BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF RADIO ELECTRONICS

ÚSTAV RADIOELEKTRONIKY

OPTIMIZING NEURAL NETWORK ARCHITECTURE FOR EEG PROCESSING USING EVOLUTIONARY ALGORITHMS

OPTIMALIZACE ARCHITEKTURY NEURONOVÝCH SÍTÍ PRO ZPRACOVÁNÍ EEG POMOCÍ EVOLUČNÍCH ALGORITMŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. Kristýna Pijáčková

SUPERVISOR
VEDOUCÍ PRÁCE

doc. Ing. Tomáš Götthans, Ph.D.

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Studentka: Bc. Kristýna Pijáčková ***ID:*** 211534

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Optimalizace architektury neuronových sítí pro zpracování EEG pomocí evolučních algoritmů

POKYNY PRO VYPRACOVÁNÍ:

Proveďte literární rešerši týkající se evolučních algoritmů pro optimalizaci hyperparametrů neuronových sítí. Popište některé dosud používané nebo vyvíjené metody. Popište princip vzniku, základní charakteristiky, a metody měření EEG signálu. Popište jak neuronové sítě mohou zlepšit aktuálně používané metody zpracování EEG. Navrhněte algoritmus pro optimalizaci hyper parametrů neuronových sítí se zaměřením na zpracování EEG signálů.

Implementujte algoritmus v programovacím jazyce Python s využitím standardně používaných knihoven pro neuronové sítě (např. Pytorch). Aplikujte algoritmus na zadané EEG data epileptických pacientů. Porovnejte výsledky z reálných dat s teoretickými předpoklady. Proveďte diskusi získaných výsledků a zhodnoťte využitelnost metody.

DOPORUČENÁ LITERATURA:

[1] NEJEDLY, P., CIMBALNIK, ET. AL. Intracerebral EEG artifact identification using convolutional neural networks. Neuroinformatics, 17(2), 225–234. https://doi.org/10.1007/s12021-018-9397-6

[2] STANLEY, K. O., CLUNE, ET. AL. Designing neural networks through neuroevolution. Nature Machine Intelligence, 1(1), 24–35.
https://doi.org/10.1038/s42256-018-0006-z

[3] STANLEY, K. O., MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. Evolutionary Computation, 10(2), 99–127. https://doi.org/10.1162/106365602320169811

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

Vedoucí práce: doc. Ing. Tomáš Götthans, Ph.D.

doc. Ing. Lucie Hudcová, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

ABSTRACT

This thesis deals with an optimization of neural network hyperparameters for EEG signal processing using evolutionary algorithms. The incorporation of evolutionary optimization can reduce reliance on human intuition and empirical knowledge when designing neural network and can thus make the process design more effective. In this work, a genetic algorithm was proposed that is suitable for hyperparameters optimization as well as neural architecture search. These methods were compared to a benchmark model designed by an engineer with expertise in iEEG processing. Data used in this work are classified into four categories and come from St. Anne's University Hospital (SAUH) and Mayo Clinic (MAYO) and were recorded on drug-resistant epileptic patients undergoing pre-surgical examination. The results of the neural architecture search method were comparable with the benchmark model. The hyperparameter optimization improved the F1 score over the original, empirically designed, model from 0.9076 to 0.9673 for the SAUH data and 0.9222 to 0.9400 for the Mayo Clinic data. The increased scores were mainly due to the increased accuracy of the classification of pathological events and noise, which may have further positive implications in applications of this model in seizure and noise detectors.

KEYWORDS

Neural network, deep learning, optimization, evolutionary algorithms, genetic algorithm, neural architecture search, hyperparameter optimization, iEEG, epilepsy

ABSTRAKT

Tato práce se zabývá optimalizací hyperparametrů neuronových sítí pro zpracování EEG signálu pomocí evolučních algoritmů. Využití evolučních optimalizace může snížit závislost na lidské intuici a empirických znalostech při návrhu neuronové sítě a může tak zefektivnit návrh neuronové sítě. V této práci byl navržen genetický algoritmus, který je vhodný pro optimalizaci hyperparametrů i pro hledání neuronové architektury. Tyto metody byly porovnány s referenčním modelem navrženým inženýrem s expertýzou v této oblasti. Data použitá v této práci jsou rozdělena do čtyř kategorií a pocházejí z Fakultní nemocnice svaté Anny v Brně (SAUH) a Mayo kliniky (MAYO) a obsahují iEEG záznamy u pacienta s epilepsií rezistentní na léky, který podstupuje předoperační vyšetření. Metoda hledání neuronové architektury dosáhla výsledků srovnatelných s referenčním modelem. Optimalizovaný model zlepšil F1 skóre oproti originálnímu, empiricky navrženému modelu z 0.9076 na 0.9673 pro data z SAUH a 0.9222 na 0.9400 pro data z Mayo kliniky. Ke zvýšenému skóre přispěla hlavně zvýšená přesnost klasifikace patologických událostí a šumu, která může mít dále pozitivní vliv v aplikacích tohoto modelu v detektoru záchvatů a šumu.

KLÍČOVÁ SLOVA

Neuronová síť, deep learning, optimizace, evoluční algoritmy, genetický algoritmus, hledání neuronové architektury, optimalizace hyperparametrů, iEEG, epilepsie

Rozšířený abstrakt

Digitalizace ve zdravotnictví hraje v dnešním světe zásadní roli při rozvoji efektivnější a přístupnější zdravotní péče. Metody strojového a hloubkového učení se stávají čím dál tím víc důležitějším nástrojem pro bioinženýry při zpracovávání biomedicínských dat, jež můžou mít různou podobu od obrazových dat, přes signálové až po textové data. S ohledem na diverzitu dat a možnosti jejich zpracování může být výběr správného modelu hloubkového učení, případně jeho parametrů, které by dosáhly optimálních výsledků poměrně komplikovaný a často probíhá metodou pokus-omyl.

Tato diplomová práce se proto zabývá optimalizací modelů neuronových sítí pomocí evolučních algorithmů a to konkrétně pro zpracování EEG signálů. K optimalizaci byl v rámci této práce navržen genetický algoritmus, jež funguje na principu darwinovské evoluční teorie. Algoritmus tak prozkoumává daný prostor s hyperparametry modelu a snaží se najít optimální řešení problému tím, že vytváří jednotlivé generace jedinců, kteří následně na základě hodnoceného skóre mají šanci přenést své geny na další potomky v nové generaci a tím doiterovat do optimálního řešení. V rámci této metody tedy funguje výběr jedinců vhodných k dalšímu křížení, kdy vybraní jedinci předávají své geny (hledané parametry) dále a následně mutace, kdy část těchto parametrů může změnit svou hodnotu a tím zajistit dostatečnou diverzitu v generacích.

V rámci této práce byly prozkoumány dvě metody optimalizace modelu hloubkového učení. Ta první je optimalizace hyperparametrů daného modelu, jako jsou learning rate, batch size, velikost kernelů a počet filtrů v konvoluční vrstvě, počet vrstev GRU, atd... a zároveň optimalizace parametrů předzpracování vstupních dat, které byly z časové oblasti transformované pomocí Fourierovy nebo vlnkové transformace. Druhá metoda je neural architecture search, tedy hledání architektury neuronové sítě jako takové. V rámci této metody byly stanovené základní pravidla pro design modelu, který je tvořen N hlavními bloky, které v sobě obsahují až n možných operací, které můžou být buď konvoluční operace, maximal nebo average pooling, nebo identity vrstva.

Vzhledem k náročnosti celkového problému byl genetický algoritmus navržený tak, aby pracoval jak synchronně, tak asynchronně, tedy aby bylo možné počítat více jedinců ve stejný čas na více GPU procesorech. K implementaci byl použit jazyk Python a knihovna Pytorch pro modely hloubkového učení, výpočet pak probíhal až na 8 GPU procesorech a byl omezen buď časově, nebo množstvím jedinců.

Data použitá v této práci byla nasbírána ve Fakultní nemocnici u Sv. Anny v Brně a na Mayo klinice. Data pochází od pacientů, kteří trpí rezistentní epilepsií, tedy nereagují na léčbu antiepileptiky, a v rámci předoperačního vyšetření podstupo-

vali intrakraniálnímu zavedení EEG elektrod. Data se v datasetu vyskytují jako 3 sekundové úseky a jsou členěné do 4 skupin - šum ze sítě, šum, fyziologická aktivita a epileptická aktivita.

K porovnání výsledků byl použitý CNN-GRU model, který byl navržený inženýrem s expertýzou v oblasti zpracování EEG dat. Výsledky této práce ukazují, že genetický algoritmus v kombinaci s optimalizací hyperparametrů CNN-GRU modelu byl schopný zásadně vylepšit výsledky tohoto modelu na obou datasetech a to z 0.9076 na 0.9673 pro data z Fakultní nemocnice u Sv. Anny a 0.9222 na 0.9400 pro data z Mayo kliniky. Ke zvýšenému skóre přispěla hlavně zvýšená přesnost klasifikace patologických událostí a šumu, která může mít dále pozitivní vliv při aplikaci tohoto modelu v detektoru záchvatů a šumu. Metoda hledání architektury neuronové sítě pomocí genetického algoritmu sice žádné výrazné zlepšení skóre oproti tomu původnímu nepřinesla, přesto však svými výsledky byla schopná konkurovat výsledkům porovnávaného CNN-GRU modelu. Tento výsledek byla schopna dosáhnout bez předchozích znalostí tohoto problému a za použití základních operací a pravidel k jejímu návrhu. Zároveň není vylučitelné, že by tato metoda po důslednějším prozkoumávání hyperparametrického prostoru mohla dosáhnout lepších výsledků.

Author's Declaration

Author:	Bc. Kristyna Pijackova
Author's ID:	211534
Paper type:	Master's Thesis
Academic year:	2022/23
Topic:	Optimizing neural network architecture for EEG processing using evolutionary algorithms

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

This thesis was done in collaboration with the Computational Neuroscience group at the Institute of Scientific Instruments in Brno. I would thus like to thank Ing. Petr Klimes PhD. and MSc. Petr Nejedly for including me in the team and providing me with their knowledge and expertise. I am also thankful to my advisor Ing. Tomas Gotthans PhD. for his support throughout this year and my studies.

Contents

List of Figures

List of Tables

1 Introduction

Digitization in healthcare in today's world plays a vital role in the development of more effective and affordable healthcare services. Access to abundant biomedical data allows for further development of diagnostic methods, improved monitoring of patient's health conditions, and better treatment of abnormalities or diseases. Traditional data processing is based on data mining and feature extraction based on the engineers' domain knowledge. However, biomedical data come from heterogeneous sources which can include biomedical signals, images, textual data, laboratory reports, and many more. This access to various and complex data sources can pose a challenge both in time spent processing big data and the lack of sufficient domain knowledge [1].

Both machine learning (ML) and deep learning (DL) are nowadays essential tools for engineers working with biomedical data. While both techniques are powerful and are used to identify patterns and trends in the data that would otherwise remain undetected, they come with their own challenges. ML methods rely on extracted features from the data and thus do not solve the above-described problem. DL is, on the other hand, able to make sense even of the raw state of the data with an optimal neural network architecture.

The current state-of-the-art in neural network architecture design for electroencephalogram (EEG) signal processing is based on a process of trial-and-error, whereby different architectures are tried out and the one that gives the best results is selected. This process is time-consuming and can be expensive, especially when considering the number of different hyperparameters that need to be optimized. A more efficient approach would be to use optimization algorithms that would search for the hyperparameters or could even design the neural network architectures.

The topic of this thesis is, therefore, focused on optimizing neural network architectures, used for processing EEG signals with genetic algorithms. The proposed GA-based approach is tested on a classification problem with a real-world dataset from drug-resistant epileptic patients consisting of different EEG events. So far this thesis covers brief introductions to EEG signals, neural architecture design, and optimization algorithms. The second part of the thesis is followed by an introduction of the dataset and methodology for hyperparameter optimization with GA of a pre-existing neural architecture. The results show that the GA-based approach is able to find the optimal solution with a much higher accuracy, which is crucial in further application of the classifier in detectors of epileptic seizures or noise in the EEG data.

2 EEG Signals

2.1 Introduction

An electroencephalogram signal allows us to observe neural activities in the brain. Electrical activities in the brain are caused by an exchange of ions across neuron membranes. The transmission of information by a nerve, called an action potential, may be initiated by many types of stimuli such as chemical activity, light, electricity, touch, and pressure,...[2].

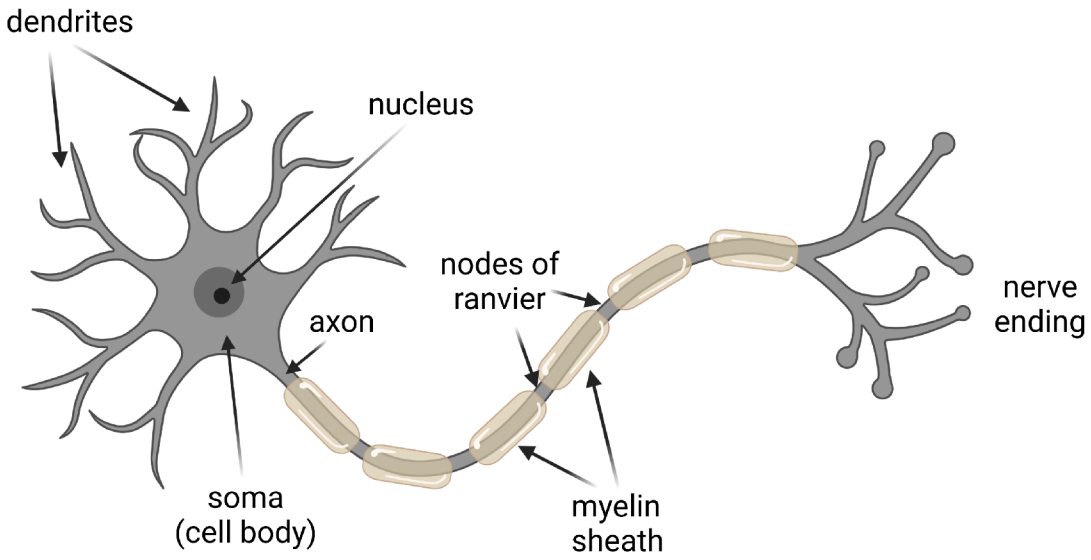


Fig. 2.1: Structure of a neuron*

Differences in electrical potentials are caused by summed postsynaptic graded potentials from pyramidal cells that create electrical dipoles between the soma (body of a neuron) and apical dendrites, which branch from neurons 2.1. The current in the brain is generated mostly by pumping the positive ions of sodium, Na+, potassium, K+, calcium, Ca++, and the negative ion of chlorine, Cl-, through the neuron membranes in the direction governed by the membrane potential and concentration gradient [3].

* Created with BioRenderer.com

The sodium pump in a nerve cell produces a gradient of potassium and sodium ions, which produce the action potential illustrated in Fig 2.2. On the inside, the excitable nerve cell is high in potassium and low in sodium. After the dendrites of the nerve cell receive the stimulus, Na+ channels open and enter the nerve cell. This depolarizes the membrane and produces a spike, that occurs once the interior potential rises from -70 mV up to -55 mV. When the threshold is reached, additional Na+ channels open and depolarize the interior of the cell membrane up to +30 mV. After that, potassium channels open and the membrane repolarizes back, usually overshooting the process to a potential level of -90 mV. This action prevents the neuron to react to a new stimulus triggered by another action potential and takes about 2 ms for the Na+/K+ pumps to reach the resting state of -70 mV [2].

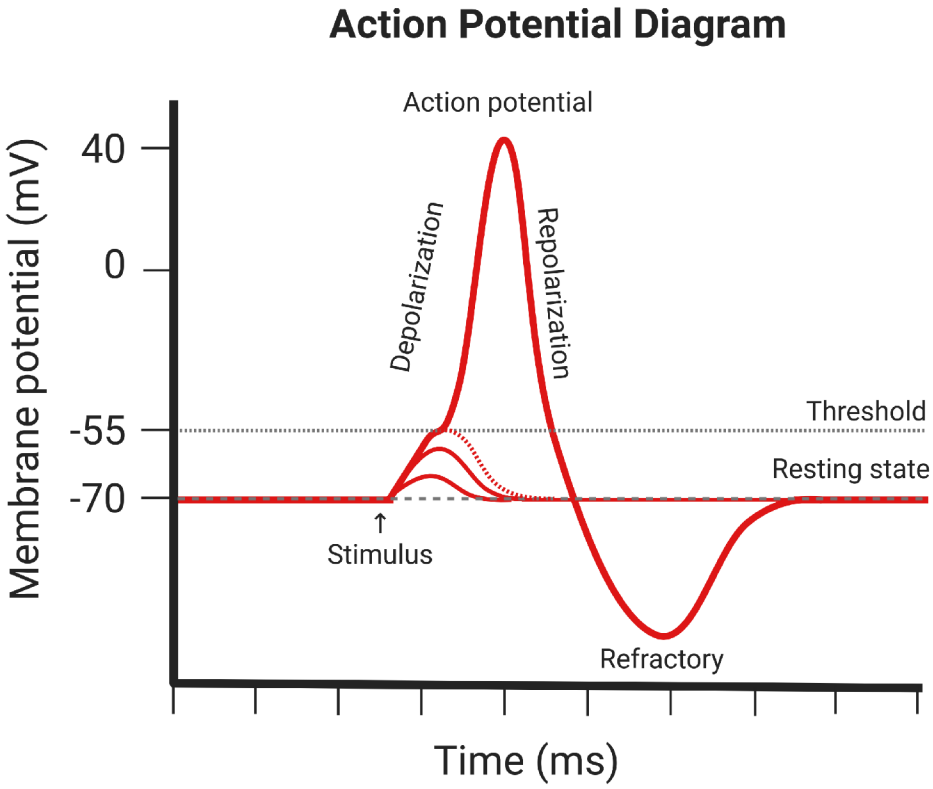


Fig. 2.2: Action potential in excitatory and inhibitory presynaptic activity (EPSP and IPSP) and postsynaptic activity*

The current generated through this process generates a magnetic field measurable by electromyogram (MEG) machines and a secondary electrical field over the scalp measurable by EEG systems [2]. The principle is illustrated in Fig 2.3.

* Created with BioRenderer.com

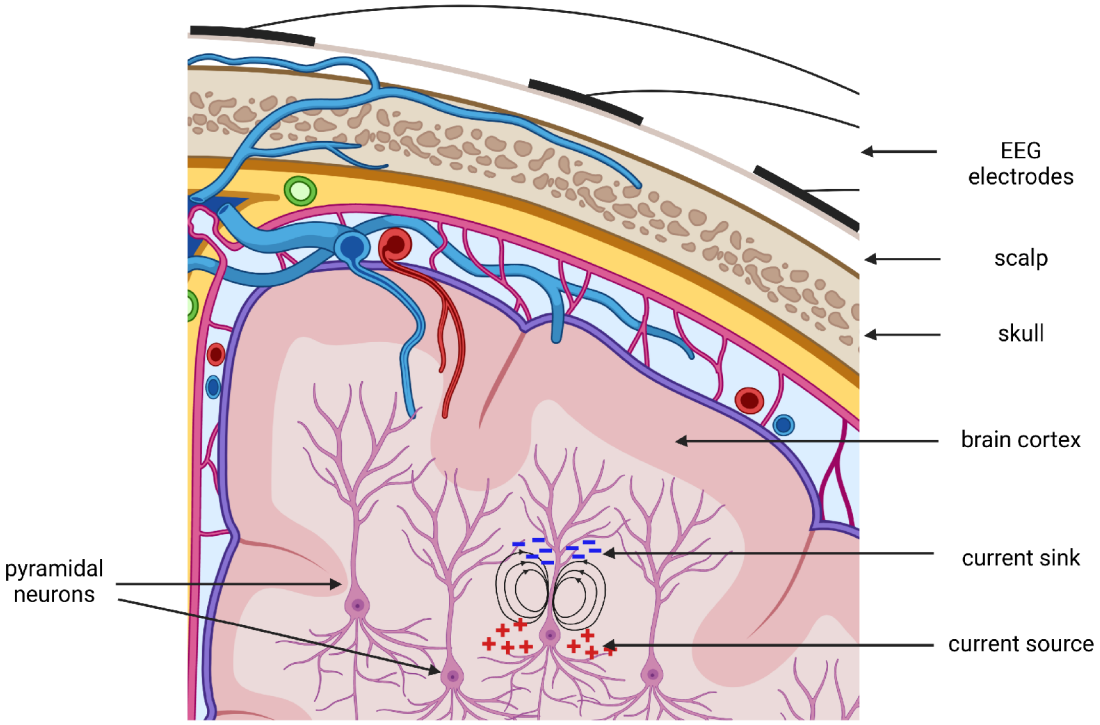


Fig. 2.3: Schematic demonstration of EEG measurement*

2.2 Signal Acquisition and Processing

The EEG signals are captured by multiple-electrode EEG acquisition systems. The EEG signal can be measured either from specific locations over the scalp, in this case, we talk about scalp EEG, or the measurement can be intracranial (iEEG), where electrodes are implanted into the skull. Scalp electrodes are the most common type and can be either disposable or reusable and applied on the scalp with or without gel.

The scalp EEG is easy to use and relatively easy to set up and is often used in clinical applications as well as in brain-computer interface applications. The non-invasive EEG can be also measured with headbands or pre-made electrode caps for easier usage. The other option for measuring the EEG signals is by implanting the needle electrodes into the skull (intracranial or iEEG) which is mainly used in epilepsy surgeries. These needle electrodes have various numbers of contacts, an example of invasive EEG monitoring can be seen in Fig. 2.4.

The signals are measured in the time domain and may be often difficult to understand and analyze. Furthermore, the EEG signals often contain artifacts, signals

* Created with BioRenderer.com

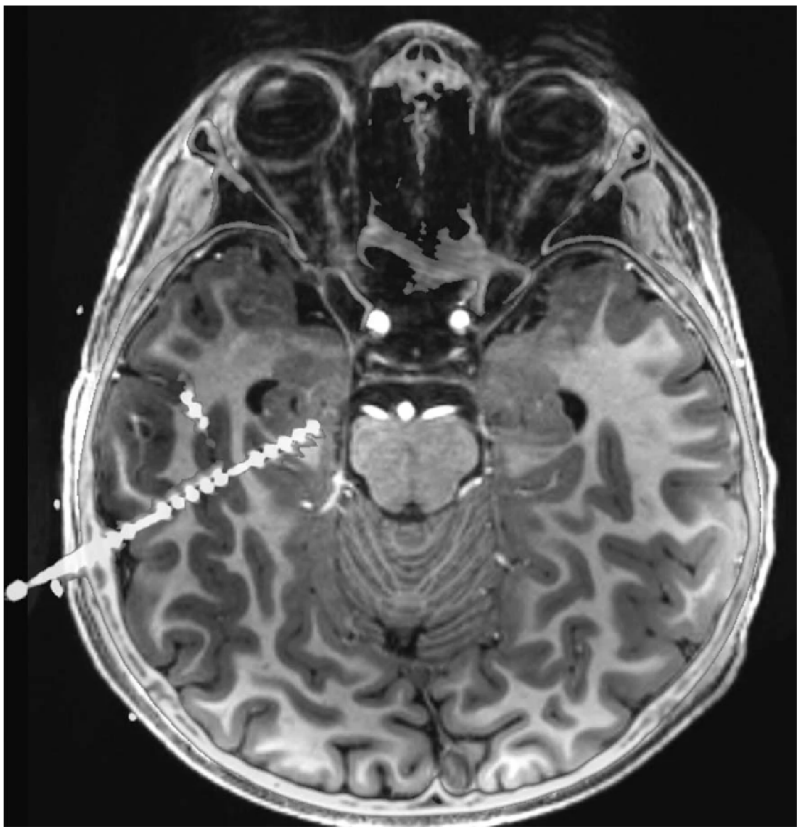


Fig. 2.4: A CT image of an invasive EEG monitoring with a stereotactic depth electrode. The post-operational CT image is co-registered with weighted pre-operative MRI with pixel intensities averaged to optimize concurrent visibility of CT electrodes and MRI tissue contrast (adapted from [5]).

that were not generated by the brain and are unwanted. The artifacts can be generated from different sources and come from external sources, acquisition instruments, or biological phenomena such as eye movement, eye blinks, head movement, muscle activity, or cardiac signals. Examples of some of the artifacts can be seen in 2.5 [6].

Therefore, signal processing tools are applied to the EEG signals for better separation and analysis and may differ based on the desired application. Signal processing in EEG originated in applications areas such as communication engineering, speech and music signal processing, and processing of other physiological signals such as ECG. Some of the commonly used processing includes filtering, denoising, transformation into a time-frequency domain by Fourier transform or wavelet transform, or independent component analysis. [2]

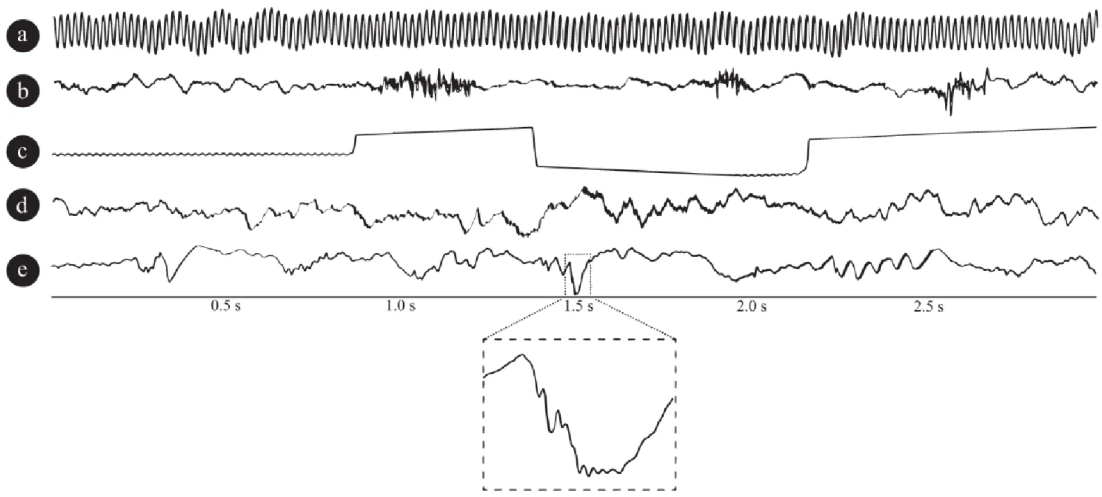


Fig. 2.5: Examples of EEG signals of different events - (a) powerline noise, (b) muscle artifact, (c) baseline jump artifact, (d) physiological signal, (e) epileptiform pathological signal with an HFO riding on spike (adapted from [5])

2.3 EEG Applications

Changes in the patterns of the EEG signals can be caused by various events. The patterns can change due to abnormalities of the central nervous system, various mental abnormalities, and drugs used for their intake. Patterns also vary with the shift of our attention and focus and much more. These patterns are often described in some EEG atlases (eg. [7, 8]) which may help with their identifications. The varying patterns allow investigation of the EEG signals for clinical problems such as monitoring cognitive engagement, production of biofeedback situation, anesthesia depth control, location of damages after head injuries, stroke, or tumor, or investigation and location of seizure origins in epilepsy.

Feature engineering and machine learning are commonly used when working with EEG, however, after successful application in other fields, deep learning finds its application to many EEG tasks as well. While machine learning needs proper feature extraction, deep learning can work with little preprocessing on the raw signals which may include filtering, denoising, and eventually some sort of transformation into a time-frequency domain. It is used to tackle various problems from detecting emotions and mental illnesses, classifying different stages of sleep, detecting anomalies in the EEG signal, and identifying seizures, or Alzheimer's disease to classifying motor imagery,... [9, 10, 11]. As the practical part of this thesis works with a dataset of epileptic events, the next section includes a brief background for better topic orientation.

2.4 iEEG in Epilepsy Diagnosis

Epilepsy is a neurological disorder characterized by unprovoked seizures, that affects around 50 to 60 million people around the world [12]. For roughly two-thirds of epileptic patients the number of seizures can be reduced with anti-seizure medications. However, around one-third of patients remain drug-resistant despite a wide array of available medications [13].

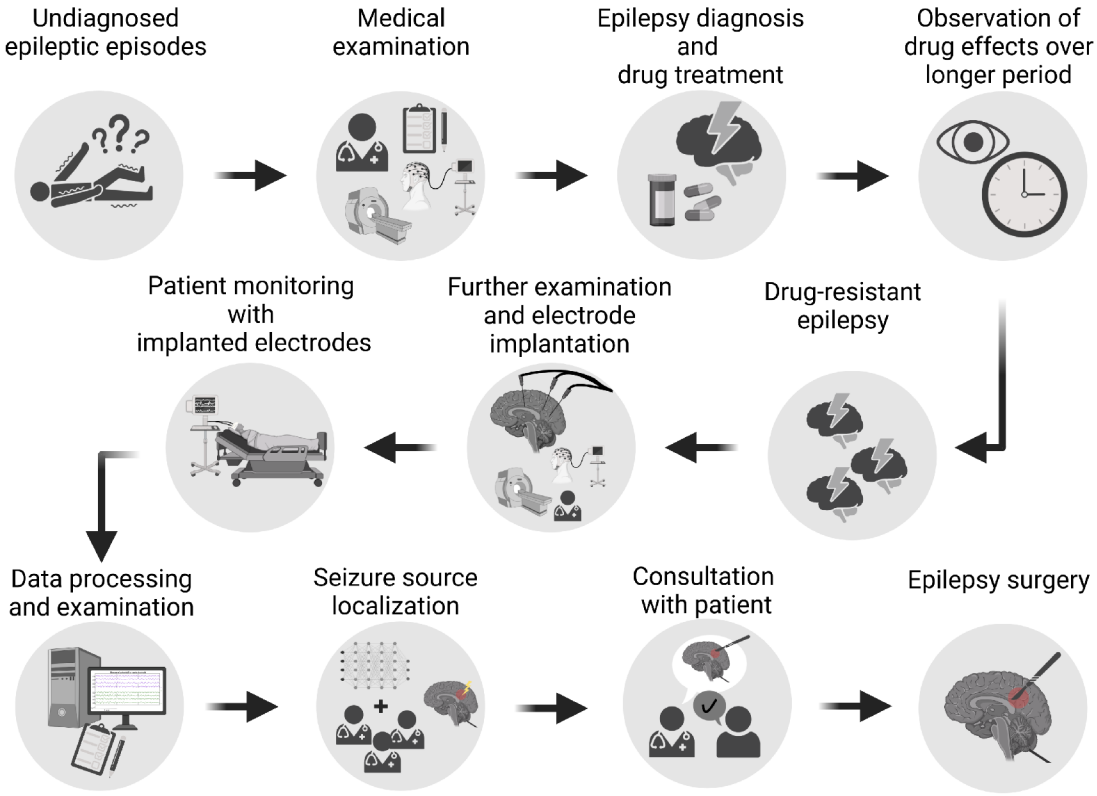


Fig. 2.6: Epilepsy - Schematic of procedures leading from the first epileptic episode to an epilepsy surgery*

Epileptic seizures in drug-resistant patients may be reduced after resection, a surgery, where a part of the brain with the epileptogenic zone responsible for the seizures is removed [14]. Before this surgery, the patients undergo numerous clinical procedures including intracranial EEG (iEEG) monitoring. This monitoring of the patient's brain activity helps to localize the pathological epileptic tissue in the brain and can last up to 4 weeks. The next step in the patient's treatment then depends

* Created with BioRenderer.com

on further investigation of the recorded signals. The iEEG recordings are manually investigated by electrophysiologist physicians for the localization of epileptic foci. This task, however, is very time-consuming and strongly dependent on the subjective experience of the physicians, and automation of the iEEG review process is thus highly aimed for. Machine learning and deep learning methods applied to this domain, see [11, 15, 16], can thus bring a lot of additional information for the doctors that can be used for better decisions about further medical procedures. The process of the first epileptic diagnosis up to an epileptic surgery is illustrated in Fig. 2.6.

3 Optimization

Optimization is an important aspect of problem-solving. It is a part of our everyday life and can be applied to almost everything from the optimization of our schedules or traveling routes to the optimization of economic systems, engineering applications, biological systems or healthcare, and many more [17].

In deep learning, optimization algorithms play an important role in the training of deep learning models, their efficiency, and their performance. Optimization algorithms such as stochastic gradient descent, Adagram, RMSProp, Adam, and others were developed to train deep learning models by continually updating model parameters and minimizing a loss function [18]. An optimization algorithm may be also applied for hyperparameter tuning of model hyperparameters and even for a neural architecture search. For this task algorithms such as grid or random search, Bayesian optimization, genetic and swarm algorithms, or neuroevolutionary strategies may be applied [19].

As the design and tuning of the neural network architectures is a topic of this thesis, this chapter includes a brief overview of possible tunable hyperparameters in a neural network as well as an overview of commonly used optimization algorithms.

3.1 Neural Network Architecture

A neural network is an algorithm that is used to model complex patterns in data. It composes of a large number of interconnected nodes, often called neurons, which are able to learn to recognize patterns of the input data. Figure 3.1 illustrates a basic structure of a neural network with an input layer, one hidden layer, and one output layer.

The layers within the neural network contain the network's knowledge and can be of many types, each suited for a different task. To name a few commonly used layers, dense layers usually process simple vector data and are often placed at the end of classification models. Recurrent layers, such as LSTM or GRU layers are good at processing sequence data as they also work with information from prior elements of the sequence. Convolutional layers are often used to extract specific features from images by applying learned filters to the input data. A pooling layer, applied after the convolutional layer reduces the spatial size of the data representations, whereas a Dropout layer can usually temporally "drop" nodes of the neural network to zero or a random value to support the generalization of the model and helps to avoid overfitting. Non-linearity is added to the network by including non-linear activation functions such as ReLU and its variations, Sigmoid, Tanh, Softmax, and many more...[18]

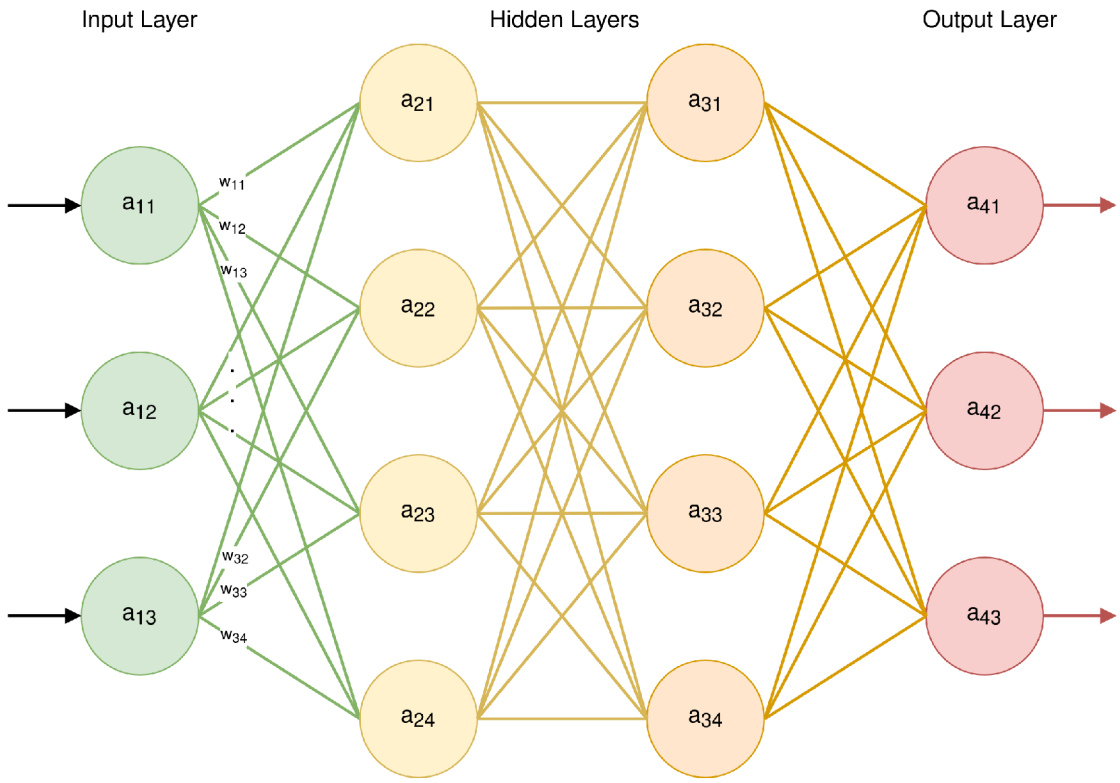


Fig. 3.1: Example of a simple neural network with 3 input features, two hidden layers, and 2 output nodes.

Final architectures are usually built empirically from these, and many more, layers. Moreover, the layers have their own set of parameters that also influence the behavior of the final model along with training hyperparameters such as cost function, learning rate, batch size, etc... The process of designing the final model can be thus a tedious task full of trial and error.

3.2 Optimization Algorithms

An application of an optimization algorithm can help with the designing process. We can see different approaches to the optimization process in the literature. The first option is tuning of hyperparameters of a given model as was done in [20, 21]. The second option would be to build an architecture of the model based on predefined building blocks consisting of defined layers. This take on the architecture optimization can be either restricted to some design limits as in [22, 23, 24], or the search space can be huge as was done in neural architecture search (NAS) in [25, 26]. Since the neural architecture search is usually extremely computationally expensive, researchers try to find ways to make the search more efficient by developing differ-

ent strategies, such as few or no shot NAS, mentioned in [27, 28]. These strategies can use some sort of predictors of the score, or they train all the networks as one supernetwork and use the shared weights as subparts. Finally, there is also a different take on designing and training neural networks with evolutionary algorithms, without gradient descent, called neuroevolution [29, 30, 31].

Grid Search

Grid search is a basic optimization algorithm that selects the best set of pre-chosen parameters based on the given task. The algorithm automates the process of creating all possible combinations of the given parameters from the limited search space. This algorithm leads to the most accurate predictions and is often used in machine learning for hyperparameter tuning. Since it does not depend on previous runs of the algorithm, the computation can be parallelized. However, as the algorithm does compute all possible combinations, the complexity of the computation increases exponentially and along with that the time needed for the search.

Random Search

Random search algorithm is an improvement on the grid search algorithm. The principle is based on a randomized hyperparameter search, where the parameters have a given value range. The algorithm then randomly selects the parameter values and continues the search process until a pre-determined computational budget is exhausted or a pre-defined accuracy was reached. While more effective than the grid search algorithm, the random search is still computationally expensive than other algorithms with guided search.

Bayesian Search

Bayesian search is based on a probabilistic approach for black-box function optimization. It aims at finding the global optimum with a minimal number of trials. It approximates the objective function with a surrogate function which is made of the predictions already evaluated hyperparameters that capture the unknown relationship between input and output. Evaluations of the black-box function train the surrogate function and an acquisition function guides the algorithm to an optimal solution. The acquisition function chooses the next set of parameters based on the surrogate function and once its evaluated adds it to the surrogate function for better approximation. These steps are repeated in several iterations, making the surrogate function more precise, unless the algorithm reaches the stopping condition.

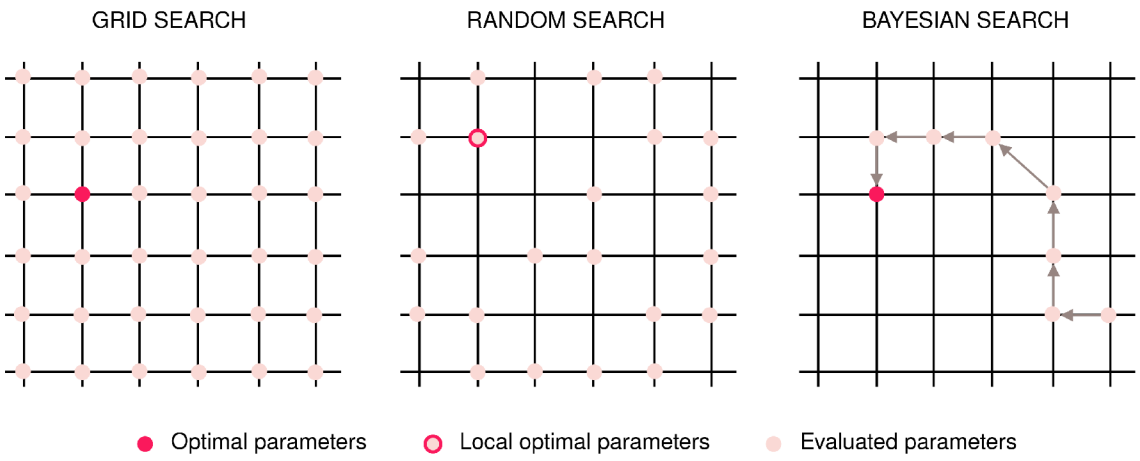


Fig. 3.2: Population of hyperparameter space (with two hyperparameters) for Grid Search, Random Search and Bayesian Search.

Genetic Algorithms

Genetic algorithm (GA), illustrated in Fig. 3.3 inspired by Darwin's evolution theory, is an optimization technique utilizing a population-based metaheuristic search algorithm, based on the survival of individuals with the fittest genes in the population. The general principle of the genetic algorithm is based on a selection of the fittest candidates, that either pass their genes onto their offspring or appear again in the new generation, and thus converge to an optimal solution.

The candidates of the GA that will become parents are chosen with a tournament selection, by being the most fit within a group of three randomly selected individuals from the generation. This way, the fitter candidates have a higher chance to become parents, but the less fit individuals are not excluded completely, which is important for diversity during the exploration phase. The individuals that won the tournament selection then become parents by forming pairs with others. This way they pass a combination of their genes onto an offspring, which will compete in the next generation. This above principle is illustrated in Fig. 3.3.

SWARM Algorithms

Swarm intelligence is based on the collective behavior of an organized group of entities. These entities may be inspired by nature such as the behavior of bees, ants, fireflies, etc... or artificial (particle swarm optimization). Such entities are able to form decentralized, self-organized organisms with swarm intelligence, even though the individuals on their own are less capable. The simulation of such behaviors leads to population-based optimization algorithms that can be used for both discrete and

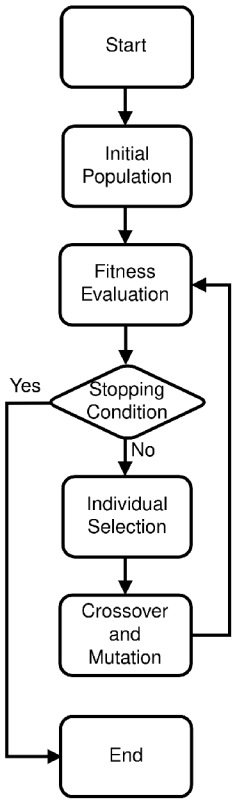


Fig. 3.3: Schematic illustration of a genetic algorithm

continuous search spaces. The search agents search through mathematical space with a set of rules defining their behavior and often depend on their personal best scores, the scores of the neighboring agents, and their past behaviors [32].

3.3 Neural Network Architecture Optimization

Manual Design

The design of neural networks and their parameters are often designed manually by engineers and experts working in the deep learning field. The design of the networks is often based on their empirical knowledge and trial-and-error process when designing the networks. For common applications of deep learning, such as computer vision or NLP (Natural Language Processing), one can use architectures designed by experts, which are often pre-trained and with the use of transfer learning easily fit the network to one's needs.

However, the problem begins when we are working with data, that does not completely fit to the common applications. The design of a proper neural network architecture fitted to the problem can become challenging as we need to analyze

what types of layers would be suitable for the given problem, how many layers to use, what hyperparameters to choose, etc... Thus, the manual fine-tuning, in this case, might take us a long time.

Hyperparameter Optimization

As mentioned in the previous subsection, fine-tuning hyperparameters by hand can be a tedious task to do. We can save a lot of time for ourselves by automizing the process with the use of some of the algorithms mentioned in 3.2.

We can then use either grid search or random search and look through either the whole search space or just random samples of it. Or we could use some of the more sophisticated algorithms such as Bayesian optimization, genetic algorithms, or swarm algorithms that are searching for the optimal hyperparameters in the search space and are trying to find the optimal solution by evolving and adjusting the hyperparameters in order to get to better solution next.

In this case, we want to pre-define what hyperparameters we would like to tune, eg. learning rate, number of layers, number of units/kernels/filters in the layers, types of activation functions, dropout rate, regularization techniques (dropout rate, L1, L2...), etc... And also to set some evaluation metric, such as validation accuracy, or validation loss (for the latter named algorithms).

Neural Architecture Search

Neural architecture search focuses on automating the design of neural architectures instead of engineers. This process is done iteratively evaluating and comparing various architectures based on a given criterion such as validation loss or validation accuracy. The NAS approach uses very often reinforcement learning or evolutionary algorithms [**?**, **?**] as optimization and search algorithms. During the search, new architectures are created and modified based on pre-defined operations, such as adding or removing new layers, changing sizes or types of layers, changing connections between layers, changing activation functions, etc...

The approach to NAS can vary, it can be more macro-level based and focus on the overall topology of the network and try out different variations based on some pre-defined building blocks, or search for those building blocks. Or the search can be more micro-level based and deal with the type and size of individual layers, activation functions, etc... Researchers also try to come up with techniques to save computational expenses by introducing few-shot or zero-shot NAS, where they pre-define evaluation functions, that either tries to find the optimal architecture by sampling and computing just a few of the architectures, or a global super-architecture and

sampling smaller architectures out of it or one, that can decide about the suitability of the architecture completely without training.

Neuroevolution

Neuroevolution [?] uses genetic algorithms to evolve whole neural network architecture. It uses evolutionary principles such as mutation, selection, and crossover. The topology of the networks is encoded in the genotype, which usually takes into account connections between different nodes and their weights. The topologies of different individuals are competing with each other and create offspring based on their fitness in the same way as in classical genetic algorithms.

4 Dataset and Data Preprocessing

4.1 Dataset

For the purposes of this thesis, two publicly available datasets with iEEG signals [5] were used. The data included in those datasets were collected at the St. Anne's University Hospital (SAUH) and the Mayo Clinic. The intracranial recordings come from patients with drug-resistant epilepsy that were undergoing pre-surgical invasive EEG monitoring. The patients were implanted with a custom acquisition system M&I; Brainscope, Czech Republic (SAUH) and a Neuralynx Cheetah system (Neuralynx Inc., Bozeman MT, USA) acquisition system (Mayo) with either depth electrodes, grids and strips, or a combination of both. The data from the SAUH were sampled with a frequency of 5 kHz and were collected from 14 patients during awake interictal resting state. The Mayo Clinic data were collected from 25 patients during the first night after electrode implantation and have a sampling frequency of 5 kHz as well.

iEEG Event	Samples SAUH	Samples MAYO
Powerline Artefacts	13,489	41,922
Noise Artefacts	32,599	41,303
Pathological Epileptic Activity	52,470	15,227
Physiological Activity	94,560	56,730
Total	193,118	155,182

Tab. 4.1: Overview of data distribution for the 3-second segment

The annotation of the iEEG recordings was done manually by three independent operators and sorted into distinctive events and following to that segmented into 3-second segments. It is based on a visual inspection in the time domain alongside power distribution matrices from manual detection for manual artifact detection [35]. The dataset contains 4 iEEG events - powerline artefacts, noise artefacts, pathological epileptic activity, and physiological activity. The overview of the number of the occurring events in the dataset can be seen in Table 4.1. The software used for the annotation was SignalPlant [33, 34], which is a free software tool for signal processing, inspection, and annotation. More information on the dataset can be found in [5].

The data segments from SAUH were randomly split into training (60% of each class) and validation (20% of each class) and testing (20% of each class) data. The validation data were used for model architecture ranking during the optimization

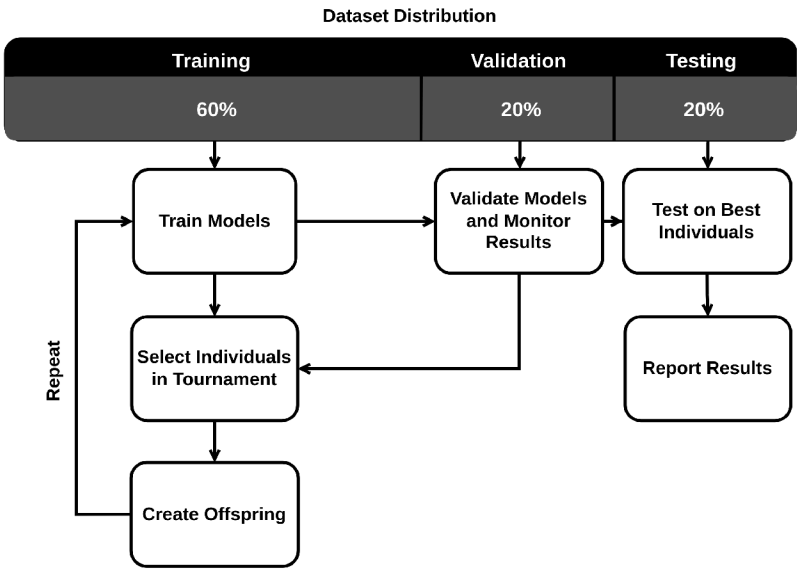


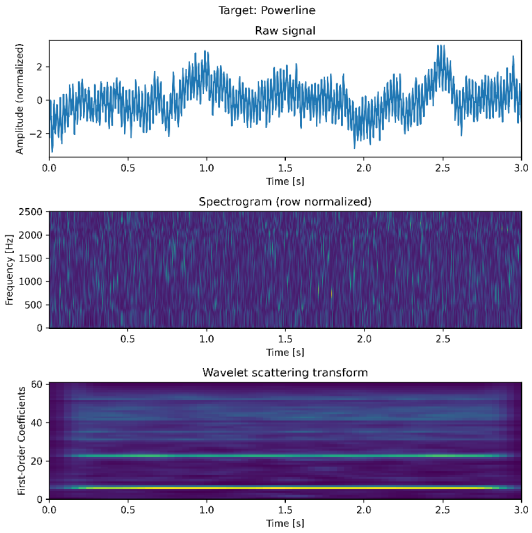
Fig. 4.1: Principle of optimization of the neural network with genetic algorithm

process. The test set was only used to obtain final scores (preventing data overfitting), as shown in Fig. 4.1. The data from SAUH were used for model optimization, development, and evaluation, while data from Mayo Clinic were solely used for final method evaluation. The proposed data split allows us to show that the developed model architecture generalizes to different institutions, which was not used for model development.

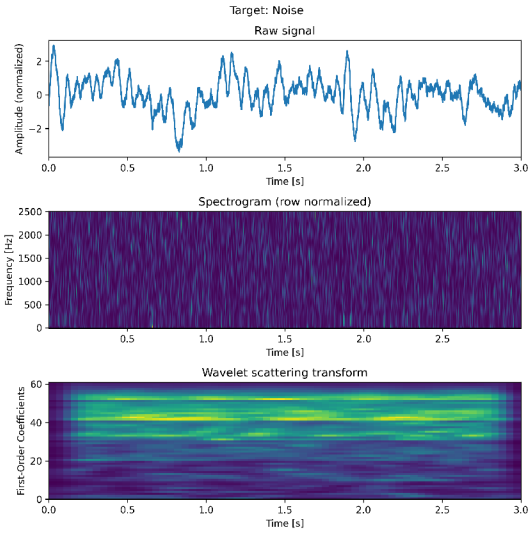
4.2 Data Preprocessing

The signal segments were converted into a time-frequency domain, either by computing the spectrogram by the Short-Time Fourier Transform (STFT) of the signal or by extracting first-order scattering coefficients from a wavelet scattering transform [36]. Before the signal representation is passed to the classification model, we normalize it by calculating the z-scored for each frequency band.

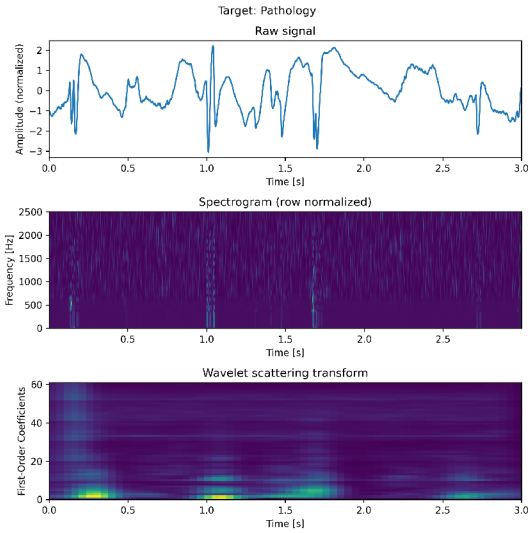
For the spectrogram computation, the SciPy library [37] was used, where the window type, length of each segment, number of overlapping segments, and length of zero-padded FFT are parameters that we want to optimize. The computation of the wavelet scattering transform, from which we extract the first-order coefficients as our model input, is done through the Kymatio library [38]. The coefficients can be changed by varying parameters J and Q, which specify an average scale as a power of two and the number of wavelets per octave respectively.



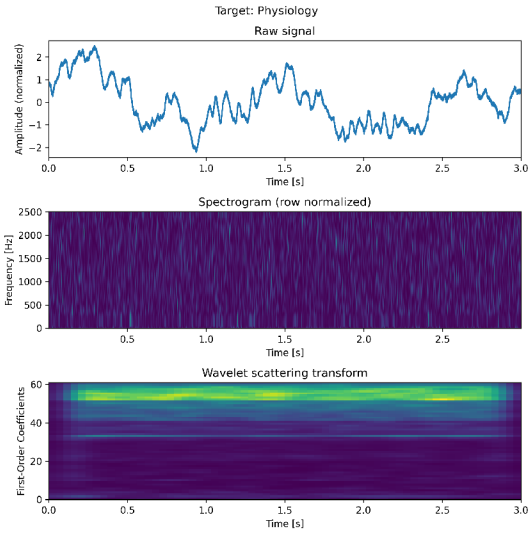
(a) Powerline Class



(b) Noise Class



(c) Pathological Class



(d) Physiology Class

Fig. 4.2: Demonstration of the preprocessing (STFT and WST) applied on the raw signal for four different events in the dataset.

5 Methodology

5.1 Genetic Algorithm

The genetic algorithm designed and used in this work was designed to work both as a synchronous and asynchronous algorithm and thus allows to distribute computing across multiple machines. In this work, mainly the asynchronous GA was used to optimize the neural network architecture for intracranial EEG processing. This approach allows for faster optimization, as more models can be trained at the same time on multiple GPUs. The fitness of each neural network architecture was measured with a kappa score, reflecting the model's classification performance compared with the expert gold standard while accounting for random chance.

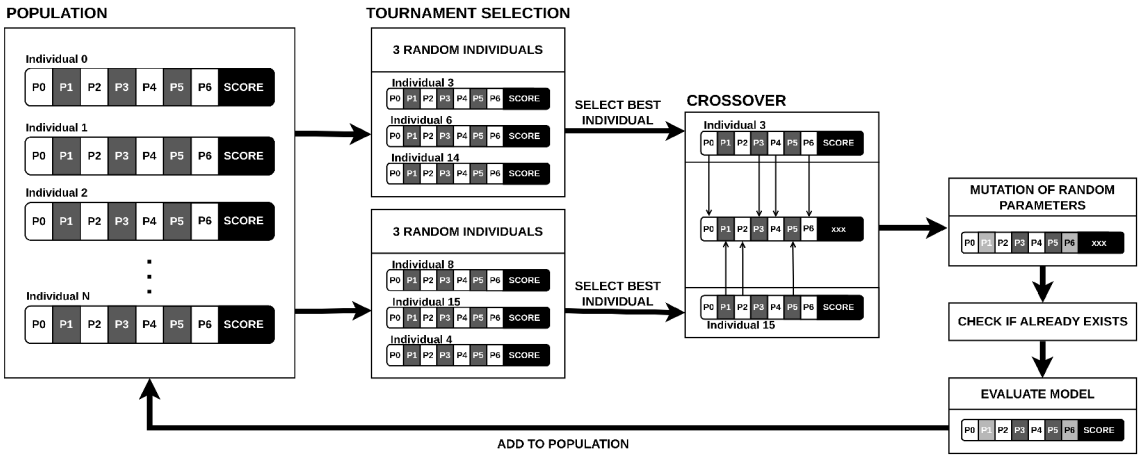


Fig. 5.1: Principle of implemented genetic algorithm

The basic principle of the algorithm is pictured in Fig. 5.1. Here, a new individual is created based on a crossover operation of two individuals that were evaluated in previous steps and were chosen to be the parent candidates based on tournament selection. The crossover operation uses uniform crossover [39], which means that the operation randomly samples the genes of both parents to create two new offspring *. The first offspring would inherit randomly selected genes from parent A and the rest of the genes would be inherited from parent B, just as is shown in 5.1. Moreover, there is also a small probability that one or more genes of the new offspring will be changed for another random valid value during a mutation. Should a genotype that was already evaluated in earlier generations appear, it is removed from the new

*Note that only one is pictured in Fig. 5.1 for simplicity. The second one would have complementary genes to that one, and the rest of the following process would remain the same.

generation and the algorithm gets a chance to replace it with a randomly generated genotype.

As mentioned, the proposed algorithm works with the synchronous and asynchronous setups. The synchronous setup traditionally creates a set of populations, computes their scores, and uses them to create new offspring to create new generation and repeat those steps, until a pre-set stopping condition is achieved. In the asynchronous setup, the candidate pool does consist of all the individuals scored by far, or of X last ones, as the population is not created in generations anymore, but as is needed when a computing machine gets free. This work uses to work with 20 last individuals enriched by 3 best ones of the whole set. This way the genetic algorithm tends to prefer the exploitation a bit more than the synchronous one. The asynchronous parallel implementation setup includes a main server that guides the optimization process and automatically assigns optimization tasks to available workers (docker images each with a single GPU). This principle is pictured in Fig 5.2

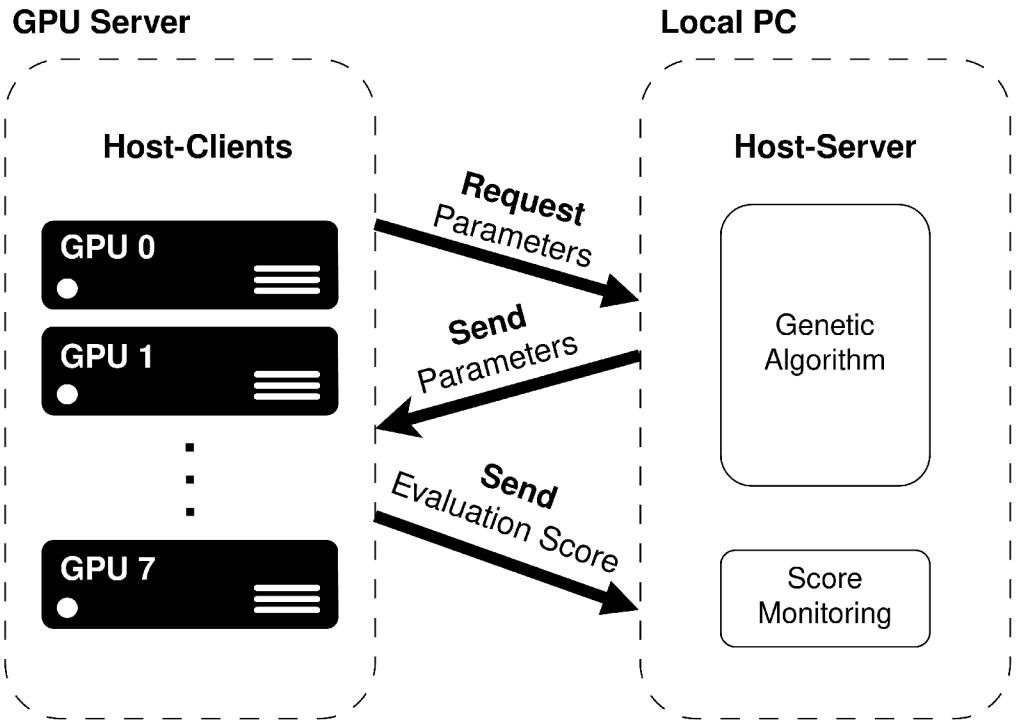


Fig. 5.2: Principle of GA distribution on multiple GPUs

5.2 Hyperparameters Optimization

The optimized parameters in this study can be classified into two groups. The first group consists of hyperparameters of the classification model i.e. size of convolutional filters, a number of convolutional filters, a learning rate, etc... The second group of hyperparameters optimizes iEEG signal preprocessing, i.e., hyperparameters (window type, FFT length, overlap, etc.) of short-time Fourier transform (STFT) and of the wavelet scattering transform (WST). The overview of the parameters can be found in Table 5.1. The parameter optimization in this work was done on a CNN-GRU architecture[†] presented in [16]. While the results achieved in the original study were remarkable, the model was designed empirically. Thus in this following study, the aim is on improving the model performance, by automatically optimizing the parameters of the given model. The model was implemented with a Pytorch library [40] while trained and optimized on a GPU server with 2x Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz with 1.41TB RAM memory and up to 8 available NVIDIA Quadro RTX 5000 GPUs.

The model itself was designed to classify different iEEG events based on a spectrogram input. The architecture of the model consists of a convolutional layer with a ReLU activation function that extracts spatial features from the spectrogram. The output of the convolutional layer is normalized with a batch normalization layer and after reshaping is passed into a GRU layer, followed by a fully connected layer with a Softmax activation function. During the training, the model uses an Adam optimizer and cross-entropy loss. The architecture of the model, with the highlighted features, that are considered for optimization can be seen in Fig. 5.3.

The optimization process was split into two parts, exploration and exploitation-focused GA. During the exploration part, the GA with an initial population of 20 searches for solutions from different parts of the search space by introducing a higher mutation rate of 0.1 and by creating a new generation with 20% of new, random individuals and 80% of offspring with inherited genotypes. To further limit the search space the influence of the model hyperparameters with constant preprocessing hyperparameters was observed. This process was done also with the signal preprocessing parameters while keeping the model architecture constant. The process was set up to run up to 30 generations.

After the exploration phase, there were around 1000 solutions available (input data optimization and model optimization) and the search continued with both parameter groups joined together. The initial population of size 20 was randomly initialized by solutions from the exploration part and combinations of the three best solutions for the model hyperparameters and the two best signal-processing param-

[†] `https://github.com/xnejed07/NoiseDetectionCNN-GRU`

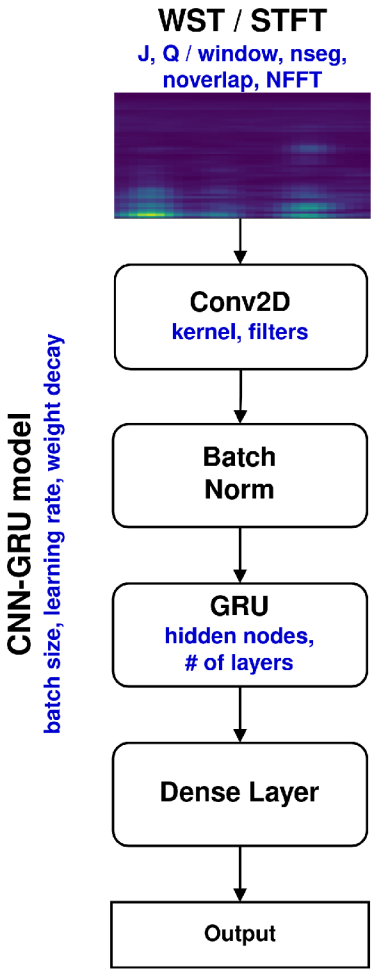


Fig. 5.3: CNN-GRU model with highlighted optimizable parameters in blue

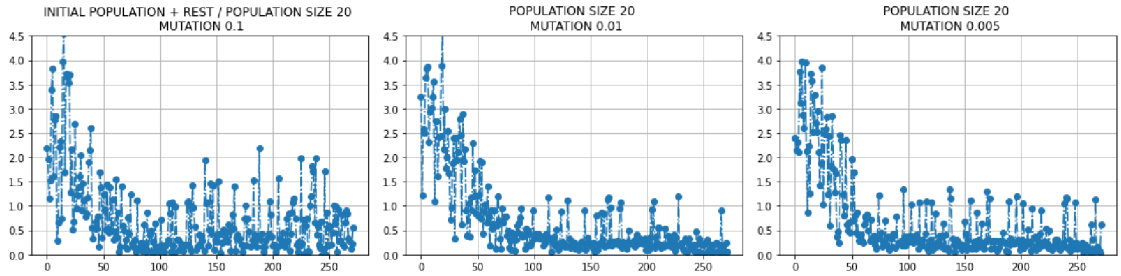
eters. This allowed us to speed up the learning process and use already explored parameters from the exploration part, while still allowing it to explore the search space. Once all of the initial genotypes have been evaluated, new ones are created when a GPU is available for the computation. To support a faster convergence of the algorithms the mutation rate was decreased to 0.05 and the 3 so far best-performing individuals to the pool of tournament candidates to improve elitism. The parameter search stops after the pool is filled with 150 evaluated individuals. The size of the population, mutation rate, and size of the global pool were chosen based on test runs of the algorithm on a simple testing function simulating our problem:

$$f(x) = \frac{(x_0 - x_p)^2}{x_{p_{max}}},$$

where x_0 is the optimal parameter, x_p is a current parameter and $x_{p_{max}}$ is the highest value from the range of parameters. An example of the influence of the mutation rate on the results with the test function can be seen in Fig.5.4.

Encoded Parameter	Benchmark Model Value	GA Value Range
Neural Network Architecture Parameters		
Number of Filters	256	{64, 128, 256, 512, 1024}
Kernel Size	3	{3; 5; 7}
Number of Hidden Nodes	128	{64, 128, 256, 512}
Number of GRU Layers	1	{1, 2, 3}
Batch Size	64	{64, 128, 256}
Learning Rate	1e-3	{1e-3; 5e-4; 1e-4}
Weight Decay	1e-4	{0; 1e-4; 1e-6}
Spectrogram Parameters (STFT)		
Window Function	Tukey	Tukey, Hann, Barlett, Flattop
Number of Segments	256	{32, 64, 128, 256, 512}
Overlapping Segments	128	Number of Segments // {2, 4, 8, 16, 32, 64}
NFFT	1024	{32, 64, 128, 256, 512, 1024}
Wavelet Scattering Parameters (WST)		
J	-	{3, 4, 5, 6, 7, 8, 9, 10}
Q	-	{2, 3, 4, 5, 6, 7, 8, 9, 10}

Tab. 5.1: Overview of optimizable parameters and their nominal value in the benchmark model



(a) Mutation rate 0.1 (b) Mutation rate 0.01 (c) Mutation rate 0.005

Fig. 5.4: Convergence of the genetic algorithm with test function with different mutation rates

5.3 Neural Architecture Search

Neural network architecture search was chosen as a second approach for architecture optimization. The aim here was to see if a genetic algorithm with preset rules for architecture build would be able to create architecture with similar or better scores than those achieved by a model from an expert engineer.

The NAS approach comes with a lot of challenges that need to be taken care of for the architecture to be valid. The dimensions of layers need to match and be compatible, the optimization problem cannot be too large, otherwise, we would have a small chance of achieving some good scores within a reasonable time, and lastly,

the architecture of the network must be encoded. Encoding of the architecture needs to be done in a way, that allows for a crossover of two completely different architectures, which still results in a valid architecture and can be also mutated.

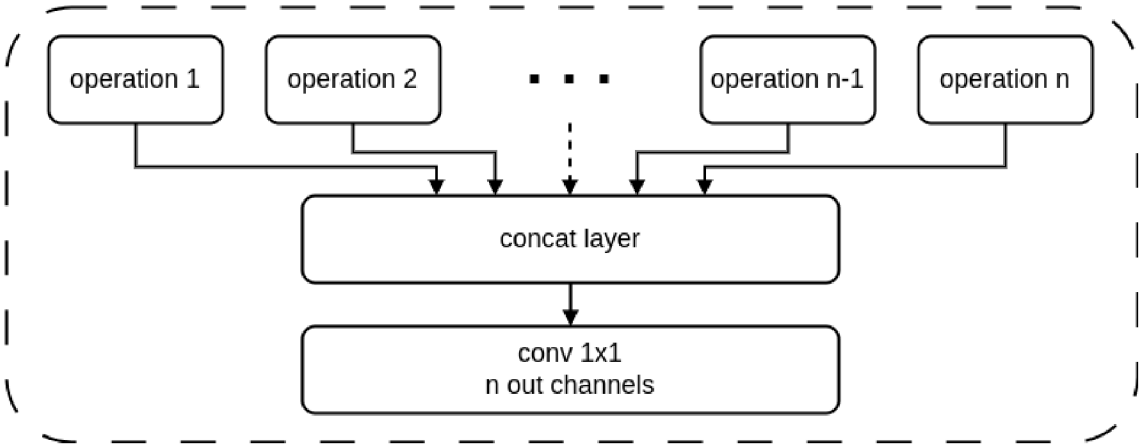


Fig. 5.5: Example of the inception-like block that is used in the architecture as main building stone

The inspiration for the NAS approach described below comes from NAS papers that used smaller building blocks such as convolutional block with 3x3 or 1x1 kernels and some average or maximum pooling operations, as well as inception blocks and the architecture building there [25, 41].

The architecture is built from main building blocks, pictured in Fig. 5.5. This block is created from n operation blocks, which can be either maximum pooling, average pooling, convolutional operation with (k x k) kernels, or identity operation, in which the output is the same as the input. The combination of those blocks is then concatenated in the following layer and is followed by a 1x1 convolutional operation with N output channels. This layer ensures for dimension matching between multiple main blocks stacked onto each other in the architecture. The last main building block is followed by an adaptive average pooling layer and a linear layer, that takes care of the classification, which can be seen in Fig. 5.6.

The Fig. 5.6 also highlights optimizable hyperparameters of the neural network. Before the start of the parameter search with the genetic algorithm, the maximal number of main building blocks N is defined, as well as the number of operation blocks in each of them and their parameters, such as available kernel sizes. Another parameter is m input channels from the first 1x1 convolutional layer, which upsizes the input signal in the channel dimensions to ensure an error-less run. The parameter n out channels is the same for all the main building blocks and ensures for correct dimension matching between them. Once the genetic algorithm is started, a preset

number of architectures is created and evaluated. From there, the principle is the same as with the hyperparameter optimization, as they both use the same genetic algorithm, which is described above in Subsection 5.1.

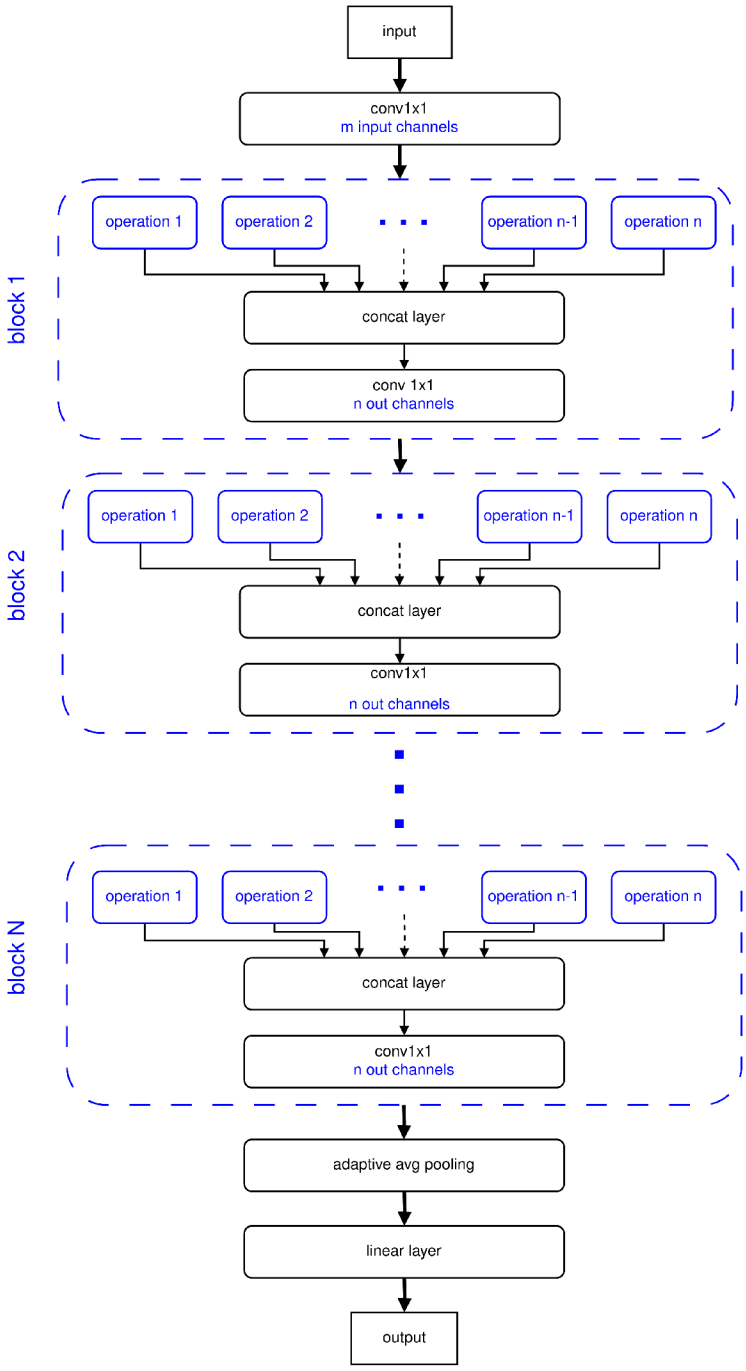


Fig. 5.6: Example of architecture build the random generator from the NAS implemented method. Blue color highlights all optimizable parameters - number of m input channels, number of n output channels in each block, N number of blocks, M number of building blocks (conv nxn, avg pool, ...) in each of the blocks.

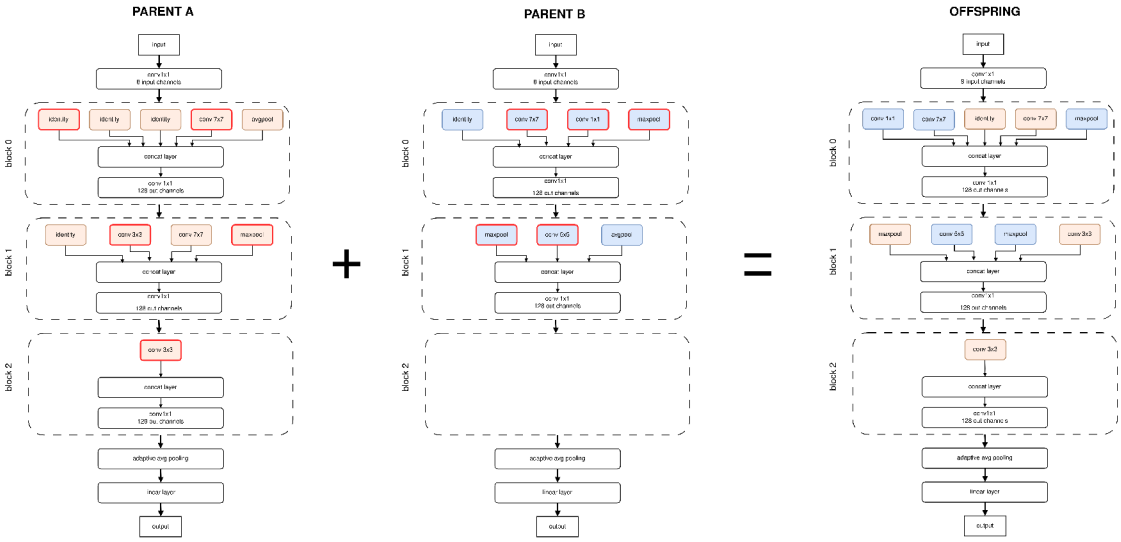


Fig. 5.7: Example of crossover between two neural networks

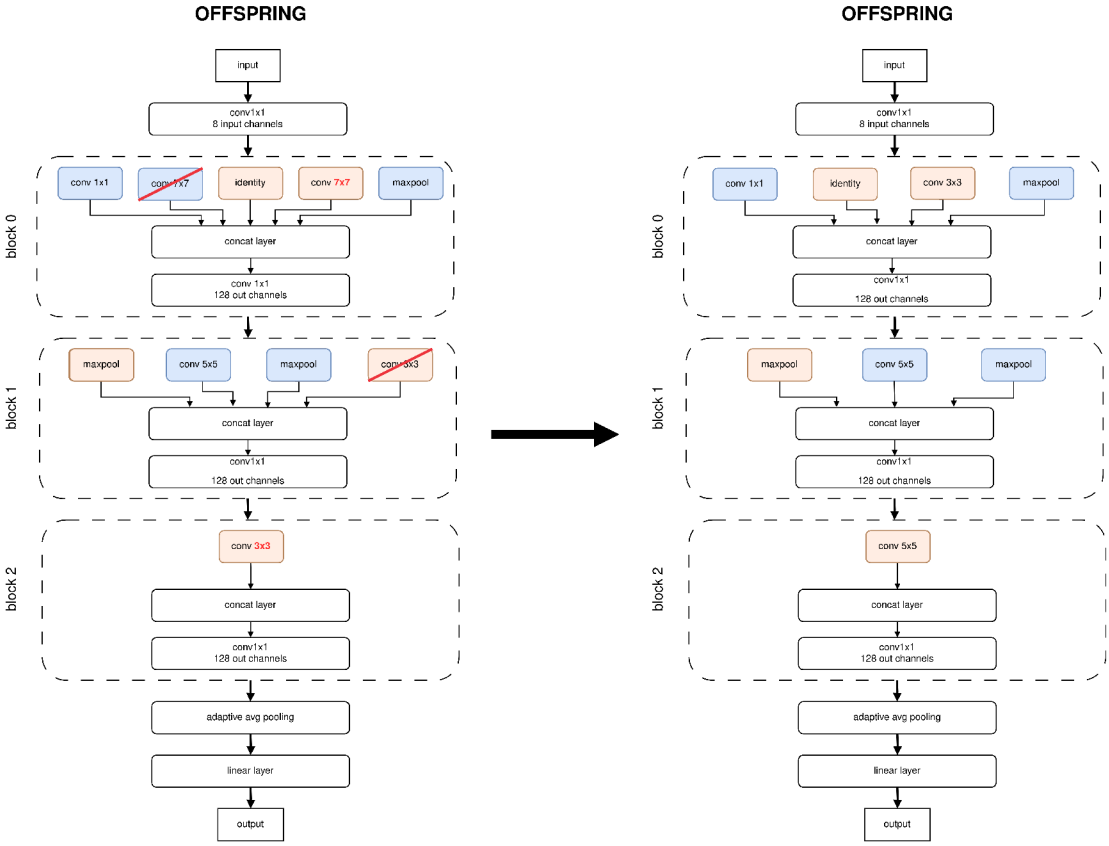


Fig. 5.8: Example of offspring mutation

The encoding of the architecture is realized with Python dictionaries that have 3 layer hierarchy. The dictionaries in the bottom layer have the same structure

with information about the number of input and output channels and kernel size. The middle layer assigns the type of operation and its values are the operation parameters from the bottom layer. The top layer has information about the block number and their names from 0 to Nth block and wraps it all up. An example of this encoding system is shown in Listing 5.1. The rest of the architecture does not change and is directly encoded in the encoder and decoder part of the code. This way of direct encoding via dictionaries allows to do crossover and mutation operations between the operations of the main blocks at the same level. Crossover operation has one additional limitation compared to the algorithm implemented for the hyperparameter optimization and that is the limitation of a maximal number of operations in each block. Here, indexes of operations from parent A and B are randomly chosen and cropped after the maximal number of operations n, which was defined before the algorithm was started, is achieved, and the rest of them are thrown away. The mutation operation has two options, both shown in Figure 5.8 either to delete random operations from the main blocks or to change their parameters, such as the number of output channels or kernel size.

Listing 5.1: Example of encoded architecture with dictionaries

```
 {
'block_0': {
    'identity_0': {'in_channels': 8, 'out_channels': 8, 'kernel_size': (7, 7)},
    'identity_1': {'in_channels': 8, 'out_channels': 8, 'kernel_size': (1, 1)},
    'identity_2': {'in_channels': 8, 'out_channels': 8, 'kernel_size': (7, 7)},
    'conv_3': {'in_channels': 8, 'out_channels': 16, 'kernel_size': (7, 7)},
    'avgpool_4': {'in_channels': 8, 'out_channels': 16, 'kernel_size': (7, 7)}
            },
'block_1': {
    'identity_0': {'in_channels': 128, 'out_channels': 128, 'kernel_size': (3, 3)},
    'conv_1': {'in_channels': 128, 'out_channels': 16, 'kernel_size': (3, 3)},
    'conv_2': {'in_channels': 128, 'out_channels': 32, 'kernel_size': (7, 7)},
    'maxpool_3': {'in_channels': 128, 'out_channels': 32, 'kernel_size': (7, 7)}
            },
'block_2': {
    'conv_0': {'in_channels': 128, 'out_channels': 32, 'kernel_size': (7, 7)}
            }
}
```

The genetic algorithm for the NAS optimization problem was initialized with a population size of 20 and mutation rate of 0.05, 4 input channels for the first covn 1x1 layer, a maximal number of 5 main blocks with a maximal number of 3 operations per block. The convolutional kernel could be of size 1x1, 3x3, and a possible number of output channels can be either 16, 32, 64, or 128. The algorithm was tested on inputs with STFT and WST data transformations and each ran for 3 days on 8 GPUs and during that time was able to evaluate 160 neural networks each.

5.4 Evaluation Metrics

For the evaluation of the neural network's score 4 metrics were monitored, kappa score, F1 score, an area under the receiver operating characteristic (AUROC), and an area under the precision-recall curve (AUPRC). All of those metrics can be used when working with imbalanced data and were also used in previous studies on this dataset [16].

Value of Kappa	Level of Agreement	Reliable Data
0 - 0.20	None to Slight	0-4%
0.21 - 0.39	Minimal	4-15%
0.40 - 0.59	Weak	15-35%
0.60 - 0.79	Moderate	35-63%
0.80 - 0.90	Strong	64-81%
> 0.90	Almost Perfect to Perfect	82-100%

Tab. 5.2: Interpretation of Cohen's Kappa Score

The kappa score was used as a fitness function to evaluate the solutions found by the genetic algorithm. The interpretability of Cohen's Kappa is more intuitive than that of the F1. Compared to a macro F1, Cohen's Kappa is a more strict metric as it also considers a probability of random agreement in the final score. This metric is used to measure how closely the classified instances match the ground truth labels while controlling the accuracy of a random classifier. [42] The interpretation of Cohen's kappa [43] is summarized in Table 5.2 and can be computed as follows:

$$\kappa = \frac{Agreement - AgreementChance}{1 - AgreementChance}.$$

n expert

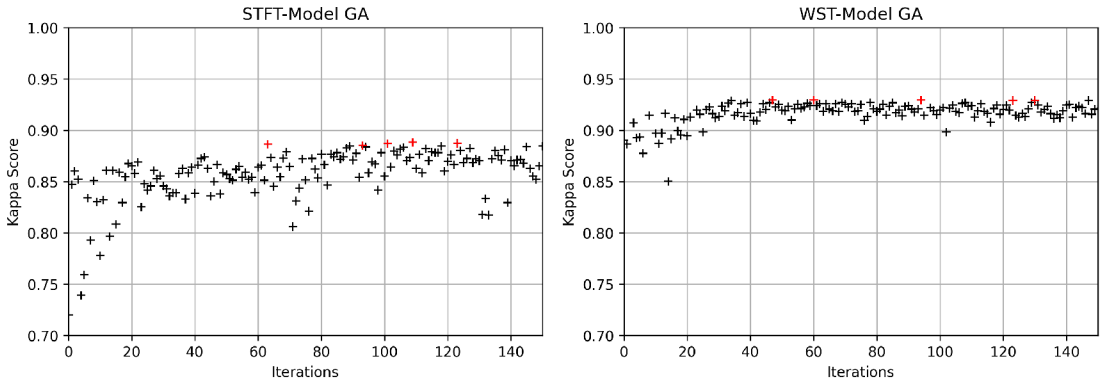
6 Results

This section reports the results of the hyperparameter tuning and neural architecture search. It offers the solution overview of the best-found parameters and results. Discussion of these results is included in the following Chapter 7.

6.1 Hyperparameter Tuning

Solutions Overview

Tables 6.1 and 6.2 offer an overview of the 5 best solutions found by the genetic algorithm for each of the STFT and the WST models and the convergence of the algorithm in the exploitation phase is shown in Fig 6.1a and 6.1b. The chosen parameters of the STFT-Model and the WST-Model used for reporting the final results are highlighted in Tables 6.1 and 6.2.



(a) (b)

Fig. 6.1: Convergence of the genetic algorithm for the STFT- and WST-model during the exploitation part

Performance Overview

An overview of final results is reported in 6.3, where the results are averaged over 5 runs on the test datasets for SAUH and MAYO. We compare the results on the benchmark model and on the best-found solution for the STFT-Model and the WST-Model parameters. We can observe an increase in all measured metrics scores after the optimization process. Tables 6.4 provide better insight into the impact of the improved performance on the classes, monitoring the F1 Score, AUROC, and AUPRC

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**
Hann	$\mathbf{2^5}$	$\mathbf{2^4}$	$\mathbf{2^7}$	$\mathbf{2^8}$	**3**	$\mathbf{2^9}$	**2**	$\mathbf{2^6}$	$\mathbf{5^{-4}}$	**0**
Hann	2^5	2^4	2^8	2^8	5	2^9	3	2^5	5^{-4}	0
Hann	2^5	2^4	2^8	2^5	3	2^9	2	2^4	5^{-4}	0
Hann	2^5	2^4	2^7	2^8	3	2^9	2	2^5	5^{-4}	0
Hann	2^5	2^4	2^8	2^8	5	2^9	2	2^7	5^{-4}	0

Tab. 6.1: Overview of top 5 solutions for the STFT-Model.
STFT parameters: P1 - type of window function, P2 - length of each segment, P3 - number of overlapping points between segments, P4 - length of used FFT with zero padding
Model parameters: P5 - number of convolutional filters, P6 - size of convolutional kernel, P7 - number of features in the hidden state of a GRU layer, P8 - number of recurrent layers, P9 - batch size, P10 - learning rate, P11 - weight decay

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
8	**10**	$\mathbf{2^8}$	**5**	$\mathbf{2^9}$	**2**	$\mathbf{2^7}$	$\mathbf{1^{-3}}$	**0**
8	10	2^9	7	2^7	3	2^6	1^{-3}	0
7	10	2^9	5	2^9	2	2^7	5^{-4}	1^{-5}
7	10	2^{10}	3	2^8	3	2^7	1^{-3}	1^{-5}
7	10	2^{10}	7	2^8	3	2^7	1^{-3}	1^{-5}

Tab. 6.2: Overview of top 5 solutions for the WST-Model.

WST parameters: P1 - maximum log-scale of the scattering transform, P2 - number of wavelets per octave for the first order
Model parameters: P3 - number of convolutional filters, P4 - size of convolutional kernel, P5 - number of features in the hidden state of a GRU layer, P6 - number of recurrent layers, P7 - batch size, P8 - learning rate, P9 - weight decay

for each class. Tables 6.5 are contingency tables with outcomes of the benchmark and the optimized model for the SAUH and MAYO data. McNemar's test based on these results shows that for both datasets, the optimized model significantly improved the final accuracy with a p-value « 0.01.

	Kappa Score	**Macro F1**	**Mean AUROC**	**Mean AUPRC**
	SAUH Dataset			
Benchmark Model [16]	0.8450	0.9076	0.9823	0.9613
STFT-Model	0.9079	0.9419	0.9922	0.9837
WST-Model	**0.9411**	**0.9673**	**0.9969**	**0.9938**
	MAYO Dataset			
Benchmark Model [16]	0.8885	0.9222	0.9897	0.9740
STFT-Model	0.9011	0.9344	0.9919	0.9805
WST-Model	**0.9123**	**0.9400**	**0.9941**	**0.9856**

Tab. 6.3: Comparison of the overall performance between the benchmark model and the best STFT and WST Models.

	F1 Score			**AUROC**			**AUPRC**		
	BM	**STFT**	**WST**	**BM**	**STFT**	**WST**	**BM**	**STFT**	**WST**
	SAUH Dataset								
Powerline	0.9934	**1.0**	**1.0**	0.9999	**1.0**	**1.0**	0.9994	**1.0**	**1.0**
Noise	0.8152	0.8780	**0.9475**	0.9687	0.9873	**0.9886**	0.8962	0.9592	**0.9886**
Pathology	0.8829	0.9217	**0.9451**	0.9763	0.9880	**0.9984**	0.9509	0.9763	**0.9881**
Physiology	0.9054	0.9311	**0.9593**	0.9675	0.9817	**0.9847**	0.9624	0.9792	**0.9925**
	MAYO Dataset								
Powerline	0.9898	0.9953	**0.9970**	0.9995	**0.9999**	**0.9999**	0.9992	**0.9999**	**0.9999**
Noise	0.8779	0.8841	**0.9042**	0.9822	0.9844	**0.9899**	0.9529	0.9577	**0.9740**
Pathology	0.9232	**0.9512**	0.9499	0.9963	**0.9984**	**0.9984**	0.9751	0.9881	**0.9893**
Physiology	0.9013	0.9051	**0.9216**	0.9807	0.9847	**0.9892**	0.9690	0.9754	**0.9818**

Tab. 6.4: Detailed results of F1 Score, AUROC, and AUPRC for each of the classification categories compared between the benchmark model and the best STFT and WST models.

6.2 Neural Architecture Search

Solution Overview

Table 6.6 offers an overview of the 3 best architectures found with the neural architecture search that are reported along with their kappa and F1 scores. The detailed results with class scores can be seen in Table 6.7. Fig. 6.2 shows the convergence of the algorithm during the optimization.

SAUH	**BM Correct**	**BM False**
WST Correct	33917	3214
WST False	626	867

MAYO	**BM Correct**	**BM False**
WST Correct	27797	1545
WST False	893	801

Tab. 6.5: Contingency tables for McNemar's Test on SAUH and MAYO datasets with p-values « 0.01

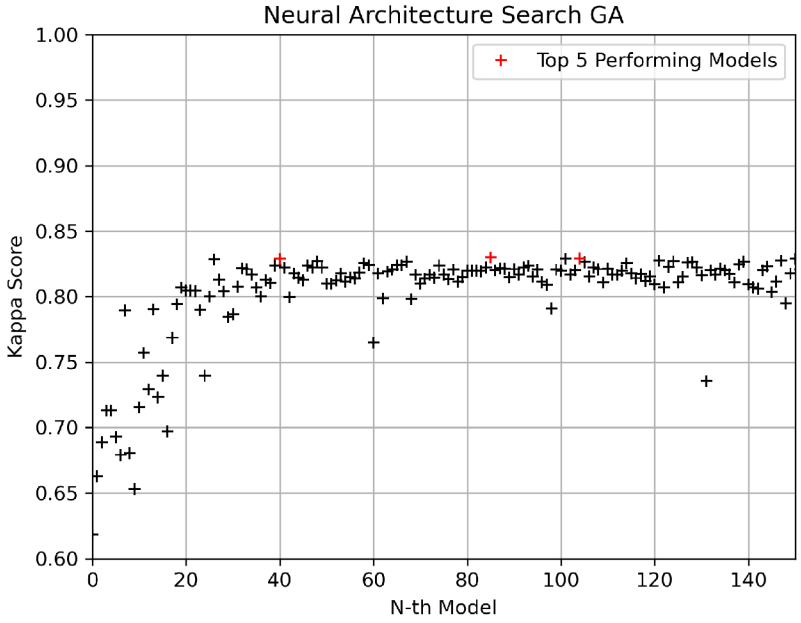


Fig. 6.2: Convergence of the genetic algorithm with Neural Architecture Search

	Operation 1	Operation 2	Operation 3
Block 1	conv 1x1 128 ch	conv 3x3 16 ch	conv 3x3 16 ch
Block 2	conv 3x3 64 ch	conv 1x1 128 ch	conv 3x3 128 ch
Block 3	maxpool	conv 1x1 64 ch	avgpool
Block 4	conv 3x3 128 ch	conv 3x3 128 ch	conv 3x3 64 ch
Block 5	maxpool	maxpool	maxpool
Kappa Score:	0.8299	**F1 Score:**	0.8997

	Operation 1	Operation 2	Operation 3
Block 1	conv 3x3 16 ch	conv 3x3 16 ch	avgpool
Block 2	avgpool	conv 3x3 16 ch	conv 1x1 128 ch
Block 3	maxpool	conv 1x1 64 ch	avgpool
Block 4	conv 3x3 64 ch	conv 3x3 128 ch	conv 3x3 64 ch
Block 5	maxpool	maxpool	maxpool
Kappa Score:	0.8294	**F1 Score:**	0.8991

	Operation 1	Operation 2	Operation 3
Block 1	conv 3x3 128 ch	conv 1x1 128 ch	maxpool
Block 2	identity	conv 3x3 16 ch	conv 3x3 64 ch
Block 3	maxpool	conv 1x1 128 ch	conv 1x1 128 ch
Block 4	maxpool	conv 3x3 64 ch	conv 3x3 128 ch
Block 5	maxpool	maxpool	maxpool
Kappa Score:	0.8290	**F1 Score:**	0.9004

Tab. 6.6: Overview of 3 best found architectures with Neural Architecture Search

	F1 Score		**AUROC**		**AUPRC**	
	BM	NAS	BM	NAS	BM	NAS
Powerline	**0.9934**	0.9987	**0.9999**	**0.9999**	**1.0**	0.9999
Noise	**0.8152**	0.8140	**0.9687**	0.9659	**0.9592**	0.8769
Pathology	0.8829	**0.8831**	0.9763	**0.9766**	**0.9509**	0.9472
Physiology	**0.9054**	0.9030	**0.9675**	0.9651	0.9624	**0.9626**

Tab. 6.7: Detailed results of F1 Score, AUROC, and AUPRC for each of the classification categories compared between the benchmark model and the best NAS model on SAUH data

7 Discussion

This thesis is focused on the optimization of neural network design for iEEG processing and follows up on the work in [16]. Chapters 5 and 6 describe the chosen approach to the optimization which was done with the help of custom designed genetic algorithm and hyperparameter tuning of a benchmark model as well as neural architecture search. The benchmark model is a CNN-GRU model designed by an engineer with expertise knowledge in the field of iEEG processing and neural networks. The optimization was distributed over multiple GPUs which was possible thanks to the GA design and took about 2 to 3 days per method, which is equivalent to 10 days of computing time on a single GPU. For the idea, it would take 180 years to do the grid search optimization in the case of the hyperparameter optimization.

Hyperparameter Tuning

In the case of the hyperparameter tuning, the result achieved by this method was superior to the results achieved by the benchmark model. The optimization was focused both on the optimization of hyperparameters of the neural network and optimization of parameters the input data pre-processed.

During the exploration part of the optimization search, higher increments in the final score were observed when optimizing the preprocessing of the input signal into the model. At this part, the optimized model hyperparameters increased the kappa score by 0.02, whilst the spectrogram and WST parameters caused an increment by 0.04 and 0.06. One of the reasons for this could be the fact that the benchmark model was designed with enough expertise, whilst the choice for the spectrogram parameters in the original was based on visual impression without further fine-tuning.

The results of the overall performance can be seen in Table 6.3 and show improvement in all observed metrics. The wavelet scattering transformation shows along with the model hyperparameter optimization shows better results than the model with STFT signal preprocessing. When trained on the SAUH dataset, the WST-Model optimization improved the mean AUROC score of 0.9823 and mean AUPRC score of 0.9613 to 0.9969 and 0.9938 respectively. The final kappa score of 0.9411 and macro F1 score of 0.9673 showed improvement as well compared to the benchmark model by 0.1 and 0.06 respectively. The performance of the model was also improved on the model that was re-trained on data from the Mayo Clinic. The improvement of the separate classes can be seen in more detail in Table 6.4. The increment of the final score is caused mainly by improved classification of noise

and physiology classes and, most importantly, also of the pathology class, whose F1 score improved from 0.8829 to 0.9451.

7.1 Neural Architecture Search

The results of the neural architecture search are comparable with the results of the benchmark model. While this is nothing superior, the NAS approach was able to find an architecture that is still comparable with the original scores with just a few design rules and basic operations such as convolution, maximum and average pooling, and identity. Since the optimization problem is quite large and we only sampled a few networks with quite limited parameters (1x1 and 3x3 convolutional kernel size only), a pre-set number of input channels and output channels from each main block, etc... it is possible, that under further exploration and tweaking of the algorithm setting and longer computation time, the NAS could find could yet find better fitting architecture for the task.

8 Conclusion

The aim of this thesis was to design an evolutionary algorithm for the optimization of neural networks that processes EEG signals. A genetic algorithm with uniform crossover and random mutation of the genes was chosen for this task. The algorithm was designed to work both synchronously or asynchronously and can thus speed up the optimization process if more GPU processors are available. Due to its design, it only needs small adjustments to upgrade it from hyperparameter optimization to neural architecture search, which was explored in this thesis.

The data used in this thesis are intracranial EEG recordings from epileptic patients undergoing pre-surgical examinations at St. Anne's University Hospital and Mayo Clinic. The results achieved within this thesis were compared to a benchmark model that was designed and optimized for this task by an engineer.

This work introduces two approaches to neural architecture optimization. The first is hyperparameter optimization, in which the hyperparameters of the benchmark model are optimized with the help of the genetic algorithm in order to find a better solution. The second approach is a neural architecture search, in which the genetic algorithm searches for optimal architecture for the given task based on some set of pre-designed rules. Both of the approaches were trained, validated, and tested on the St. Anne's Hospital dataset. The optimization process took about 10 days of computing time*.

The results of the hyperparameter optimization were much more promising than the results of the neural architecture search. The NAS approach resulted in a model that had results comparable to the benchmark model. The hyperparameter tuning approach on the other hand was able to significantly improve the final score of the neural network and most importantly, improved the score on pathology and noise classes. The result of the hyperparameter tuning was also tested on the dataset from Mayo Clinic, where the results improved as well with the optimized version of the model.

While the neural architecture search did not result in any superior results, it had comparable results with the benchmark model, even though it had no prior knowledge about the problem and only a few simple building blocks and operations and a set of rules for architecture design at hand. The hyperparameter optimization showed great improvement in the final classification scores and can thus further positively impact the detection of pathological activity or noise detection in iEEG signal processing.

*Calculated for computation on a single GPU processor.

Bibliography

[1] Miotto, R., Wang, F., Wang, S., Jiang, X., & Dudley, J. T. (2018). Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics, 19(6), 1236-1246.

[2] Saeid S. and Chambers J. A. EEG signal processing. Chichester: Wiley, 2007. xxii, 289 s. ISBN 978-0-470-02581-9

[3] Attwood, H. L., and MacKay, W. A., Essentials of Neurophysiology, B. C. Decker, Hamilton, Canada, 1989.

[4] Li, R., Yang, D., Fang, F., Hong, K. S., Reiss, A. L., & Zhang, Y. (2022). Concurrent fNIRS and EEG for brain function investigation: a systematic, methodology-focused review. Sensors, 22(15), 5865.

[5] Nejedly, P., Kremen, V., Sladky, V., Cimbalnik, J., Klimes, P., Plesinger, F., ... & Worrell, G. (2020). Multicenter intracranial EEG dataset for classification of graphoelements and artifactual signals. Scientific data, 7(1), 1-7.

[6] Nejedly, P., Cimbalnik, J., Klimes, P., Plesinger, F., Halamek, J., Kremen, V., ... & Jurak, P. (2019). Intracerebral EEG artifact identification using convolutional neural networks. Neuroinformatics, 17(2), 225-234.

[7] Gibbs, F. A., & Gibbs, E. L. (1941). Atlas of electroencephalography.

[8] Frauscher, B., Von Ellenrieder, N., Zelmann, R., Doležalová, I., Minotti, L., Olivier, A., ... & Gotman, J. (2018). Atlas of the normal intracranial electroencephalogram: neurophysiological awake activity in different cortical areas. Brain, 141(4), 1130-1144.

[9] Craik, A., He, Y., & Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: a review. Journal of neural engineering, 16(3), 031001.

[10] Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T. H., & Faubert, J. (2019). Deep learning-based electroencephalography analysis: a systematic review. Journal of neural engineering, 16(5), 051001.

[11] Shoeibi, A., Khodatars, M., Ghassemi, N., Jafari, M., Moridian, P., Alizadehsani, R., ... & Acharya, U. R. (2021). Epileptic seizures detection using deep learning techniques: a review. International Journal of Environmental Research and Public Health, 18(11), 5780.

[12] Beghi, E. (2020). The epidemiology of epilepsy. Neuroepidemiology, 54(2), 185-191.

[13] Asadi-Pooya, A. A., Stewart, G. R., Abrams, D. J., & Sharan, A. (2017). Prevalence and incidence of drug-resistant mesial temporal lobe epilepsy in the United States. World neurosurgery, 99, 662-666.

[14] Rosenow, F., & Lüders, H. (2001). Presurgical evaluation of epilepsy. Brain, 124(9), 1683-1700.

[15] Nejedly, P., Cimbalnik, J., Klimes, P., Plesinger, F., Halamek, J., Kremen, V., ... & Jurak, P. (2019). Intracerebral EEG artifact identification using convolutional neural networks. Neuroinformatics, 17(2), 225-234.

[16] Nejedly, P., Kremen, V., Sladky, V., Cimbalnik, J., Klimes, P., Plesinger, F., ... & Worrell, G. (2019). Exploiting graphoelements and convolutional neural networks with long short term memory for classification of the human electroencephalogram. Scientific reports, 9(1), 1-9.

[17] Simon, D. (2013). Evolutionary optimization algorithms. John Wiley & Sons.

[18] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. arXiv preprint arXiv:2106.11342.

[19] Yu, T., & Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689.

[20] Zhou, M. (2021). Heuristic Hyperparameter Optimization for Convolutional Neural Networks using Genetic Algorithm. arXiv preprint arXiv:2112.07087.

[21] Aszemi, N. M., & Dominic, P. D. D. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. International Journal of Advanced Computer Science and Applications, 10(6).

[22] Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 50(9), 3840-3854.

[23] Xie, L., & Yuille, A. (2017). Genetic cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1379-1388).

[24] Gao, Z., Li, Y., Yang, Y., Wang, X., Dong, N., & Chiang, H. D. (2020). A GPSO-optimized convolutional neural networks for EEG-based emotion recognition. Neurocomputing, 380, 225-235.

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

[26] Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2018). Regularized Evolution for Image Classifier Architecture Search.(2018). arXiv preprint arXiv:1802.01548.

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

[28] Zhao, Y., Wang, L., Tian, Y., Fonseca, R., & Guo, T. (2021, July). Few-shot neural architecture search. In International Conference on Machine Learning (pp. 12707-12718). PMLR.

[29] Stanley, K. O., Clune, J., Lehman, J., & Miikkulainen, R. (2019). Designing neural networks through neuroevolution. Nature Machine Intelligence, 1(1), 24-35.

[30] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.

[31] Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567.

[32] Darwish, A., Hassanien, A. E., & Das, S. (2020). A survey of swarm and evolutionary computing approaches for deep learning. Artificial intelligence review, 53(3), 1767-1812.

[33] Plesinger, F., Jurco, J., Halamek, J. & Jurak, P. SignalPlant: an open signal processing software platform. Physiol. Meas. 37, N38–48 (2016)

[34] Nejedly, P., Plesinger, F., Halamek, J. & Jurak, P. CudaFilters: A SignalPlant library for GPU-accelerated FFT and FIR filtering. Softw. Pract. Exp. 48, 3–9 (2018)

[35] Brázdil, M. et al. Very high-frequency oscillations: Novel biomarkers of the epileptogenic zone. Ann. Neurol. 82, 299–310 (2017).

[36] Andén, J., & Mallat, S. (2014). Deep scattering spectrum. IEEE Transactions on Signal Processing, 62(16), 4114-4128.

[37] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & Van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272.

[38] Andreux, M., Angles, T., Exarchakis, G., Leonarduzzi, R., Rochette, G., Thiry, L., ... & Eickenberg, M. (2020). Kymatio: Scattering Transforms in Python. J. Mach. Learn. Res., 21(60), 1-6.

[39] Umbarkar, Walchand College of Engineering, Sheth, and Government College of Engineering, Karad. 2015. "Crossover Operators in Genetic Algorithms: A Review." ICTACT Journal on Soft Computing 06 (01): 1083–92.

[40] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... & Lerer, A. (2017). Automatic differentiation in pytorch.

[41] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions (2014). arXiv preprint arXiv:1409.4842. 2014;10.

[42] Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and psychological measurement, 20(1), 37-46.

[43] McHugh, M. L. (2012). Interrater reliability: the kappa statistic. Biochemia medica, 22(3), 276-282.

Symbols and abbreviations

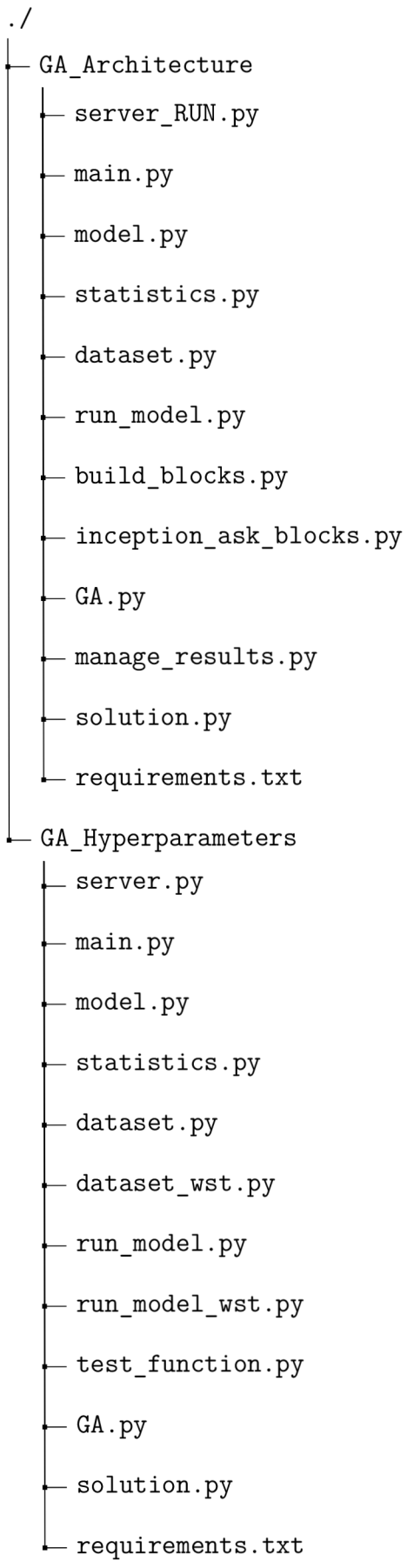
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Curve
BM	Benchmark Model
DL	Deep Learning
EEG	Electroencephalogram
FFT	Fast Fourier Transform
GA	Genetic Algorithm
GPU	Graphics Processing Unit
iEEG	Intracranial Electroencephalogram
MAYO	Mayo Clinic
ML	Machine Learning
NAS	Neural Architecture Search
NLP	Natural Language Processing
SAUH	St. Anne's University Hospital
STFT	Short-Time Fourier Transform
WST	Wavelet Scattering Transform

Appendix

The electronic appendix contains a list of source codes used for the realization of this thesis. The data usod for the thesis are publicly available[†] and the source codes are available on GitHub[‡].

[†]https://springernature.figshare.com/collections/Multicenter_intracranial_EEG_dataset_for-_classification_of_graphoelements_and_artifactual_signals/4681208

[‡]https://github.com/KristynaPijackova/Diploma-Thesis-GA-for-iEEG


./
GA_Architecture
server_RUN.py
main.py
model.py
statistics.py
dataset.py
run_model.py
build_blocks.py
inception_ask_blocks.py
GA.py
manage_results.py
solution.py
requirements.txt
GA_Hyperparameters
server.py
main.py
model.py
statistics.py
dataset.py
dataset_wst.py
run_model.py
run_model_wst.py
test_function.py
GA.py
solution.py
requirements.txt

Ethical Statement

This study was carried out in accordance with the approval of the Mayo Clinic Institutional Review Board with written informed consent from all subjects. The protocol was approved by the Mayo Clinic Institutional Review Board and St. Anne's University Hospital Research Ethics Committee and the Ethics Committee of Masaryk University. All subjects gave written informed consent in accordance with the Declaration of Helsinki. All methods were performed in accordance with the relevant guidelines and regulations.