficult tasks, considering the high EERs achieved by previous experiments of other approaches (above 10% on average). In the subsequent phases of this research, we investigated these results in more detail. The next section discusses the results of the NAS experiments.

5.3 Neural Architecture Search

As explained in Chapter 4 - Section 4.8, we experimented with two different approaches for Neural Architecture Search: a RANDOM approach to draw a baseline to our primary approach, which is the REINFORCE approach based on RL. The main goal is to discover the best ***config*** *(m1,m2,m3,m4) and (n1,n2,n3,n4)*; these eight numbers represent the branches length - in terms of the number of convolutional layers and the number of fully connected layers, respectively - of each branch for the four predefined groups of tasks.

5.3.1 NATS-Bench Dataset Experiments

The NATS-Bench dataset has two search spaces: Size Search Space (SSS) and Topology Search Space (TSS), with the purpose of benchmarking different methods of NAS with no training and evaluation of architectures needed during the searching process, only queries to the previously recorded NAS dataset. As explained in Chapter 4, we tested three different methods with this dataset: RANDOM, Regularized Evolution, and our version of Reinforcement Learning. In this section, we report and discuss the results of each method by search

ICAO Requisite	**Exp. I**	**Exp. II**	**Exp. III**
Mouth	27.00%	9.00%	3.67%
Rotation	28.00%	21.68%	18.75%
Looking Away	18.00%	10.40%	11.96%
Eyes Closed	4.00%	0.00%	3.75%
Close	0.00%	0.00%	**0.00%**
Hat	12.00%	1.40%	1.61%
Dark Glasses	3.00%	0.00%	**0.00%**
Frames Heavy	0.00%	0.00%	**0.00%**
Frame Eyes	15.00%	2.42%	4.83%
Flash Lenses	15.00%	4.70%	3.86%
Veil	0.00%	0.00%	4.82%
Reflection	19.00%	12.15%	13.17%
Light	21.00%	13.69%	8.22%
Shadow Face	18.00%	13.14%	12.13%
Shadow Head	10.00%	8.97%	6.16%
Blurred	18.00%	13.86%	9.67%
Ink Mark	-	-	-
Skin Tone	0.00%	0.00%	**0.00%**
Washed Out	22.00%	17.65%	19.10%
Pixelation	0.00%	0.00%	**0.00%**
Hair Eyes	35.00%	26.24%	27.39%
Background	6.00%	6.78%	4.89%
Red Eyes	13.00%	12.46%	10.42%
Mean EER	12.91%	7.93%	7.47%
EER std dv.	10.36%	7.83%	7.32%
Median EER	14.00%	7.88%	**4.86%**

Table 5.4: Handcrafted 3 MTL experiments results. The best results are highlighted in bold.

space and dataset (CIFAR-10, CIFAR-100, and ImageNet16-120).

Performance in Terms of Accuracy

First of all, we noticed that all three methods are able to achieve good results if the search process can occur in enough time. Figures 5.6 and 5.7 show the validation and test accuracies of the best architectures found by the NAS methods when considering different time budgets across all datasets in SSS search space, while Figures 5.8 and 5.9 show the results for the best architectures found in the TSS search space. Note that, in the case of architectures found with the same validation or test accuracy during a search, we consider the best the one found first, so the speed of the methods in finding the best architecture earlier is recognized.

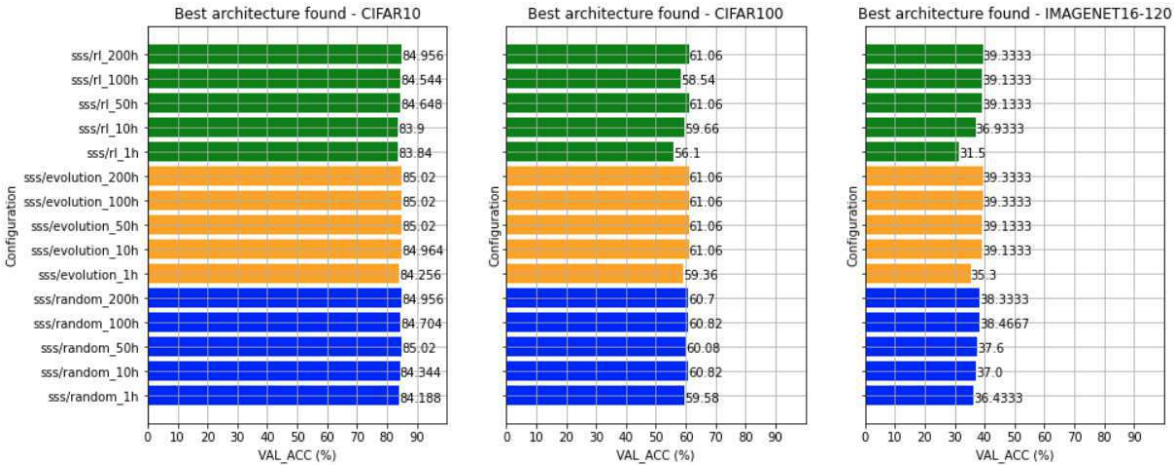


Figure 5.6: Validation accuracy of best architecture found by each NAS method in different time budgets in the SSS search space.

The Size Search Space (SSS) is more restricted and easier for the methods to propose good architectures; considering that a ResNet-like architecture is the basis for the proposed ones, all methods, independently of the time budget and dataset, reach nice and similar results in terms of accuracy, as we observe in Figures 5.6 and 5.7. Previous research demonstrates how the RANDOM approach can achieve competitive results if compared with NAS algorithms [Wistuba et al., 2019; Yu et al., 2019; Liashchynskyi and Liashchynskyi, 2019] especially when running on simple search spaces.

When it comes to searching for the best topology, the Topology Search Space (TSS) poses a greater challenge, leading to more varied results across different approaches. For the CIFAR-10 dataset, the proposed RL method struggled to achieve the best results within 50 hours of search time, especially compared to the state-of-the-art method (SOTA) - Reg-

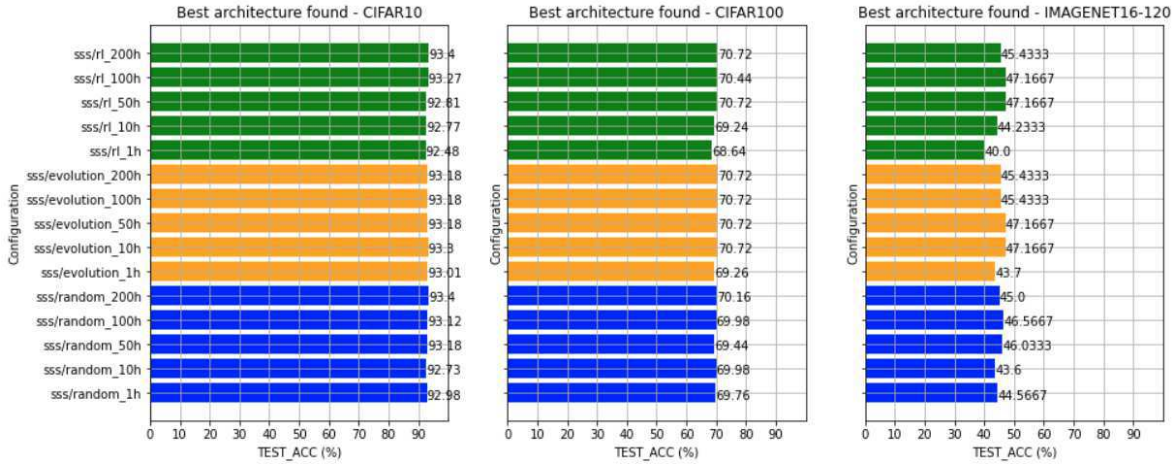


Figure 5.7: Test accuracy of best architecture found by each NAS method in different time budgets in the SSS search space.

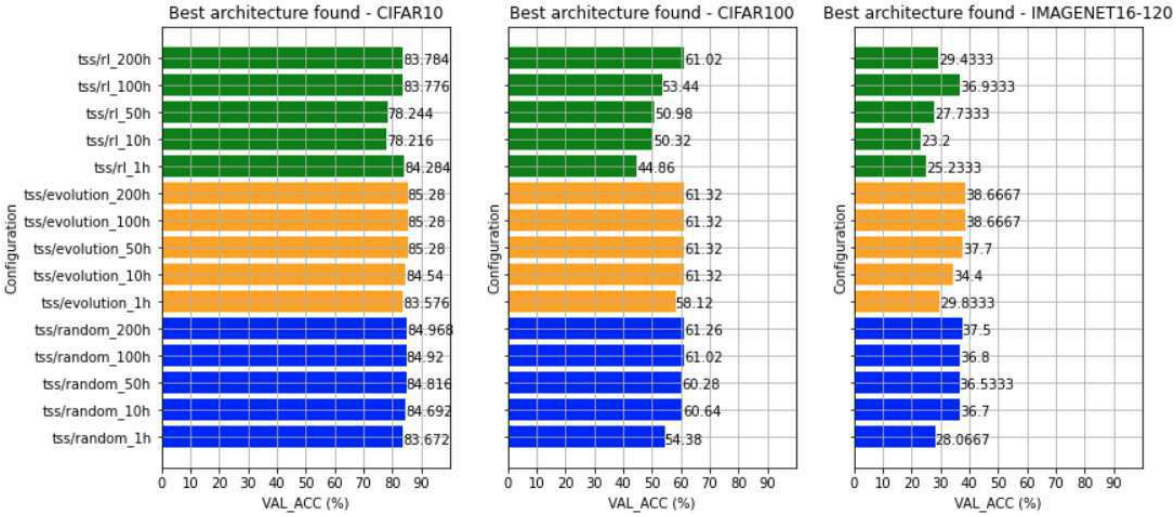


Figure 5.8: Validation accuracy of best architecture found by each NAS method in different time budgets in the TSS search space.

ularized Evolution, which delivered acceptable results with only 10 hours of search. This remained true for all three datasets, even the more difficult ones like ImageNet16-120. However, the RL method achieved near-SOTA or SOTA results within a time budget of 200 hours for all cases studied.

While RE is a valuable approach, it can encounter difficulties when it comes to making precise architectural decisions. Moreover, it may lack flexibility in certain problem domains due to its heavy reliance on random mutations and selections [Liu et al., 2023], which may

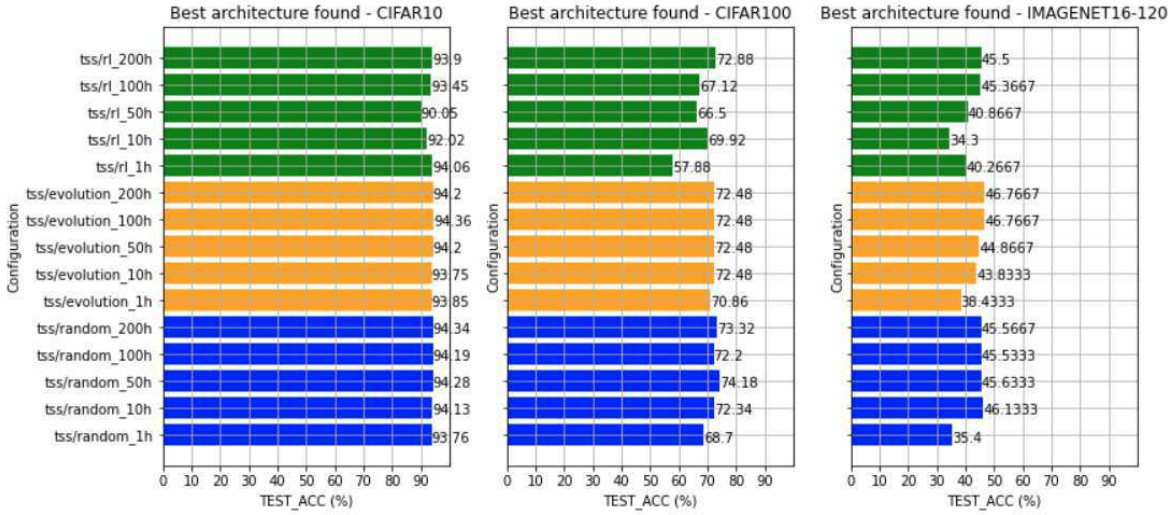


Figure 5.9: Test accuracy of best architecture found by each NAS method in different time budgets in the TSS search space.

not efficiently adapt to the complexity and nuances of those domains.

In contrast to RL methods, which can learn from past experiences and adjust their search strategy based on learned insights, RE methods have a more limited capacity to adjust their exploration strategy during the search process. This can result in the algorithm getting stuck in sub-optimal regions of the search space and missing out on potentially better architectures. Additionally, the method may converge on local optima and may not explore architectural diversity as effectively as RL-based methods. As a result, RE may require more generations to discover optimal architectures, particularly in complex search spaces, taking more time to finish the neural architecture search process.

Performance in Terms of Time

It is possible to confirm our conclusions by examining the time taken by each approach to find the best architectures on different datasets and time budgets. Figures 5.10 and 5.11 depict this. Despite the validation and test accuracies being very close to each other, different performances were observed when evaluating time.

In the ImageNet16-120 dataset, our RL approach yielded superior results in the SSS search space. Regardless of the time budget, our method found the optimal architecture earlier than the RE and RANDOM approaches. In four out of the five time budgets evaluated,

our RL approach was able to find the best architecture in less than 10 hours, and even in the one instance where it took longer, it still outperformed the RE and RANDOM approaches in the same time budget. Notably, our RL approach was highly efficient in finding the best architecture in less than 1 hour in three cases. On average, it takes 12.2 hours for the RL approach to reach the optimal architecture, while the RE approach takes three times longer, with 37.6 hours.

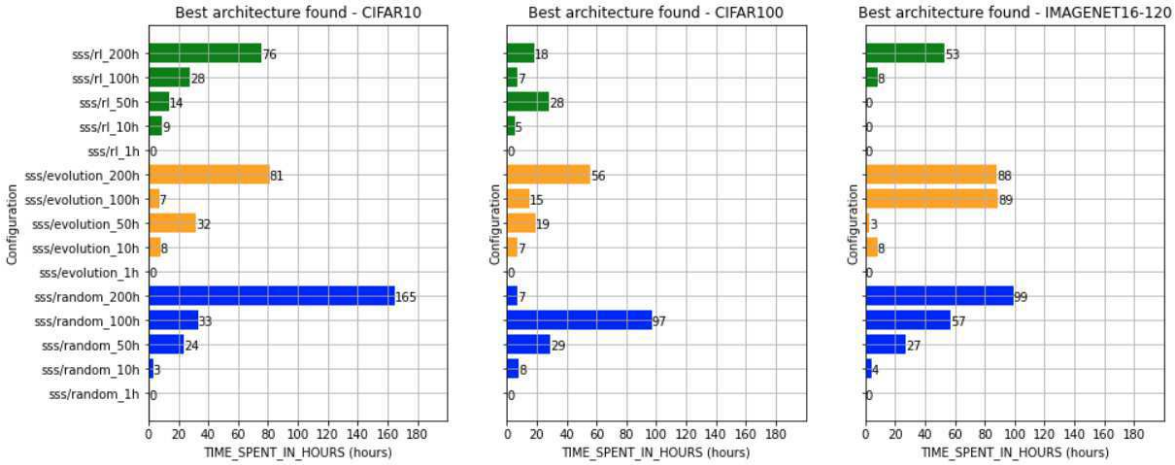


Figure 5.10: Time spent in hours to find the best architecture by each NAS method in different time budgets in the SSS search space.

Our RL method has proven to be effective on the CIFAR-100 dataset, with results comparable to other methods. In four out of five scenarios, the RL method identified the best architecture in less than 20 hours, which is comparable to the RE algorithm. However, it is worth noting that even the worst time result obtained with the RL method still required half the time it takes to get the worst result with the RE algorithm. The RL method takes 11.6 hours for this dataset to reach the best architecture, while the RE method takes 19.4 hours.

The CIFAR-10 dataset results show that our method reached the best architecture in less than 30 hours in four out of five scenarios, which is comparable to other approaches. The RL approach took an average of 25.4 hours, while the RE method took 25.6 hours to reach the same outcome.

Similarly, when we look at the TSS search space results, the RL approach took no more than 32 hours to find the best architectures in the ImageNet16-120 dataset. It took 19.4 hours on average, while the RE approach took 65.8 hours. For the CIFAR-100 dataset, the

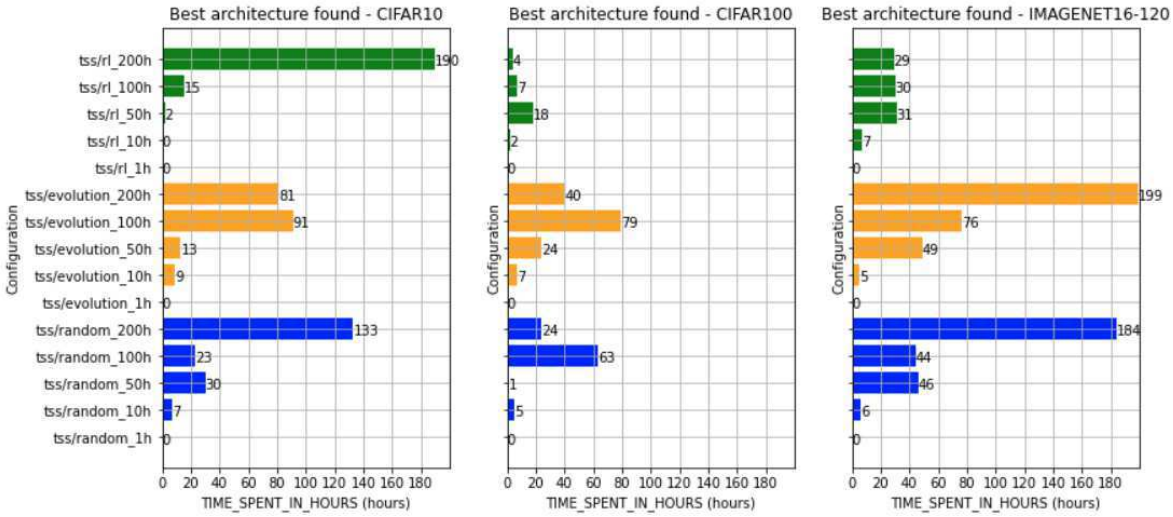


Figure 5.11: Time spent in hours to find the best architecture by each NAS method in different time budgets in the TSS search space.

RL approach again reached the best results earlier, with less than 20 hours required to find the best architecture. On average, the RL technique took 6.2 hours, while the RE took 30 hours.

In relation to the CIFAR-10 dataset within the NATS-Bench dataset, we observed that the RE approach took, on average, 38.8 hours to find the best architecture, while the RL approach took 41.4 hours. It is important to note that an outlier value was observed with the 200-hour budget, where the RL method took an exceptionally long time of 190 hours to find the best architecture.

Tables 5.5 and 5.6 summarize the results regarding the average time taken by each method to find the best architecture. This set of results highlights the significant advantage of the proposed method over the RE and RANDOM methods in most cases. This is particularly crucial in the context of NAS, where training and evaluation time are both critical and expensive. The sooner the method can identify the optimal architecture, the better it is.

Stability in Agent Training

The graph in Figure 5.12 displays the decreasing loss of the LSTM agent over time. The loss stays low throughout the neural architecture search, which is essential for the network

Dataset/Method	CIFAR-10	CIFAR-100	ImageNet16-120
RANDOM	45h	28.2h	37.4h
RE	25.6h	19.4h	37.6h
RL	**25.4h**	**11.6h**	**12.2h**

Table 5.5: Average time to find the best architecture by each method in every dataset on NATS-Bench in the SSS search space. The best results are highlighted in bold.

Dataset/Method	CIFAR-10	CIFAR-100	ImageNet16-120
RANDOM	**38.6h**	18.6h	56h
RE	38.8h	30h	65.8h
RL	41.4h	**6.2h**	**19.4h**

Table 5.6: Average time to find the best architecture by each method in every dataset on NATS-Bench in the TSS search space. The best results are highlighted in bold.

to propose better architectures and find the best one more quickly. The stability and low loss values were achieved by adding to the LSTM training two types of regularization based on l_2 norm. The kernel and recurrent regularizers were also crucial in achieving this stability.

5.3.2 Traditional Datasets Case Study

This section exposes the results of the neural architecture search executed over the four traditional datasets we used to execute a complete neural architecture search: CIFAR-10, MNIST, FASHION-MNIST, and Celeb-A. In the next section, we present the results of the ICAO-FVC dataset. We searched for the best architecture using our RL-based method on each dataset, followed by the training and evaluation of the best-found neural network. Table 5.7 details the obtained results.

MNIST The RL-based approach proposed achieved an accuracy of 99.85% in the MNIST dataset, which is close to the state-of-the-art results obtained by other methods such as 99.84% [An et al., 2020], 99.87% [Byerly et al., 2020], and 99.91% [Hirata and Takahashi, 2020]. It is worth noting that the associated error of about 0.1%-0.2% can be considered the test data noise and is hard to achieve with existing methods and neural networks [Byerly

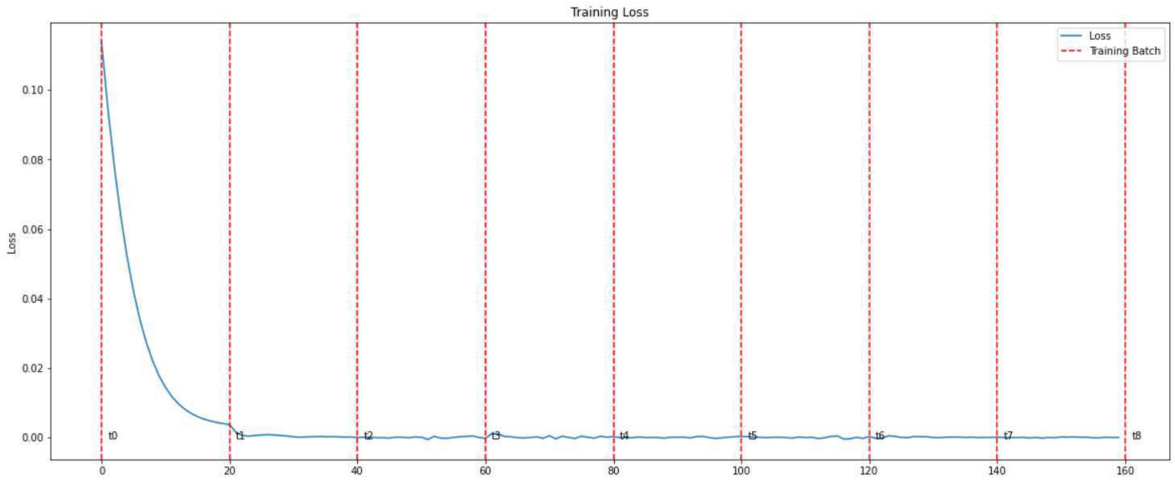


Figure 5.12: Training loss curve of the LSTM agent. The red dashed lines indicate batch accumulations that define the different training periods. It is worth noting that the first training session occurs at $t1$, and subsequent trainings occur at different times ($t2$ to $t7$) with distinct batches.

Dataset	Accuracy	EER mean	EER median	EER std. dv.	Total time
MNIST	99.85%	0.15%	0.11%	0.11%	12h
CIFAR-10	95.77%	4.24%	3.19%	1.99%	10h
Celeb-A	88.04%	11.97%	11.01%	7.64%	44h
FASH.-MNIST	97.66%	2.36%	1.55%	1.99%	13h

Table 5.7: Results of RL-based method on the traditional datasets. The total time refers to the time spent by the neural architecture search added to the time required to train for 50 epochs and the final evaluation (inference) of the found network.

et al., 2020]. Also, the best architecture was found in the first ten iterations of the neural architecture search, which may be considered an early stage of the search process. These results are summarized in Figure 5.8.

CIFAR-10 In our experiments with the CIFAR-10 dataset, we found that the results we obtained were not as good as those of other works that also used neural architecture search methods and networks with a similar number of parameters. Our final network had approximately 6.2 million parameters and achieved an accuracy of 95.77%, using an architecture discovered in the first ten iterations of our search process.

Method	**Accuracy**
Sanghyeon An *et al.* [An et al., 2020]	99.84%
Adam Byerly *et al.* [Byerly et al., 2020]	99.87%
Daiki Hirata and Norikazu Takahashi [Hirata and Takahashi, 2020]	**99.91%**
Proposed RL-based Approach	99.85%

Table 5.8: Comparison of results in the MNIST dataset. The best results are highlighted in bold.

When we compare our results with those in the literature, we see that Zhichao Lu *et al.* [Lu et al., 2020] achieved around 98% accuracy on the CIFAR-10 dataset using a method based on Evolutionary Algorithms and transfer learning. Han Cai *et al.* [Cai et al., 2019] also achieved around 98% top-1 accuracy with a method based on Reinforcement Learning and gradient descent, similar to ours. Additionally, Xiangxiang Chu *et al.* [Chu and Zhang, 2020] achieved 97.61% accuracy using an implementation derived from the work called DARTS [Liu et al., 2018c]. They proposed injecting some random noise to address the issue of performance collapse due to the aggregation of skip-connections.

Although our results are not better than these state-of-the-art results, our proposed approach is competitive, being around 2.23 percentage points of the top-performing models. Table 5.9 summarizes these reported results.

Method	**Accuracy**	**#params**
Zhichao Lu *et al.* [Lu et al., 2020]	**∼98%**	-
Han Cai *et al.* [Cai et al., 2019]	**∼98%**	5.7M
Xiangxiang Chu *et al.* [Chu and Zhang, 2020]	97.61%	4.3M
Proposed RL-based Approach	95.77%	6.2M

Table 5.9: Comparison of results in the CIFAR-10 dataset. The best results are highlighted in bold. The size of each model is depicted in terms of the number of parameters in the *#params* column.

Celeb-A Our performance in relation to the Celeb-A dataset was the lowest results achieved in terms of accuracy. The best network the NAS method found only reached

88.04% accuracy and had 7.4M parameters. If we compare this to the state-of-the-art results in the literature with a similar or smaller number of network parameters, we can see the scenario in Table 5.10 [Guo et al., 2020].

Method	**Accuracy**	**#params**
Branch-VGG [Lu et al., 2017]	90.79%	2.09M
LearnToBranch-VGG [Guo et al., 2020]	91.55%	1.94M
LearnToBranch-Deep-Wide [Guo et al., 2020]	**91.62%**	6.33M
GNAS [Huang et al., 2018]	91.36%	6.41M
Proposed RL-based Approach	88.04%	7.42M

Table 5.10: Comparison of results in the Celeb-A dataset. The best result is highlighted in bold.

FASHION-MNIST Results on the FASHION-MNIST dataset are depicted in Table 5.11. Our approach achieved better results than state-of-the-art works on the same dataset. In fact, our discovered network outperformed the Inception-v3 network [Gavrikov and Keuper, 2022]. Other recent works have focused on different forms of data augmentation to improve network accuracy [Harris et al., 2020; Zhong et al., 2017], while others fine-tune the weakly supervised DARTS approach [Tanveer et al., 2020]. In another work [Foret et al., 2020], a technique was developed that simultaneously minimizes the loss value and loss sharpness during gradient descent to improve model generalizability.

Our RL-based approach found the best architecture on a later search iteration. However, it's worth noting that networks with over 97% accuracy emerged since the first iterations of the search, demonstrating the high capacity of the method to find suitable architectures in a short period of time.

5.3.3 ICAO Dataset Case Study

In this section, we discuss the results of our experiment with the ICAO dataset and compare them with the results reported in the literature. The general results achieved in the ICAO-FVC dataset are shown in Table 5.12. Our NAS RL-based model performed promisingly, achieving results just 1.5 percentage points below the best MTL result (Handcrafted

Method	**Accuracy**
Harris *et al.* [Harris et al., 2020]	96,36%
Tanveer *et al.* [Tanveer et al., 2020]	96.91%
Foret *et al.* [Foret et al., 2020]	96.41%
Zhong *et al.* [Zhong et al., 2018]	96,35%
Gavrikov and Keuper [Gavrikov and Keuper, 2022]	94.44%
Proposed RL-based approach	**97.66%**

Table 5.11: Comparison of results in the FASHION-MNIST dataset. The best result is highlighted in bold.

Approach 3). The entire search, training, and evaluation process took 19 hours.

Dataset	**Accuracy**	**EER mean**	**EER median**	**EER std. dv.**	**Total time**
ICAO-FVC	91.98%	7.79%	6.4%	7.81%	19h

Table 5.12: Results of the proposed RL-based method on the ICAO-FVC dataset. The total time refers to the time of neural architecture search added to the following train for 50 epochs and the final evaluation of the found network.

We cannot compare our results directly with other works reported in the literature as we have not yet submitted our model to the FVC-Ongoing competition. Once we do that, we can verify the efficacy of our model on the same test dataset as other solutions developed by private companies, academic institutes, or independent developers.

However, we need to consider two caveats. First, the FVC-Ongoing test set is different from ours, however it has a similar distribution to the validation set provided as guaranteed by the competition organization and that was used as our test set in our experiments. Second, we could not evaluate the requisite Ink Mark because of the lack of samples of this class in our test set. Therefore, we decided to assess the performance of our method by comparing it with some top solutions submitted to the competition platform. We only considered solutions that evaluate all 23 ICAO requisites so that we can compare more fairly with our model, which evaluates all 23 requisites simultaneously. Table 5.13 shows the results of BioLab [Ferrara et al., 2012], BioTest [BioTest, 2017], Biopass Face [Vsoft, 2017], and ICAONet [Gualberto et al., 2022], along with the best results obtained by us on MTL (Experiment III) and on

NAS with RL approach.

The IcaoNet method did achieve good results in 9 of 23 ICAO requisites. However, it had a high EER in other requisites, such as Looking Away, Red Eyes, and Dark Glasses. These results caused the method to have a high average EER, despite having a relatively low median EER. When we analyze the results obtained by Biopass Face and BioLab, we see that each achieves the best result in three different requirements, showing a certain complementarity between these solutions. The lowest median EER identified was from Biopass Face (3.10%).

In relation to our proposed methods, namely Multi-Task Learning (MTL) and Neural Architecture Search (NAS), we observed promising results when we compared specific requisites with the existing solutions. In fact, for some requisites highlighted in bold in Table 5.13, these approaches have the lowest Equal Error Rate (EER). For instance, the requisites Close, Hat, Frames Heavy, Skin Tone, Pixelation, Veil, and Dark Glasses showed zero or near-zero EER when using the MTL and NAS-based methods, surpassing even the state-of-the-art methods.

It is worth noting that both networks - handcrafted and automatically generated - achieved good results in practically the same tasks. This is perhaps because the NAS search space was modeled considering the Handcrafted Approach 3 network as a reference, resulting in the networks generated with similar architectures. However, if a different search space embraces other types of layers, convolutions, network blocks, and operations, the NAS method could achieve better results in the other tasks.

Furthermore, our solution had competitive results when considering the EER median and mean as reference metrics. This was in comparison to other solutions that also analyzed the 23 ICAO requisites. In the future, we plan to submit our model to the FVC-Ongoing platform, gather more precise results, and provide a deeper analysis of this topic relative to the ICAO-FVC dataset and requisites. We have not submitted yet because there are some challenges that we must overcome, such as: (i) we have to submit a single Windows executable file; (ii) this executable has a maximum file size, which is very restrictive; (iii) there is a time limit of execution of the test dataset on the FVC platform that cannot be surpassed.

In the next chapter, we summarize our research steps, the achieved results, and future research directions.

Task	BioLab	BioTest	BPFace	IcaoNet	MTL III	NAS RL
Mouth	6.20%	5.00%	3.80%	**2.1%**	3.67%	9.68%
Rotation	12.70%	12.60%	10.70%	**5.4%**	18.75%	11.88%
Looking Away	20.60%	24.60%	13.30%	49.0%	**11.96%**	13.61%
Eyes Closed	4.60%	6.70%	4.60%	**1.7%**	3.75%	6.16%
Close	21.60%	15.40%	1.20%	1.2%	**0.00%**	**0.00%**
Hat	14.00%	16.50%	9.80%	7.3%	**1.61%**	2.88%
Dark Glasses	1.90%	2.10%	1.80%	29.0%	**0.00%**	**0.00%**
Frames Heavy	5.80%	3.30%	2.10%	13.7%	**0.00%**	**0.00%**
Frame Eyes	6.30%	4.00%	10.70%	**0.8%**	4.83%	7.49%
Flash Lenses	**2.10%**	2.30%	2.70%	8.4%	3.86%	5.57%
Veil	2.50%	3.70%	1.40%	4.6%	4.82%	**0.00%**
Reflection	**0.60%**	1.20%	1.40%	1.0%	13.17%	12.15%
Light	4.20%	4.60%	**3.10%**	8.2%	8.22%	12.74%
Shadow Face	13.10%	15.90%	9.90%	**3.3%**	12.13%	13.61%
Shadow Head	**2.30%**	2.40%	5.40%	3.3%	6.16%	6.43%
Blurred	5.20%	30.50%	1.60%	**0.4%**	9.67%	5.62%
Ink Mark	3.40%	3.60%	4.80%	**0.8%**	-	-
Skin Tone	4.00%	5.10%	1.90%	9.5%	**0.00%**	**0.00%**
Washed Out	9.60%	9.20%	**0.00%**	2.3%	19.10%	15.63%
Pixelation	1.30%	32.40%	1.30%	5.7%	**0.00%**	**0.00%**
Hair Eyes	12.80%	12.40%	13.00%	**0.0%**	27.39%	35.06%
Background	5.20%	3.70%	5.20%	**2.3%**	4.89%	6.36%
Red Eyes	7.40%	10.30%	**1.70%**	41.4%	10.42%	6.60%
Mean EER	7.28%	9.89%	**4.84%**	8.75%	7.47%	7.79%
EER std dev.	5.90%	9.05%	**4.18%**	12.83%	7.32%	7.81%
Median EER	5.20%	5.10%	**3.10%**	3.3%	4.86%	6.4%

Table 5.13: Comparison between submitted solutions to FVC-Ongoing competition and our approaches results. Note the platform uses its own test set, different from ours. The best results are highlighted in bold.

Chapter 6

Concluding Remarks

This chapter summarizes the research with the achieved results, shows some conclusions, describes how we answered the research questions and depicts future research steps.

6.1 Research Summary and Conclusions

This thesis proposed and experimentally evaluated a new NAS approach to speed up the process of finding new architectures suitable to address various problems, with a special interest in MTL scenarios. The key and most difficult problem investigated was to find an architecture to check the compliance of face images with respect to ICAO requirements. Our investigation also included the following additional scenarios to show the generality of the proposed approach: MNIST, FASHION-MNIST, Celeb-A, and CIFAR-10.

We conducted comprehensive experiments to compare and analyze NAS, MTL, and STL approaches, with the goal of combining NAS and MTL to create multi-task learning neural networks with the same problem-solving capability as a handcrafted network. The experiments showed the advantages of using MTL networks over STL ones through an explainability technique called Grad-CAM. Besides that, the experiments demonstrated how the shared network branches and the division of tasks into related groups may be beneficial in a MTL network. The knowledge acquired on the handcrafted design of MTL networks was used to create a basic architectural setup and a search space that was used by the NAS technique.

As a result of our analyses, we proposed a NAS method that uses Reinforcement Learning and the REINFORCE algorithm to train a Network Controller (LSTM) to identify the most

effective neural network architecture, which is then evaluated for its ability to solve multiple tasks simultaneously on a particular dataset. We fine-tuned the proposed approach on the NATS-Bench NAS dataset/benchmark and found that the method can find architectures with the same accuracy as a state-of-the-art method in NAS (Regularized Evolution) but with an exceptional search speed.

We evaluated the RL-based NAS method on the FVC-ICAO dataset and traditional datasets (MNIST, Fashion-MNIST, Celeb-A, and CIFAR-10), comparing it to STL and MTL methods and related methods found in the literature. The results of the NAS REINFORCE approach are competitive, with a median EER of 6.4% in the FVC-ICAO dataset. Additionally, we obtained competitive results for all traditional datasets, comparable to the state-of-the-art works. In special, our method presented above SOTA results in the dataset FASHION-MNIST.

Finally, we published the research, including the methodology, the experiments, and the results achieved, in a conference paper in 2023 [Gadelha et al., 2023].

6.2 Revisiting our Research Questions

The proposed Neural Architecture Search (NAS) approach demonstrates a high degree of generalization to different tasks and efficacy across multiple datasets, even surpassing the state-of-the-art in some cases. The NAS approach can find good architectures faster than the Regularized Evolution technique, which is currently considered state-of-the-art while maintaining similar efficiency in terms of accuracy. The technique we developed showcases the potential of NAS to automatically design Multi-Task Learning (MTL) architectures, including a base network and transfer learning. Our experimental methodology gave us insights by comparing manually designed architectures using the STL and MTL approaches to networks generated automatically using NAS. The development of this technique and the experimental evaluation provide answers to the research questions raised in Chapter 1:

- RQ 1: How to apply NAS to design MTL architectures automatically for image classification tasks?

- RQ 2: How do the architectures discovered through NAS differ from handcrafted ar-

chitectures designed using the STL and MTL paradigms, and what insights can be gained from this comparison?

We successfully answered our first research question by demonstrating the effectiveness of our NAS technique, which is based on RL. We proposed a task branching schema using a pre-trained network as the foundation to create a search space for our NAS approach. Our approach allowed us to use a Long Short-Term Memory (LSTM) neural network as the learning agent to identify efficient MTL architectures through RL.

Our study found that the architectures discovered using NAS are not significantly different from the MTL Handcrafted III architecture, as the search space was designed based on it. Therefore, the networks found by NAS do not differ much from the best architecture manually designed. However, it is important to note that although the handcrafted architecture has shown a final result that was better in the ICAO-FVC dataset, the achieved results in individual tasks were equally the best possible (0% EER) in the MTL and in the NAS approaches.

There are some criticisms that can be made about using NAS for designing MTL networks. Firstly, the final discovered architecture may be difficult to interpret due to the particular search space used, thereby making it hard to understand the network design in the same way that a human expert can on handcrafted networks. Secondly, NAS-based architectures are highly dependent on the quality of the training data used, and the use of poor-quality data may lead to the discovery of low-performance networks. Therefore, even though NAS can be used to automate network discovery, it should be used with caution and under the supervision of a human expert. It should not be seen as a replacement for the entire process of network discovery.

6.3 Future Research

In the experiments presented in this thesis, the architectures resulting from the proposed NAS approach did not produce better results than the handcrafted architecture in relation to accuracy and equal error rate. However, we believe there is potential to uncover new and better architectures by adding new types of layers, operations, attention mechanisms, and hyperparameters into the search space. These new architectures could surpass the best ones

found for each multi-task dataset evaluated in this thesis. This represents a key area for future research and could even involve the newer transformer-based architectures, being necessary only to produce an appropriate search space.

In addition, further research can expand the method's degrees of freedom by allowing it to determine the task grouping into branches within the discovered networks. This has the potential to improve the final results by evaluating relative tasks together in the same task branches, enabling them to share common knowledge within the networks. As discussed in the previous chapter about the MTL experimental results, this could significantly improve the overall performance.

Another future work is to improve interpretability techniques like Grad-CAM but adapted to the MTL context so we can better evaluate the network classification when solving multiple tasks simultaneously. This would help us better understand the network's decision process and improve the choices of new architectures made by our NAS method based on this analysis.

One last idea is to replace the LSTM model with a newer transformer-based network that is faster to train and may have lower computational costs.

References

[Ahn et al., 2019] Ahn, C., Kim, E., and Oh, S. (2019). Deep elastic networks with model selection for multi-task learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:6528–6537.

[An et al., 2020] An, S., Lee, M., Park, S., Yang, H., and So, J. (2020). An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition. *arXiv*.

[Angeline et al., 1994] Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1994). An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):54–65.

[Benavente, 1998] Benavente, R. (1998). The ar face database. Computer Vision Center (CVC) Technical Report.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, pages 1–9.

[Bergstra et al., 2012] Bergstra, J., Ca, J. B., and Ca, Y. B. (2012). Random search for hyper-parameter optimization yoshua bengio.

[BioTest, 2017] BioTest (2017). Result of algorithm biotest 1.3.8 on ficv-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=2787`. Online. Visited: 09/05/2022.

[Boslaugh, 2008] Boslaugh, S. (2008). *Statistics in a Nutshell: A Desktop Quick Reference*. O'Reilly Media, 2nd edition edition.

[Bruggemann et al., 2020] Bruggemann, D., Kanakis, M., Georgoulis, S., and van Gool, L. (2020). Automated search for resource-efficient branched multi-task networks. *British Machine Vision Conference*.

[Byerly et al., 2020] Byerly, A., Kalganova, T., and Dear, I. (2020). No Routing Needed between Capsules. *arXiv*, pages 1–13.

[Cai et al., 2019] Cai, H., Zhu, L., and Han, S. (2019). Proxylessnas: Direct neural architecture search on target task and hardware. *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[Caruana, 1997] Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1):41–75.

[Chen et al., 2018] Chen, L. C., Collins, M. D., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., and Shlens, J. (2018). Searching for efficient multi-scale architectures for dense image prediction. *Advances in Neural Information Processing Systems*, 2018-December(Nips):8699–8710.

[Chen et al., 2020] Chen, Y. C., Hsu, J. Y., Lee, C. K., and Lee, H. Y. (2020). DARTS-ASR: Differentiable architecture search for multilingual speech recognition and adaptation. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2020-Octob:1803–1807.

[Choromanska et al., 2015] Choromanska, A., Henaff, M., and Mathieu, M. (2015). The Loss Surfaces of Multilayer Networks. *arXiv*.

[Chrabaszcz et al., 2017] Chrabaszcz, P., Loshchilov, I., and Hutter, F. (2017). A downsampled variant of imagenet as an alternative to the cifar datasets. *CoRR*.

[Chu and Zhang, 2020] Chu, X. and Zhang, B. (2020). Noisy differentiable architecture search. *arXiv*.

[Domhan et al., 2015] Domhan, T., Springenberg, T., and Hutter, F. (2015). Speeding Up Automatic Hyperparameter Optimization of Deep Neural Netowrks by Extrapolation of Learning Curves. *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3460–3468.

[Dong et al., 2022] Dong, X., Liu, L., Musial, K., and Gabrys, B. (2022). Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:3634–3646.

[Dong and Yang, 2020] Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *CoRR*.

[Duan et al., 2021] Duan, Y., Chen, X., Xu, H., Chen, Z., Liang, X., Zhang, T., and Li, Z. (2021). Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. *CoRR*, abs/2105.11871.

[Elsken et al., 2019] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21.

[Fan et al., 2023] Fan, Y., Alon, D., Shen, J., Peng, D., Kumar, K., Long, Y., Wang, X., Iliopoulos, F., Juan, D.-C., and Vee, E. (2023). Layernas: Neural architecture search in polynomial complexity. *arXiv*.

[Ferrara et al., 2012] Ferrara, M., Franco, A., Maio, D., and Maltoni, D. (2012). Face image conformance to iso/icao standards in machine readable travel documents. *IEEE Transactions on Information Forensics and Security*, 7:1204–1213.

[Ferrara et al., 2022] Ferrara, M., Franco, A., Maio, D., and Maltoni, D. (2022). FVC-ongoing. benchmark area: Face image iso compliance verification. `https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/BenchmarkAreas/BenchmarkAreaFICV.aspx`. Visited: 14-11-2018.

[Floreano et al., 2008] Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

[Foret et al., 2020] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. *CoRR*.

[Gadelha et al., 2023] Gadelha, G., Gomes, H. M., and Batista, L. V. (2023). Neural architecture search in the context of deep multi-task learning. pages 684–691. SCITEPRESS.

[Gao et al., 2020] Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., and Liu, W. (2020). Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning.

[Gao et al., 2018] Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. (2018). NDDR-CNN: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. *arXiv*, pages 3205–3214.

[Gavrikov and Keuper, 2022] Gavrikov, P. and Keuper, J. (2022). CNN filter DB: An empirical investigation of trained convolutional filters. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Graesser and Keng, 2020] Graesser, L. and Keng, W. L. (2020). *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley, 1 edition.

[Gualberto et al., 2022] Gualberto, A., Gomes, H., and Batista, L. (2022). A collaborative deep multitask learning network for face image compliance to iso/iec 19794-5 standard. *Expert Systems with Applications*, 198.

[Guido, 2017] Guido, A. C. M. S. (2017). *Introduction to Machine Learning with Python*. O'Reilly, third edit edition.

[Guo et al., 2020] Guo, P., Lee, C.-Y., and Ulbricht, D. (2020). Learning to Branch for Multi-Task Learning. *arXiv*.

[Harris et al., 2020] Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., and Hare, J. (2020). Fmix: Enhancing mixed sample data augmentation. *CoRR*, abs/2002.12047.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity Mappings in Deep Residual Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9908 LNCS:630–645.

[Hirata and Takahashi, 2020] Hirata, D. and Takahashi, N. (2020). Ensemble learning in CNN augmented with fully connected subnetworks. *arXiv*.

[Hu et al., 2021] Hu, S., Xie, X., Liu, S., Cui, M., Geng, M., Liu, X., and Meng, H. (2021). Neural architecture search for LF-MMI trained time delay neural networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2021-June, pages 6758–6762. Institute of Electrical and Electronics Engineers Inc.

[Huang et al., 2018] Huang, S., Li, X., Cheng, Z. Q., Zhang, Z., and Hauptmann, A. (2018). Gnas: A greedy neural architecture search method for multi-attribute learning. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 2049–2057. Association for Computing Machinery, Inc.

[Hutter et al., 2020] Hutter, F., Lars, K., and Vanshchoren, J. (2020). *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

[ICAO, 2015] ICAO (2015). Doc 9303 - Machine Readable Travel Documents - Part 1: Introduction - 7th Edition.

[ICAO, 2022] ICAO (2022). ICAO. International Civil Aviation Organization. `https://www.icao.int/about-icao/Pages/default.aspx`. Visited: 16-11-2018.

[ISO, 2017] ISO (2017). ISO/IEC 19754-5 information technology — biometric data interchange formats — part 5: Face image data. `https://www.iso.org/standard/50867.html`. Visited: 17/11/2018.

[Istrate et al., 2018] Istrate, R., Scheidegger, F., Mariani, G., Nikolopoulos, D., Bekas, C., and Malossi, A. C. I. (2018). Tapas: Train-less accuracy predictor for architecture search. *CoRR*, abs/1806.00250.

[Jin et al., 2018] Jin, H., Song, Q., and Hu, X. (2018). Auto-Keras: Efficient Neural Architecture Search with Network Morphism. *arXiv.org*.

[Kaiser et al., 2017] Kaiser, Ł., Brain, G., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J. (2017). One Model To Rule Them All - Google. *arXiv*.

[Kandasamy et al., 2018] Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., and Xing, E. P. (2018). Neural Architecture Search with Bayesian Optimisation and Optimal Transport. *arXiv*.

[Karpathy et al., 2014] Karpathy, A., Joulin, A., and Fei-Fei, L. (2014). Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. *arXiv*.

[Kasiński et al., 2008] Kasiński, A., Schmidt, A., Kasinski, A., and Florek, A. (2008). The put face database. *Image Processing and Communications*, 13(3-4):59–64.

[Kokiopoulou et al., 2019] Kokiopoulou, E., Hauth, A., Sbaiz, L., Gesmundo, A., Bartok, G., and Berent, J. (2019). Fast task-aware architecture inference. *CoRR*, abs/1902.05781.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Technical report, MIT.

[Kumar et al., 2019] Kumar, A., Kaur, A., and Kumar, M. (2019). Face detection techniques: a review. *Artificial Intelligence Review*, 52(2):927–948.

[Leang et al., 2020] Leang, I., Sistu, G., Burger, F., Bursuc, A., and Yogamani, S. (2020). Dynamic Task Weighting Methods for Multi-task Networks in Autonomous Driving Systems. *Proceedings of 23rd International Conference on Intelligent Transportation Systems*, 1:1–8.

[LeCun et al., 1998] LeCun, Y., Cortes, C., and Burges, C. J. (1998). THE MNIST DATABASE of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. Online. Visited: 21/03/2022.

[Li et al., 2023] Li, G., Hoang, D., Bhardwaj, K., Lin, M., Wang, Z., and Marculescu, R. (2023). Zero-shot neural architecture search: Challenges, solutions, and opportunities.

[Li et al., 2022] Li, J., Sun, A., Han, J., and Li, C. (2022). A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

[Liashchynskyi and Liashchynskyi, 2019] Liashchynskyi, P. and Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for nas. *CoRR*, abs/1912.06059.

[Liu et al., 2018a] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-j., Fei-fei, L., Yuille, A., Huang, J., and Murphy, K. (2018a). Progressive Neural Architecture Search (ProgressiveNAS). *proceedings of the 15th European Conference on Computer Vision (ECCV 2018)*, pages 1–20.

[Liu et al., 2018b] Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. (2018b). Hierarchical Representations for Efficient Architecture Search. *ILCR 2018*, pages 1–13.

[Liu et al., 2018c] Liu, H., Simonyan, K., and Yang, Y. (2018c). Darts: Differentiable architecture search. *arXiv*, pages 1–13.

[Liu et al., 2023] Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G. G., and Tan, K. C. (2023). A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):550–570.

[Liu et al., 2015] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:3730–3738.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:3431–3440.

[Lu et al., 2017] Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. (2017). Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1131–1140.

[Lu et al., 2020] Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., and Boddeti, V. N. (2020). Neural architecture transfer. *CoRR*, abs/2005.05859.

[Maltoni et al., 2009] Maltoni, D., Maio, D., Jain, A. K., and Parbhakar, S. (2009). *Handbook of Fingerprint Recognition*. Springer, 2nd edition.

[Maziarz et al., 2019] Maziarz, K., Kokiopoulou, E., Gesmundo, A., Sbaiz, L., Bartok, G., and Berent, J. (2019). Flexible Multi-task Networks by Learning Parameter Allocation. *arXiv*.

[Misra et al., 2016] Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-Stitch Networks for Multi-task Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:3994–4003.

[Nekrasov et al., 2019] Nekrasov, V., Dharmasiri, T., Spek, A., Drummond, T., Shen, C., and Reid, I. (2019). Real-Time Joint Semantic Segmentation and Depth Estimation Using Asymmetric Annotations. *arXiv*.

[Peng et al., 2020] Peng, D., Dong, X., Real, E., Tan, M., Lu, Y., Bender, G., Liu, H., Kraft, A., Liang, C., and Le, Q. (2020). Pyglove: Symbolic programming for automated machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 96–108.

[Pham et al., 2018] Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. (2018). Efficient Neural Architecture Search via parameter Sharing. *35th International Conference on Machine Learning, ICML 2018*, 9:6522–6531.

[Phillips et al., 2005] Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. (2005). Overview of the face recognition grand challenge. In *CVPR 2005*, volume I, pages 947–954. IEEE Computer Society.

[Qin et al., 2023] Qin, Y., Wang, X., Zhang, Z., Chen, H., and Zhu, W. (2023). Multi-task graph neural architecture search with task-aware collaboration and curriculum. *Advances in Neural Information Processing Systems*.

[Real et al., 2019] Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. *Proceedings EAAI 2019*, pages 4780–4789.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*, pages 1–6.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv*, pages 1–8.

[Ruder, 2017] Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv*, pages 1–14.

[Sable, 2019] Sable, A. (2019). Overview of Neural Architecture Search. `https://blog.paperspace.com/overview-of-neural-architecture-search`. Visited: 10-11-2020.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

[Selvaraju et al., 2020] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359.

[Shen et al., 2023] Shen, X., Wang, Y., Lin, M., Huang, Y., Tang, H., Sun, X., and Wang, Y. (2023). Deepmad: Mathematical architecture design for deep convolutional neural network. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6163–6173, Los Alamitos, CA, USA. IEEE Computer Society.

[Shen et al., 2017] Shen, Y., Yun, H., Lipton, Z. C., Kronrod, Y., and Anandkumar, A. (2017). Deep active learning for named entity recognition. *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP 2017 at the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 252–256.

[Shi et al., 2023] Shi, H., Ren, S., Zhang, T., and Pan, S. J. (2023). Deep multitask learning with progressive parameter sharing. *International Conference on Computer Vision*.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, pages 1–14.

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127.

[Steppan, 2022] Steppan, J. (2022). `https://commons.wikimedia.org/w/index.php?curid=64810040`. Online. Visited: 22/03/2022.

[Sun et al., 2020] Sun, R., Li, D., Liang, S., Ding, T., and Srikant, R. (2020). The Global Landscape of Neural Networks: An Overview. *IEEE Signal Processing Magazine*, 37(5):95–108.

[Sun et al., 2019] Sun, X., Panda, R., and Feris, R. (2019). AdaShare: Learning what to share for efflcient deep multi-task learning. *arXiv*, pages 1–19.

[Sun et al., 2018] Sun, X., Wu, P., and Hoi, S. C. (2018). Face detection using deep learning: An improved faster RCNN approach. *Neurocomputing*, 299:42–50.

[Suteu and Guo, 2020] Suteu, M. and Guo, Y.-k. (2020). Regularizing Deep Multi-Task Networks Using Orthogonal Gradients. *arXiv*, pages 1–11.

[Szegedy et al., 2015] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:2818–2826.

[Tanveer et al., 2020] Tanveer, M. S., Khan, M. U. K., and Kyung, C.-M. (2020). Fine-tuning darts for image classification. *CoRR*, abs/2006.09042.

[Torralba et al., 2007] Torralba, A., Murphy, K. P., and Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29.

[Vandenhende et al., 2019] Vandenhende, S., Georgoulis, S., De Brabandere, B., and Van Gool, L. (2019). Branched Multi-Task Networks: Deciding What Layers To Share. *arXiv*, pages 1–19.

[Vandenhende et al., 2020] Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L. (2020). Multi-Task Learning for Dense Prediction Tasks: A Survey. *arXiv*, pages 1–20.

[Vsoft, 2017] Vsoft (2017). Result of algorithm biopass face 5.6 on ficv-1.0. `https://biolab.csr.unibo.it/FvcOnGoing/UI/Form/AlgResult.aspx?algId=6336`. [Online. Visited December-2020].

[Wang et al., 2023] Wang, S., Xie, T., Cheng, J., Zhang, X., and Liu, H. (2023). Mdl-nas: A joint multi-domain learning framework for vision transformer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20094–20104.

[Williams, 1992] Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256.

[Wistuba et al., 2019] Wistuba, M., Rawat, A., and Pedapati, T. (2019). A survey on neural architecture search. *CoRR*, abs/1905.01392.

[Wong et al., 2018] Wong, C., Houlsby, N., Lu, Y., and Gesmundo, A. (2018). Transfer learning with neural automl. *CoRR*, abs/1803.02780.

[Wu et al., 2022] Wu, L., Cui, P., Pei, J., and Zhao, L. (2022). *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv*, pages 1–6.

[Xie et al., 2019] Xie, S., Zheng, H., Liu, C., and Lin, L. (2019). Stochastic neural architecture search. *ICLR*, pages 1–17.

[Xu et al., 2023] Xu, Y., Yang, Y., and Zhang, L. (2023). Multi-task learning with knowledge distillation for dense prediction. In *2023 IEEE/CVF International Conference on*

Computer Vision (ICCV), pages 21493–21502, Los Alamitos, CA, USA. IEEE Computer Society.

[Yang et al., 2015] Yang, S., Luo, P., Loy, C. C., and Tang, X. (2015). From Facial Parts Responses to Face Detection: A Deep Learning Approach. *arXiv*, pages 3676–3684.

[Ying et al., 2019] Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., and Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. *CoRR*, abs/1902.09635.

[Yu et al., 2019] Yu, K., Sciuto, C., Jaggi, M., Musat, C., and Salzmann, M. (2019). Evaluating the search phase of neural architecture search. *CoRR*, abs/1902.08142.

[Yu et al., 2016] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2016). Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473.

[Zela et al., 2020] Zela, A., Siems, J., and Hutter, F. (2020). Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *CoRR*, abs/2001.10422.

[Zhang et al., 2022] Zhang, L., Liu, X., and Guan, H. (2022). A Tree-Structured Multi-Task Model Recommender. *arXiv*, pages 1–22.

[Zhang et al., 2023] Zhang, L., Liu, X., and Guan, H. (2023). A tree-structured multitask model architectures recommendation system. *IEEE Transactions on Neural Networks and Learning Systems*.

[Zhang and Yang, 2017] Zhang, Y. and Yang, Q. (2017). A Survey on Multi-Task Learning. *arXiv*.

[Zhong et al., 2018] Zhong, Z., Yan, J., Wu, W., Shao, J., and Liu, C. L. (2018). Practical Block-Wise Neural Network Architecture Generation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2423–2432.

[Zhong et al., 2017] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random erasing data augmentation. *CoRR*, abs/1708.04896.

[Zoph and Le, 2017] Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–16.

[Zoph et al., 2018] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8697–8710.

Appendix A

Complementary results

This appendix presents intermediary experimental data, plots, and results obtained during the experiments performed and used to corroborate the discussions presented previously in Chapters 4 and 5.

A.1 Grad-CAM

Figures A.1 and A.2 show the single false negative (FN) and false positive (FP) images, respectively, for the *Dark Glasses* requisite in the STL experiment. Observe that in both cases we may have labelling issues or a lack of objective thresholds on the requisite definition, once in the first case we have a person using glasses with transparent lenses that may not cover the eyes region; and in the other case, the ground truth label is positive, however the person is using dark tinted lenses glasses.

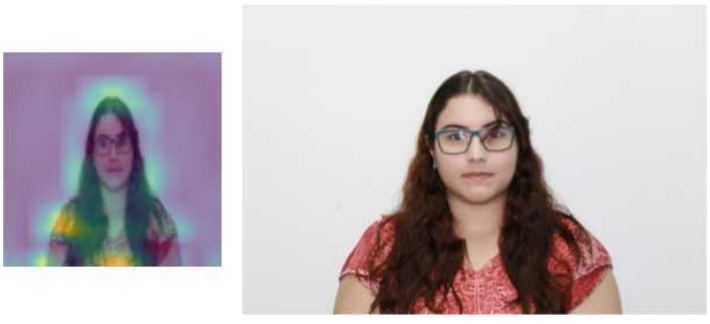


Figure A.1: Grad-CAM with the single false negative of STL network training of Dark Glasses requisite next to the original image on the right side.

Figure A.2: Grad-CAM with false positive of STL network training of Dark Glasses requisite next to the original image on the right side.

Following, Figures A.4 and A.3 show grids of images with the true negatives (TN) and true positives (TP), respectively.

Figure A.3: Grad-CAM with true positives of STL network training of Dark Glasses requisite.

A.2 Training Curves

Figure A.5 and A.6 show the training curves (accuracy vs epochs and loss vs epochs) for the MTL Handcrafted Experiment III and MTL Handcrafted Experiment II.

Figure A.4: Grad-CAM with true negatives of STL network training of Dark Glasses requisite.

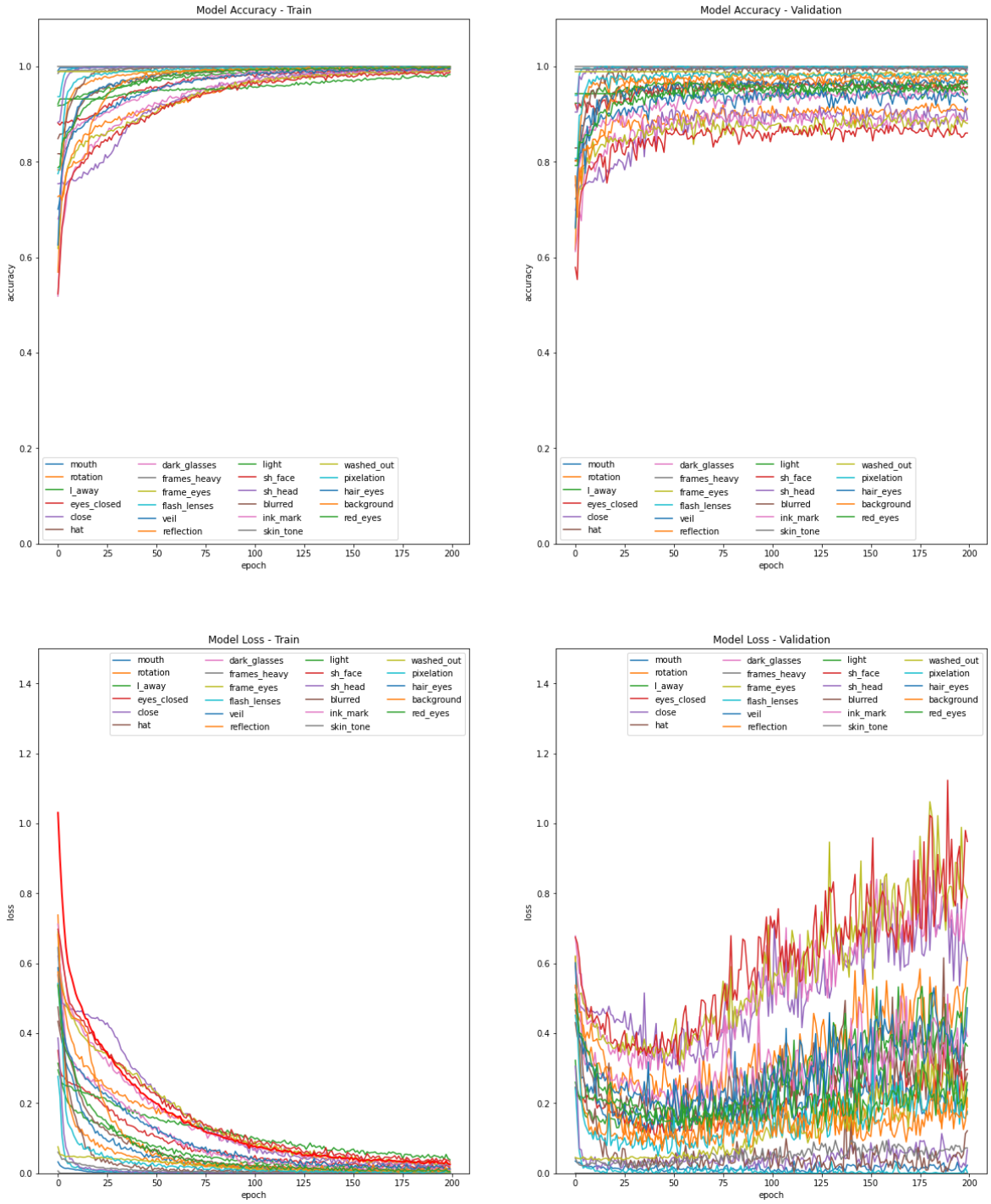


Figure A.5: Training curves of MTL Handcrafted 3 approach in experiment III with 200 epochs. The red line is the weighted value between all ICAO requisites. The metrics scores are presented in the range 0.0 to 1.0.

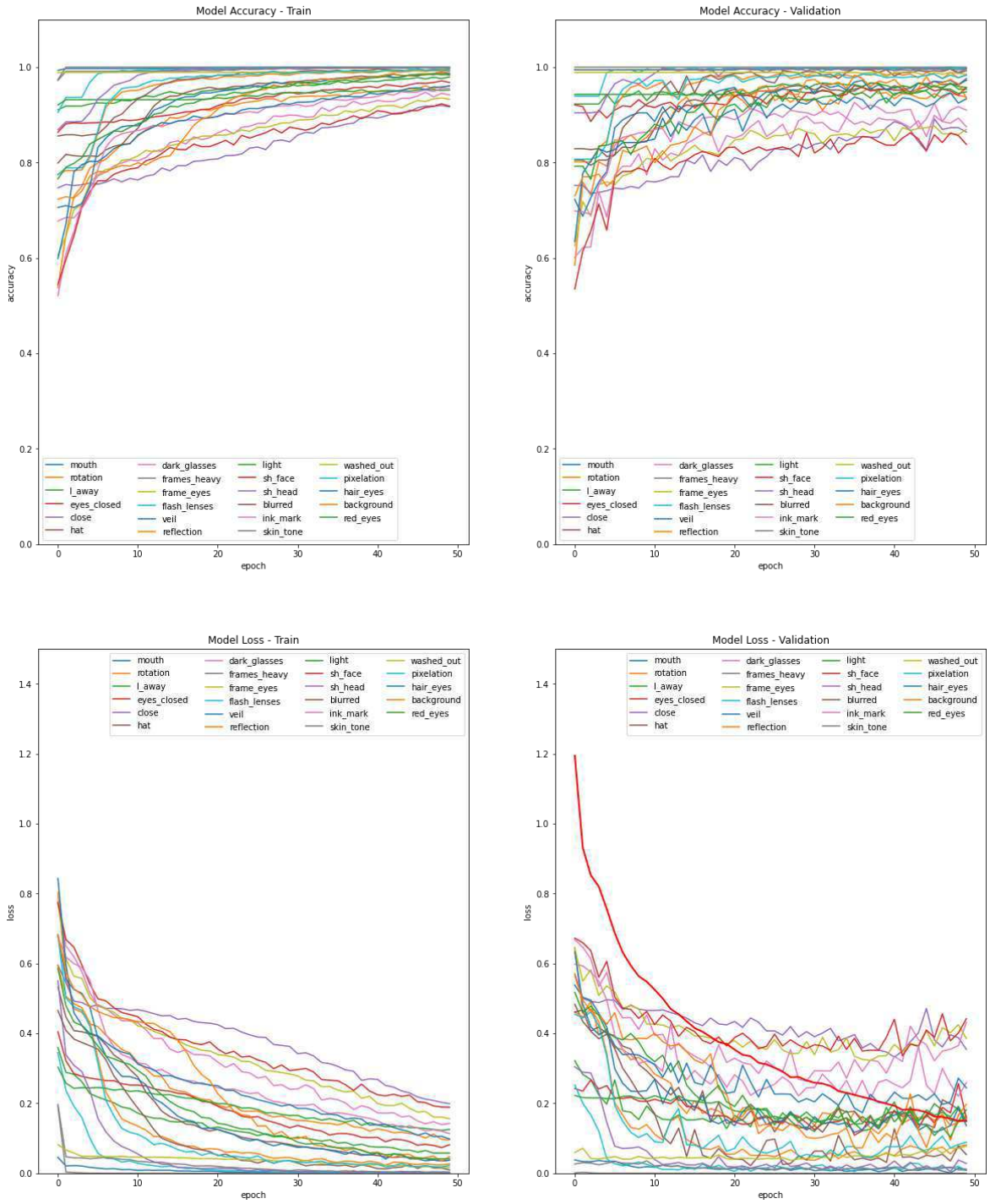


Figure A.6: Training curves of MTL Handcrafted 2 approach in experiment II with 50 epochs. The red line is the weighted value between all ICAO requisites. The metrics scores are presented in the range 0.0 to 1.0.