Silesian University of Technology

Faculty of Biomedical Engineering
Department of Biosensors and Processing of Biomedical Signals

Doctoral Dissertation

Deep learning applications in biomedical engineering

Author: mgr inż. Konrad Duraj
Scientific supervisor: dr hab. inż. Paweł Kostka, prof. PŚ

Gliwice, June 2023

Abstract

In the past decade, artificial intelligence has been the subject of intense media hype and research investigation. Deep learning, in particular, has emerged as the go-to technology for information processing. The ability of the neural networks to learn from data, approximate highly nonlinear patterns, and processing unstructured data make it possibly the most powerful technology nowadays. A sector that can benefit the most from advancements in this field is biomedical engineering. This thesis presents novel applications for this technology in the biomedical engineering domain and also discusses what is missing to fully apply these methods in real world scenarios. In the last chapter, a new paradigm is proposed which addresses some of the discussed shortcomings. The dissertation is divided into an introduction and theoretical analysis, a set of publications representing the contribution of the author to applying deep learning in biomedical applications, and the proposition of a new methodology.

Keywords: deep learning; biomedical engineering; fuzzy logic; fuzzy inference systems.

Acknowledgments

Throughout my doctoral studies, I have received significant support and assistance. I would like to express my gratitude.

- In the first place, I would like to thank my supervisor Eng. Paweł Kostka, Ph.D., for his professional mentoring, inspiration, and motivation. His support and feedback pushed me to sharpen my thinking. I especially want to thank him for allowing me to explore different topics, which helped me find a research direction that I want to explore in my future work.
- I would like to thank the head of my department, professor Ewaryst Tkacz, for constant support and allowing me to pursue my research interest in his department.
- To my friends and colleagues, thank you for being with me along the way. I want to give a special shout-out to M.Sc. Eng. Natalia Piaseczna for her energy, understanding and help.
- My gratitude also goes to my family, my parents, my sister, my grandmother, and my brother-in-law for their encouragement and support. Because of you, I am where I am. Thank you.

I dedicate this dissertation to my grandfather Ignacy.

Contents

Chapter 1

Introduction

1.1 History of artificial intelligence

Despite being studied for many years, artificial intelligence (AI) is still one of the most enigmatic topics in computer science. This is because the subject is so broad and hazy. AI includes everything from extremely intelligent machines to search algorithms used in board games. It can be applied in almost any way we use computers nowadays. Although it is difficult to pinpoint, the roots of AI can probably be traced back to the 1940s, specifically 1942, when the American Science Fiction writer Isaac Asimov published his short story Runaround [1]. In 1950, Alan Turing, a mathematician widely referred to as the father of computer science, published his seminal article entitled "Computing Machinery and Intelligence", where he described how to create intelligent machines and in particular how to test their intelligence. This Turing test is still considered today as a benchmark for identifying the intelligence of an artificial system. The test was set up as an experiment to validate that a human interacting with a machine through language will not be able to distinguish the machine from other humans[1]. The term artificial intelligence was first coined by McCarthy in 1956 when he held the first academic conference on the subject. Together with Marvin Minsky (another computer scientist),organized an eight-week workshop on artificial intelligence called *Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI)* at Darthmouth College in New Hampshire. The objective of DSRPAI was to bring together researchers from various fields to create a new research area that aims to build machines capable of simulating human intelligence [1].

1.1.1 AI spring

Following the Dartmouth Conference, the field of artificial intelligence has experienced substantial success for nearly two decades. An early example is the famous ELIZA computer program, created between 1964 and 1966 by Joseph Weizenbaum at the Massachusetts Institute of Technology [1]. ELIZA was a natural language processing tool capable of conducting a conversation with a human and one of the first programs capable of attempting to pass the Turing test mentioned above [1]. In 1970, Marvin Minsky gave an interview to Life Magazine in which he stated that a machine with the general intelligence of an average human being could be developed in three to eight years [1].

1.1.2 AI winter

Three years after the aforementioned claim made by Marvin Minsky, immense investments in AI development have provoked sharp criticism from the US Congress. In the same year, British mathematician James Lighthill questioned the optimistic prognosis offered by AI researchers in a paper commissioned by the British Science Research Council. In games like chess, machines would only ever be able to compete at the level of a "seasoned amateur", according to Lighthill, and they would never be capable of using common sense reasoning. As a consequence, the British government stopped funding research into AI. This period started the AI Winter. Although the Japanese government began to heavily fund AI research in the 1980s, to which the US DARPA responded by a funding increase as well, no further advances were made in the following years [1].

1.2 Expert systems

An expert system seeks to capture the knowledge of a human expert-an individual whose knowledge within a highly specialized area is recognized to be superior [2]. To understand how expert systems operate, it is necessary to understand a few fundamental ideas.

1.2.1 Knowledge engineering

The process of collecting knowledge from human experts and representing it in a computer is called 'knowledge engineering'. There are really two problems here: acquiring and representing human knowledge in the computer system. Knowledge acquisition is the process of extracting knowledge from the human expert [2]. Knowledge representation is a process of converting acquired knowledge into a form that can be accessed and processed using a computer.

1.2.2 Knowledge base

Software engineering can collaborate with human subject matter experts to understand a particular area of concern. Next, they use the many programming languages and techniques available to AI to encode this knowledge, building a knowledge base. The notion of a knowledge base leads to the concepts of knowledge-based systems and knowledge-based programming. The field of knowledge-based systems is closely related to the field of expert systems. A knowledge-based system may be thought of as a system that possesses general knowledge, perhaps extracted from multiple individuals and perhaps from general reference works within a particular problem domain [2]. There are several types of data structures that can represent a knowledge base.

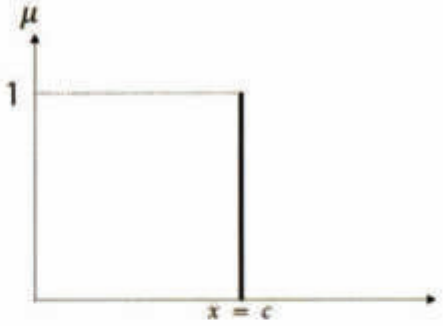
- **Hierarchical tree structure** - this knowledge base organizes the information into a tree-like structure where each node in the tree represents a category or a sub-category,
- **Relational database** - this knowledge base organizes the information in a form of rows and columns,

- **Graphs** - a knowledge graph represents objects as nodes and their relation to each other as links. This can be useful for representing complex and abstract relationships between different pieces of information,
- **Fuzzy base** - a system that utilizes fuzzy logic to represent knowledge via linguistic variables and fuzzy rules that apply to them. It was created to mimic the human way of dealing with ambiguity in the data.

1.2.3 Fuzzy logic

Fuzzy logic is a mathematical framework for dealing with uncertainty and imprecision in decision-making. It was first introduced by Lotfi A. Zadeh, a mathematician and computer scientist, in 1965 [3]. Zadeh came to the conclusion that many real-world issues are too intricate to be explained by conventional binary logic, which only accepts true or false values. Instead, he suggested "fuzzy logic," a more flexible method that accepts varying degrees of truth and partial degrees of membership in categories. Depending on how well a specific item or notion fits into several categories, fuzzy logic assigns varying degrees of membership to those categories. Numbers between 0 and 1 are used to symbolize these degrees of membership, with 0 denoting "not a member at all" and 1 denoting "completely a member." Functions that fuzzify an input feature are called membership functions. The most popular membership functions used nowadays are listed below:

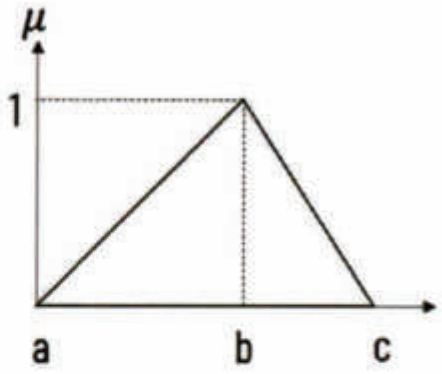
1. **Singleton** - assigns membership value of 1 to a particular value of x (input variable) and assigns 0 to the rest. It is represented by an impulse function. A mathematical formulation is presented by 1.1. The graphical representation of this function is presented in Figure 1.1



$$\mu(x) = \begin{cases} 1 & \text{if } x = c \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

Figure 1.1: Singleton membership function [4]

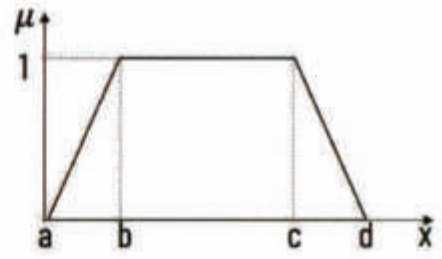
2. **Triangular** - assigns the membership value following a triangular-shaped function. It is defined by three parameters: a lower bound a, an upper bound b, and a peak value c. The membership function increases linearly from a to c, reaches its maximum value of 1 at c, and then decreases linearly from c to b. The mathematical formula is described as 2. The graphical interpretation is presented in the figure 2



$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{c-a} & \text{if } a < x \leq c \\ \frac{b-x}{b-c} & \text{if } c < x < b \\ 0 & \text{if } x \geq b \end{cases} \tag{1.2}$$

Figure 1.2: Triangular membership function [4]

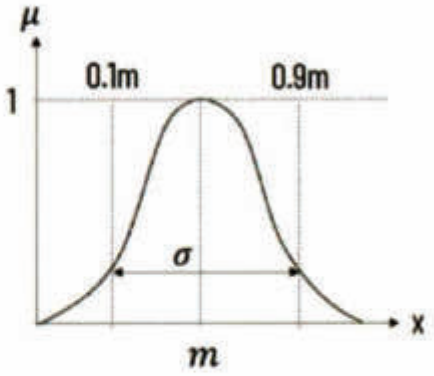
3. **Trapezoidal** - this membership function is defined by four parameters: a lower bound a, a left plateau b, a right plateau c, and an upper bound d. The membership function is 1 in the range between b and c, and increases linearly from a to b and from c to d. The mathematical formula is described as 3. The graphical interpretation is presented in Figure 1.3



$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a < x \leq b \\ 1 & \text{if } b < x \leq c \\ \frac{d-x}{d-c} & \text{if } c < x < d \\ 0 & \text{if } x \geq d \end{cases} \tag{1.3}$$

Figure 1.3: Trapezoidal membership function [4]

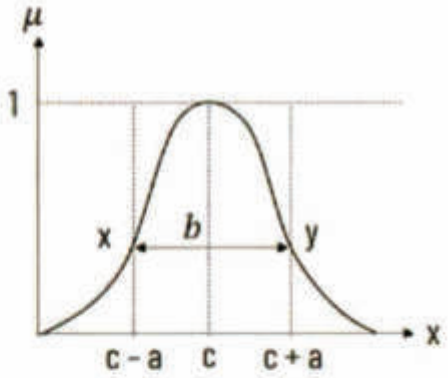
4. **Gaussian** - this membership function is defined by two parameters: a mean value m and a standard deviation σ. The mathematical formula is described as 4. The graphical interpretation is presented in Figure 1.4



$$\mu(x) = exp(-\frac{(x-m)^2}{2\sigma^2}) \qquad (1.4)$$

Figure 1.4: Gaussian membership function [4]

5. **Generalized Bell** - this membership function is defined by three parameters: a, b, c. This function has a maximum value of 1 at $x = c$, and gradually decreases to 0 as x moves further away from c in either direction. The parameter a determines the width of the curve, while the parameter b controls the shape of the curve, allowing it to be symmetric (when $b = 1$) or skewed (when $b \neq 1$). The parameter c determines the center of the curve. The mathematical formula is described as 1.5. The graphical interpretation is presented in Figure 1.5



$$\mu(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \qquad (1.5)$$

Figure 1.5: Generalized Bell membership function [4]

Fuzzy logic is particularly useful for systems that involve human judgment or perception, where there is often a great deal of ambiguity or uncertainty. Some popular applications of fuzzy logic include control systems for industrial processes, image and signal processing, and decision making in fields such as finance and medicine [5].Systems that use fuzzy logic to determine what to do next are called fuzzy systems. A set of fuzzy rules that describe how several inputs should be combined to produce an output often makes up these systems. The rules are typically stated as "if-then" phrases, such as "lower the flow rate if the temperature is too high." The system then combines these rules using fuzzy logic to create the final result, which can be either a numerical number or a collection of language phrases (e.g. "hot," "medium" or "cold") [6].

1.2.4 The downfall of expert systems

Expert systems were once hailed as a breakthrough technology that would revolutionize the way people solve complex problems. However, their popularity declined in the early 2000s, and today they are being used less and less. One of the main reasons for the downfall of expert systems is their inability to learn from data; they rely on the predetermined set of rules and expertise provided by the human experts. Thus, these systems at their peak will be as good as the software engineers who created them and are incapable of providing new ideas in the scope of scientific knowledge. They are incapable of processing unstructured data such as:

- images,
- text,
- time-series,
- graphs,
- videos.

which decreased their utilization even more. Overall, expert systems' demise can be largely linked to their limited capacity for learning and adaption, their reliance on human subject matter experts, and the development of more sophisticated and adaptable technology.

1.3 Machine learning

Expert systems were the first attempt to make computers solve problems in our daily life. The biggest flaw in their development pipeline was the fact that they required humans at every step, from data collection, through feature extraction, to decision-making at the very end. Researchers working in the field of AI saw that flaw and developed a series of solutions that would automate at least one of these steps, decision making. Those algorithms expect a series of features describing a particular problem and try to map it onto the decision space by adjusting the decision boundary, which is a hyperplane that separates the input space into different regions based on a given classification rule. When given enough examples those statistical models can actually become a good estimator for future data points. There are several modes of learning (optimization methods) and potential methods for each of them.

1.3.1 Modes of learning

There are three main modes of learning. Two of them concern how much we help the machine make the correct predictions. The third is about teaching an algorithm to do things/explore the environment.

1. **Supervised learning** - this mode of learning is often described as the student-teacher mode. In supervised learning, the data set is a collection labeled examples

$(x_i, y_i)_{i=1}^N$. Each element x_i among N is called a feature vector. A feature vector is a vector in which each dimension $j = 1, ..., D$ contains a value that describes the example in some way. Each feature is denoted as $x^{(j)}$. The goal of a supervised learning algorithm is to use the data set to produce a model that takes a feature vector x as input and outputs information that allows to label this feature vector. For example, the model created using the data set of people could take as input a feature vector describing a person and output a probability that the person has cancer [7].

2. **Unsupervised learning** - in this mode the data set is a collection of unlabeled examples $x_{i_{i=1}}^N$. The goal of an unsupervised learning algorithm is to create a model that takes a feature vector and either transforms it into another vector (often called an embedding) or into a value that can be used to solve a particular problem. For example, in clustering, the model returns an identifier for a specific cluster for each feature vector in a data set. In dimensionality reduction, the output of a model is a feature vector that has fewer features than the input x. In outlier detection, the output is a real number that indicates how specific input vector is different from the "typical" example in a data set [7].

3. **Reinforcement learning** - this mode of learning is special because it is driven by an agent (machine learning model) interacting with the environment (simulation); thus, by exploring it, the model can learn solving different tasks. The machine can execute actions in every state. Different actions bring different rewards and could also move the machine to another state of the environment. The goal of a reinforcement learning algorithm is to learn a policy. A policy is a function f (similar to the model in supervised learning) that takes the feature vector of a state as input and outputs an optimal action to execute in that state. The action is optimal if it maximizes the expected average reward [7].

1.3.2 Popular algorithms

There exist several algorithms for each mode of learning. Most popular of them are listed below.

- **Linear Regression** - it is one of the oldest and most widely practiced data analysis methods. In many real data settings, least squares linear regression leads to performance on par with state-of-the-art (and often far more complicated) methods while remaining amenable to interpretation [8]. It works by finding through optimization process, a set of parameters a, b that minimizes and the objective function, a measure of error between prediction and target value (for example, mean squared error). Given a linear model as:

$$f_{a,b}(x) = ax + b \tag{1.6}$$

we want to minimize the objective function that measures an error. The objective function can be defined as:

$$L = \frac{1}{N}\sum_{i=1}^{N}[y_i - f_{a,b}(x_i)]^2 \tag{1.7}$$

It is worth noting that for the linear regression case, there exist a number of methods to find the optimal set of parameters that does not require complex optimization. The above solution is just an example.

- **Logistic Regression** - similarly to linear regression, this method optimizes a set of two parameters to perform the classification task. The most widely used function is a sigmoid function which can be mathematically expressed as:

$$f_{a,b}(x) = \frac{1}{1 + e^{-(ax+b)}} \tag{1.8}$$

In logistic regression, instead of using a squared loss and trying to minimize empirical risk, we maximize the likelihood of our training set according to the model. In statistics, the likelihood function defines how likely the observation (an example) is according to our model. The optimization criterion in logistic regression is called maximum likelihood. Instead of minimizing the average loss, as in linear regression, we now maximize the likelihood of training data according to our model [7]:

$$L = \prod_{i=1}^{N} f_{a,b}(x_i)^{y_i}(1 - f_{a,b}(x_i))^{(1-y_i)} \tag{1.9}$$

- **Decision Tree** - a decision tree is a model that predicts a label associated with a feature vector by traveling from a root node of a tree to a leaf. At each node in the root-to-leaf path, the successor child is chosen on the basis of splitting the input space. Usually, the splitting is based on one of the features of x or on a predefined set of splitting rules. A leaf contains a specific label [9].

- **Support Vector Machines** - the goal of this algorithm is to search for "large margin" separators. Roughly speaking, a half-space separates a training set with a large margin if all the examples are not only on the correct side of the separating hyperplane but also far away from it. Restricting the algorithm to output a large margin separator can yield a small sample complexity even if the dimensionality of the feature space is high (and even infinite) [9].

- **K-Nearest Neighbors** is a nonparametric learning algorithm. Contrary to other learning algorithms that allow one to discard the training data after the model is built, kNN keeps all the training examples in memory. Once a new, previously unseen example x comes in, the kNN algorithm finds k training examples closest to x and returns the majority label (in case of classification) or the average label (in case of regression). The closeness of two points is given by a distance function. For example, the Euclidean distance seen above is frequently used in practice. Another popular choice of the distance function is the negative cosine similarity [7].

1.3.3 Where machine learning fails

Although the immense success that methods of machine learning have brought in the Internet and big data era, they still were not enough to solve many of computer science problems. Machine learning algorithms still suffer from the inability to deal with the data in its raw form (audio, video, images, text). Moreover, these solutions still rely on the features provided by humans, thus assuming that software engineers and computer scientists are capable of creating a set of instructions that will extract meaningful information for a given problem.

1.4 Deep learning

Conventional machine-learning techniques were limited in their ability to process natural data in raw form. For decades, the construction of a pattern recognition or machine learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns [10]. In recent years, deep learning has emerged as a leading approach to AI.

Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but nonlinear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned [10]. The success of A. Krizhevsky, I. Sutskever and G. Hinton ImageNet paper [11], which described a deep neural network that can categorize images with previously impossible accuracy, achieving 16.4 % top-5 error rate started a new "revolution" in AI community. Their team was the only team that used a neural network in 2012. The next year, the deep learning computer vision revolution was in full force with the vast majority of teams entering using deep neural networks, and the winning error rate dropped again substantially to 11. 7%. The human error on this task is just above 5% if the human practices for 20 hours. During the years 2011 to 2017, the winning Imagenet error rate dropped sharply from 26% in 2011 to 2.3% in 2017 [12].

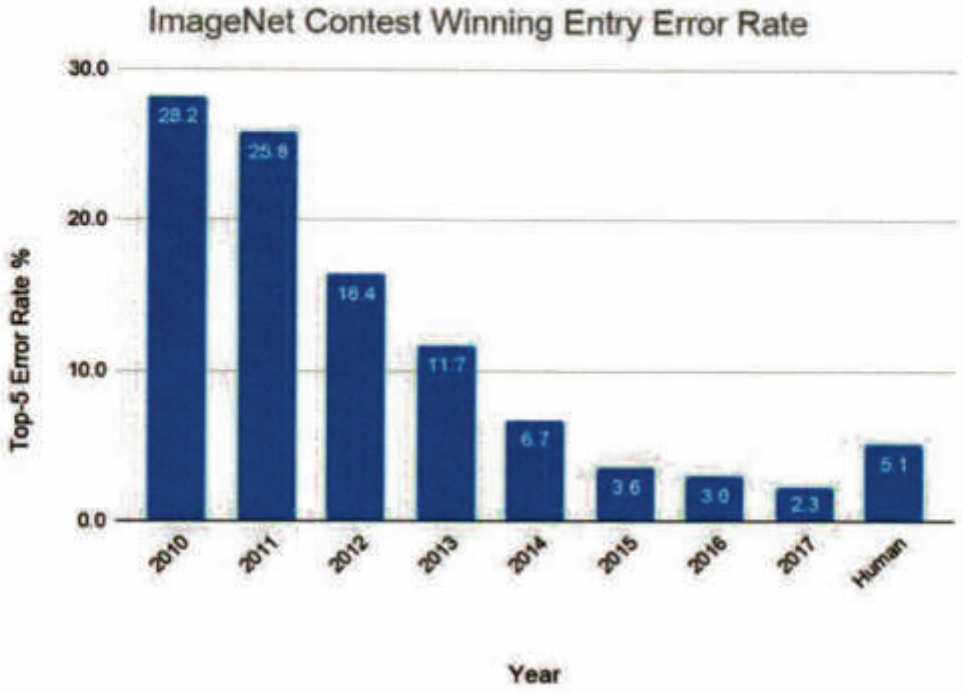


Figure 1.6: ImageNet top-5 error rate over time [12].

Deep learning is making substantial progress in addressing problems that have long defied the best efforts of the artificial intelligence community. Because of its success in identifying complex structures in high-dimensional data, it can be used in a wide range of scientific and commercial applications. Examples of applications are listed below.

- language modelling [13],
- machine translation [14],
- text generation [13],
- image classification [15, 16, 17],
- object detection [18, 19, 20],
- semantic segmentation [21, 22],
- time series forecasting [23],
- speech recognition [24],
- graph classification [25],
- drug discovery [26]
- and many others.

With increasing amounts of data, deep learning has surpassed the conventional machine learning models. Modern architectures have yet to reach their limits on some tasks. The comparison between deep learning models and machine learning models can be seen in Figure 1.7.

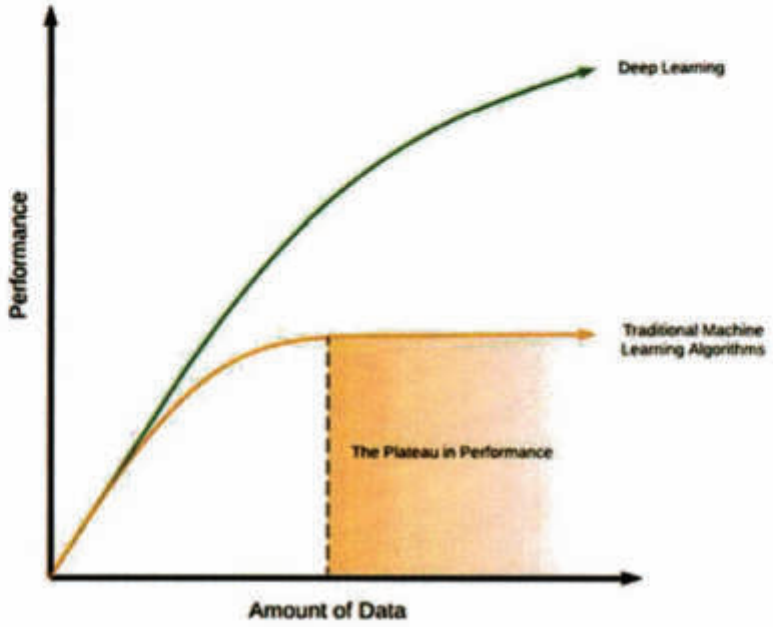


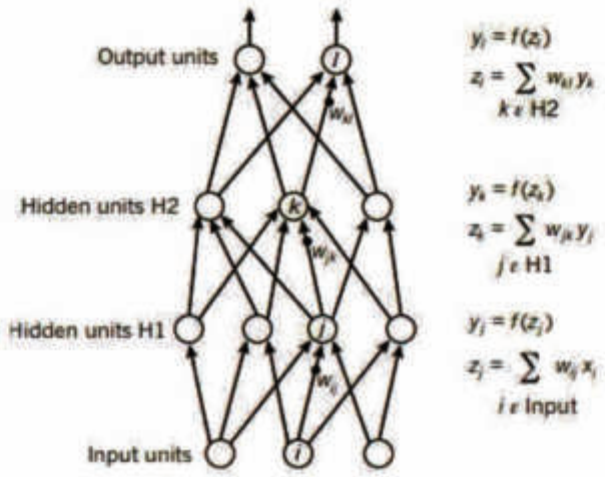
Figure 1.7: Performence comparison between neural networks and machine learning models [27].

1.4.1 Deep learning paradigm

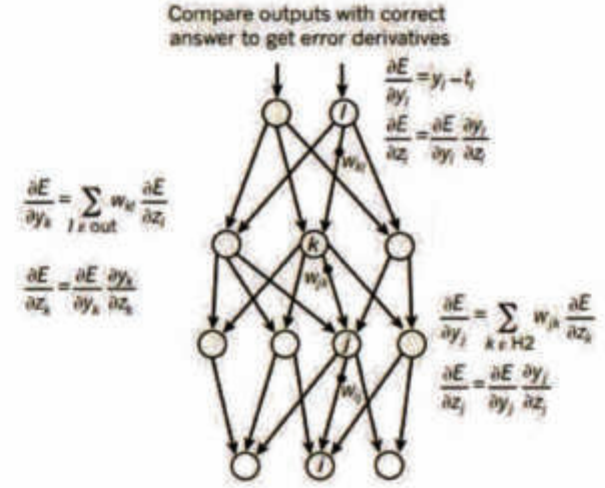
Deep learning went one step further than traditional machine learning methods. Instead of actively trying to design features for classification task, it automates feature extraction process and decision making in a single architecture. The machine generates an output in the form of a vector of scores, one for each category, after being presented with examples of data during training. Prior to training, it is improbable that the targeted category would have the best score of all categories. The error (or distance) between the output scores and the desired pattern is calculated using an objective function. To reduce this inaccuracy, the machine then alters its internal adjustable parameters. These programmable variables, frequently referred to as weights, are values that control the machine's input-output functionality. There may be hundreds of millions of these configurable weights and hundreds of millions of tagged samples in a typical deep learning system. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector [10].

Nowadays one of the most popular optimization procedures is called stochastic gradient descent, which was first formalized in the paper [28]. After computing an error

value for a subset of the whole data set, we compute the gradient of the objective function with respect to all the weights of the neural network. Computation of the output is often referred to as "forward propagation" and propagating the error signal through the network as "backward propagation" or "backpropagation" for short. Both forward 1.8a and backward 1.8b procedures are illustrated in Figure 1.8.



(a)



(b)

Figure 1.8: Computation procedure of neural network illustrated [10].

Estimating gradients for compositions of functions can be computed using the chain rule. The partial derivative $\frac{\partial y}{\partial x}$ measures the influence of a variable x on another y. The $\nabla_\theta J$ denotes the gradient vector of a scalar J with respect to some vector of variables θ. The computation graph of gradient estimation for composition of functions is illustrated in Figure 1.9.

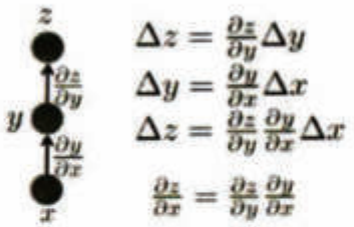


Figure 1.9: Gradient estimation for simple composition of functions [29].

What we are interested in is how a small change in x influences z. The objective function $z = J(g(\theta))$ is what we optimize for. Taking g as a scalar, the gradient can be decomposed by chain rule as follows:

$$\nabla_\theta J(g(\theta)) = \nabla_{g(\theta)} J(g(\theta)) \frac{\partial g(\theta)}{\partial \theta} \tag{1.10}$$

If g would be a vector, the equation 1.10 can be rewritten as shown in equation 1.11

$$\nabla_\theta J(g(\theta)) = \sum_i \frac{\partial J((g(\theta))}{\partial g_i(\theta)} \frac{\partial g_i(\theta)}{\partial \theta} \tag{1.11}$$

which sums the influences of θ on $J(g(\theta))$ through all intermediate variables $g_i(\theta)$.

1.4.2 Useful modules and methods

To make deep neural networks work and work efficiently, a lot of engineering has been introduced to the training procedure. Things like

- **Data pre-processing** - techniques such as Min-Max [30] formulated as 1.12, or Standarization 1.13

$$X' = \frac{X - min_x}{max_x - min_x} \tag{1.12}$$

$$X = (X - \mu_x)/\sigma_x \tag{1.13}$$

  where:

  - μ - mean,
  - σ-standard deviation

- **Regularization** - methods such as dropout [31], vibration [32], l1 and l2 [33], data augmentation [34]

  - **Data augmentation** - describe a set of transformations that when applied to existing data, expand it. Often used in image classification [34].

- **Dropout** - is a technique where during training some nodes are being "dropped-off" from the network and there is no computation between these nodes; see 1.10

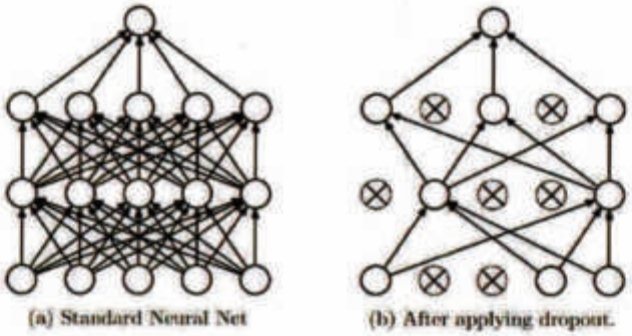


Figure 1.10: Dropout illustrated [31].

- **L1** - sometimes called *lasso regression*, adds an absolute value of magnitude of the coefficient as penalty term to the loss 1.14

$$L = L + \lambda \sum_{i=1}^{p} |w_i| \tag{1.14}$$

- **L2** - sometimes called *ridge regression*, adds a squared magnitude of coefficient as penalty term to the loss 1.15

$$L = L + \lambda \sum_{i=1}^{p} w_i^2 \tag{1.15}$$

where:

  * λ - a hyperparameter that determines how severe the penalty is,
  * w - coefficients of the system.

- **Batch normalization** - an idea that during training we normalize the output of each consecutive layer before passing it to the next layer [35]. It can be formulated as 1.16. There are several benefits from this approach:
  - Longer training, but faster convergence,
  - Allows to use higher learning rates which further speeds-up the training procedure,
  - Stabilizes the training using different weight initialization methods,
  - Simplifies the creation of deeper networks,
  - Improves generalization results.

$$X'' = \gamma * X' + \beta = \gamma * [(X - E[X])/\sqrt{\sigma}] + \beta \tag{1.16}$$

where:

- γ - scaling factor,
- β - shifting factor,
- σ - variance of the processed batch,
- $E[X]$ - mean of the processed batch.

- **Layer Normalization** - this idea is similar to batch normalization [35], but instead of normalizing the output of a layer, we normalize the layer itself [36].
- **Continuous activation functions** - nowadays they are standard, but in the early days of neural networks researchers tried to optimize neural networks with discrete, logical activation, which made it impossible to train them via backpropagation. Some of the most popular activation functions are listed below.
  - **Rectified linear unit (ReLu)** - transformation of the output of the previous layer that contains only its positive parts. It is formulated as 1.17

$$\text{ReLU}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } x > 0 \end{cases} \tag{1.17}$$

  - **Hyperbolic tangent** - transforms the input data to be in between $[-1, 1]$ range,
  - **Sigmoid** - transforms the input data to be in between $[0, 1]$ range with formula 1.18. Often used for binary classification problems,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.18}$$

  - **Softmax** - transforms the input vector into a probability distribution 1.19. It is often used in multiclass/multi-label classification problems.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{1.19}$$

- **Objective functions** - they are necessary to train neural networks or any learning system. There exist a number of different objective functions that one may want to use depending on the task at hand.
  - **Squared Error** - most commonly used in the 1980s and 1990s. Today, it is mostly used when dealing with regression problems [29]. The mathematical formulation is described as 1.20,

$$L(f_\theta(x), y) = ||f_\theta(x) - y||^2 \tag{1.20}$$

- **Cross Entropy** - when the target is a discrete label (i.e. for classification), other objective functions such as the Bernoulli negative log-likelihood have been found to be more appropriate than the squared error. In the binary case, this loss looks like 1.21

$$L(f_\theta(x), y) = -y \log f_\theta(x) - (1-y) \log(1 - f_\theta(x)) \tag{1.21}$$

- **Contrastive** - this objective function takes the output of the network for a positive example and calculates its distance to an example of the same class and contrasts that with the distance to negative examples. In other words, the loss is low if positive samples are encoded to similar (closer) representations, and negative examples are encoded to different (farther) representations. It uses the cosine distance to calculate the similarity between samples 1.22. It is formulated as 1.23

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{1.22}$$

$$L(z_i, z_j) = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \exp(sim(z_i, z_k)/\tau)_{[k \neq i]}} \tag{1.23}$$

1.4.3 Popular architectures

Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Given a sufficiently large model and a data set of labeled training samples, deep learning can be used to complete most tasks that involve quickly translating an input data into an output vector [29]. For input of varying data type (text, image), there are several architectures that can be used separately or together to solve a particular task.

- **Multilayer Perceptron** - it is a feedforward neural network made up of numerous layers of connected nodes, where each layer is in charge of handling a distinct feature of processing input data. The input layer is the first layer that receives the input data. After passing through a number of hidden layers, the input data finally reaches the output layer. MLP is an effective tool for a variety of applications because it can simulate intricate nonlinear interactions between inputs and outputs. Figure 1.11 shows a vanilla MLP architecture with a single hidden layer.

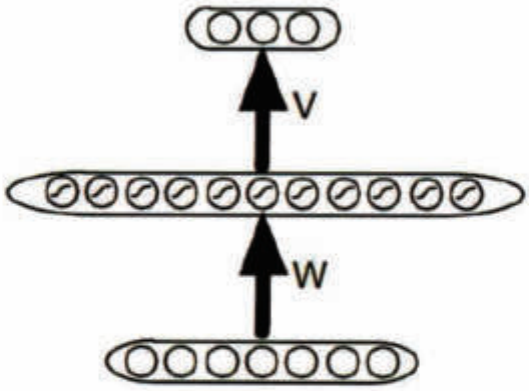


Figure 1.11: Shallow MLP with one hidden layer [29].

MLPs can learn powerful non-linear transformations: in fact, with enough hidden units, they can represent arbitrarily complex but smooth function. In fact, they can be universal approximators and represent arbitrarily complex but smooth functions if there are enough hidden units [29]. A MLP can distort the input space to make the data classes linearly separable [10].

- **Convolutional Neural Networks** (*CNNs*) - were designed to process data with a specified grid structure.They are a particular subtype of neural network. Time-series data, which may be depicted as a 1D grid capturing samples at predetermined intervals, and image data, which can be represented as a 2D grid of pixels, or a grid of voxels for 3D medical images (MRI, CT). The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution instead of general matrix multiplication at at least one of their layers [29].

  A convolution of two functions is an operation that defines an integral of the product of these functions 1.24.

$$s(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1.24}$$

The convolution operation is often denoted with the "*" (asterik operator) 1.25.

$$s(t) = (f * g)(t) \tag{1.25}$$

The discrete form for the convolution operation is described by equation 1.26

$$s[t] = (f * g)(t) = \sum_{m=-\infty}^{\infty} f[m]g[t-m] \tag{1.26}$$

The first argument to convolution, in this case the function f, is frequently referred to as the input and the second argument, in this case the function g, as the kernel

in the context of convolutional networks. The output is known as the feature map. For a two-dimensional input, the convolution operation is defined like 1.27

$$s[i,j] = \sum_m \sum_n f[m,n]g[i-m, j-n] \tag{1.27}$$

The architecture of a typical CNN (see 1.12) - is structured as a series of stages. The first stages are made up of two types of layers: convolutional layers and pooling layers [10]. The neurons of a convolutional layer are arranged in feature maps, and each neuron is connected to specific local patches in the feature maps of the layer below using a filter bank, which is a collection of weights. After that, a non-linearity function is applied to the local weighted sum's output. All neurons in a feature map share the same filter bank [10]. The role of the convolution layer is to detect local features from the previous layer. The pooling operation is used to merge semantically similar features into one feature map [10] and reduce the dimensionality of the data.

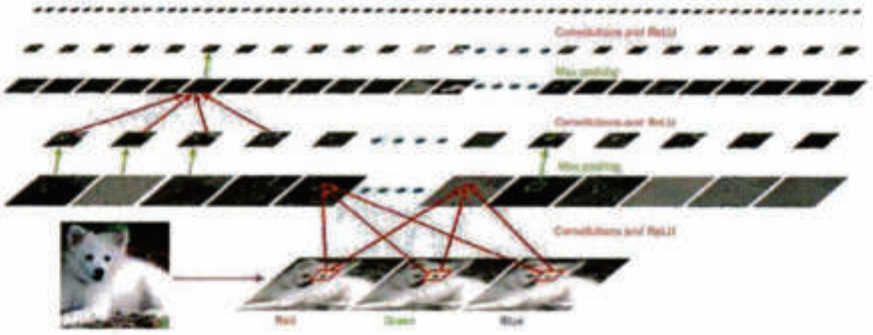


Figure 1.12: CNN feature extractor [10].

The convolutional neural network leverages three important ideas that make it an efficient learning system:

1. **Sparse interactions** - multilayer perceptron uses a matrix multiplication to describe the interaction between each input unit and each output unit. This means that every output unit interacts with every input unit. However, convolutional networks typically have sparse interactions. This is achieved by making the kernel smaller than the input [29]. As a result, we need to keep fewer parameters, which lowers the model's memory requirements and boosts its statistical effectiveness. Moreover, it means that fewer operations are needed to compute the outcome.
2. **Parameter sharing** - refers to using the same parameter for more than one function in a model [29]. In MLP each element of the weight matrix is used exactly once when computing the output of a layer. It is multiplied by one element of the input [29]. On the other hand, in convolutional neural networks, each kernel in a layer is used at every position of the input function. This means that instead of learning a separate set of parameters for every

location, we learn only one set. This feature reduces the required memory capacity for all trainable parameters.

3. **Equivariance to translation** - this property is directly related to the properties of *parameter sharing* and *sparse interactions* properties. The function f is equivariant to the function g if the changes in the input of the function f cause the same changes in the output of the function g 1.28. Taking into account the input image, if we move the object in the input, its representation will move the same amount in the output. This is useful when we know that the same local function is useful everywhere in the input.

$$f(g(x)) = g(f(x)) \tag{1.28}$$

- **Reccurent Neural Networks** (*RNNs*) - this type of neural network was specifically designed for sequence processing - where the prediction of the future depends on an arbitrary number of steps into the past. RNNs process an input sequence one element at a time, maintaining in their hidden units a *state vector* that contains information about all past elements of the sequence [10]. The RNNs suffer from the problem of *vanishing* or *exploding* gradients. Because at each time step, gradients computed for the particular neuron either shrink or grow, which makes them problematic to train on a long sequences of data. Also, because they process one input sample at a time, they are difficult to parallalize, which makes their training more expensive.

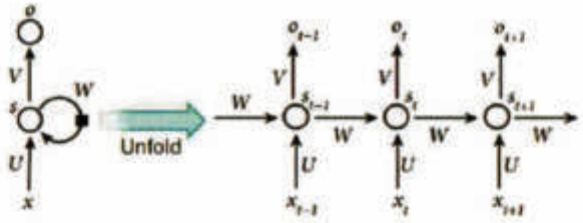


Figure 1.13: A recurrent neural network and the unfolding in time of the computation involved in its forward computation [10].

- **Transformers** - one of the most recent revolutions in the field of deep learning was the transformer architecture, introduced in the paper [37]. The transformer presented in figure 1.14 is a deep neural network that, in its core, replaced the recurrent connections with the self-attention mechanism. Self-attention allows the model to capture longer dependencies between tokens. It was introduced as the *seq2seq* model, which task was to translate between languages. That is why it was first designed as the *encoder-decoder* type model. The overall architecture is using stacked *self-attention* and *point-wise*, *fully-connected* layers for both the encoder and the decoder.

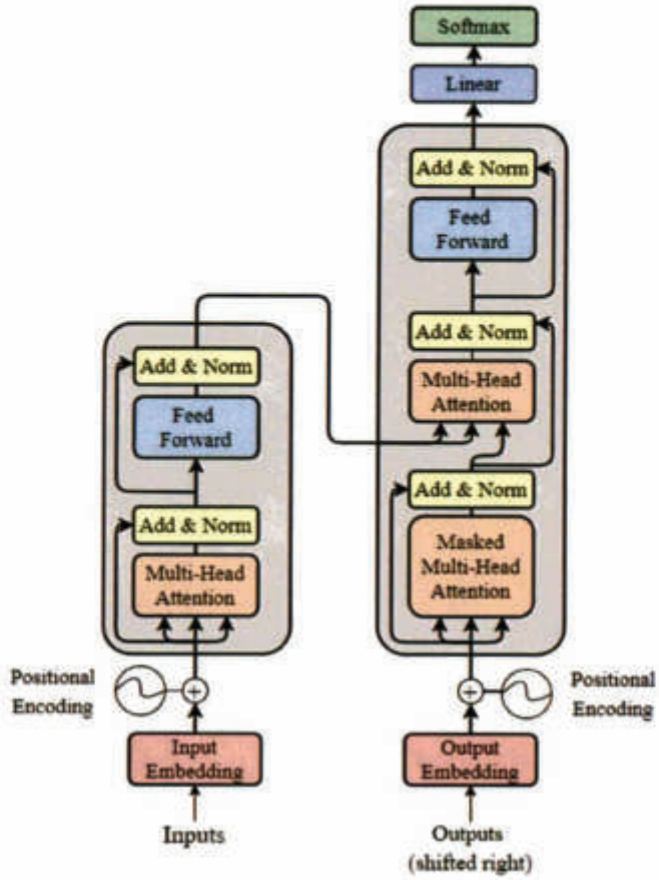


Figure 1.14: Transformer architecture [37].

The main building blocks in the transformer architecture are as follows:

1. **Positional Encoding** - because this architecture does not utilize recurrent connections and no convolution operation, in order for a model to make use of the order of the input tokens, the authors proposed injecting additional information in a form of what is called positional encoding. In the original paper, they used sine 1.29 and cosine 1.30 functions at different frequencies to denote the relative position of a token in a sequence [37].

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{1.29}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{1.30}$$

2. **Scaled Dot-Product Attention** - the attention mechanism can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The attention score is calculated as a weighted sum of the values, where the weight assigned to each value is calculated using a compatibility function of the query with the

corresponding key [37]. In the case of transformers, the authors have applied the scaling factor called d_k which refers to the number of keys used for the particular sequence. After that, they apply the *softmax* 1.19 activation function to normalize the output across all tokens in a sequence. This attention can be expressed as 1.31. The multi-head attention refers to the concatenation procedure of different attention heads.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.31}$$

3. **Position-wise MLP** - in addition to scaled dot-product attention, each of the layers contains a multilayer perceptron, which is applied to each position. It projects the vector produced by the multi-head attention module to a higher-dimensional representation, making those representations richer in information.

1.4.4 Summary

Deep learning has transformed the world in the last decade. With the advent of neural networks, which power can be utilized by aggregating exponentially growing digital information and computing capabilities, new applications have become possible.

One domain that can be revolutionized using this technology is medicine and medical sciences. Medical data are often high-dimensional information with highly nonlinear relationships. Some of the potential use cases are listed below.

- **Medical Imaging** - neural networks can be used to examine medical images such as X-rays [38], CT scans [39], and MRIs [40].
- **Medical Signal Processing** - neural networks can be used to examine biomedical signals, such as EEG signals [41].
- **Drug Discovery** - developing new drugs is an expensive and time-consuming process. Neural networks can help speed up this process by predicting the properties of the newly discovered drug or even designing them from scratch [42].

In general, deep learning has the potential to revolutionize medicine and healthcare by enhancing patient therapies, diagnosis, and care. We may anticipate seeing more deep learning applications in healthcare and medicine as new neural network models are created and more data become available.

Chapter 2

Aim of the doctoral thesis

The main goal of this thesis is to evaluate different applications in which deep learning can be helpful to the field of biomedical engineering. It also shows that while deep learning is an incredible piece of technology, it has its shortcomings, which are crucial, especially considering possible applications in the medical domain. Chapter 3 describes different techniques that have been published as possible solutions to a given problem in the biomedical engineering domain. Chapter 4 proposes a new paradigm in which some of the shortcomings are addressed. Each of the selected papers has four sections that describe:

- paper title,
- exact reference,
- paper contribution - describes the novelty of a given paper as well as possible use cases of the solution,
- author contribution - describes the involvment of the author to the given paper.

Chapter 3

Deep learning in biomedical engineering - selected papers

3.1 Predicting Molecule Toxicity Using Deep Learning

3.1.1 Citation

Exact reference: [43]

3.1.2 Paper Contribution

Predicting molecule properties is widely considered one of the most important tasks in the discovery of computational drugs and materials, as many methods rely on predicted molecular properties to evaluate, select and generate molecules. It is one of the critical tasks. With the development of deep neural networks, molecular representation learning exhibits a great advantage over feature engineering-based methods. This paper introduces a new method to approach toxicity prediction.

Novelty aspects:

- introducing the hybrid CNN-RNN architecture that allows for extracting features in both local receptive field as well as in the sequential manner,
- prevents overfitting by using *class-weighting*,
- prevents overfitting by using *mean-false-error* objective,
- prevents overfitting by using *focal-loss* objective,
- prevents overfitting by using *cyclical learning rate*,

Possible use cases:

- laboratory system which evaluates the toxicity of the produced molecules,
- laboratory system that evaluates the toxicity of the hypothetical molecules generated via another algorithm,
- extracting the knowledge learned by the model to create new hypothesis that can be validated experimentally.

3.1.3 Author's Contribution

The author conceptualized the project, surveyed the related literature, developed and tested the methodology for data pre-processing, training and evaluating a deep learning model, analyzed the results and written parts of the manuscript.

Predicting Molecule Toxicity Using Deep Learning

Konrad M. Duraj(✉) and Natalia J. Piaseczna

Department of Biosensors and Processing of Biomedical Signals, Faculty of Biomedical Engineering, Silesian University of Technology, Roosevelta 40, Zabrze 41-800, Poland
konrad.duraj@polsl.pl

Abstract. Knowledge about certain features of molecules is crucial in many fields such as drug design. Toxicity of a compound gives information on the degree to which it can be harmful to a living organism. Experimental evaluation of a molecule's toxicity requires specialised staff and equipment and generates high costs. Deep learning is a tool used in many fields and can also be helpful in the case of predicting the molecule's toxicity. For the dataset we used a free database called 'SMILES Toxicity' which is available on kaggle. This dataset consists of 6698 non-toxic molecules and 964 toxic ones. Its imbalance creates the problem called 'overfitting'. In this article, we describe several methods for overcoming the overfitting problem. We developed two models with well-balanced true positives to false positives ratio. With the first model, we achieved 96% accuracy on the training set, 89% on the validation set and 65% on the test set. The second model scored the following values of accuracy: 80% on the training set, 78% on the validation set and 77% on the test set.

Keywords: Toxicity prediction · SMILES · Deep learning

1 Introduction

We are being exposed to a large number of chemicals everyday - through our environment, food, medicine, etc. Knowledge about the properties of certain substances is crucial to insulate our bodies from exposition to dangerous factors. The degree to which a substance can harm an organism is called toxicity. Chemical experiments evaluating the toxicity of a substance can be expensive and time consuming. This evaluation is one of the most essential steps in the process of designing drugs [1,2].

Molecule toxicity prediction is an important area of research considering its potential in helping millions of people. Given the recent advances in the field of machine learning and especially its subfield called deep learning, we can now use these techniques to help researchers and doctors all around the world make

N. Piaseczna et al. (Eds.): EMBS ICS 2020, AISC 1360, pp. 18–25, 2022.
https://doi.org/10.1007/978-3-030-88976-0_3

better predictions and speed-up the chemical/drug development process [3]. By leveraging these methods we can now analyze the chemical structure of the molecules in the form of text, images or graphs [4].

For this project, we have developed a deep recurrent neural network which can distinguish between toxic molecules from non-toxic ones. The main objective of this work is to develop a deep learning model which will be able to distinguish between toxic and non-toxic molecules from their character-based representation. Our motivation is based on three particular aspects:

- compound toxicity prediction is a challenging and critical task in drug discovery,
- current methods are restricted by the availability of experimental facilities, their resources and access to trained personnel,
- those methods are high cost in terms of both money and time,
- creating a fast, explainable and accurate way to determine the molecule's characteristics has a potential to help millions of people.

2 Materials and Methods

2.1 Dataset

For the dataset, we used 'SMILES Toxicity' which is available on the popular data science website 'kaggle.com' [5]. It contains the molecules' names, their 'SMILE'-format representation and labels (if the molecule is toxic or non-toxic). The 'SMILE' strings are also provided in the form of one-hot-encoded vectors, which can be almost directly passed to a deep neural network model.

The SMILES (Simplified Molecular-Input Line-Entry System) are ASCII strings that describe the atoms, bonds, and connectivity of a compound in a way that is both precise and easy to store. Nevertheless, it lacks the molecule's 3D information, which may be crucial in certain tasks.

The dataset contains of:

- the training set: 6760 non-toxic molecules and 937 toxic molecules,
- the test set: 238 non-toxic molecules and 27 toxic molecules.

The problem with the aforementioned dataset is the imbalance between classes. This kind of disproportion in the dataset creates the neural network's problem called 'overfitting' - that means that we could not obtain a well-balanced model, but the one that predicts only 'non-toxic' category.

The aforementioned problem is quite common and there are several ways to overcome it. We can divide these methods into two categories: data-driven and algorithm-driven. Data-driven methods apply transformations to create new, synthetic data that would resemble the minority class. This process is called data augmentation. Algorithm-driven methods prevent overfitting by manipulating the neural network's loss function and optimization process, which helps to compensate for the unequal distribution of categories.

2.2 Model I

The first attempt to overcome the overfitting problem was to create a new training set in which we reduced the number of non-toxic elements to 4500. Additionally, we have created new instances of toxic molecules by copying the already existing ones in the dataset. As a result of this operation, we obtained 3240 toxic molecules in the training set. While conducting the experiments, we manipulated these proportions as well and chose the ones giving the most promising results. We also divided the original training set into training and validation sets. In that step, we have made the recurrent neural network actually trying to optimize by analyzing sequences of one-hot-encoded molecules. The test set remained the same so that we can objectively measure the performance of the developed model.

The created classifier is a deep, bidirectional, recurrent neural network which starts with an embedding layer to prepare the network for the sequential data format. It is followed by two bidirectional long short-term memory (LSTM) networks which are initialized with 64 and 32 cells, respectively. This model

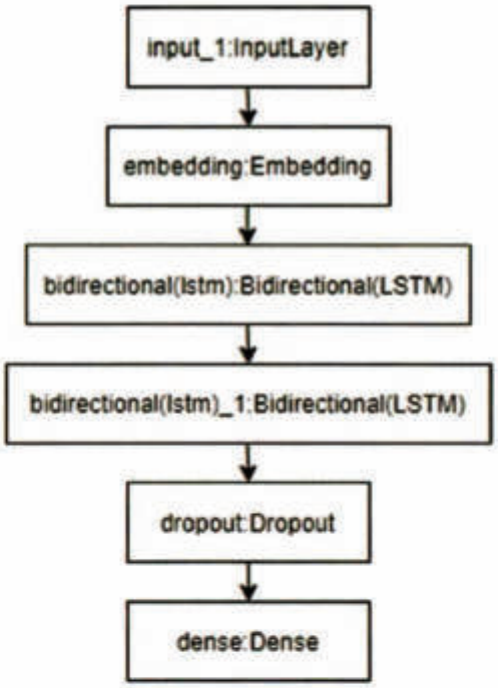


Fig. 1. Architecture of the model I

ends with a dropout layer and the final classifying layer with a single neuron activated by the sigmoid function. The architecture of the model is presented in Fig. 1.

We have decided to implement a recurrent-type network because of the character-based data type and the fact that these networks work well when analyzing time-dependent/sequential data. We have come to the conclusion that the SMILE files should also be analyzed from the beginning to the end and vice-versa because we cannot consider some part of a molecule to be the start or an end of it, that is why the recurrent units were set to learn from both these forms.

2.3 Model II

There are several methods that prevent overfitting by manipulating the neural network's loss function and optimization process. They include:

1. Loss function-based:
   - **class weighting**,
   - Mean False Error (MFE),
   - focal loss;
2. Optimization-based:
   - **cyclical/decaying learning rate**.

To create a more reliable model, we decided to apply a set of the aforementioned techniques. We chose class weighting and cyclical learning rate. These techniques help to compensate for the unequal distribution of categories.

Cyclical learning rate is a policy of learning rate adjustment that increases the learning rate of a base value in a cyclical nature. This method helps the neural network converge faster [6]. The cycles of learning rate prevents the model from getting stuck in the local minima which helps in our case to avoid the overfitting problem. Class weighting applies penalization for the misclassified examples with a certain weight. The higher the class weight, the more emphasis is put on this certain class. For our problem, we used 3.0 and 0.5 weights for toxic and non-toxic classes, respectively.

The implemented model is a hybrid of long short-term memory (LSTM) and convolution network (CNN). It contains:

- bidirectional, recurrent layer - extracting information from data in a sequential manner,
- batch normalization - regularization method,
- pooling layer - data reduction method,
- convolutional layer - extracting features in a spatial manner,
- neural layer - network's classification head.

This model was trained using the original dataset without any alterations. Total number of parameters was 329 665. The architecture of the model is presented in the Fig. 2.

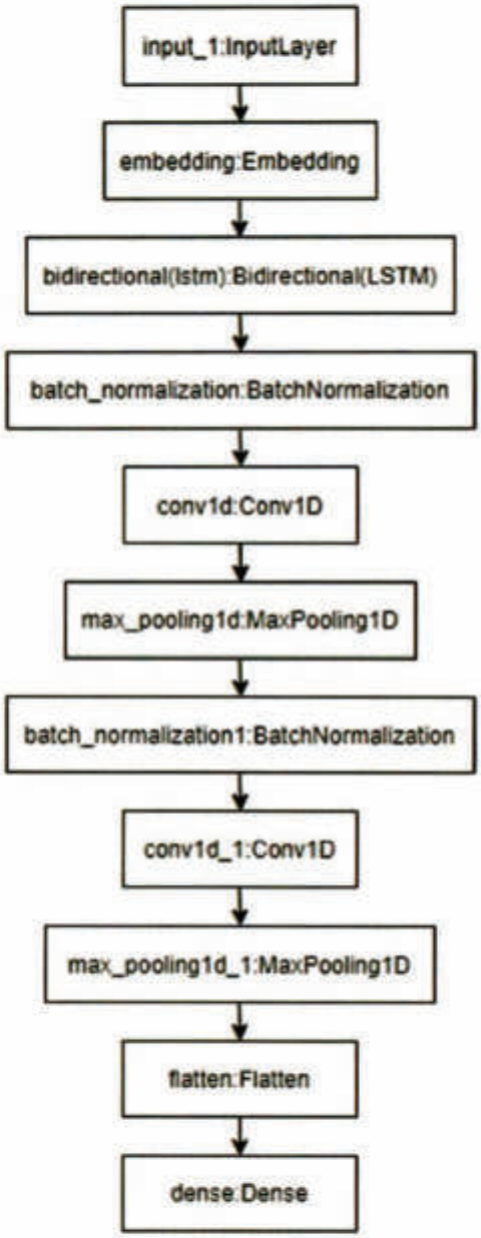


Fig. 2. Architecture of the model II

3 Results

3.1 Model I

With the first model, we have achieved 96% accuracy on the training set, 89% on the validation set (partly synthetic), and on the test set the model scored 65%. Although this model did not achieve high accuracy, it was the best-balanced one between true positives and true negatives (TP: 16; TN: 154; FP: 11; FN: 84.). Results obtained using other models (with different structure, hyper-parameters or dataset proportions) performed better on the test set (up to 81% accuracy) but they were also imbalanced between true positive and true negative predictions.

The test set contained 265 samples in which only 27 represented toxic molecules, so the model with 16 accurate identifications was almost 60% times correct. Unfortunately, this model yielded a much lower area under the ROC curve score than the second one (0.54), which is close to random prediction, which automatically disqualifies this process as the potential solution.

3.2 Model II

The model achieved 80% accuracy and 0.45 loss on the train set. On the validation set, model scored 78% accuracy and 0.49 loss. Learning curves are presented in the Fig. 3.

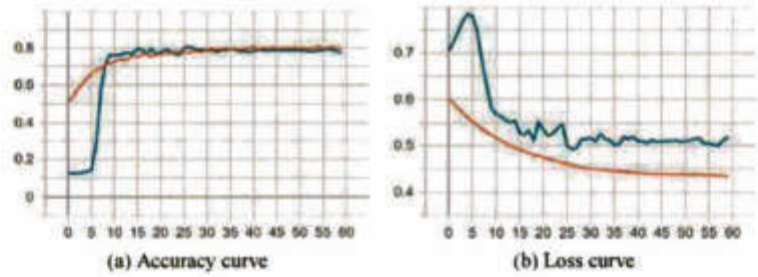


Fig. 3. Learning curves

In the Fig. 4 the metric curves are presented. Model scored:

- precision: 33% on the train set and 30% on the validation set,
- recall: 64% on the train set and 58% on the validation set,
- area under the ROC Curve: 0.8 for the train set and 0. 74 for the validation set.

The developed model was evaluated using the provided test set which contained 265 samples in total, to which 27 of them represented toxic elements and 238 non-toxic ones. Out of all the experiments, this was the model with most well-balanced true positives to true negatives ratio and most stable training process. The model yielded the following results on the test set:

- Accuracy - 77%,
- Precision - 24%,
- Recall - 52%,
- Area under ROC curve - 70%,
- Specificity - 79%,
- True positives - 14,
- True negatives - 190,
- False positives - 13,
- False negatives - 48.

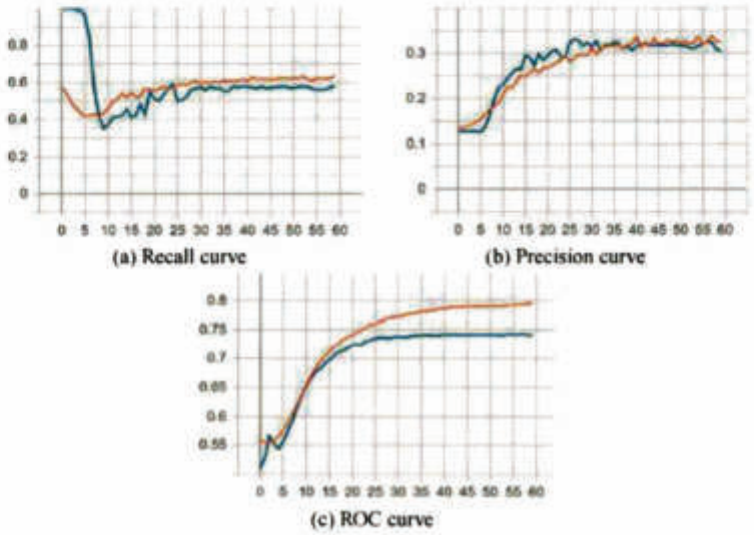


Fig. 4. Metric curves

As shown in the figures above, this solution is more robust then the first one, which can be said based on the area under ROC curve score.

4 Conclusion and Discussion

We developed deep-learning models for evaluating the toxicity of molecules based on the 'SMILES Toxicity' database. In our approach, we took into account several methods for overcoming the overfitting problem. These methods included:

- extending the original toxic molecules set by replicating its instances,
- manipulating the loss function (class weighting) and optimization process (cyclical learning rate).

With these models, we achieved achieved well-balanced true positives to true negatives ratio.

Thanks to altering the dataset, the first model scored high accuracy on the train and validation set, however, this method is insufficient. On the test set, the model scored 65% accuracy – this value tells us that we did not avoid overfitting and the learning process could still be improved.

With the second model, we used the provided dataset without any alterations and yet we managed to achieve better results on the test set. The use of class weighting and cyclical learning rate allowed us to reach a more stabilized learning process which makes the model more reliable.

The use of artificial intelligence (AI) systems becomes important in many fields, especially where there is a need of expert's knowledge. Thanks to these systems, the facilities with smaller resources could use the knowledge of an expert in the form of an application. We can now train deep-learning models using the knowledge of a specialist and create a low-cost tool which will help people with making important decisions.

In the future, we plan to combine 'SMILES Toxicity' and another dataset called 'Tox21' to perform experiments using the created models on them. This way we can achieve more balanced classes with unique instances. We also plan to apply additional techniques to the imbalanced dataset problem that will stabilize the training process and explore mechanisms such as attention and active learning to create a more robust model.

References

1. Parasuraman, S.: Toxicological screening. J. Pharmacol. Pharmacotherap. **2**(2), 74–79, (2011)
2. Chen, J., Cheong, H.-H., Siu, S.W.I.: BESTox: a convolutional neural network regression model based on binary-encoded SMILES for acute oral toxicity prediction of chemical compounds. In: Martín-Vide, C., Vega-Rodríguez, M.A., Wheeler, T. (eds.) AlCoB 2020. LNCS, vol. 12099, pp. 155–166. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42266-0_12
3. Karim, A.: Toxicity prediction by multimodal deep learning. In: Ohara, K., Bai, Q. (eds.) Knowledge Management and Acquisition for Intelligent Systems, pp. 142–152. Springer International Publishing, Cham (2019)
4. Hirohara, M., Saito, Y., Koda, Y., Sato, K., Sakakibara, Y.: Convolutional neural network based on smiles representation of compounds for detecting chemical motif. In: Proceedings of the 29th International Conference on Genome Informatics (GIW 2018): Bioinformatics, vol. 19. BMC Bioinformatics (2018)
5. Fanconi, C.: Smiles toxicity, August 2019. https://www.kaggle.com/fanconic/smiles-toxicity
6. Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472 (2017)

3.2 Recognition of Drivers' Activity Based on 1D Convolutional Neural Network

3.2.1 Citation

Exact reference: [44]

3.2.2 Paper Contribution

Driving a car is a complex activity that involves movements of the whole body. Many studies on driver behavior are conducted to improve road traffic safety. This paper attempts to develop a classifier of scenarios related to learning to drive based on the data obtained in real road traffic conditions through smart glasses.

Novelty aspects:

- constructing first (to our knowledge) data set for driver action recognition based on smart-glasses,
- developing a classifier that can accurately recognize driver's action based solely on EOG signal.

Possible use cases:

- additional safety system for young and inexperienced drivers,
- automatic assessment of a correctly performed maneuver.

3.2.3 Author's Contribution

The author was responsible for the pre-processing of the data set, developing a methodology, including model development, training and evaluation, and visualization of the results. Parts of the manuscript concerning those elements were also written by the author.

electronics

MDPI

Article

Recognition of Drivers' Activity Based on 1D Convolutional Neural Network

Rafał J. Doniec [1], Szymon Sieciński [1,*], Konrad M. Duraj [1], Natalia J. Piaseczna [1], Katarzyna Mocny-Pachońska [2] and Ewaryst J. Tkacz [1]

[1] Department of Biosensors and Processing of Biomedical Signals, Faculty of Biomedical Engineering, Silesian University of Technology, Roosevelta 40, 41-800 Zabrze, Poland; rafal.doniec@polsl.pl (R.J.D.); konrad.duraj@polsl.pl (K.M.D.); natalia.piaseczna@polsl.pl (N.J.P.); etkacz@polsl.pl (E.J.T.)

[2] Department of Conservative Dentistry with Endodontics, Faculty of Medical Science, Medical University of Silesia, Pl. Akademicki 17, 41-902 Bytom, Poland; kpachonska@sum.edu.pl

* Correspondence: szymon.siecinski@polsl.pl

Received: 3 November 2020; Accepted: 21 November 2020; Published: 25 November 2020

Abstract: Background and objective: Driving a car is a complex activity which involves movements of the whole body. Many studies on drivers' behavior are conducted to improve road traffic safety. Such studies involve the registration and processing of multiple signals, such as electroencephalography (EEG), electrooculography (EOG) and the images of the driver's face. In our research, we attempt to develop a classifier of scenarios related to learning to drive based on the data obtained in real road traffic conditions via smart glasses. In our approach, we try to minimize the number of signals which can be used to recognize the activities performed while driving a car. Material and methods: We attempt to evaluate the drivers' activities using both electrooculography (EOG) and a deep learning approach. To acquire data we used JINS MEME smart glasses furnished with 3-point EOG electrodes, 3-axial accelerometer and 3-axial gyroscope. Sensor data were acquired on 20 drivers (ten experienced and ten learner drivers) on the same 28.7 km route under real road conditions in southern Poland. The drivers performed several tasks while wearing the smart glasses and the tasks were linked to the signal during the drive. For the recognition of four activities (parking, driving through a roundabout, city traffic and driving through an intersection), we used one-dimensional convolutional neural network (1D CNN). Results: The maximum accuracy was 95.6% on validation set and 99.8% on training set. The results prove that the model based on 1D CNN can classify the actions performed by drivers accurately. Conclusions: We have proved the feasibility of recognizing drivers' activity based solely on EOG data, regardless of the driving experience and style. Our findings may be useful in the objective assessment of driving skills and thus, improving driving safety.

Keywords: activity recognition; car driving; classification; electrooculography; convolutional neural network; smart glasses; wearable devices

1. Introduction

Driving a car is a complex activity which involves movements of the whole body [1]. The decisions and behavior of drivers regarding the surrounding traffic are crucial for road safety [2]. The factors which affect the road traffic safety can be divided into two categories: the environmental factors and the state of the driver. The environmental factors include weather and road conditions. We define the state

of the driver as driver's alertness, concentration (focus), cognitive abilities and the fact of performing secondary tasks.

To solve the problem of specifying driver's activities, we apply recognition based on tracking eye movements, as most human activities require eyeball movements [3–5]. The analysis of eye movements may help understand the reasons and determine the beginning and the end of activity. However, most eye movements are involuntary and remain out of conscious control [6].

The mainstream method for tracking eye movements in human behavior research is analyzing images registered with a camera [7], because of its advantages: small individual differences, and providing a non-contact of eye measurements. The drawbacks of using a camera to track eye movements are the trade-off between processing time and detection accuracy and the susceptibility to lighting conditions, skin color and sunglasses [8]. Moreover, cameras mounted in vehicles cannot be used to detect the state of a driver outside the vehicle [9]. An alternative method is electrooculography (EOG), a technique for measuring the resting electrical potential between the cornea and retina of the human eye [10]. The EOG signal is registered by electrodes placed around the eyes. The extracted EOG signal is then processed in order to detect the eyeball movements [4,5,11–13].

The reason for choosing EOG to detect eyeball movements was the availability of JINS MEME ES_R smart glasses which can register electrooculograms in a non-invasive way (without attaching the electrodes to the body) while performing various activities (including driving a car) [9,14] and the fact that the eyeball movements contain the most information related to activities related to driving a car [2,4,15]. Another reason for using only electrooculography was to design the system for recognizing drivers' activities regardless of their style of driving and experience and to find the minimal number of attributes required to recognize such activities [16].

To the best of our knowledge, the problem of extracting and selecting appropriate features from electrooculograms in recognition of drivers' behavior has not been thoroughly investigated due to the cumbersome placement of electrodes and the breadth of the topic, which indicates a clear need for further research. The studies conducted by Niwa et al. [9] and Doniec et al. [17] are the only known study on drivers' behavior which used JINS MEME smart glasses. Another recent study on recognizing the gaze on the left turn was conducted by Stapel et al. [18].

Because the average accuracy of the classifier based on k-means clustering and BFS reported in [17] was 85% when analyzing four activities related to driving (driving on the motorway, parking, urban traffic, traffic in the neighborhood), we attempted to improve the accuracy by changing the approach to classification.

In recent years, deep learning techniques, such as recurrent neural network (RNN) [19,20], convolutional neural network (CNN) [7,19], generative adversarial network (GAN) [21], long short-term memory (LSTM) network [20,21], have found their use in classifying the state of the driver based on various signals, such as electroencephalography (EEG) [19,20,22,23], images [20] and EOG [21].

In our study, we apply a one-dimensional (1D) convolutional neural network (CNN) to perform classification on raw EOG signals without crafting features prior to classification [24]. Another advantage of using 1D CNN is the ability to retrain the model on new data sets by using transfer learning [25].

The purpose of this study was to examine whether it is possible to classify drivers' activities in real road conditions based on raw EOG signals and 1D CNN. The performance of the 1D CNN built for classification was evaluated as precision, recall and F_1-score.

The structure is as follows: material and methods are described in Section 2, which includes the experiment setup, data preprocessing and classification. The results in Section 3 consist of the loss and accuracy graphs, the confusion matrix, and receiver operating characteristic (ROC) curves of the proposed 1D CNN model. Finally, we state that, based on the ROC curve and other performance metrics presented in

the confusion matrix, the model was trained without overfitting and underfitting. In Section 4, we conclude the paper by discussing the significance of the results and advantages and limitations of our approach.

2. Materials and Methods

2.1. Experiment Setup

The study was conducted in real road conditions in accordance with Chapter 4 of the Act on Vehicle Drivers of the Republic of Poland [26] on two groups of volunteers: ten experienced drivers (age between 40 and 68) with minimum ten years of driving experience and ten learner drivers who attended driving lessons at a local driving school (age between 18 and 46). The participants gave consent to participate in the study. The candidates for drivers made a statement on their health in a questionnaire submitted to the driving school. Although the candidates for drivers may not reveal the actual health status in the questionnaires, we assumed that the study group did not have any health conditions which may cause a direct driving hazard.

In Poland, the drivers and candidates for drivers are subject to medical examination based on Article 39j of the Act on Road Transportation of the Republic of Poland [27] and Chapter 2 of the Act on Vehicle Drivers of the Republic of Poland [26]. The medical examinations of drivers include the examination of vision, hearing, and balance, the state of cardiovascular and respiratory system, kidneys, nervous system, including epilepsy, obstructive sleep apnea, mental health, symptoms of alcohol abuse, the use of drugs that may affect the ability to drive and other health conditions that may cause a driving hazard.

To avoid distracting the driver during the field study, data were acquired with JINS MEME ES_R smart glasses. The device consists of a three-point electrooculography (EOG) and six-axis inertial measurement unit (IMU) with a gyroscope and an accelerometer. JINS MEME smart glasses acquire ten channels of data: the acceleration and rotation in X, Y and Z axes, and four EOG channels: electric potentials on the right and left electrodes and the vertical and horizontal difference between them. All signals are sampled with the frequency of 100 Hz. The data are transmitted to a computer via Bluetooth or USB and can be exported to CSV file. Electrooculograms were recorded with three-point EOG sensor which consist of left, right and bridge electrodes and were converted into four lead (channel) recording of EOG signal: left, right, horizontal and vertical [14].

Each participant had to perform the same set of tasks while wearing the smart glasses. Sensor data were acquired and linked to scenarios related to driving by the observer sitting on a back seat (see Figure 1). Each learner driver (learner) drove the same car adapted for driving lessons and marked with the L sign. The learners drove the car under the supervision of a driving instructor, whereas other drivers drove their own cars. Sensor data from learner drivers were obtained thanks to the cooperation with a local driving school.

All participants completed their tasks on the same route of 28.7 km in Tarnowskie Góry, Radzionków, Bytom, and Piekary Śląskie in southern Poland presented in Figure 2. The tasks to be completed during the drive were based on the regulations on practical driving tests [28] and included:

- journey through a motorway;
- drive straight ahead in city traffic;
- passage of a section straight ahead outside of the urban area;
- drive straight ahead in residential traffic;
- driving through a roundabout (right turn, driving straight ahead and left turn);
- driving through a crossroads (right turn, driving straight ahead and left turn);
- parking (parallel, perpendicular, angled).

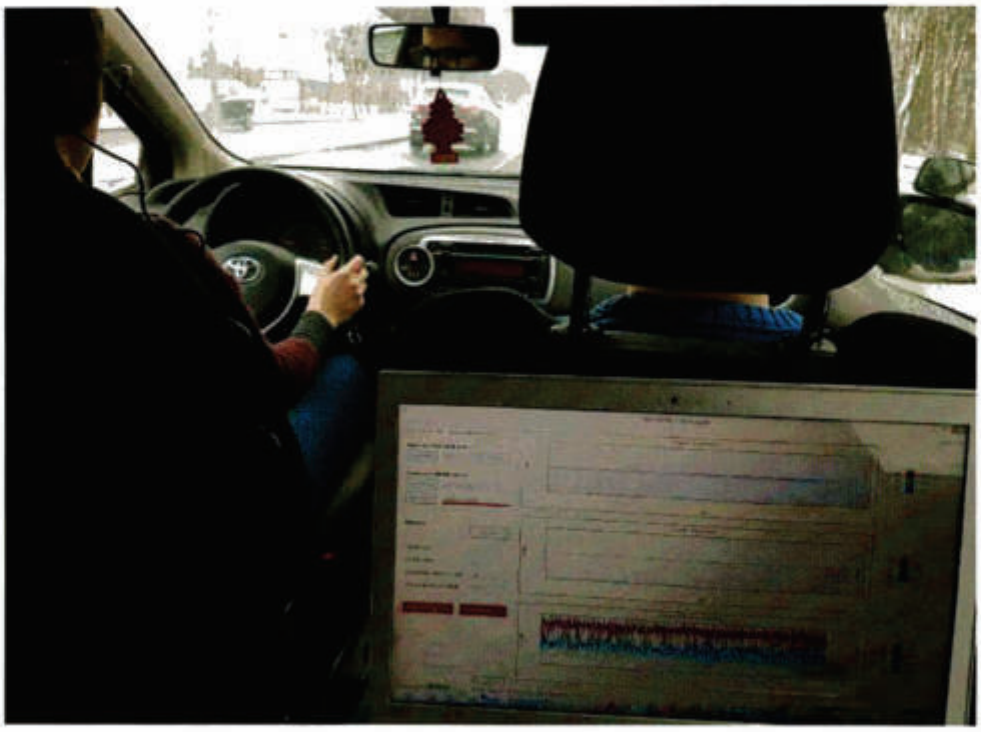


Figure 1. Experiment setup. The driver (left) is wearing JINS smart glasses connected to the computer (foreground) while recording the signals.

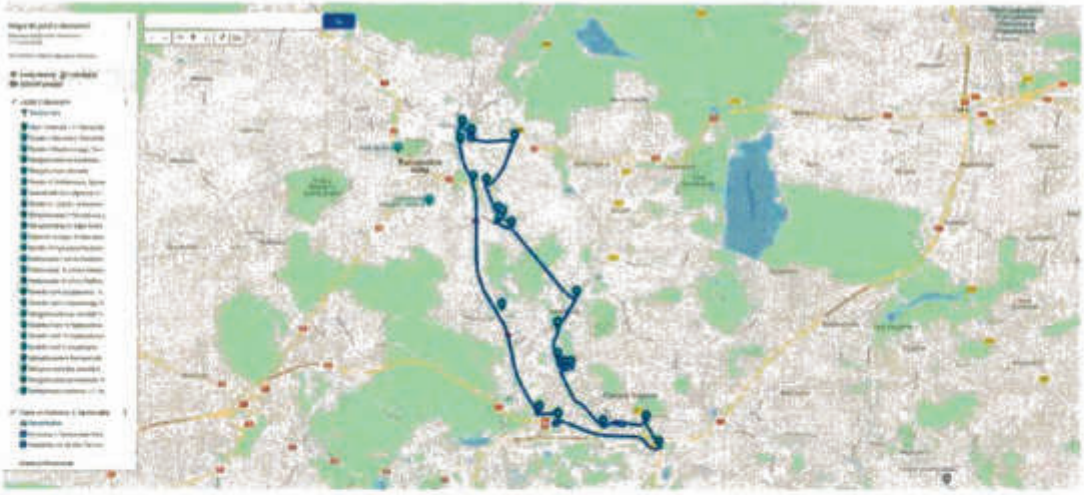
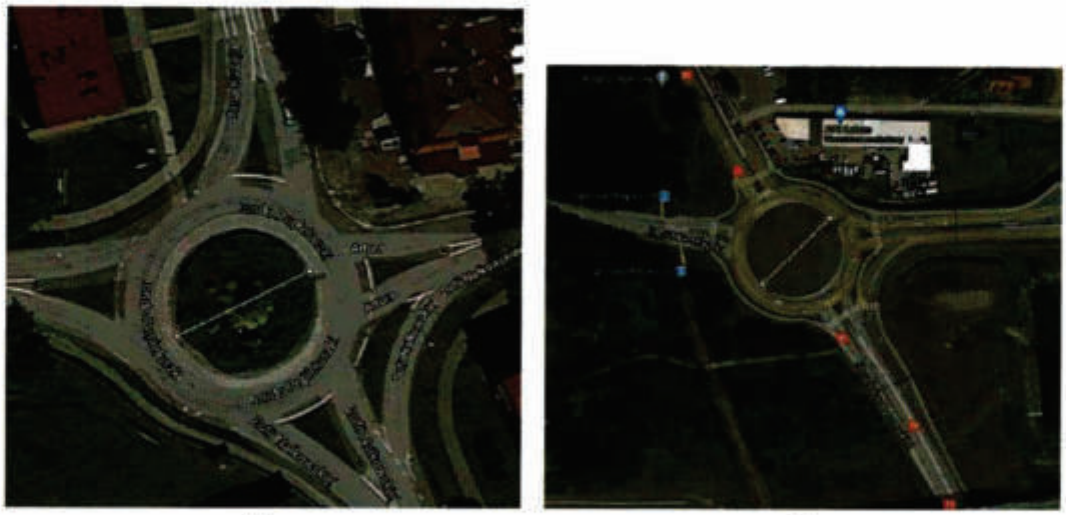


Figure 2. The test route chosen for the study (map data: Google).

The route included roundabouts and parking lots shown as satellite images in Figures 3 and 4. Figure 3 presents the bird's eye view of two roundabouts (small and large) and Figure 4 presents the bird's eye view of public parking spaces with no ticketing along the Artura street in Radzionków (Poland).



(a) (b)

Figure 3. Bird's eye view of two roundabouts: (**a**) small roundabout (the diameter of circle island: 20 m); (**b**) large roundabout (the diameter of circular island: 54 m). Map and satellite images: Google, CNES, Airbus, Maxar Technologies, 2020.

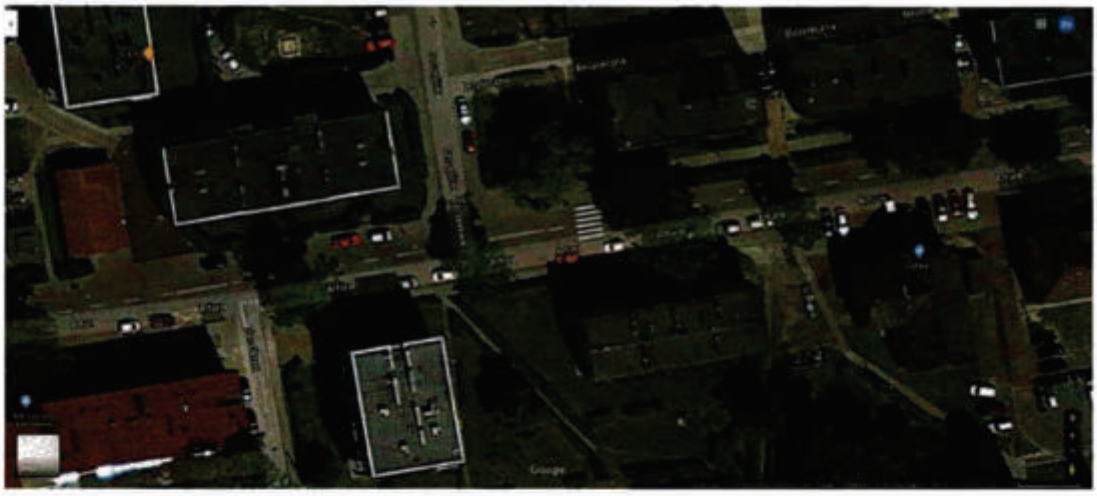


Figure 4. Bird's eye view of the public parking lots at the Artura Street in Radzionków, Poland (map and satellite image: Google, CNES, Airbus, Maxar Technologies, 2020).

Each activity was labeled manually by the researcher during the drive. A pilot (driving instructor in case of learner drivers and in other cases—the researcher) asked the driver to start performing a particular activity, and at the same time, recording began. When the task was completed, the pilot asked to stop the recording. The file with recorded data was named according to the registered activity.

The average time of completing all the tasks during the experiment was 75 min. Experienced drivers generally completed the route faster than learner drivers, regardless of road conditions [17].

The experiment was carried out following the rules of the Declaration of Helsinki of 1975, revised 2013, and with the permit issued by the Provincial Police Department in Katowice. The participants

gave their informed consent for inclusion in the study. Study protocol was approved on 16 October 2018 by the Bioethics Committee of the Medical University of Silesia in Katowice (the resolution number KNW/0022/KB1/18). The identity of learner drivers is confidential under the agreement with the driving school, and the same rule applies for the experienced drivers. Experimental data as Supplementary Materials were made publicly available at IEEE DataPort [29].

2.2. Data Preprocessing

The data set consists of 520 labeled recordings of electrooculograms, acceleration and gyration signals from both experienced and inexperienced drivers acquired with JINS_MEME ES_R smart glasses and is available at the IEEE DataPort [29]. The reason for analyzing the recordings acquired form both experienced and inexperienced drivers is based on the fact that there are no significant differences in overall cognitive and motor skills [30].

The recordings were divided into four scenarios (categories): parking, driving through a roundabout, driving in city traffic and driving through an intersection chosen based on the classification accuracy reported in [17]. By considering only EOG signals, we can distinguish specific patterns associated with analyzed activities, regardless of style of driving, driving dynamics and experience visible in acceleration and gyration.

The recordings were divided into each category as follows:

- parking: 120 recordings,
- driving through a roundabout: 120 recordings,
- driving in city traffic: 160 recordings,
- driving through an intersection: 120 recordings.

Each of these activities can be further divided into categories described in the Section 2.1. We focused on recognizing 4 activities to verify the feasibility of classification based on 1D CNN.

The data were preprocessed before classification in Python 3.7.5 with Pandas, NumPy, Matplotlib and Scikit-learn. The first step was to unify the length of the signals because the length of the original signal vector varies from 320 to 32,357 samples (considering all of the categories). In order to unify the input signals, we took the maximum length value across all the recordings and tiled the shorter vectors to match that value. This approach turned out to be the easiest and the fastest way to standardize input data without losing the inherent characteristics of the signal for each category.

The second step of preprocessing was normalizing the input values to prevent the occurrence of the exploding/vanishing gradient problem [31]. After normalization, the data were divided into training and validation sets. The best performance was achieved with validation split equal to 20%. The number of samples in both sets and for each category was shown in the Figures 5 and 6.

2.3. Classification

In this study, we propose an approach to driver activity recognition using a one-dimensional convolutional neural network (1D CNN). This neural network model has proven its effectiveness in signal classification, yielding state-of-the-art results [32,33]. Because biological signals have non-linear characteristics, convolutional neural networks are an adequate choice, as they are precisely developed for recognizing non-linear patterns in the data [34]. Considering that there have not been established patterns in EOG signals related to driver's activity, we applied 1D CNN due to the ability of automatic extraction of features. By using convolutional layers, we can also visualize the set of filters after training and try to learn which characteristics of the input signal are related to certain activity.

The architecture of the proposed model is shown in Figure 7.

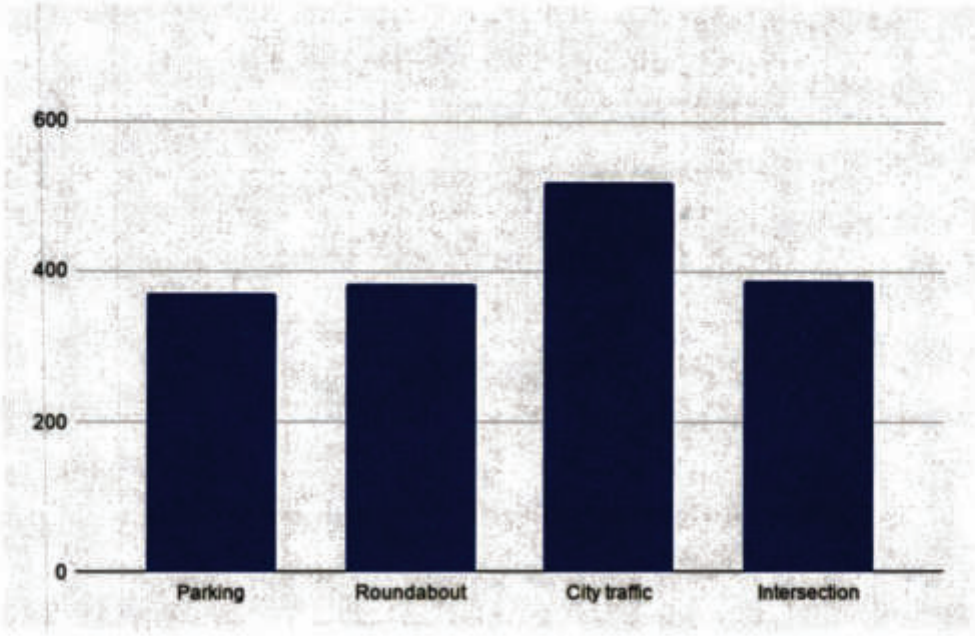


Figure 5. Number of samples in each category in training set.

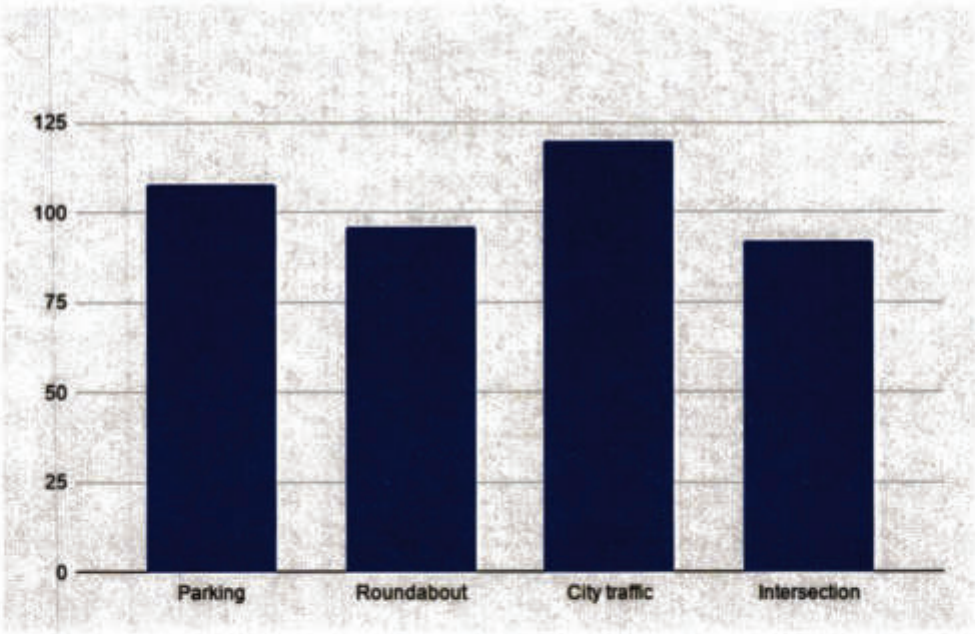


Figure 6. Number of samples in each category in validation set.

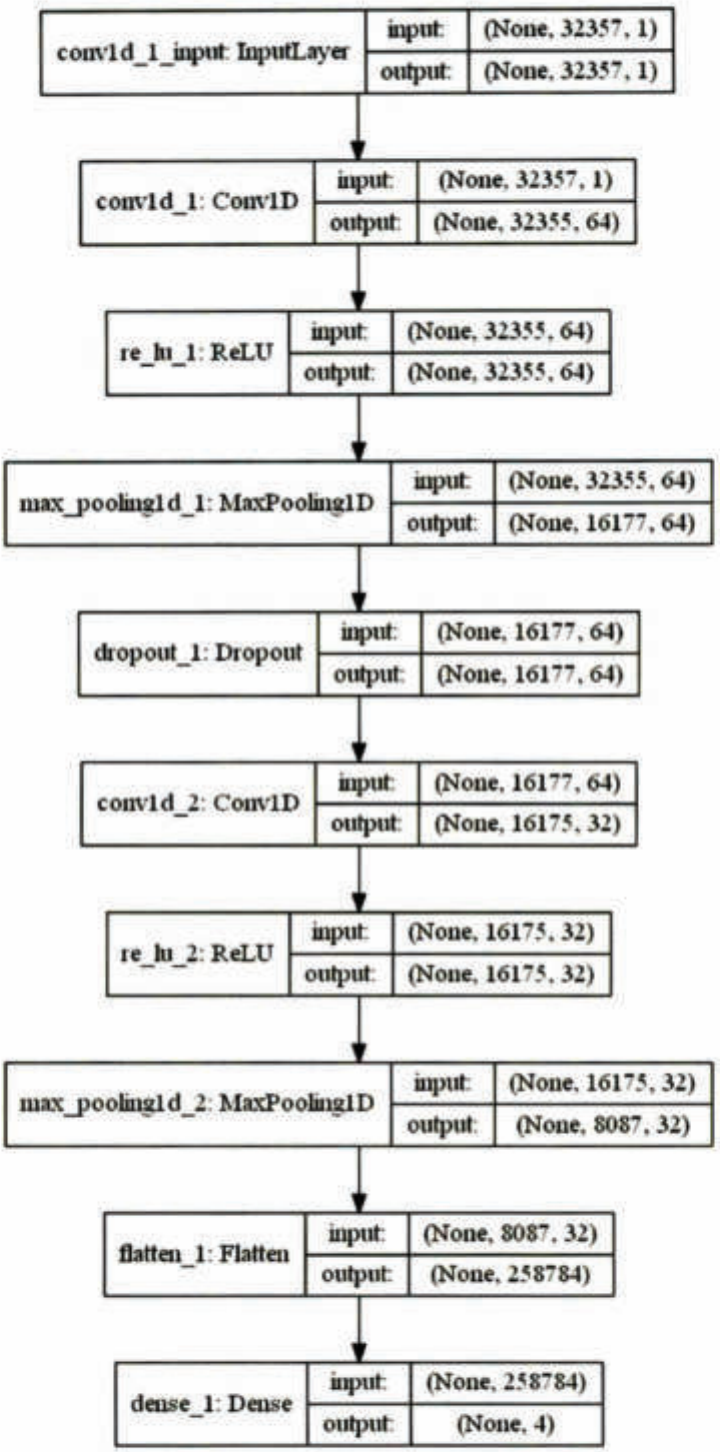


Figure 7. Architecture of the proposed 1D convolutional neural network (CNN) model generated by Keras and graphviz. `None` is the variable batch size.

The architecture consists of the following elements:

- Convolution with max pooling block—first convolution layer produces 64 feature maps which are then processed by activation function in order to capture non-linear patterns and followed by pooling layer with kernel size of two to reduce the extracted information. The second convolution layer generates 32 feature maps with kernel of size three (as in the first block), followed again by rectified linear unit (ReLU) and pooling layer [35]. Although the kernel size of convolutional layer may be much higher in the case of 1D CNN than in their two-dimensional (2D) counterpart, the best results were achieved with the smaller kernel.
- Dropout layer—the dropout layer was set with the rate of 0.5. This layer turned out to be the key element because it prevents overfitting at the beginning of the training phase [36].
- Dense and flatten blocks—after obtaining the data from the second convolutional pooling block, the feature maps are mapped into their one-dimensional representation and classified with the single and final layer consisting of four neurons followed by softmax probability activation function.

The conducted training process lasted 100 epochs, the batch size was 20 and the decaying learning rate was 0.001 at the start. The 1D CNN model was developed in Python 3.7.5 using Keras library with Tensorflow version 2.1 as the backend. To accelerate the computation, we set up the GPU (graphics processing unit) support with Nvidia CUDA (Compute Unified Device Architecture) version 10.1.

Data preprocessing and classification of 520 labeled recordings was run on the Nvidia GTX 1060 with 6 GB of VRAM (Video RAM) and training the proposed model for 100 epochs took circa 10–15 min. The source code of classifier was made publicly available at IEEE DataPort [29].

3. Results

This section provides and describes the results of classification of four analyzed driving scenarios registered in 520 labeled recordings using 1D CNN.

The accuracy on the training set was 99.8% on and 95.6% on the validation set. The performance of the training process was presented as the learning curves and the decaying learning rate curve on Figures 8–10.

The accuracy curve shows the correctness of the model's performance among the epochs. Both train and test accuracy achieved high values (above 90%) after circa 40 epochs.

The loss function is the sum of errors made after each epoch. After circa 40 iterations, loss stabilizes (circa 0 for training set and below 0.2 for validation set).

Figure 10 shows how the learning rate was changed among iterations. In this case, the learning rate was halved after five epochs if the validation loss did not decrease.

The performance of the proposed classifier is presented in the form of confusion matrix (Figure 11) and receiver operating characteristic (ROC) curves for each activity (see Figure 12).

The confusion matrix presents the numbers of cases classified to a specific group (predicted label) in comparison with their real classification (true label). The correctness of the classification is as follows:

- for parking—ten out of 108 signals were classified incorrectly (two as driving through a roundabout, four as driving in city traffic and four as driving through an intersection);
- for driving through a roundabout—two out of 96 signals were classified incorrectly (one as parking and one as driving in city traffic);
- for driving in city traffic—three out of 120 signals were classified incorrectly (as one of each group);
- for driving through an intersection—one out of 92 signals was classified incorrectly (as driving in city traffic).

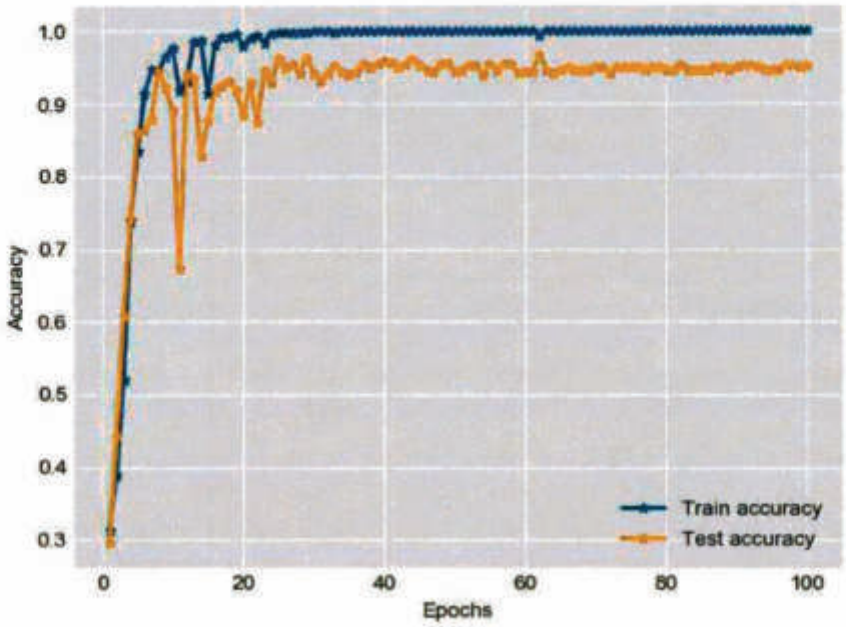


Figure 8. Accuracy of the proposed 1D CNN model.

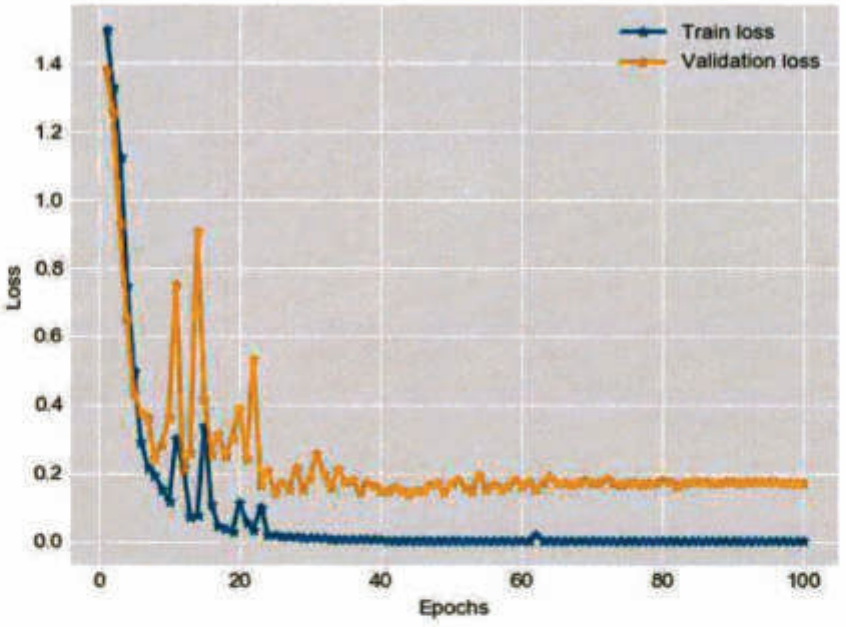


Figure 9. Loss of the proposed 1D CNN model.

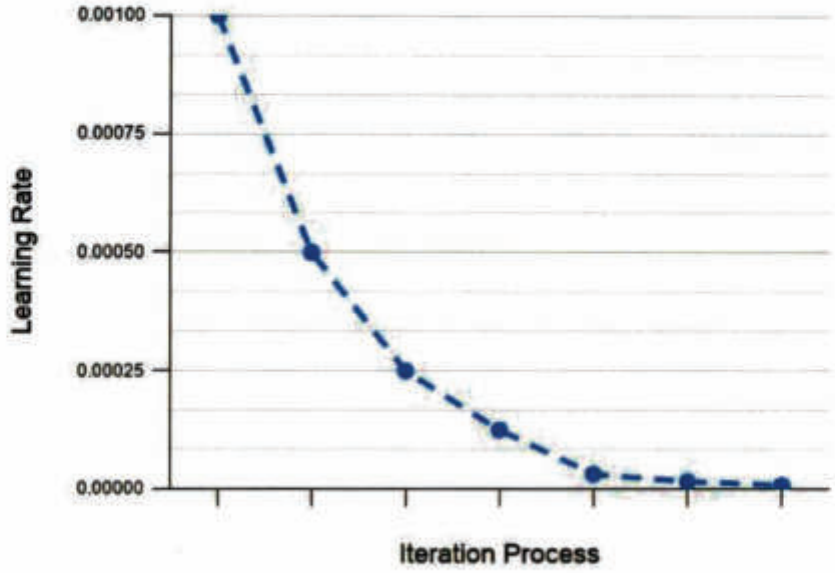


Figure 10. Learning rate decay.

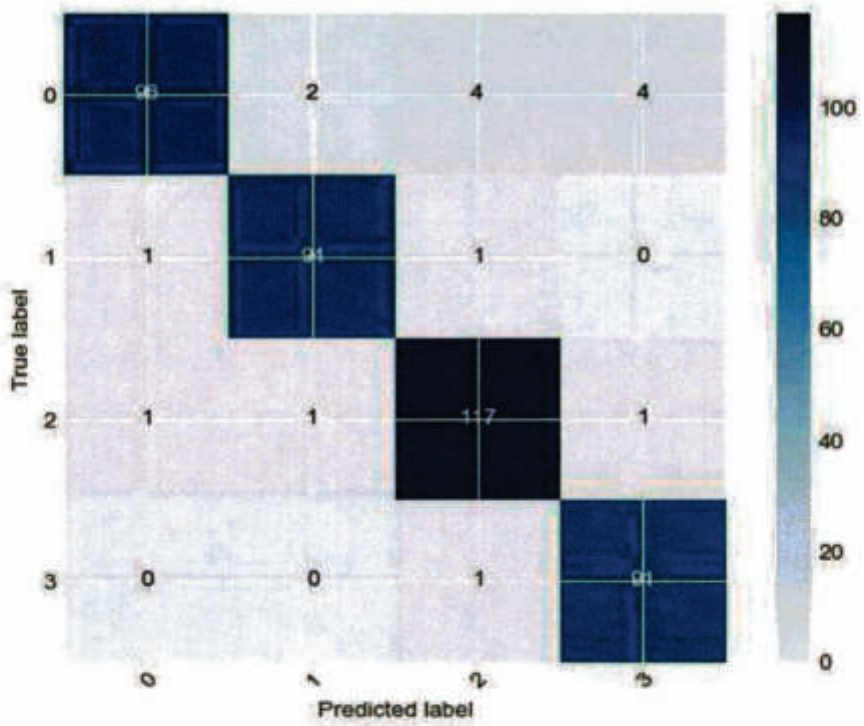


Figure 11. Confusion matrix of the classifier.

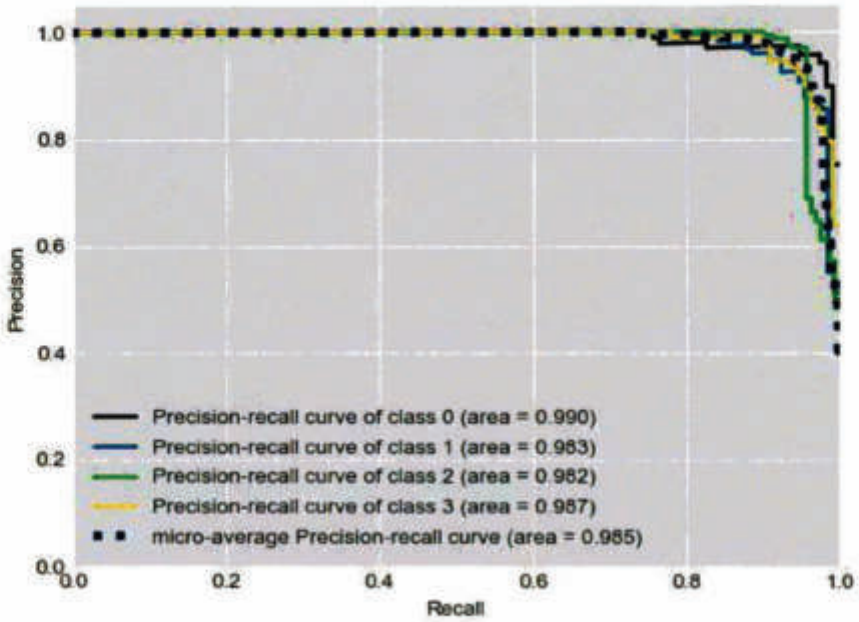


Figure 12. Precision recall curves of the classifier.

Based on the confusion matrix, the following parameters (adapted for multi-category classification) were calculated:

- Precision is the proportion of positive samples out of the retrieved samples (true positive and false positive).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

- Recall (sensitivity) measures how accurate is the model within all the positive samples.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

- F_1 score—combination of the two aforementioned metrics which rises when both precision and recall increases.

$$F_{1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

The calculated values of aforementioned metrics are presented in the Table 1.

Table 1. The performance of the classification of four driving scenarios.

Category	Precision	Recall	F_1-Score
0—parking	0.98	0.907	0.942
1—roundabout	0.97	0.98	0.97
2—city traffic	0.95	0.975	0.96
3—intersection	0.95	0.98	0.968

The highest precision was obtained for parking (0.98), the highest recall was obtained for driving through a roundabout and driving through and intersection (0.98 in both cases) and the highest F_1-score

for driving through a roundabout. The lowest precision was obtained for city traffic, and the lowest recall and F_1-score was obtained for parking.

The ROC curve measures the capability of distinguishing given classes by the analyzed model and is presented as true positive rate to false positive rate. With the multi-class problem, we measure this ratio for each single category against all the other categories [37]. The receiver operator characteristic curve shows that the created model was well trained, without overfitting and underfitting. It also shows that there are no unbalanced ratios between true positives and false positives, which again leads to the conclusion that the model was not biased.

Based on the confusion matrix and ROC curve, we can observe that most of the misclassified samples belong to the parking activity (see Figure 11). The reason is the fact that parking can be performed in different ways (angle, perpendicular, parallel) and is also associated with frequent movement of head and eyeballs reflected in the EOG signal. However, the EOG signals linked to parking may resemble the other activities, such as driving on a roundabout.

4. Discussion

We built the classifier of driver's activity based on 1D CNN. Its accuracy was 99.8% on the training set and 95.6% on the validation set. The accuracy of our classifier is higher than in other studies, especially:

- Doniec et al.'s study, which used BFS approach with 2-fold cross validation on seven activities (62% driving on a highway, parking in front, parallel parking, slope parking, driving around a roundabout, driving in city traffic, driving in residential traffic) and four activities (85% for driving on a freeway, city traffic, parking, driving in a residential area) [17];
- Jiang et al. study which used k-nearest neighbors (kNN) and SVM classifying approach on signals from wearable devices (90%) [38];
- Vora et al.'s study which used driver's eye tracking in video recording and CNN-based classifier (95.2%) [25];
- Galarza et al.'s study, which used a video recording of the driver's face, statistical model and Google API-based classifier (93.37%) [39];
- Mulhall et al.'s study, which used binary logistic regression for recognition of lane departures (73%) and microsleeps (96%) during driving in real road conditions [40].

This classifier has proven its high accuracy in classifying four driving scenarios (parking, driving through a roundabout, driving in city traffic and driving through an intersection). Its inherent ability to capture non-linear patterns in the sensor data makes 1D CNN a powerful tool in processing biologically related signals. The main drawback of this approach is the fact that it needs a fixed size input. The performance of 1D CNN deep learning model is at least 14%, better than BFS approach with soft assignment to specific configurations for the same data set. The results obtained for both data sets (training and validation sets) emphasize two points: the superiority of automatically learned functions over manually created ones used in [17], and the stability of 1D CNN deep learning architectures.

The type of dominant features fed to the classifier in [17] depend on the size of the sliding window and the BFS entropy of the extracted data frames. Moreover, the 1D CNN model is among the most efficient methods, which underlines the stability of this model and suggests its good ability to generalize on various data sets [41], including medical data [33]. The accuracy of studies on the drivers' behavior was based on monitoring one or two signals; the accuracy ranged from 60% to 80% [42].

In this study, we have proven the feasibility of driver's activity recognition based solely on EOG data, regardless of his/her experience and style of driving, which can be determined based on accelerometer and gyroscope data. Therefore, this approach may be applied to numerous real-world scenarios, such as building a system that may help improve the driving skills and driving safety, especially in smart vehicles.

Due to the increasing number of vehicles on roads, changing the paradigm of the driver training process is necessary to prevent the growth of road accidents and fatalities. The opportunity to measure the drivers' perception can provide valuable insight into drivers' attention. Accurate and inexpensive driver assistant systems may help encourage safe driving. However, real-time monitoring of behavior and driving conditions imposes technical challenges and the need for monitoring the state of the driver, especially dizziness caused by long trips, extreme changes in lighting, reflections of the glasses or the weather conditions on the road.

In future, we will address the limitations of our study: the need of providing fixed-sized input signals and misclassification of some recordings linked to parking activity. We propose developing a variable-size model which will operate on global pooling instead of a flatten layer to overcome the first limitation. With that approach, we may propose a new method of determining the length of the input vector based on the information from additional sensors. To overcome the second limitation, we propose differentiating variants of parking and providing more robust architecture. Although we achieved satisfactory results with 1D CNN, we consider ensemble classifiers or multi-input deep learning models for recognizing a wider range of activities in the long term.

Supplementary Materials: The research data are available online at IEEE DataPort: https://dx.doi.org/10.21227/q163-w472.

Author Contributions: Conceptualization: R.J.D. and S.S.; Investigation: R.J.D. and K.M.-P.; Methodology: R.J.D., S.S. and K.M.D.; Writing—Original Draft Preparation: R.J.D., S.S. and N.J.P.; Writing—Review & Editing: R.J.D., S.S., K.M.D., N.J.P. and E.J.T.; Validation: K.M.D.; Visualization, K.M.D.; Supervision, E.J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would like to thank the volunteers who participated in the study and the driving school for providing the opportunity to acquire signals on learner drivers. We also would like to thank the reviewers for useful comments.

Conflicts of Interest: The authors declare no conflict of interest.

Ethical Statements: The study was conducted in accordance with the Declaration of Helsinki, and the protocol was approved by the Bioethics Committee of the Medical University of Silesia on 16 October 2018 (KNW/0022/KB1/18).

Abbreviations

The following abbreviations are used in this manuscript:

1D	One-dimensional
2D	Two-dimensional
BFS	Best fit sequence[s]
CNN	Convolutional neural network
CUDA	Compute unified device architecture
ECG	Electrocardiogram
EEG	Electroencephalogram
EOG	Electrooculography
GAN	Generative adversarial network
GPU	Graphics processing unit
IMU	Inertial measurement unit
kNN	K-nearest neighbors
LSTM	Long short-term memory
MARS	Masking action relevant stimuli
RNN	Recurrent neural network

ReLU	Rectified linear unit
ROC	Receiver operating characteristic
SVM	Support vector machine
VRAM	Video RAM

References

1. Salvucci, D.D. Modeling Driver Behavior in a Cognitive Architecture. *Hum. Factors* **2006**, *48*, 362–380. [CrossRef] [PubMed]
2. Braunagel, C.; Geisler, D.; Rosenstiel, W.; Kasneci, E. Online Recognition of Driver-Activity Based on Visual Scanpath Classification. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 23–36. [CrossRef]
3. Bulling, A.; Ward, J.A.; Gellersen, H.; Tröster, G. Eye movement analysis for activity recognition. In Proceedings of the 11th International Conference on Ubiquitous Computing, Orlando, FL, USA, 30 September–3 October 2009; pp. 41–50. [CrossRef]
4. Bulling, A.; Ward, J.A.; Gellersen, H.; Troster, G. Eye movement analysis for activity recognition using electrooculography. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 741–753. [CrossRef] [PubMed]
5. Huda, K.; Hossain, M.S.; Ahmad, M. Recognition of reading activity from the saccadic samples of electrooculography data. In Proceedings of the 2015 International Conference on Electrical Electronic Engineering (ICEEE), Rajashi, Bangladesh, 4–6 November 2015; pp. 73–76. [CrossRef]
6. D'Souza, S.; Natarajan, S. Recognition of EOG based reading task using AR features. In Proceedings of the International Conference on Circuits, Communication, Control and Computing (I4C), Bangalore, India, 20–22 November 2014; pp. 113–117. [CrossRef]
7. Xing, Y.; Lv, C.; Wang, H.; Cao, D.; Velenis, E.; Wang, F.Y. Driver Activity Recognition for Intelligent Vehicles: A Deep Learning Approach. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5379–5390. [CrossRef]
8. Sigari, M.H.; Pourshahabi, M.R.; Soryani, M.; Fathy, M. A Review on Driver Face Monitoring Systems for Fatigue and Distraction Detection. *Int. J. Adv. Sci. Technol.* **2014**, *64*, 73–100. [CrossRef]
9. Niwa, S.; Yuki, M.; Noro, T.; Shioya, S.; Inoue, K. *A Wearable Device for Traffic Safety—A Study on Estimating Drowsiness with Eyewear, JINS MEME*; SAE Technical Paper Series; SAE International: Detroit, MI, USA, 2016. [CrossRef]
10. Joseph, D.P.; Miller, S.S. Apical and basal membrane ion transport mechanisms in bovine retinal pigment epithelium. *J. Physiol.* **1991**, *435*, 439–463. [CrossRef]
11. Lagodzinski, P.; Shirahama, K.; Grzegorzek, M. Codebook-based electrooculography data analysis towards cognitive activity recognition. *Comput. Biol. Med.* **2017**, *95*. [CrossRef]
12. Grzegorzek, M. *Sensor Data Understanding*; Logos Verlag Berlin GmbH: Berlin, Germany, 2017.
13. Shirahama, K.; Köping, L.; Grzegorzek, M. Codebook Approach for Sensor-Based Human Activity Recognition. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, UbiComp '16, Heidelberg, Germany, 12–16 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 197–200. [CrossRef]
14. JINS MEME. JINS MEME Glasses Specifications. Available online: https://www.cnet.com/reviews/jins-meme-preview/ (accessed on 17 June 2020).
15. Braunagel, C.; Kasneci, E.; Stolzmann, W.; Rosenstiel, W. Driver-activity recognition in the context of conditionally autonomous driving. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC), Las Palmas, Spain, 15–18 September 2015; pp. 1652–1657. [CrossRef]
16. Khushaba, R.N.; Kodagoda, S.; Lal, S.; Dissanayake, G. Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm. *IEEE Trans. Biomed. Eng.* **2011**, *58*, 121–131. [CrossRef]
17. Doniec, R.; Sieciński, S.; Piaseczna, N.; Mocny-Pachońska, K.; Lang, M.; Szymczyk, J. The Classifier Algorithm for Recognition of Basic Driving Scenarios. In *Information Technology in Biomedicine*; Piętka, E., Badura, P., Kawa, J., Więcławek, W., Eds.; Springer: Cham, Switzerland, 2020; pp. 359–367. [CrossRef]

18. Stapel, J.; Hassnaoui, M.E.; Happee, R. Measuring Driver Perception: Combining Eye-Tracking and Automated Road Scene Perception. *Hum. Factors J. Hum. Factors Ergon. Soc.* **2020**. [CrossRef]
19. Gao, Z.K.; Li, Y.L.; Yang, Y.X.; Ma, C. A recurrence network-based convolutional neural network for fatigue driving detection from EEG. *Chaos Interdiscip. J. Nonlinear Sci.* **2019**, *29*, 113126. [CrossRef]
20. Karuppusamy, N.S.; Kang, B.Y. Multimodal System to Detect Driver Fatigue Using EEG, Gyroscope, and Image Processing. *IEEE Access* **2020**, *8*, 129645–129667. [CrossRef]
21. Jiao, Y.; Deng, Y.; Luo, Y.; Lu, B.L. Driver sleepiness detection from EEG and EOG signals using GAN and LSTM networks. *Neurocomputing* **2020**, *408*, 100–111. [CrossRef]
22. Shin, J.; Kim, S.; Yoon, T.; Joo, C.; Jung, H.I. Smart Fatigue Phone: Real-time estimation of driver fatigue using smartphone-based cortisol detection. *Biosens. Bioelectron.* **2019**, *136*, 106–111. [CrossRef] [PubMed]
23. Gao, Z.; Wang, X.; Yang, Y.; Mu, C.; Cai, Q.; Dang, W.; Zuo, S. EEG-Based Spatio–Temporal Convolutional Neural Network for Driver Fatigue Evaluation. *IEEE Trans. Neural Networks Learn. Syst.* **2019**, *30*, 2755–2763. [CrossRef]
24. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*. [CrossRef]
25. Vora, S.; Rangesh, A.; Trivedi, M.M. Driver Gaze Zone Estimation Using Convolutional Neural Networks: A General Framework and Ablative Analysis. *IEEE Trans. Intell. Veh.* **2018**, *3*, 254–265. [CrossRef]
26. Act of 5 January 2011 on Vehicle Drivers. Journal of Laws of the Republic of Poland (Dz.U. 2011 nr 30 poz. 151). Available online: http://prawo.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20110300151 (accessed on 24 November 2020).
27. Act of 6 September 2001 on the road traffic. Journal of Laws of the Republic of Poland (Dz.U. 1997 nr 28 poz. 152). Available online: http://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20011251371/U/D20011371Lj.pdf (accessed on 24 November 2020).
28. Regulation of the Minister of Infrastructure of 28 June 2019 on Examining Applicants for Driving Licenses, Training, Examining and Obtaining Qualifications by Examiners and Samples of Documents Used in These Matters. Journal of Laws of the Republic of Poland (Dz.U. 2019 poz. 1206). Available online: http://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20190001206 (accessed on 24 November 2020).
29. Doniec, R.; Duraj, K.; Mocny-Pachońska, K.; Piaseczna, N.; Sieciński, S.; Tkacz, E. Drivers' Activity Tracking With JINS MEME Smart Glasses. 2020. Available online: https://ieee-dataport.org/documents/drivers-activity-tracking-jins-meme-smart-glasses (accessed on 24 November 2020).
30. van Leeuwen, P.M.; de Groot, S.; Happee, R.; de Winter, J.C.F. Differences between racing and non-racing drivers: A simulator study using eye-tracking. *PLoS ONE* **2017**, *12*, e0186871. [CrossRef]
31. Philipp, G.; Song, D.; Carbonell, J.G. The exploding gradient problem demystified—Definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv* **2017**, arXiv:1712.05577.
32. Kiranyaz, S.; Ince, T.; Abdeljaber, O.; Avci, O.; Gabbouj, M. 1-D Convolutional Neural Networks for Signal Processing Applications. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 8360–8364. [CrossRef]
33. Amiri, P.; Abbasi, H.; Derakhshan, A.; Gharib, B.; Nooralishahi, B.; Mirzaaghayan, M. Potential Prognostic Markers in the Heart Rate Variability Features for Early Diagnosis of Sepsis in the Pediatric Intensive Care Unit using Convolutional Neural Network Classifiers. In Proceedings of the 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Montreal, QC, Canada, 20–24 July 2020; pp. 1031–1034. [CrossRef]
34. Zubarev, I.; Zetter, R.; Halme, H.L.; Parkkonen, L. Adaptive neural network classifier for decoding MEG signals. *NeuroImage* **2019**, *197*, 425–434. [CrossRef]
35. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef]
36. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958. [CrossRef]
37. Kumar, R.; Indrayan, A. Receiver operating characteristic (ROC) curve for medical researchers. *Indian Pediatr.* **2011**, *48*, 277–287. [CrossRef] [PubMed]

38. Jiang, L.; Lin, X.; Liu, X.; Bi, C.; Xing, G. SafeDrive: Detecting Distracted Driving Behaviors Using Wrist-Worn Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2018**, *1*, 144:1–144:22. [CrossRef]
39. Galarza, E.E.; Egas, F.D.; Silva, F.M.; Velasco, P.M.; Galarza, E.D. Real Time Driver Drowsiness Detection Based on Driver's Face Image Behavior Using a System of Human Computer Interaction Implemented in a Smartphone. In Proceedings of the International Conference on Information Technology & Systems (ICITS 2018), Libertad City, Ecuador, 10–12 January 2018; Rocha, Á., Guarda, T., Eds.; Springer: Cham, Switzerland, 2018; pp. 563–572. [CrossRef]
40. Mulhall, M.D.; Cori, J.; Sletten, T.L.; Kuo, J.; Lenné, M.G.; Magee, M.; Spina, M.A.; Collins, A.; Anderson, C.; Rajaratnam, S.M.; et al. A pre-drive ocular assessment predicts alertness and driving impairment: A naturalistic driving study in shift workers. *Accid. Anal. Prev.* **2020**, *135*, 105386. [CrossRef] [PubMed]
41. Li, F.; Shirahama, K.; Nisar, M.; Köping, L.; Grzegorzek, M. Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. *Sensors* **2018**, *18*, 679. [CrossRef] [PubMed]
42. Choi, M.; Koo, G.; Seo, M.; Kim, S.W. Wearable Device-Based System to Monitor a Driver's Stress, Fatigue, and Drowsiness. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 634–645. [CrossRef]

3.3 Heartbeat Detection in Seismocardiograms with Semantic Segmentation

3.3.1 Citation

Exact reference: [45]

3.3.2 Paper Contribution

Because it is recognized as a representative measure of cardiac function, heartbeat detection is a crucial component of cardiac signal analysis. Location of the QRS complex on an electrocardiogram is the gold standard for heartbeat detection. Seismocardiography (SCG), which uses vibrations to measure heart rate, is replacing electrocardiography as a reliable method of doing so due to the advancement of sensors and information and communication technologies (ICT).

Novelty aspects:

- developed an accurate method for processing and extracting information from seismocardiography signals,
- developed a deep neural network which is efficient and can be run on edge devices.

Possible use cases:

- edge device that can accurately measure and evaluate the cardiac function,
- extracting additional information about mechanical function of a heart.

3.3.3 Author's Contribution

The author conceptualized the project, surveyed the related literature, developed and tested the methodology for data pre-processing, training and evaluating a deep learning model, analyzed the results and written parts of the manuscript.

2022 44th Annual International Conference of
the IEEE Engineering in Medicine & Biology Society (EMBC)
Scottish Event Campus, Glasgow, UK, July 11-15, 2022

Heartbeat Detection in Seismocardiograms with Semantic Segmentation*

Konrad M. Duraj, Szymon Siecinski, Rafal J. Doniec,
Natalia J. Piaseczna, Pawel S. Kostka, and Ewaryst J. Tkacz[1]

Abstract— **Heartbeat detection is an essential part of cardiac signal analysis because it is recognized as a representative measure of cardiac function. The gold standard for heartbeat detection is to locate QRS complexes in electrocardiograms. Due to the development of sensors and information and communication technologies (ICT), seismocardiography (SCG) is becoming a viable alternative to electrocardiography to monitor heart rate. In this work, we propose a system for detecting the heartbeat based on seismocardiograms using deep learning methods. The study was carried out with a publicly available data set (CEBS) that contains simultaneous measurements of ECG, breathing signal, and seismocardiograms. Our approach to heartbeat detection in seismocardiograms uses a model based on a ResNet-based convolutional neural network and contains a squeeze and excitation unit. Our model scored state-of-the-art results (Jaccard and F1 score above 97%) on the test dataset, demonstrating its high reliability.**

I. INTRODUCTION

Heartbeat detection is one of the essential parts of cardiac signal analysis because it is recognized as a representative measure of cardiac function [1]. Dynamic changes in the interval between consecutive heartbeats are known as heart rate variability (HRV) [2]–[4].

The gold standard for heartbeat detection is the detection of QRS complexes in the electrocardiogram [2], [5]. Although electrocardiography (ECG) is a widely available, inexpensive, easy to use, and inexpensive method of diagnosing the state of the heart, technological advances in sensors and information and communication technologies (ICT) make seismocardiography (SCG) a viable alternative [5]–[9].

Seismocardiography (SCG) is the registration of mechanical activity of the heart with an accelerometer placed on the chest wall in the xyphoid process [6], [10]–[12]. The most popular applications of seismocardiography are heart rate measurement [13]–[18] and HRV analysis [5], [16], [19], [20].

Conventional segmentation of cardiac signals consists of extracting specific features and intervals using various algorithms proposed by experts [8], [21]. An alternative approach is to use semantic segmentation and apply deep neural networks. Semantic segmentation is an approach to detect, classify, and annotate various internally consistent subsequences of time signals based on changes between defined states [22].

Although the use of machine learning in seismocardiography has been reported in several studies [23]–[28], the use of semantic segmentation for heartbeat detection in seismocardiograms with deep neural networks was mentioned only in [28] to the best of our knowledge.

The purpose of our study was to address the aforementioned gap by developing and evaluating the performance of a deep neural network based on UNet that takes a raw SCG signal and returns the signal with annotated heart beats by applying the semantic segmentation approach.

II. MATERIAL AND METHODS

A. Data Set

The study was carried out on the data set "Combined Measurement of ECG, Breathing and Seismocardiogram" (CEBS) publicly available at PhysioNet.org [13], [29]. The data set consists of 60 simultaneous recordings of ECG signals (Leads I and II), breathing signal, and seismocardiograms (on the z axis) acquired from 20 healthy volunteers of Caucasian race who were awake and remained in a supine position on a bed. The recordings were divided into three groups according to the data acquisition phase: b001-b020 (before playing the music, 5 minutes), m001-m020 (while playing the music for 50 minutes) and p001-p020 (after playing the music, 5 minutes) [13], [29]–[31].

Each signal was recorded with a sampling frequency of 5 kHz. ECG signal and the breathing wave were acquired with the Biopac MP36 data acquisition system, and the SCG signal was recorded with the ST Microelectronics LIS334ALH triaxial accelerometer. The bandwidth of the ECG and SCG signals were 0.05 Hz–150 Hz and 0.5–100 Hz [13], [30], [31].

B. Data preprocessing

1) Heartbeat Detection in ECG: The location of heartbeats on the electrocardiogram is defined as the location of R waves [32]. Heartbeats in ECG signals were detected with the Pan-Tompkins algorithm, which consists of the following steps: bandpass filtering, differentiation, sample squaring, smoothing of the signal with a moving average filter, correlation analysis, and thresholding [33].

*This study was carried out under the project "InterPOWER - Silesian University of Technology as a modern European technical university", co-financed by the European Union under Measure 3.5 Comprehensive programs of universities III Priority Axis Higher education for the economy and development of the Operational Program Knowledge Education Development 2014–2020

[1]Konrad M. Duraj, Szymon Siecinski, Rafal J. Doniec, Natalia J. Piaseczna, Pawel S. Kostka, and Ewaryst J. Tkacz are with the Department of Biosensors and Processing of Biomedical Signals, Faculty of Biomedical Engineering, Silesian University of Technology, Roosevelta 40, 41-800 Zabrze, Poland. {kduraj,ssiecinski,rdoniec,npiaseczna,pkostka,etkacz}@polsl.pl

2) Reference Heartbeat Annotation in SCG: The heartbeat in the SCG signal is considered to be the occurrence of an aortic valve opening (AO) wave [13], [34], [35]. The reference heartbeats in the SCG signals were determined as the occurrence of AO waves based on the windowing method described in [5], [36] and consist of several steps. The first step was bandpass filtration with the third-order Butterworth filter with a passband of 4–50 Hz. The next step was smoothing the SCG signals with a moving average filter with a window width of 15 ms. Finally, the locations of AO waves were determined as the local maxima of the SCG signals within 100 ms after the R waves in ECG signals [5], [16], [36], [37]. The locations of the reference heartbeats in the SCG signals were converted to binary masks.

$$AO\_mask_i = \begin{cases} 1 & \text{the sample belongs to a reference} \\ & \text{heartbeat within 150ms margin} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where AO_mask is the binary signal consisting of N samples, which is equal to the number of samples of the input signals and serves as the "ground truth" in the learning process.

C. Segmentation of Heartbeats with a Deep Neural Network

1) Data Preprocessing: The first step was to normalize the signal amplitude to $[0; 1]$ with *Min-Max* normalization, expressed as:

$$x' = \frac{x - min(x)}{max(x) - min(x)}. \quad (2)$$

The normalized signal is then divided into observations containing 5000 samples. If the particular signal is not divisible by the interval length, then the appropriate number of zeros is added or subtracted. The data sets consist of full SCG signals from individual patients that resulted in a total of 35973 observations in the data set.

The constructed data set was divided into training, validation and test sets in the proportions shown in Figure 1. The signals from a given patient were included in only one of the sets.

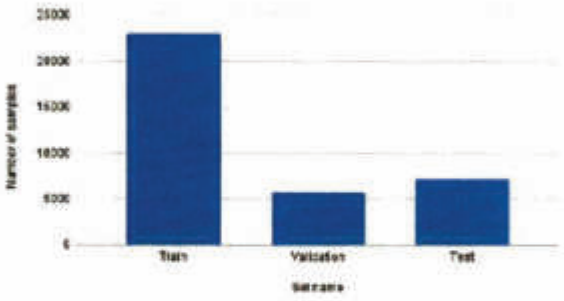


Fig. 1: Data distribution among train, validation, and test set.

This means that the signal representing each patient has been assigned to a given set and was not included in another set.

2) Architecture: The input of the network consists of subsignals derived from the original input signals. The ground truth for the network was obtained as presented in Sections II-B.2 and II-C.1. The output of the designed deep neural network has the same shape as the input.

The encoder part of our network is a UNet-like architecture [38] of a one-dimensional convolutional neural network with skip connections as applied in the ResNet architecture [39]. It was designed with four sequentially connected blocks. The single block consisted of a convolutional unit followed by two residual units, with two additional convolutional blocks upon which a squeeze and excitation unit was applied.

The outputs of the second and third blocks were concatenated with the averaged pooled input data. The decoder part consists of three upsampling layers followed by concatenation with encoder feature maps and convolutional units. The output was generated with the 77 kernel convolution window.

3) Model Training: The training procedure was conducted using the Adam optimization algorithm [40] and binary cross-entropy as the loss function. The model was trained with a mini-batch size of 32 and a starting learning rate of 0.0001. The hyperparameters were tuned empirically. To optimize training, the weights of a model with the best score were saved, the learning rate was reduced by half each 3 epoch if the validation loss was not improving, and the training was stopped in case of overfitting. The model was set to be trained for 100 epochs, but after 18 epochs, the model training was stopped due to overfitting. The model with the best validation loss was saved. The described model was trained on a NVIDIA RTX 2080Ti graphics card with 12GB of video RAM. The model was designed with Tensorflow and Keras deep learning libraries.

III. RESULTS

Table I presents the results achieved for both the training and validation sets.

TABLE I: Performance measures of the designed deep neural network.

Dataset	Train	Validation	Test
Accuracy	0.998	0.992	0.993
AUC	0.999	0.996	0.997
Specificity	0.999	0.994	0.995
Sensitivity	1.0	0.999	0.999
Precision	0.987	0.97	0.969
Recall	0.986	0.97	0.967
Loss	0.000784	0.022	0.022

The performance of our model was also expressed as the Jaccard index ($J_{(A,B)}$) and the F1 score ($F1_{(A,B)}$) to determine the similarity and differences between the ground truth and prediction. They are defined as follows:

$$J_{(A,B)} = \frac{|A \cap B|}{|A \cup B|}, \quad (3)$$

$$F1_{(A,B)} = \frac{2 \times |A \cap B|}{|A| + |B|}, \tag{4}$$

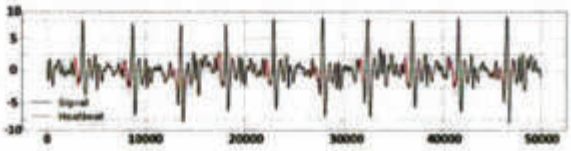
where: $A \cap B$ is the intersection of sets A and B, $A \cup B$ is the union of sets A and B [41].

These metrics were calculated with three different methods of averaging: micro-, marco-, and weighted averaging. The results are presented in Table II.

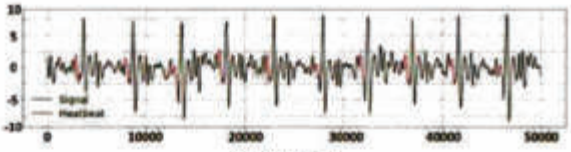
TABLE II: Performance on the test set

Averaging	Jaccard	F1 score
Micro-averaging	0.986	0.992
Macro-averaging	0.971	0.981
Weighted-averaging	0.987	0.992

In the worst case scenario (Jaccard score of 97.1%), the model segments the signal with an offset within circa five samples (while the fragments that correspond to the heartbeat are circa 500 samples).



(a) Ground truth



(b) Prediction

Fig. 2: Comparison between the original labels and predicted labels

Figure 2 presents a comparison of the heartbeats indicated by the original labels and predicted with our deep neural network on a 10 s fragment of the SCG signal.

According to the metrics shown in Table I, the model achieved state-of-the-art results after just one iteration. Further learning did not significantly improve the results; therefore, the process has stopped after 18 iterations to prevent overfitting. Performance metrics indicate that the model is fully capable of segmenting heartbeats into SCG signals with greater repeatability than human experts.

IV. DISCUSSION

We have designed and evaluated a U-Net-based deep neural network with a squeeze and excitation block that is capable of detecting heartbeats using a semantic segmentation approach. The reported performance (sensitivity of 1.000 for the training set and 0.999 for the validation set, precision of 0.987 for the training set and 0.97 for the validation set) proves its high reliability and is similar to the performance metrics reported by Suresh et al. (sensitivity of 0.98 and precision of 0.98) in [28] on the same dataset, except subjects 4 and 18.

Despite similar performance, our approach is not the same as in [28]; the differences lie in the preprocessing of input signals (min-max normalization versus dividing into overlapping time windows), blocks of the deep neural network, parameters of the neural network, and the output signal (seismocardiogram with annotated heartbeats versus distance transform-like output).

Such a good performance of heartbeat detection with a deep neural network is achievable because deep neural networks are good at modeling diverse phenomena. However, due to overparameterization and scalability, deep neural networks are computationally expensive due to their design [42]. Moreover, normal heartbeats are repetitive (quasi-periodic) features with similar characteristics that are relatively easy to capture by neural networks.

The limitations of our study are the study group limited to 20 healthy subjects between 19 and 30 years of age [13], [29]–[31], clear signals acquired in the supine position, and the dependence of heartbeat detection in SCG signals on the R waves in ECG.

In future research, we consider a wider range of subjects with different cardiovascular conditions, applying a heartbeat detector that does not depend on the parallel ECG signal, and improving the validation process by additional analysis, such as cross-validation and live tests.

ACKNOWLEDGMENT

We would like to thank the authors of the "Combined Measurement of ECG, Breathing and Seismocardiogram" (CEBS) Database for sharing the SCG signals.

REFERENCES

[1] B. Pomeranz, R. J. Macaulay, M. A. Caudill, I. Kutz, D. Adam, D. Gordon, K. M. Kilborn, A. C. Barger, D. C. Shannon, R. J. Cohen, and al. et, "Assessment of autonomic function in humans by heart rate spectral analysis," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 248, no. 1, pp. H151–H153, Jan 1985.

[2] Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology, "Heart rate variability. standards of measurement, physiological interpretation, and clinical use," *Circulation*, vol. 93, pp. 1043–1065, March 1996.

[3] G. E. Billman, "Heart rate variability - a historical perspective," *Frontiers in Physiology*, vol. 2, 2011.

[4] G. Ernst, "Hidden signals–the history and methods of heart rate variability," *Frontiers in Public Health*, vol. 5, Oct 2017.

[5] M. J. Tadi, E. Lehtonen, T. Koivisto, M. Pänkäälä, A. Paasio, and M. Teräs, "Seismocardiography: Toward heart rate variability (HRV) estimation," in *2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings*, Turin, Italy, May 2015, pp. 261–266.

[6] J. M. Zanetti and K. Tavakolian, "Seismocardiography: Past, present and future," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2013, pp. 7004–7007.

[7] M. Scarpetta, M. Spadavecchia, G. Andria, M. A. Ragolia, and N. Giaquinto, "Simultaneous measurement of heartbeat intervals and respiratory signal using a smartphone," in *2021 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Lausanne, Switzerland, June 2021, pp. 1–5.

[8] D. Rai, H. K. Thakkar, S. S. Rajput, J. Santamaria, C. Bhatt, and F. Roca, "A comprehensive review on seismocardiogram: Current advancements on acquisition, annotation, and applications," *Mathematics*, vol. 9, no. 18, p. 2243, Sep 2021.

[9] X. Han, X. Wu, J. Wang, H. Li, K. Cao, H. Cao, K. Zhong, and X. Yang, "The latest progress and development trend in the research of ballistocardiography (BCG) and seismocardiogram (SCG) in the field of health care," *Applied Sciences*, vol. 11, no. 19, p. 8896, Sep 2021.

[10] D. Salerno and J. Zanetti, "Seismocardiography: A new technique for recording cardiac vibrations. concept, method, and initial observations," *Journal of Cardiovascular Technology*, vol. 9, no. 2, pp. 111–118, January 1990.

[11] O. T. Inan, P. F. Migeotte, K. S. Park, M. Etemadi, K. Tavakolian, R. Casanella, J. Zanetti, J. Tank, I. Funtova, G. K. Prisk, and M. di Rienzo, "Ballistocardiography and seismocardiography: A review of recent advances," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1414–1427, July 2015.

[12] A. Taebi, B. Solar, A. Bomar, R. Sandler, and H. Mansy, "Recent advances in seismocardiography," *Vibration*, vol. 2, no. 1, p. 64–86, Jan 2019.

[13] M. A. García-González, A. Argelagós-Palau, M. Fernández-Chimeno, and J. Ramos-Castro, "A comparison of heartbeat detectors for the seismocardiogram," in *Computing in Cardiology 2013*, Sep 2013, pp. 461–464.

[14] G. Shafiq, S. Tatinati, W. T. Ang, and K. C. Veluvolu, "Automatic identification of systolic time intervals in seismocardiogram," *Scientific Reports*, vol. 6, no. 37524, November 2016, article.

[15] M. J. Tadi, E. Lehtonen, T. Hurnanen, J. Koskinen, J. Eriksson, M. Pänkäälä, M. Teräs, and T. Koivisto, "A real-time approach for heart rate monitoring using a Hilbert transform in seismocardiograms," *Physiological Measurement*, vol. 37, no. 11, pp. 1885–1909, Sep 2016.

[16] S. Sieciński, P. S. Kostka, and E. J. Tkacz, "Heart rate variability analysis on electrocardiograms, seismocardiograms and gyrocardiograms on healthy volunteers," *Sensors*, vol. 20, no. 16, p. 4522, Aug 2020.

[17] F. Cocconcelli, N. Mora, G. Matrella, and P. Ciampolini, "Seismocardiography-based detection of heartbeats for continuous monitoring of vital signs," in *2019 11th Computer Science and Electronic Engineering (CEEC)*, Colchester, UK, Sep. 2019, pp. 53–58.

[18] P.-Y. Hsu, P.-H. Hsu, T.-H. Lee, and H.-L. Liu, "Heart rate and respiratory rate monitoring using seismocardiography," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. Guadalajara, Mexico: IEEE, Nov 2021.

[19] J. Ramos-Castro, J. Moreno, H. Miranda-Vidal, M. A. García-González, M. Fernández-Chimeno, G. Rodas, and L. Capdevila, "Heart rate variability analysis using a seismocardiogram signal," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2012, pp. 5642–5645.

[20] A. Laurin, A. Blaber, and K. Tavakolian, "Seismocardiograms return valid heart rate variability indices," in *Computing in Cardiology 2013*, September 2013, pp. 413–416.

[21] A. N. Londhe and M. Atulkar, "Semantic segmentation of ECG waves using hybrid channel-mix convolutional and bidirectional LSTM," *Biomedical Signal Processing and Control*, vol. 63, p. 102162, Jan 2021.

[22] S. Gharghabi, C.-C. M. Yeh, Y. Ding, W. Ding, P. Hibbing, S. LaMunion, A. Kaplan, S. E. Crouter, and E. Keogh, "Domain agnostic online semantic segmentation for multi-dimensional time series," *Data Mining and Knowledge Discovery*, vol. 33, no. 1, pp. 96–130, Sep 2018.

[23] H. Lee and M. Whang, "Heart rate estimated from body movements at six degrees of freedom by convolutional neural networks," *Sensors*, vol. 18, no. 5, May 2018.

[24] S. Mehrang, M. Jafari Tadi, M. Kaisti, O. Lahdenoja, T. Vasankari, T. Kiviniemi, J. Airaksinen, T. Koivisto, and M. Pänkäälä, "Machine learning based classification of myocardial infarction conditions using smartphone-derived seismo- and gyrocardiography," in *2018 Computing in Cardiology Conference (CinC)*, vol. 45, September 2018, pp. 1–4.

[25] Z. Iftikhar, O. Lahdenoja, M. J. Tadi, T. Hurnanen, T. Vasankari, T. Kiviniemi, J. Airaksinen, T. Koivisto, and M. Pänkäälä, "Multiclass classifier based cardiovascular condition detection using smartphone mechanocardiography," *Scientific Reports*, vol. 8, no. 1, June 2018.

[26] M. J. Tadi, S. Mehrang, M. Kaisti, O. Lahdenoja, T. Hurnanen, J. Jaakkola, S. Jaakkola, T. Vasankari, T. Kiviniemi, J. Airaksinen, T. Knuutila, E. Lehtonen, T. Koivisto, and M. Pänkäälä, "Comprehensive analysis of cardiogenic vibrations for automated detection of atrial fibrillation using smartphone mechanocardiograms," *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2230–2242, Mar 2019.

[27] S. Mehrang, O. Lahdenoja, M. Kaisti, M. J. Tadi, T. Hurnanen, A. Airola, T. Knuutila, J. Jaakkola, S. Jaakkola, T. Vasankari, T. Kiviniemi, J. Airaksinen, T. Koivisto, and M. Pänkäälä, "Classification of atrial fibrillation and acute decompensated heart failure using smartphone mechanocardiography: A multilabel learning approach," *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7957–7968, July 2020.

[28] P. Suresh, N. Narayanan, C. V. Pranav, and V. Vijayaraghavan, "End-to-end deep learning for reliable cardiac activity monitoring using seismocardiograms," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, Dec 2020, pp. 1369–1375.

[29] M. A. García-González, A. Argelagós-Palau, M. Fernández-Chimeno, and J. Ramos-Castro, "Combined measurement of ecg, breathing and seismocardiograms (cebs database)," 2013. [Online]. Available: https://physionet.org/content/cebsdb/

[30] M. A. García-González, A. Argelagós, M. Fernández-Chimeno, and J. Ramos-Castro, "Differences in qrs locations due to ecg lead: Relationship with breathing," in *XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013*, L. M. Roa Romero, Ed. Cham: Springer International Publishing, 2014, pp. 962–964.

[31] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[32] I. I. Christov, "Real time electrocardiogram qrs detection using combined adaptive threshold," *BioMedical Engineering OnLine*, vol. 3, no. 1, p. 28, 2004.

[33] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, March 1985.

[34] J. M. Zanetti, M. O. Poliac, and R. S. Crow, "Seismocardiography: waveform identification and noise analysis," in *[1991] Proceedings Computers in Cardiology*, Venice, Italy, September 1991, pp. 49–52.

[35] C. Yang and N. Tavassolian, "Combined seismo- and gyrocardiography: A more comprehensive evaluation of heart-induced chest vibrations," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 5, pp. 1466–1475, September 2018.

[36] K. Pandia, O. T. Inan, G. T. A. Kovacs, and L. Giovangrandi, "Extracting respiratory information from seismocardiogram signals acquired on the chest using a miniature accelerometer," *Physiological measurement*, vol. 33, no. 10, p. 1643—1660, 10 2012.

[37] K. Sørensen, S. E. Schmidt, A. S. Jensen, P. Søgaard, and J. J. Struijk, "Definition of fiducial points in the normal seismocardiogram," *Scientific Reports*, vol. 8, no. 1, p. 15455, 2018.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 770–778.

[40] Y. Bengio and Y. LeCun, Eds., *Adam: A Method for Stochastic Optimization*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[41] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb 1912.

[42] N. C. Thompson, K. Greenwald, K. L. Lee, and G. F. Manso, "The computational limits of deep learning," *MIT Initiative on the Digital Economy Research Brief*, vol. 4, Sep 2020. [Online]. Available: https://ide.mit.edu/wp-content/uploads/2020/09/RBN.Thompson.pdf

3.4 Semantic Segmentation of 12-Lead ECG Using 1D Residual U-Net with Squeeze-Excitation Blocks

3.4.1 Citation

Exact reference: [46]

3.4.2 Paper Contribution

ECG evaluation is perhaps the most widely used biomedical signal for the performance of diagnostic measurements. This signal has its reflection in the mechanical action of the heart and can inform us about the physiological condition of this organ. The analysis of an electrocardiogram is a complex process that requires specific knowledge. This paper describes an approach of using deep neural networks for extracting ECG segments such as: T-Wave, P-Wave, and QRS complex.

Novelty aspects:

- developed an architecture capable of accurate segmentation of relevant segments of ECG waveform,
- developed method can work in parallel on all of the 12-leads, which makes this method not only effective but also efficient,
- developed a deep neural network which is efficient and can be run on edge devices.

Possible use cases:

- edge device that can accurately measure and evaluate the cardiac function,
- precise information extraction for all channels in a parallel manner (efficiency),
- can be easily integrated with other software used to evaluate the state of the cardiac function for in-house monitoring.

3.4.3 Author's Contribution

The author conceptualized the project, surveyed the related literature, developed and tested the methodology for data pre-processing, training and evaluating a deep learning model, analyzed the results and written parts of the manuscript.

applied sciences

MDPI

Article

Semantic Segmentation of 12-Lead ECG Using 1D Residual U-Net with Squeeze-Excitation Blocks

Konrad Duraj, Natalia Piaseczna *, Paweł Kostka and Ewaryst Tkacz

Department of Biosensors and Processing of Biomedical Signals, Faculty of Biomedical Engineering, Silesian University of Technology, Roosevelta 40, 41-800 Zabrze, Poland; kduraj@polsl.pl (K.D.); pkostka@polsl.pl (P.K.); etkacz@polsl.pl (E.T.)
* Correspondence: npiaseczna@polsl.pl

Abstract: Analyzing biomedical data is a complex task that requires specialized knowledge. The development of knowledge and technology in the field of deep machine learning creates an opportunity to try and transfer human knowledge to the computer. In turn, this fact influences the development of systems for the automatic evaluation of the patient's health based on data acquired from sensors. Electrocardiography (ECG) is a technique that enables visualizing the electrical activity of the heart in a noninvasive way, using electrodes placed on the surface of the skin. This signal carries a lot of information about the condition of heart muscle. The aim of this work is to create a system for semantic segmentation of the ECG signal. For this purpose, we used a database from Lobachevsky University available on Physionet, containing 200, 10-second, and 12-lead ECG signals with annotations, and applied one-dimensional U-Net with the addition of squeeze-excitation blocks. The created model achieved a set of parameters indicating high performance (for the test set: accuracy—0.95, AUC—0.99, specificity—0.95, sensitivity—0.99) in extracting characteristic parts of ECG signal such as P and T-waves and QRS complex, regardless of the lead.

Keywords: ECG; signal segmentation; deep learning

Citation: Duraj, K.; Piaseczna, N.; Kostka, P.; Tkacz, E. Semantic Segmentation of 12-Lead ECG Using 1D Residual U-Net with Squeeze-Excitation Blocks. *Appl. Sci.* **2022**, *12*, 3332. https://doi.org/10.3390/app12073332

Academic Editor: Josué Álvarez Borrego

Received: 14 February 2022
Accepted: 23 March 2022
Published: 25 March 2022

1. Introduction

Electrocardiography (ECG) is a golden diagnostic standard for measuring the electrical activity of the heart. This signal has its reflection in the mechanical action of the heart and can inform us about the physiological condition of this organ. The analysis of an electrocardiogram is a complex process that requires specific knowledge. Engineers try to help physicians by creating expert systems. This process requires transfer of the specialist's knowledge to a computer by feeding it with examples and a set of rules about how to treat specific cases. There are many possibilities of creation of an expert system—one of the most developed is using machine learning-based technologies.

One of the subfields of machine learning methods is deep learning (DL)—containing many hidden layers in the Artificial Neural Network (ANN) structure, which mimics the performance of the human brain [1].

The convolutional neural network, CNN for short, is a specialized type of neural network designed for working with two-dimensional data. It creates rich representations of the input data by sequentially stacking the convolution operations over the image. To reduce the data size, CNNs are often equipped with pooling layers. The filter values are being updated using a backpropagation algorithm. Figure 1 presents a sample of 1D CNN.

These structures can be applied for analyses of physiological signals, such as ECG. The conventional deep convolutional networks are designed to operate exclusively on two-dimensional data. As described in [2], 1D CNNs are more advantageous than their 2D counterparts in dealing with one-dimensional data.

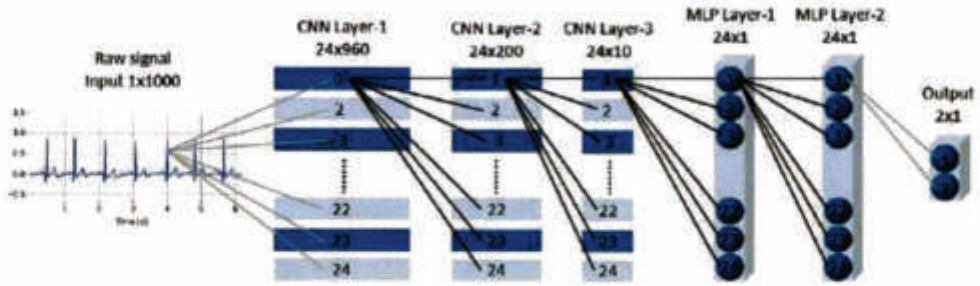


Figure 1. A sample 1D CNN with three-convolutional and two-linear layers [2].

Data annotation is a time-consuming activity, so especially in the era of big data, systems that automatically (but also reliably) label the data are absolutely necessary.

Related Work

The application of deep learning techniques in processing different physiological signals was summarized by Faust et al. They analyzed 53 publications from 1 January 2008 to 31 December 2017, 17 concerning ECGs [3]. Another review of using DL methods for ECG data was performed by Hong et al. in [4]. They evaluated 191 articles published between 1 January 2010 and 29 February 2020. The authors of this work confirm that the application of DL methods in ECG analysis (including signal annotation) is becoming an increasingly frequently discussed topic.

The most frequently discussed issue in relation to the ECG signal is undoubtedly the anomaly detection. Novotna et al. used DL methods for premature ventricular contractions (PVCs) localization. On the CPSC2018 (China Physiological Signal Challenge 2018) database, they achieved the dice coefficient 0.947 for the model with a max pooling layer [5].

Du et al. presented a fine-grained multilabel ECG framework to detect correlated cardiac abnormalities in clinical ECG data using convolutional neural network (CNN) and recurrent neural network (RNN) [6]. The authors of [7] used U-Net with bidirectional LSTM for the automated detection of QRS complexes. They achieved an accuracy of 78.73% and 98.29% on the CPSC2019 (China Physiological Signal Challenge 2019) and mitdb datasets, respectively.

In their work, Weinman and Conrad used transfer learning for improving CNNs classification of heart rhythm and atrial fibrillation (AF) from short ECG signals [8]. The authors explored both unsupervised and supervised pretraining of CNN on the Icentia11K (Icentia11K contains ECG data from 11,000 patients, who wore a monitoring device for up to two weeks resulting in 630,000 h of ECG signal with over 2,700,000,000 beats labeled by the device and then again by a specialist) dataset. They showed that pretraining improves CNN's performance by over 6% and scored the maximum of F_1-score 0.926 for beat classification.

Another interesting issue is the analysis of the ECG signal in terms of detecting individual features in physiological biometrics. Zheng et al., in their work, used BP and DNN for ECG-based identification [9]. They created and tested their model on two databases: mitdb (MIT arrhythmia database) and a self-collected one. The database consisted of signals obtained during various emotions. The authors achieved a 94.39% recognition rate on the combined datasets.

An automatic segmentation of a signal, which is nothing more than finding the reference points within it, is crucial in order to perform automatic interpretation of the signal. There are many different algorithms for segmentation of the ECG signal that use many signal processing methods and work on different databases. In their work, Beraza and Romero compared algorithms for ECG segmentation [10]. The authors performed signal segmentation on ECG using nine different algorithms [11–19] and PhysioNet's

QT database [20]. They achieved the best results while using probabilistic methods and methods based on wavelet transform.

In our study, we propose an alternative approach for the segmentation of a one-dimensional signal segmentation such as ECG. We created a tool for semantic segmentation that used a one-dimensional U-Net containing squeeze-excitation blocks.

2. Materials and Methods

2.1. Dataset

In this study, we used the dataset provided by Lobachevsky University [21], available on the Physionet website. The dataset consists of 200, 10-second, and 12-lead ECG signals, representing different morphologies. Each of the signals comes with an annotation describing the starting, stopping, and peak points of the P and T waves and QRS complexes. The data are compatible with the popular wfdb toolbox. For a more detailed description, please refer to [21].

Data Preparation

In the provided dataset, some of the annotations were incorrect, containing an annotation without the peak point or with it being mislabeled. These samples were removed from the data along with its paired signals. In total, there were 2377 signals with a length of 5000 samples. As mentioned, the annotation came in the form of point collection. To create suitable masks for our model, we transformed the points collection into masks matching the shape of the input signal. For the specific fragments corresponding to the annotations, we applied a label in the form of an integer number as follows:

- "0"—background;
- "1"—QRS block;
- "2"—T-wave;
- "3"— P-wave.

Then, we took the generated sparse masks that are one-hot-encoded, and the final labels have the shape $k \times 5000 \times 4$, where k indicates the number of samples. The input signals were normalized using the min–max scaling technique, which is defined as:

$$x' = \frac{x - min(x)}{max(x) - min(x)}. \quad (1)$$

Then, the dataset was split into training, validation, and testing sets as follows: first, we divided the data into a training and testing set with 80% and 20%; then, we extracted 20% of the training set into the validation set. The distribution of the dataset is presented in the Figure 2.

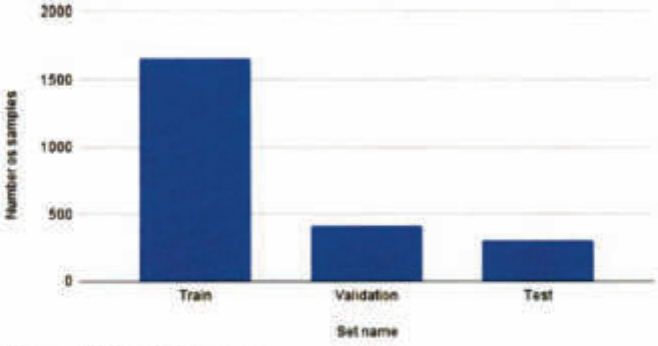


Figure 2. Data distribution.

The distributions between categories across the dataset are as follows:

- Background—7,309,154 points,

- QRS complex—894,721 points;
- P-wave—1,516,929 points;
- T-wave—814,196 points.

2.2. Model Architecture

For our study, we chose the U-Net architecture design proposed by Ronnenberger et al. [22] and adapted it for the task of 1D semantic segmentation. Inspired by the paper [23], we have decided to incorporate residual and squeeze-excitation blocks into our model. A diagram of the model architecture is presented in Figure 3. A full diagram of the model is available as supplementary material (File S1).

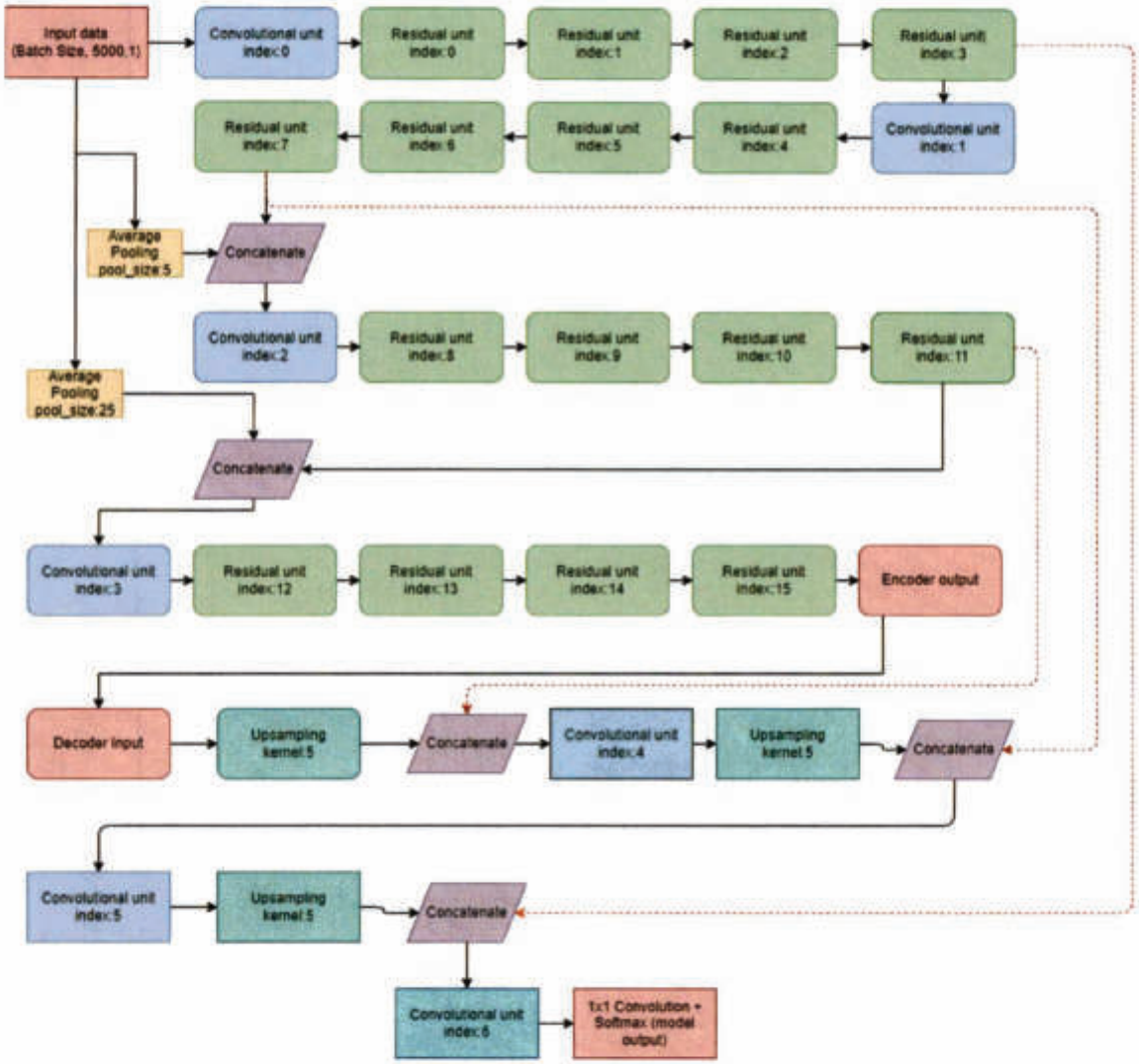


Figure 3. Diagram of the model.

2.2.1. Encoder

The encoder part of our network is a modified ResNet [24] convolutional neural network. It was designed with four sequentially connected blocks. The single block was made of 3 functional units:

1. Convolutional unit—this unit consists of a one-dimensional convolutional layer followed by batch normalization [25];
2. Squeeze-exciting unit—is a cell designed to improve the representational power of a network by enabling it to perform channel-wise feature calibration. The inputs to this block are the feature maps generated by the convolutional unit. Each of the channels is being "squeezed" into a single numeric value using global average pooling. This value will be passed through two feed-forward layers activated by the ReLU and sigmoid functions to add nonlinearity and give each channel a smooth gating function. Then, the output of the sigmoid function is weighted by the input feature maps to get the excitation [26];
3. Residual unit—this unit stacks two convolutional units, and on top of them, the squeeze-exciting unit is being stacked. The output is being added to the input feature maps.

The encoder consists of four of these blocks as a single convolutional unit followed by four consecutive residual blocks. The outputs of the second and third block are being concatenated with the averaged pooled input data.

2.2.2. Decoder

The decoder part consists of up-sampling layers followed by concatenation with encoder feature maps and convolutional units. In total, there were 34,305,156 parameters from which 34,816 were not trainable.

2.3. *Training Process*

The training procedure was conducted using the Adam optimization algorithm [27] and categorical cross-entropy as the loss function. We have examined two versions of our model—one with and one without squeeze-exciting blocks. Both models were trained with the following parameters:

- 32 mini-batch size;
- starting learning rate of 0.0005.

The hyperparametrs were tuned empirically. For this procedure, we have defined a set of callbacks to prevent overfitting and optimize learning:

- Model checkpoint—saving the model that achieves the best validation loss score;
- Reduce learning rate—dynamic learning rate set to decrease by half each 3 epochs when the validation loss is not improving;
- Early stopping—stopping the training procedure when the model overfits the data (after 3 epochs).

The model was set to be trained for 100 epochs, but after 20 epochs, the early stopping callback stopped the learning process due to overfitting. The model with the best validation loss was saved.

The described model was trained on an NVIDIA RTX 2080Ti graphics card with 12 GB of video RAM. The model was designed with the help of Tensorflow and Keras deep learning libraries.

3. Results

This section provides numerical and visual results of the training, validation, and testing of the proposed models. We compared the models with and without the addition of squeeze-exciting blocks.

3.1. *Numerical Results*

To examine the overall model performance, we calculated standard metrics, such as precision, recall, and area under the ROC curve. The numerical results are presented for two versions of the model—with and without squeeze-exciting blocks. The comparison between these two models is shown in Table 1.

When evaluating on the test set, the Jaccard index [28] and the F1-score were also defined to properly test the trained models (see Table 2). The Jaccard index $J_{(A,B)}$ and F1-score $F1_{(A,B)}$ are defined as follows:

$$J_{(A,B)} = \frac{|A \cap B|}{|A \cup B|}, \tag{2}$$

$$F1_{(A,B)} = \frac{2 \cdot |A \cap B|}{|A| + |B|}, \tag{3}$$

where $A \cap B$ is the intersection of sets A and B, and $A \cup B$ is the union of sets A and B.

Table 1. Comparison between metrics achieved without and with squeeze-exciting blocks for the train, validation, and test datasets.

	Without Squeeze-Exciting			With Squeeze-Exciting		
Dataset	**Train**	**Validation**	**Test**	**Train**	**Validation**	**Test**
Accuracy	0.97	0.92	0.92	0.97	0.95	0.95
AUC	0.99	0.98	0.98	0.99	0.99	0.99
Specificity	1.0	0.99	0.97	0.99	0.99	0.95
Sensitivity	1.0	0.99	0.99	0.99	0.99	0.99
Recall	0.97	0.92	0.92	0.98	0.95	0.95
Precision	0.97	0.92	0.92	0.97	0.95	0.95
Loss	0.06	0.25	0.39	0.06	0.14	0.14

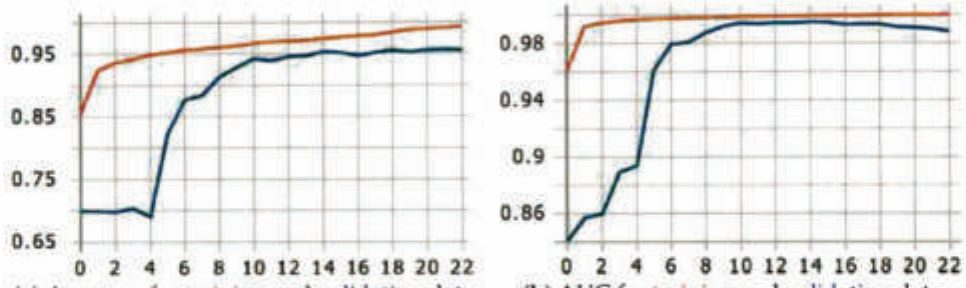
Table 2. Comparison of the Jaccard index and F1-score achieved without and with squeeze-exciting blocks for the test dataset.

	Jaccard Index			F1-Score		
	Macro	**Micro**	**Weighted**	**Macro**	**Micro**	**Weighted**
Without squeeze-exciting	0.8	0.86	0.86	0.87	0.92	0.92
With squeeze-exciting	0.87	0.91	0.91	0.93	0.96	0.96

As we are dealing with a multi-class segmentation problem, the aforementioned indicators were calculated in three different variants: micro, macro, and weighted. With the 'micro' variant, we calculate metrics globally, i.e., by counting the total true positives, true negatives, false negatives, and false positives. For the 'macro' variant, we calculate metrics for each label, and then, we calculate their unweighted mean. This does not take label imbalance into account. In case of the 'weighted' variant, we calculate metrics for each label and find their average, which is weighted by the number of true instances for each label. This alters 'macro' to account for label imbalance.

Based on the results presented in Table 2, we can see that the squeeze-excitation blocks enhance the overall performance of the model. They also provide a better generalization of the problem, which can be seen by analyzing the loss function across different sets.

The learning curves are presented for the model that contains squeeze-excitation blocks since it yields better results (see Figures 4–8).



(a) Accuracy for training and validation data (b) AUC for training and validation data

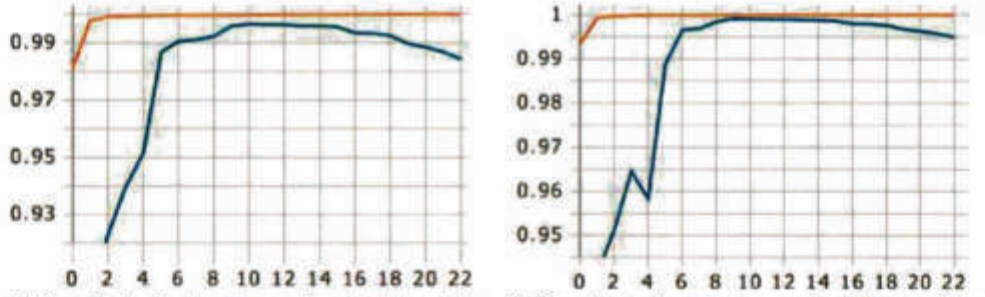
Figure 4. Learning curves—accuracy and AUC among the epochs.

Accuracy is a percent of correct predictions. It describes the performance of the model among classes. Accuracy is given by the following formula [29]:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \quad (4)$$

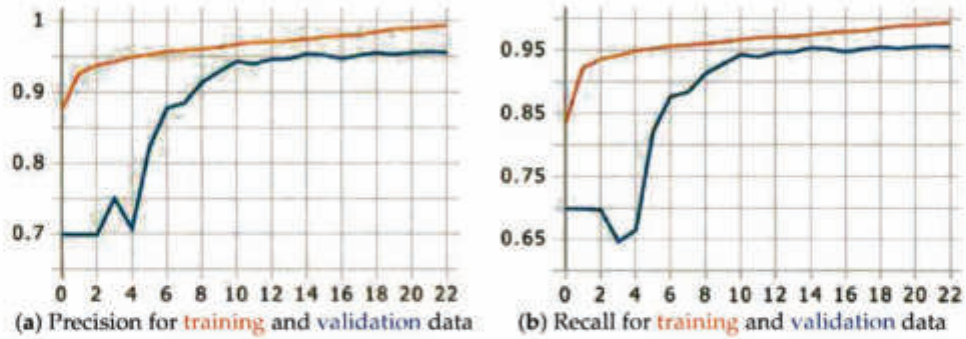
where *TP*—true positive value, *TN*—true negative value, *FP*—false positive value, *FN*—false negative value (applies to Formulas (4)–(8)).

AUC, Area Under the ROC Curve, provides information on how well the model distinguishes between the classes. The desired value is an AUC as close as possible to 1.



(a) Specificity for training and validation data (b) Sensitivity for training and validation data

Figure 5. Learning curves—specificity and sensitivity among the epochs.



(a) Precision for training and validation data (b) Recall for training and validation data

Figure 6. Learning curves—precision and recall among the epochs.

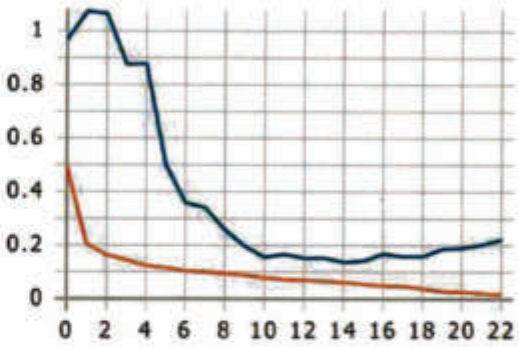


Figure 7. Loss on training and validation data among the epochs.

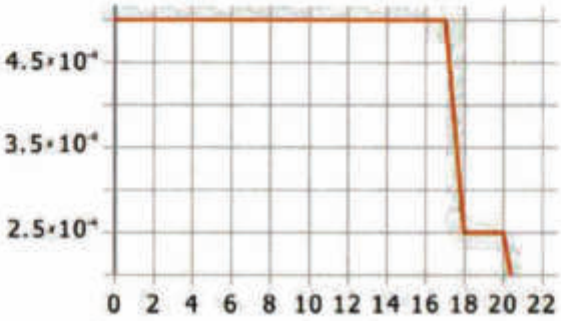


Figure 8. Learning rate among the epochs.

Specificity is a metric that describes the model's ability to correctly predict the true negative value. Specificity is given by the following formula [29]:

$$Specificity = \frac{TN}{TN + FP}. \quad (5)$$

Sensitivity is a metric that describes the model's ability to correctly predict the true positive value. Sensitivity is given by the following formula [29]:

$$Sensitivity = \frac{TP}{TP + FN}. \quad (6)$$

In this case, to determine the sensitivity, we first compute specificity at 200 different thresholds ($threshold_n = \frac{threshold_{n-1}}{2}; n = 1, 2, 3, \ldots, 200; threshold_0 = 1$). Then, the sensitivity is computed at the chosen threshold [30].

When the model achieves high specificity and sensitivity values after the learning cycle, that means that the results are reliable.

Precision is a ratio between the number of true positive samples and all samples classified as positive (i.e., the sum of true and false positive samples). This metric shows the model's ability to correctly classify positive samples. Precision is given by the following formula [29]:

$$Precision = \frac{TP}{TP + FP}. \quad (7)$$

Recall is a metric that defines how well the model detects the positive samples. It is calculated as a ratio between true positive samples and all positive samples (that is, the sum of true positive and false negative samples).

$$Recall = \frac{TP}{TP + FN}. \quad (8)$$

Loss, in other words, penalty for a bad prediction, is a value that indicates how bad the model prediction was on a single sample. In simple terms—the lower the loss, the better the model.

We have calculated the percentage of correctly classified samples for each class and presented it in the form of confusion matrices. Figure 9 shows confusion matrices for each class for the test dataset.

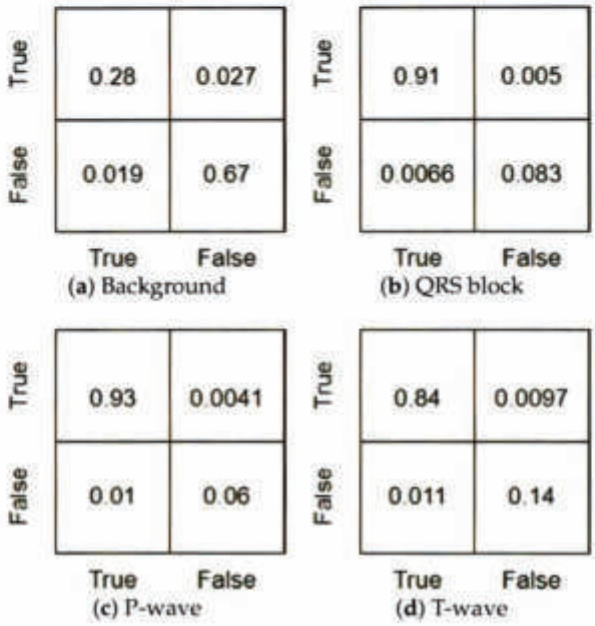


Figure 9. Confusion matrices for each class for the test dataset.

The percentage of correctly classified samples can be calculated as the sum of the values on the main diagonal. For the test dataset, the model scored the following values for each class:

- background—95%;
- QRS block—99.3%;
- P wave—99%;
- T wave—98%.

3.2. *Visual Results*

To compare predictions of the created model with the original annotations, we chose three examples of signals from the test dataset. As the model containing the squeeze-exciting blocks achieved better results, only its predictions were visualized.

Figure 10 show signals with masks presented as follows:

- black—background (0);
- red—QRS block (1);
- blue—T wave (2);
- cyan—P wave (3).

In the figure, we can see that the model correctly segmented all the aforementioned ECG fragments regardless of the signal lead.

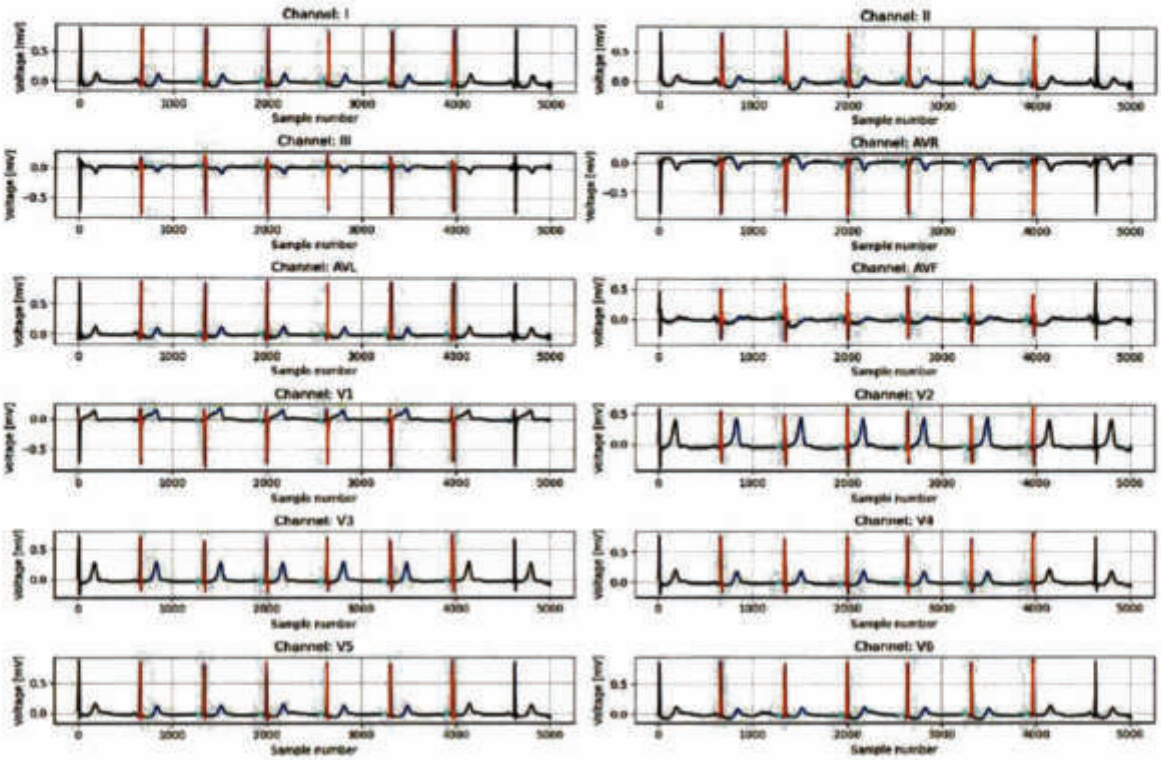


Figure 10. The result of the segmentation using the model on all twelve leads of the ECG signal.

4. Discussion

In this work, we presented a solution for extracting P and T-waves and QRS complexes from 12-lead ECG signals using methods that are mostly used in the image processing domain. The created model achieves a high set of performance parameters (accuracy, AUC, specificity, and sensitivity) independently on all signal leads. Thanks to this, our solution can be used in any type of electrode configuration as a basis for heart rhythm and heart rate variability (HRV) classification as well as many other parameters resulting from specific fragments of ECG. Using our method, we obtained results comparable or better than those presented in the literature [3–19].

According to the World Health Organization (WHO), the top cause of death worldwide in 2019 was ischemic heart disease [31], also showing the biggest increase in deaths since 2000. This disease has its representation in ECG as ST-segment elevation or depression. Taking into account the growing popularity of wearable devices (e.g., a smart watch or a smart band equipped with a simple ECG recorder), we have a chance for earlier detection of cardiovascular disorders by applying real-time segmentation methods and ECG analysis on an everyday basis for high-risk patients.

The deep learning approach seems to be an effective solution to the problem of medical signal segmentation. Considering the ability of neural networks to capture highly nonlinear patterns and their efficient feature extraction process, it can be used to help diagnose patients based on the signals captured from their body. These solutions are yielding state-of-the-art results and can be adopted as preliminary diagnosis systems.

For solving the problem of semantic segmentation of ECG, we have chosen to use U-Net like architecture. This solution has a few advantages. First, convolutions are less computationally expensive than for example the transformer model, which adopts point-wise dependencies discovery, whose complexity increases quadratically with the length of the time series, which given a time series of a 5000-samples length can easily become

intractable. In turn, the lower computation cost can be advantageous when trying to incorporate neural network models onto microprocessors to work directly on hardware. Given the nature of the problem—recognizing sequentially repeating signals—using convolution seems to be a suitable solution since it is sensitive to a local neighborhood of values.

A significant challenge in creating a system for ECG segmentation is finding an appropriate database for training the model or classifier. For this purpose, signal generators can be used, such as the one proposed by Stabenau et al. [32]. Artificial signals can be used to pretrain models that then are tuned to a real data.

In the future, we plan to replace the up-sampling layers of the decoder with one-dimensional transposed convolution layers that have learnable parameters. Based on the success of the transformer architecture [33] in natural language processing, we also consider replacing the convolution-based feature extractor with the mentioned architecture. In the future work, besides architectural changes, we also plan to extend our dataset with new categories that represent both healthy P and T-waves, ORS-complexes, and pathological. Another direction worth exploring in the future research is the use of a semantic segmentation approach to detect more complex, and non-sequential events in biomedical signals comparing a wider range of architectures, from 1D convolutional networks, Performer [34], and Linformer [35] and new architectures that seem to be a bridge between the two [36].

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app12073332/s1. File S1: Full diagram of the proposed model.

Author Contributions: Conceptualization, K.D. and P.K.; methodology, K.D. and N.P.; software, K.D.; validation, K.D. and P.K.; formal analysis, K.D. and N.P.; investigation, K.D. and E.T.; resources, K.D.; data curation, K.D.; writing—original draft preparation, K.D. and N.P.; writing—review and editing, K.D., N.P., P.K. and E.T.; visualization, K.D. and N.P.; supervision, P.K. and E.T.; project administration, N.P.; funding acquisition, K.D. and E.T. All authors have read and agreed to the published version of the manuscript.

Funding: This study was carried out under the project "InterPOWER—Silesian University of Technology as a modern European technical university", co-financed by the European Union under Measure 3.5 Comprehensive programs of universities III Priority Axis Higher education for the economy and development of the Operational Program Knowledge Education Development 2014–2020.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AF	Atrial Fibrillation
ANN	Artificial Neural Network
AUC	Area Under the Curve
CNN	Convolutional Neural Network
DL	Deep Learning
ECG	Electrocardiography
HRV	Hear Rate Variability
PVC	Premature Ventricular Contractions
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
WHO	World Health Organization

References

1. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2016.
2. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D.J. 1D convolutional neural networks and applications: A survey. *Mech. Syst. Signal Process.* **2021**, *151*, 107398. [CrossRef]
3. Faust, O.; Hagiwara, Y.; Hong, T.J.; Lih, O.S.; Acharya, U.R. Deep learning for healthcare applications based on physiological signals: A review. *Comput. Methods Programs Biomed.* **2018**, *161*, 1–13. [CrossRef] [PubMed]

4. Hong, S.; Zhou, Y.; Shang, J.; Xiao, C.; Sun, J. Opportunities and Challenges of Deep Learning Methods for Electrocardiogram Data: A Systematic Review. *arXiv* **2020**, arXiv:2001.01550.
5. Novotna, P.; Vicar, T.; Hejc, J.; Ronzhina, M.; Kolarova, J. Deep-Learning Premature Contraction Localization in 12-lead ECG From Whole Signal Annotations. In Proceedings of the Computing in Cardiology, Rimini, Italy, 13–16 September 2020. [CrossRef]
6. Du, N.; Cao, Q.; Yu, L.; Liu, N.; Zhong, E.; Liu, Z.; Shen, Y.; Chen, K. FM-ECG: A fine-grained multi-label framework for ECG image classification. *Inf. Sci.* **2021**, *549*, 164–177. [CrossRef]
7. He, R.; Liu, Y.; Wang, K.; Zhao, N.; Yuan, Y.; Li, Q.; Zhang, H. Automatic Detection of QRS Complexes Using Dual Channels Based on U-Net and Bidirectional Long Short-Term Memory. *IEEE J. Biomed. Health Inform.* **2021**, *25*, 1052–1061. [CrossRef]
8. Weimann, K.; Conrad, T.O.F. Transfer learning for ECG classification. *Sci. Rep.* **2021**, *11*, 5251. [CrossRef]
9. Zheng, G.; Ji, S.; Dai, M.; Sun, Y. ECG Based Identification by Deep Learning. In *Biometric Recognition*; Zhou, J., Wang, Y., Sun, Z., Xu, Y., Shen, L., Feng, J., Shan, S., Qiao, Y., Guo, Z., Yu, S., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 10568, pp. 503–510. [CrossRef]
10. Beraza, I.; Romero, I. Comparative study of algorithms for ECG segmentation. *Biomed. Signal Process. Control* **2017**, *34*, 166–173. [CrossRef]
11. Laguna, P.; Jané, R.; Caminal, P. Automatic Detection of Wave Boundaries in Multilead ECG Signals: Validation with the CSE Database. *Comput. Biomed. Res.* **1994**, *27*, 45–60. [CrossRef]
12. Martinez, J.; Almeida, R.; Olmos, S.; Rocha, A.; Laguna, P. A Wavelet-Based ECG Delineator: Evaluation on Standard Databases. *IEEE Trans. Biomed. Eng.* **2004**, *51*, 570–581. [CrossRef]
13. Singh, Y.N.; Gupta, P. ECG to Individual Identification. In Proceedings of the 2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems, Washington, DC, USA, 29 September–1 October 2008; pp. 1–8. [CrossRef]
14. Di Marco, L.Y.; Chiari, L. A wavelet-based ECG delineation algorithm for 32-bit integer online processing. *Biomed. Eng. Online* **2011**, *10*, 23. [CrossRef]
15. Sun, Y.; Chan, K.L.; Krishnan, S.M. Characteristic wave detection in ECG signal using morphological transform. *BMC Cardiovasc. Disord.* **2005**, *5*, 28. [CrossRef] [PubMed]
16. Martínez, A.; Alcaraz, R.; Rieta, J.J. Application of the phasor transform for automatic delineation of single-lead ECG fiducial points. *Physiol. Meas.* **2010**, *31*, 1467–1485. [CrossRef] [PubMed]
17. Vázquez-Seisdedos, C.R.; Neto, J.E.; Marañón Reyes, E.J.; Klautau, A.; Limão de Oliveira, R.C. New approach for T-wave end detection on electrocardiogram: Performance in noisy conditions. *BioMed. Eng. Online* **2011**, *10*, 77. [CrossRef] [PubMed]
18. Vitek, M.; Hrubes, J.; Kozumplik, J. A Wavelet-Based ECG Delineation with Improved P Wave Offset Detection Accuracy. *Anal. Biomed. Signals Images* **2010**, *20*, 160–165.
19. Hughes, N.P.; Tarassenko, L.; Roberts, S.J. Markov Models for Automated ECG Interval Analysis. In Proceedings of the NIPS 2003, Vancouver, BC, Canada, 8–13 December 2003.
20. Laguna, P.; Mark, R.; Goldberg, A.; Moody, G. A database for evaluation of algorithms for measurement of QT and other waveform intervals in the ECG. In Proceedings of the Computers in Cardiology 1997, Lund, Sweden, 7–10 September 1997; IEEE: Lund, Sweden, 1997; pp. 673–676. [CrossRef]
21. Kalyakulina, A.; Yusipov, I.; Moskalenko, V.; Nikolskiy, A.; Kosonogov, K.; Zolotykh, N.; Ivanchenko, M. Lobachevsky University Electrocardiography Database. Type: Dataset. Available online: https://physionet.org/content/ludb/1.0.0/ (accessed on 10 July 2021).
22. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:1505.04597.
23. Yan, W.; Hua, Y. Deep Residual SENet for Foliage Recognition. In *Transactions on Edutainment XVI*; Pan, Z., Cheok, A.D., Müller, W., Zhang, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 11782, pp. 92–104. [CrossRef]
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
25. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.
26. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. *arXiv* **2019**, arXiv:1709.01507.
27. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
28. Jaccard, P. The distribution of the flora in the alpine zone. 1. *New Phytol.* **1912**, *11*, 37–50. [CrossRef]
29. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2021**, *17*, 168–192. [CrossRef]
30. Keras—Sensitivity at Specificity | TensorFlow Core v2.8.0. Available online: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity (accessed on 14 March 2022).
31. The Top 10 Causes of Death. Available online: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death (accessed on 25 June 2021).
32. Stabenau, H.F.; Bridge, C.P.; Waks, J.W. ECGAug: A novel method of generating augmented annotated electrocardiogram QRST complexes and rhythm strips. *Comput. Biol. Med.* **2021**, *134*, 104408. [CrossRef] [PubMed]
33. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
34. Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. Rethinking Attention with Performers. *arXiv* **2020**, arXiv:2009.14794.

35. Wang, S.; Li, B.Z.; Khabsa, M.; Fang, H.; Ma, H. Linformer: Self-Attention with Linear Complexity. *arXiv* **2020**, arXiv:2006.04768.
36. Li, D.; Hu, J.; Wang, C.; Li, X.; She, Q.; Zhu, L.; Zhang, T.; Chen, Q. Involution: Inverting the Inherence of Convolution for Visual Recognition. *arXiv* **2021**, arXiv:2103.06255.

Chapter 4

A new paradigm

Deep learning systems in healthcare must be transparent and interpretable in a way that encourages public confidence in the algorithms and models that will be used to make critical decisions. This is especially true for healthcare applications, where explainability is essential for clinical judgments. When a deep learning model produces a prediction or delivers a diagnosis, it is vital that physicians are able to understand the reasoning behind the decision-making process in order to provide the best possible care to patients. The adoption of these systems in healthcare can suffer if a model comes to a conclusion that cannot be justified or understood. This might result in mistrust and skepticism toward the technology. In biomedical sciences, it is crucial that decisions made by neural networks can be traced back to specific features, their combinations, and data points in training data. This is important for identifying possible biases or weaknesses in the models and for confirming their precision and applicability in various clinical scenarios. Without it can be difficult to ensure the safety of deep learning systems in healthcare. Interpretable models can bridge the gap between clinicians and data scientists, which in turn can provide better care for the patient.

4.1 Limits of deep learning

Although deep neural networks have achieved spectacular results, they also suffer from many flaws. In the paper [47] authors showed that deep neural networks can be easily fooled while performing image classification by adding noise to the input data, which did not affect the perceived object as seen by humans. Garry Marcus, a professor of psychology and neural sciences at New York University, has written a paper on problems that deep learning has not yet solved [48]. He pointed out several limitations, such as

- They are data hungry - neural networks perform well under three conditions:

  1. The data set on which they were trained is large and annotated,
  2. The data set on which they were trained is diverse,
  3. The architecture of the model is large enough.

  Human or animal intelligence needs a few examples to solve a particular task, which is why Marcus thinks that more work should be put into unsupervised deep learning.

- They do not create compositional representations - this can be understood better when considering image classification task. Although architectures such as convolutional neural networks do build up a higher-level feature representation for determining an object's category, they do not deconstruct such an object into its distinctive parts.
- They are not sufficiently transparent,
- The prior knowledge cannot be well integrated,
- They are unable to perform symbolic manipulation,
- They are unable to distinguish causation from correlation.

In recent years, there has been a development of so-called explainable artificial intelligence (XAI) methods. These methods were supposed to allow developers of deep learning algorithms to gain insight into the inner workings of a neural network. Explainability is essential in deep learning models to ensure accountability, transparency, and trustworthiness.

One popular technique for explainability is SHAP (SHapley Additive exPlanations) [49], which is a game-theoretic approach to explain the output of any machine learning model. SHAP provides a unified approach to explain the output of any model by calculating the importance of each feature in the prediction. This technique allows us to identify the features that are most important in making the prediction, which can help us gain insight into how the model works.

Another popular explainability algorithm is LIME (Local Interpretable Model-Agnostic Explanations) [50], which is a method for explaining the predictions of any black-box model by approximating it with a simpler model. LIME creates a local approximation of the model at the instance level by creating a linear model that approximates the complex model's behavior. This method helps us identify which features of the input data influenced the prediction, which can be helpful in identifying data biases.

GradCAM (Gradient-weighted Class Activation Mapping) [51] is another explainability algorithm that can be used to identify the most important parts of an image that led to a particular prediction. GradCAM works by using the gradients of a specific class with respect to the final convolutional layer to generate a heat map of the important regions in the image. This technique allows its users to understand which parts of the image the model is focusing on and which features of the image are most important in making the prediction.

Although algorithms such as SHAP, LIME, and GradCAM are useful techniques for explainability in deep learning models, they are not always sufficient alone. These methods have the drawback that they only offer a partial understanding of how the model functions. For instance, SHAP and LIME are often used to highlight the significance of input variables in a prediction, but they do not necessarily shed light on the model's decision-making process. Similarly to this, GradCAM is a useful tool for locating critical areas in a picture, although it might not fully explain how the model came to its conclusion.

Moreover, these methods might not always apply to explain the behavior of more intricate models, such as deep neural networks with numerous layers. In these scenarios,

the explanations offered by these methodologies could be challenging to understand, and the insights that follow could be narrow. Additionally, these techniques may not always capture the full complexity of the decision-making process, particularly when dealing with more nuanced or abstract concepts.

Furthermore, these approaches might not always be able to spot biases or ethical issues in the model. Although explainability is crucial for ensuring that models are reliable and open, it does not necessarily guarantee that its predictions are ethical or free from bias. For example, SHAP or LIME may identify certain features as being important in a prediction, but they may not necessarily identify whether these features are biased or discriminatory.

4.2 Best of both worlds

Both neural networks and fuzzy systems have certain disadvantages which almost completely disappear by combining both concepts. They can be used for solving a problem (e.g. pattern recognition, regression, or classification) if there does not exist any mathematical model of the given problem.

Neural networks and deep learning techniques come especially useful when dealing with high-dimensional, complex, nonlinear data, where feature extraction done by handcrafted algorithms does not scale. That being said, they can only be applied when a sufficient amount of observed examples are provided. On the one hand, no expert knowledge of the problem needs to be given. On the other hand, it is not straightforward to extract comprehensible rules from the neural network structure.

In contrast, a fuzzy system demands linguistic rules instead of learning examples as prior knowledge. Furthermore, the input and output variables have to be described linguistically. If the knowledge is incomplete, wrong, or contradictory, then the fuzzy system must be tuned. Since there is no formal approach for it, the tuning is performed in a heuristic way. This is usually very time-consuming and error-prone.

From the description of both deep learning and fuzzy logic, one might see that from combining the best of both worlds, a system capable of both human-like reasoning and learning might emerge. Both pros and cons of both were summarized in a table 4.1.

In summary, current notions of explaining the decision-making process of a deep neural network are insufficient for real-world scenarios. With the mentioned methods we can only get a glimpse of the prediction-relevant information but we do not get a whole picture, which is crucial considering the possible applications of these models in biomedical engineering and life sciences.

From the description of both deep learning and fuzzy logic, one might see that by combining the best of both worlds, a system capable of both human-like reasoning and learning might emerge. In the past, researches have tried to integrate both algorithms into a single structure that would enable learning and expressing knowledge in a fuzzy logic manner.

Deep learning	Fuzzy Systems
Highly complex models	Simple models
Require large amounts of data	Can work with small data sets
Can be sensitive to data bias	Can handle data bias
Difficult to interpret	Transparent and interpretable
Require significant computational resources	Can be computationally efficient
Can be prone to overfitting	Robust to noisy data
Cannot be easily modified	Can be easily modified
Learns from input exampels	Everything is defined explicitly
Difficult to extract knowledge	Knowledge is easily extracted
Difficult to use prior/expert knowledge	Expert knowledge is easy to incorporate
Does not perform reasoning in human-like manner	Performs reasoning the resembles humans

Table 4.1: Comparison of deep learning and fuzzy systems

4.3 Neuro-Fuzzy Inference Systems

Fuzzy inference systems represent an important part of fuzzy logic. In most practical applications (i.e. control), such systems perform crisp non-linear mapping, which is specified in the form of fuzzy rules encoding expert or common sense knowledge about the problem at hand [52]. There are three main types differentiated in the available literature.

4.3.1 Cooperative Fuzzy Inference Neural Networks

In this setting, an artificial neural network and a fuzzy logic inference system work independently from each other. The neural network tries to learn parameters of a fuzzy system - either membership functions or fuzzy rules. Once the fuzzy inference system parameters are obtained, the neural network goes to the background [53]. There exist four different settings for this type of neuro-fuzzy inference system:

- Fuzzy neural network learns the fuzzy set from the given training data. This is usually done by fitting the membership functions with a neural network. The fuzzy sets are then determined offline. They are then utilized to form the fuzzy system by fuzzy rules that are also given (not learned) [54]. This type of cooperative FINN is described in Figure 4.1a.

- Neuro-fuzzy system determines fuzzy rules from training data by a neural network. Here, as well, neural networks learn offline before the fuzzy system is initialized. The rule learning is usually done by clustering on self-organizing maps [55, 54]. This type of cooperative FINN is described in Figure 4.1b.

- Neuro-fuzzy system learns all membership function parameters online, i.e., while the fuzzy system is applied. Thus, fuzzy rules and membership functions must

initially be defined beforehand. Moreover, the error has to be measured to improve and guide the learning step [54]. This type of cooperative FINN is described in Figure 4.1c.

- The neural-fuzzy system determines rule weights for all fuzzy rules by a neural network. This can be done online and offline. A rule weight is interpreted as the influence of a rule. They are multiplied by the output of the rule [54]. This type of cooperative FINNs is described in Figure 4.1d.

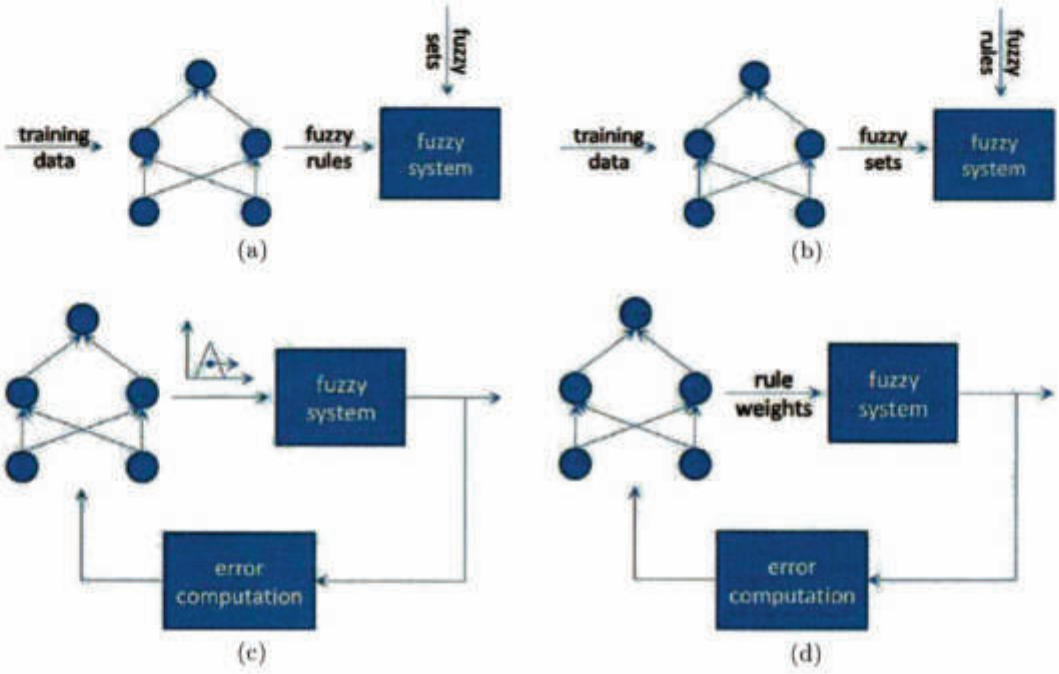


Figure 4.1: Cooperative FINNs types [54].

4.3.2 Concurent Fuzzy Inference Neural Networks

In this setting neural network aids the fuzzy system continuously (or vice versa) to obtain the required parameters especially if the input variables of the controller cannot be measured directly. These kinds of combinations do not update the fuzzy system but only help to enhance the performance of the overall system. Learning occurs only in the neural network, and the fuzzy system remains unchanged during this phase [53].

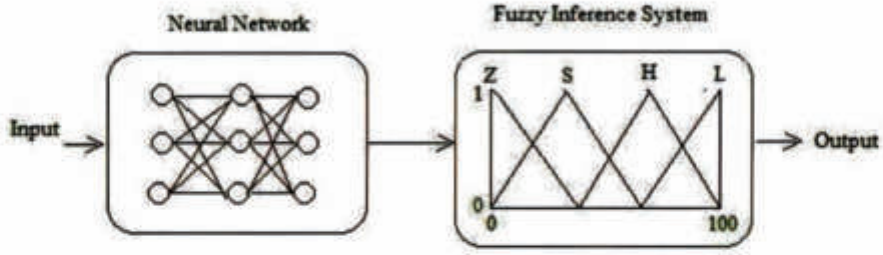


Figure 4.2: Concurent FINN [53].

4.3.3 Hybrid Fuzzy Inference Neural Networks

In this setting, the fuzzy system and neural network are one fully merged system. The fuzzy system utilizes the learning algorithm since it is depicted in a special neural network-like structure. These hybrid models have supervised learning abilities. They are made up of nodes, which depend on parameters that are updated based on certain learning algorithm to reduce error measure and links that always show the direction of information flow. The adaptive network received much attention from various fields as a result of its enticing features such as speedy convergence, accurate learning, ease of use, tolerating uncertainties, and imprecise information. The learning algorithm for updating the fuzzy inference system (FIS) parameters can be a back-propagation or hybrid learning algorithm [53]. Hybrid neuro-fuzzy systems are homogeneous and usually resemble neural networks. Here, the fuzzy system is interpreted as a special kind of neural network. The advantage of such hybrid FINN is its architecture, since both fuzzy system and neural network do not have to communicate with each other anymore. They are one fully fused entity. These systems can learn online and offline. The rule base of a fuzzy system is interpreted as a neural network. Fuzzy sets can be regarded as weights, whereas the input and output variables and the rules are modeled as neurons. Neurons can be included or deleted in the learning step. Finally, the neurons of the network represent the fuzzy knowledge base. Obviously, the major drawbacks of both underlying systems are overcome [54]. A diagram of the information flow in such a system is presented in Figure 4.3.

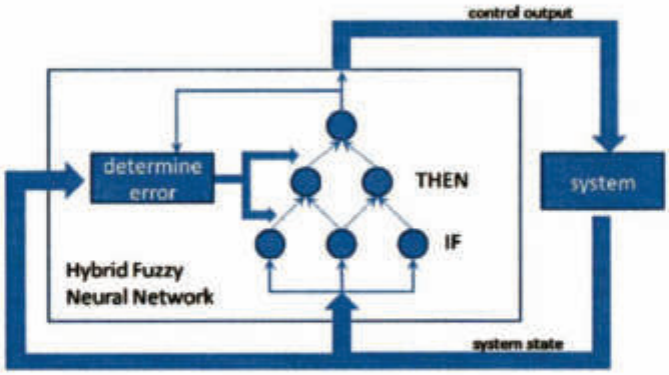


Figure 4.3: Hybrid FINN [54].

In the late 1990s and early 2000s artificial intelligence research community focused

on developing these hybrid models. As a result, few variations became mainstream.

- *Mamdani Integrated Fuzzy Inference System* uses the method of gradient descent. to learn the parameters of a membership function [28]. In this method, you try to find the best values of a parameter by minimizing a cost function. It consists of five functional layers:
  - **Input Layer** - this layer consists of direct inputs. Each node in this layer consists of individual inputs that are passed directly to the nodes of the next layer [56],
  - **Fuzzification Layer** - this layer is responsible for converting crisp inputs passed from the input layer into Fuzzy Sets. It tries to find the degree of membership of each individual input value in the fuzzy set. Fuzzy clustering approaches are used to define the number and type of membership functions for each input variable. Throughout the process of backpropagation, the membership function's numbers and types of the membership function will continue to change to fine-tune the entire system [56],
  - **Rule Antecedent Layer** - this layer is used to define the antecedents of the rule base The output of this layer is the firing strength of the corresponding fuzzy rule [56],
  - **Rule Consequent Layer** - it is used to determine the consequents for each rule antecedent. It helps to define the membership of each antecedent to the output value. The number of nodes in this layer is equal to the number of rules in the previous layer [56],
  - **Defuzzification Layer** - in this layer, all the rule consequents are combined using the T-Co-Norm operation [57] and then converted into crisp outputs using the defuzzification approaches [56].

The architecture of the Mamdani Integrate FIS is depicted on Figure 4.4

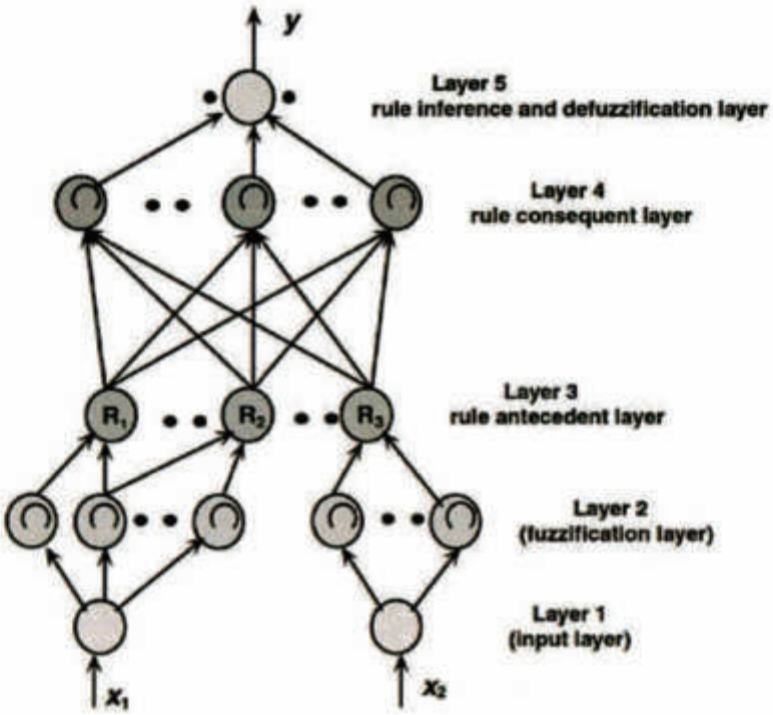


Figure 4.4: Mamdani integrated architecture [56].

- *Takagi-Sugeno Integrated Fuzzy Inference System* - works similarly to the Mamdani Integrated FIS. The difference is that it has an additional layer for rule strength normalization. It works by determining the firing strength of each rule and then dividing it by the sum of all the firing strengths of the rules following the equation 4.1.

$$w_i' = \frac{w_i}{\sum_{i=0}^{n} w_i} \tag{4.1}$$

The fifth layer is used to determine the consequents of the rule using the least-squares estimation method 4.2

$$w_i' f_i = w_i'(p_i x_1 + q_i x_2 + r_i) \tag{4.2}$$

In this equation, p, q, and r define the consequent parameters. The final layer is used to aggregate all the output from the previous layer 4.3

$$y = \frac{\sum_{i=0}^{n} w_i f_i}{\sum_{i=0}^{n} w_i} \tag{4.3}$$

The diagram for Takagi-Sugen FIS is presented by the Figure 4.5

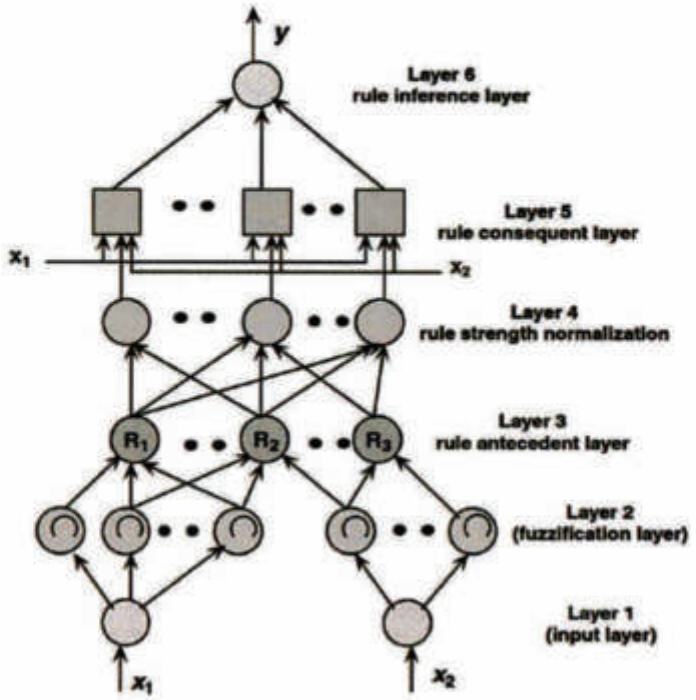


Figure 4.5: Takagi-Sugeno integrated architecture [56].

- *Tsukamoto Fuzzy Inference Neural Network* - in this model, instead of having a constant or linear output of fuzzy memberships, the system outputs a monotonic membership function, which is defuzzified using the weighted average approach. Since it is a weighted average approach, the process becomes very fast, and hence time is not wasted during the detailed process of defuzzification. The output of the Tsukamoto fuzzy inference sysetm is always crisp, no matter what the input types are [56]. However, the Tsukamoto fuzzy model is not used often since it is not as transparent as either the Mamdani or Sugeno fuzzy models.

- *Adaptive Neuro Fuzzy Inference Systems (ANFIS)* - using this system, researches could represent both the Sugeno and Tsukamoto systems, hence its name, adaptive network, as with one network, you can represent multiple networks with minor changes [56]. The Mamdani Fuzzy Inference System can also be represented using ANIFS. It is the most widely used neuro-fuzzy system for modeling extremely complex nonlinear systems [58]. The most challenging part of designing this system is determining the type and number of membership functions, suitable to the process or system. ANFIS is generally very efficient until the number of inputs is less than five [58]. The overall architecture is presented on Figure 4.6

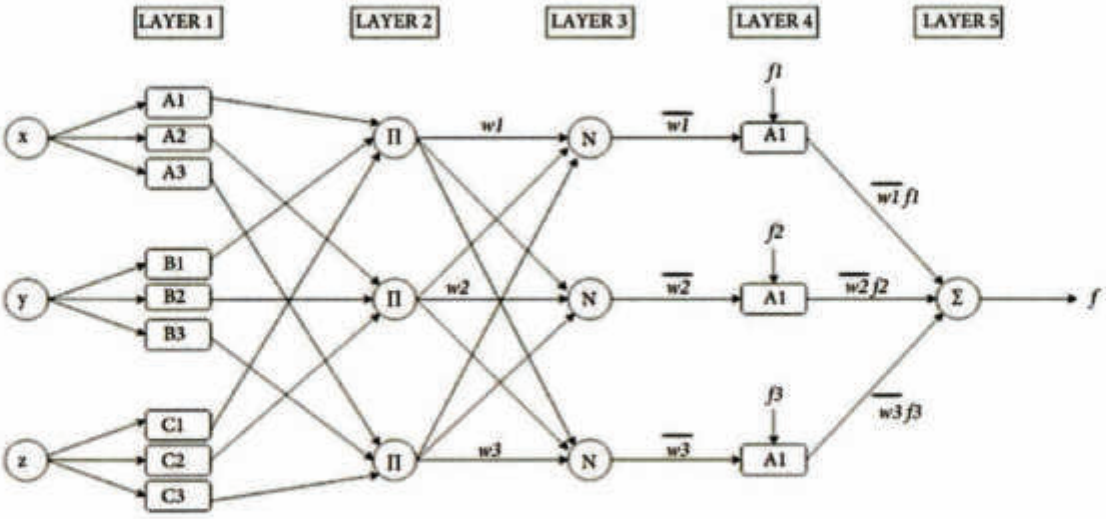


Figure 4.6: ANFIS architecture [58].

4.3.4 Limits of Neuro-Fuzzy Inference Systems

Neuro-Fuzzy Inference Systems were once one of the most powerful and exacting techniques in machine learning. But their limits were quickly uncovered, and they have fallen behind more efficient and scalable machine and deep learning methods. The first big problem is the fact that they still cannot process the unstructured data, such as text and images. Algorithms such as *ANFIS* cannot be adapted to more widely used architectures such as convolutional neural networks, recurrent neural networks, or transformers. Another reason lies in their architecture. By merging multilayer neural network with fuzzy logic system into a single object, researchers have putted a constraint regarding the deepness of the neural network. From recent developments in the deep learning field, it has become known that those algorithms given enough data create levels of abstraction which create linear separation in the input data space. By constraining the number of layers of a neural network, we are basically making it unable to correctly separate data points, thus worsening their overall performance.

4.4 Fuzzy Representation Learning

"Prediction is the essence of intelligence." —Yann LeCun [59]

4.4.1 Intuition

Many of the leading scientists in the field of artificial intelligence and neuroscience believe that the essence of intelligence is the ability to make predictions. Although it is hard to argue against it because at the end of the day prediction is necessary for any learning system to improve, it also seems that something is missing. When we humans make predictions, we do not make them directly from sensory input. Instead, after receiving the sensory stimulus, we seem to evaluate the sensory input against our internal world model to finally make predictions [60]. When tuned, this so-called world

model works as an inference engine to evaluate course of actions or to make predictions about observed phenomenon directly.

The progression of learning algorithms over the last few decades has been based on further automation. We went from implementing entire logic of the system (expert systems) to automating the decision making process based on features computed in the feature extraction process (machine learning), to finally automating both feature extraction and decision making process (deep learning). On the one hand, we have learned that knowledge derived from data, through the learning process, is the only way forward. On the other hand, by automating almost the entire process, we have abandoned crucial elements of any real-world system, such as transparency, explainability, knowledge derivation, and others which were discussed in greater detail in chapter 4.1.

This chapter proposes a new paradigm, which in its principles resembles the neuro-fuzzy inference system, but mitigates the constraints put upon these systems. In its essence, this system is about making deep neural networks create hypothesis instead of making predictions. The hypothesis/explanation of the problem takes the form of a fuzzy inference system. This solution utilizes the power of making the models deeper, can be applied to any neural network architecture, and provides fully-explainable, interpretable, and transparent solution. In their current state, neural networks use the biases in the data to make their predictions, which leads to biased predictions - essentially, deep neural network trained on the data set with high imbalance, low diversity and insufficient data set size will make predictions correlated with the categories they were trained on, but their prediction will not reflect the actual task we wanted them to solve. Instead of being prediction machines which will "tell" its trainer anything that he/she wants to "hear", their decision making process should resemble more the scientific method, in which prediction about the future or the state of the world is made through stating a hypothesis.

In summary, prediction is necessary for any learning system, but it should not be the goal of the learning system. Sensory information should be evaluated on the basis of the stated hypothesis, which in turn generates a prediction. Thus, prediction is just a by-product of the learning system, the explanation is what we should be optimizing for.

4.4.2 Algorithm

The main idea behind this algorithm is that a deep neural network based on input data must produce a complete fuzzy logic system which is then used to make a prediction. There are no rules specified by humans; the system converts information from input to fuzzy system to prediction in the end. The steps of the algorithm are as follows:

1. **Feature extraction** - given the input data, we pass it through the deep neural network to extract relevant features and obtain the latent representation of the feature vector,

2. **Defining fuzzification of the features** - the neural network, from its latent representation of the input data produces a matrix describing the fuzzification process for features. The matrix is of shape: *number of features* x *granularity* x *parameters* and is called an *FGP* matrix,

3. **Fuzzy adjectivity scores for features** - this step focuses on computing the adjectivity score of the input features for all G adjectivity/membership functions in a univariate manner. A fuzzy adjective is a vector describing particular feature membership to all of G adjectivity functions,

4. **Crisp adjectivity scores** - in this step, we compute the crisp values for all fuzzy-adjectinvess vectors, for all input features. This operation produces a vector called an *adjective vector*. This operation is carried out using a softargmax function [61].

5. **Defining fuzzification of the classes** - the neural network, from its latent representation of the input data produces a matrix describing the fuzzification process for class membership. The matrix is of shape: *number of classes* x *class granularity* x *number of features* x *parameters* and is called an *CMFP* matrix,

6. **Fuzzy adjectivity scores for classes** - this step focuses on computing the membership score of the adjectives for all M membership functions in a multivariate manner. A fuzzy class is a vector describing the membership of a particular class to all the M membership functions,

7. **Summation** - we compute the summation for M-membership functions,

8. **Output computation** - compute the *softargmax* for the summed membership values to get the output class.

The figure 4.7 illustrates the architecture of such a system.

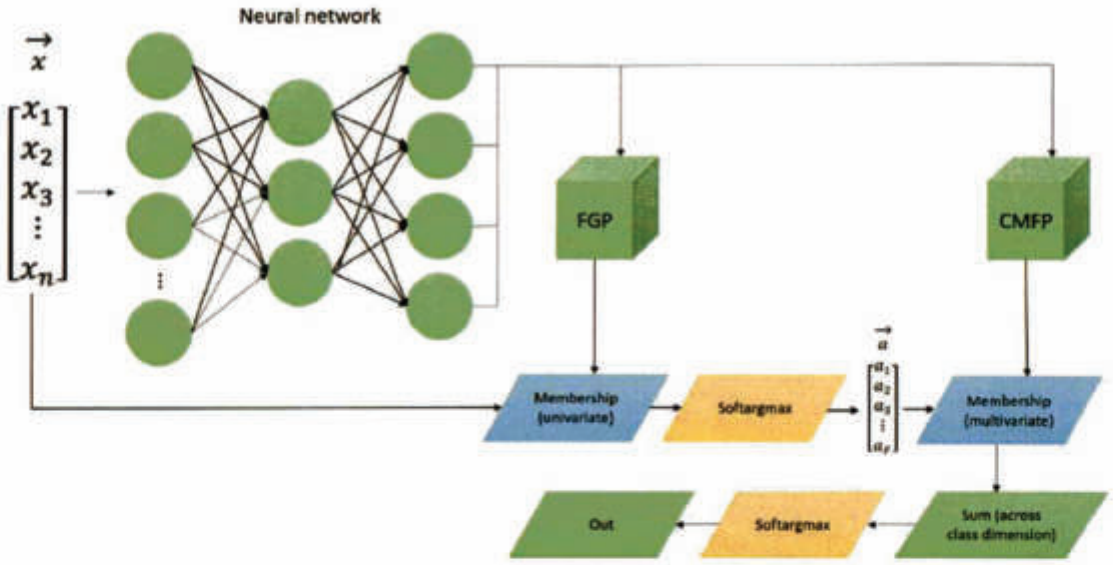


Figure 4.7: Overall system architecture.

For both feature fuzzy adjectivity scores and class memberships, various functions can be used. While implementing the Gaussian function was employed. For features we compute univariate version, and for class membership the multivariate case was used. The Gaussian membership function is described by Equation 4.4

$$e^{-0.5[(X-\mu)^T \sum^{-1}(X-\mu)]}, \tag{4.4}$$

where:

- X - denotes the input vector,
- μ - denotes the vector describing the centers of the Gaussian functions,
- $\sum$ - denotes the covariance matrix.

Individual covariances were set to zeros to make computation easier. The σ denotes the variance of a Gaussian function, which is defined as 4.5.

$$\sum = \begin{bmatrix} \sigma_{11}^2 & 0 & \dots & 0 \\ 0 & \sigma_{22}^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{KK}^2 \end{bmatrix} \tag{4.5}$$

For computing the vector of adjectives and final prediction, the *softargmax* [61] function was used instead of *argmax*, for two reasons:

- it is differentiable, which is a crucial property for modules used along deep neural networks,
- it allows for a degree of uncertainty - if two values are close together, given a specific parameterization of this function, it allows for measuring the performance of a model.

The softargmax function is defined by equation 4.6:

$$f(x) = \sum_{i=0} \frac{e^{\beta * x_i}}{\sum_{j=0} e^{\beta * x_j}} * i, \tag{4.6}$$

where:

- β - adjustable parameter, the higher it is, the more precise the output is,
- x - input vector.

4.4.3 An example solution

As a proof of concept, the XOR logic gate (exclusive OR) was adapted as a problem to be solved. The XOR is a logic gate that gives a true (high-state) output when the number of true inputs is odd; otherwise the state is false (low-state). It is also considered as the simplest nonlinear problem. To describe this problem in the setting of this algorithm, the granularity of feature membership functions (number of adjectives to describe a feature) was set to two (each feature can be either high or low). The granularity of the class memberships was also set to two because each combination of feature adjectives can be assigned to two possible states within one class (each class can be in a high or low state). In summary, the FGP feature descriptor and the CMFP

class descriptor have the shapes *2x2x2* and *2x2x2x2* respectively. The neural network was a standard feedforward neural network with two hidden layers, each consisting of 128 neurons followed by the ReLU activation function [62].

The model was trained using the binary cross-entropy function, which is a standard for training deep neural networks for the classification task [63].

The model was trained for 100 epochs, resulting in a final loss of 0.1231. The evolution of the fuzzy system is presented in a series of figures 4.8-4.14.

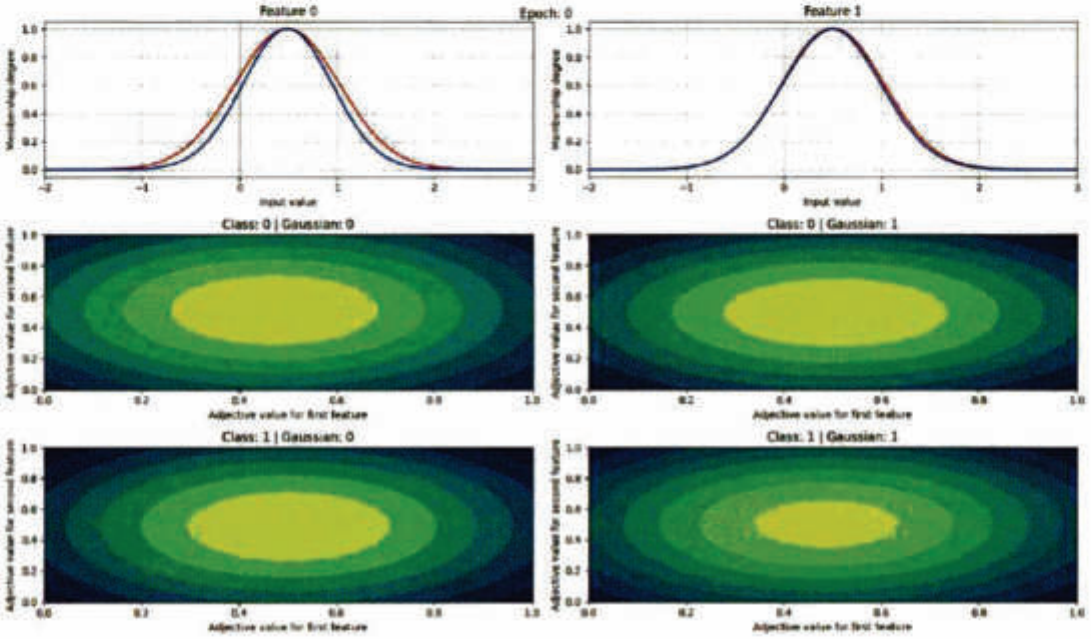


Figure 4.8: Fuzzy representation at first epoch.

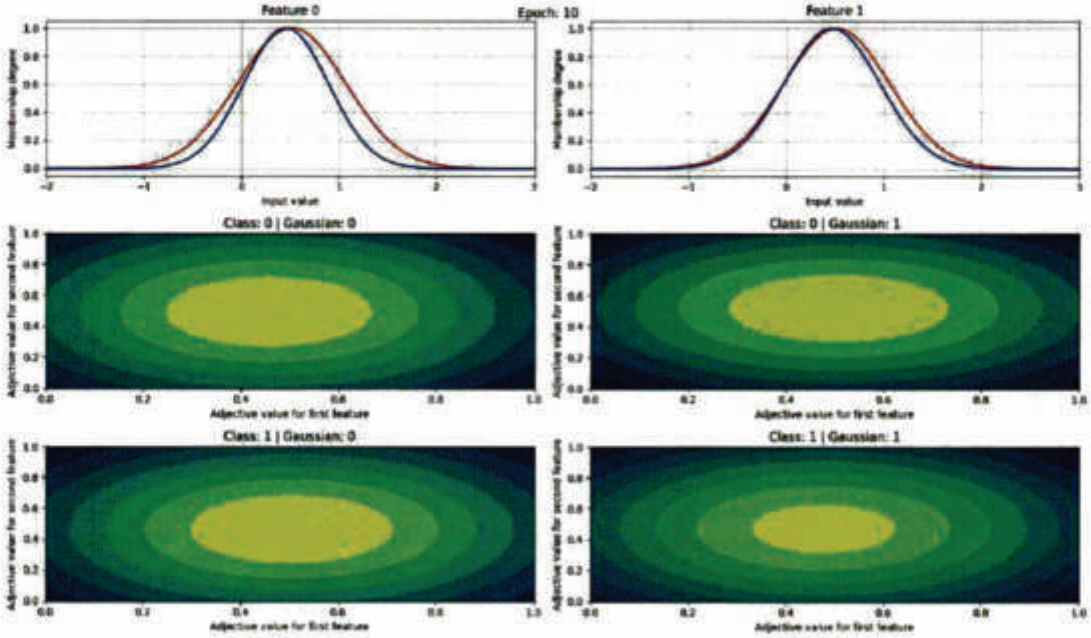


Figure 4.9: Fuzzy representation at 10-th epoch.

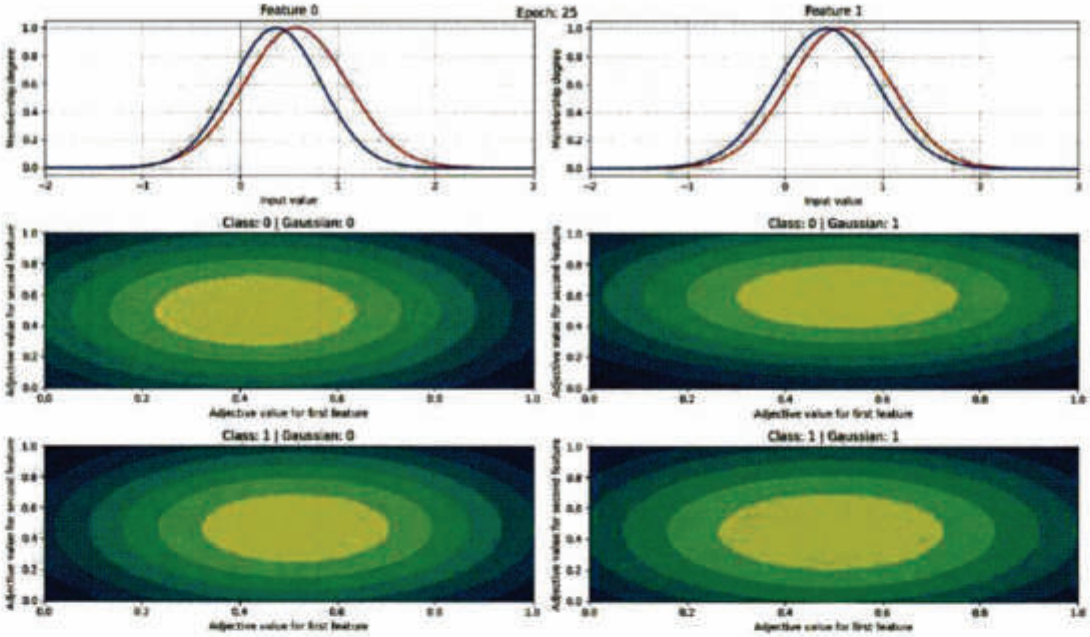


Figure 4.10: Fuzzy representation at 25-th epoch.

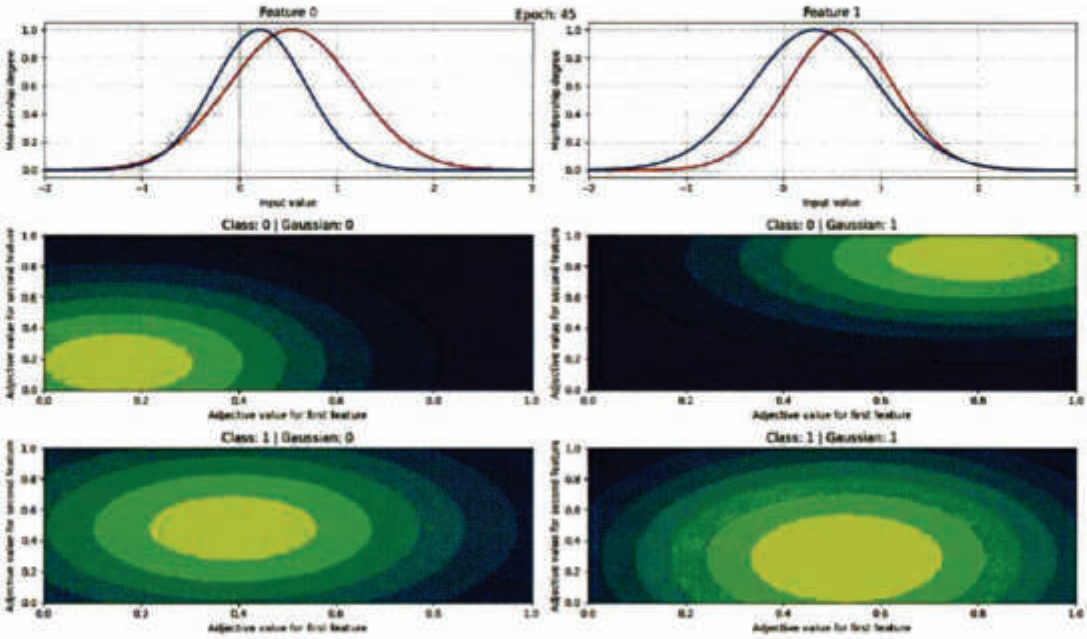


Figure 4.11: Fuzzy representation at 45-th epoch.

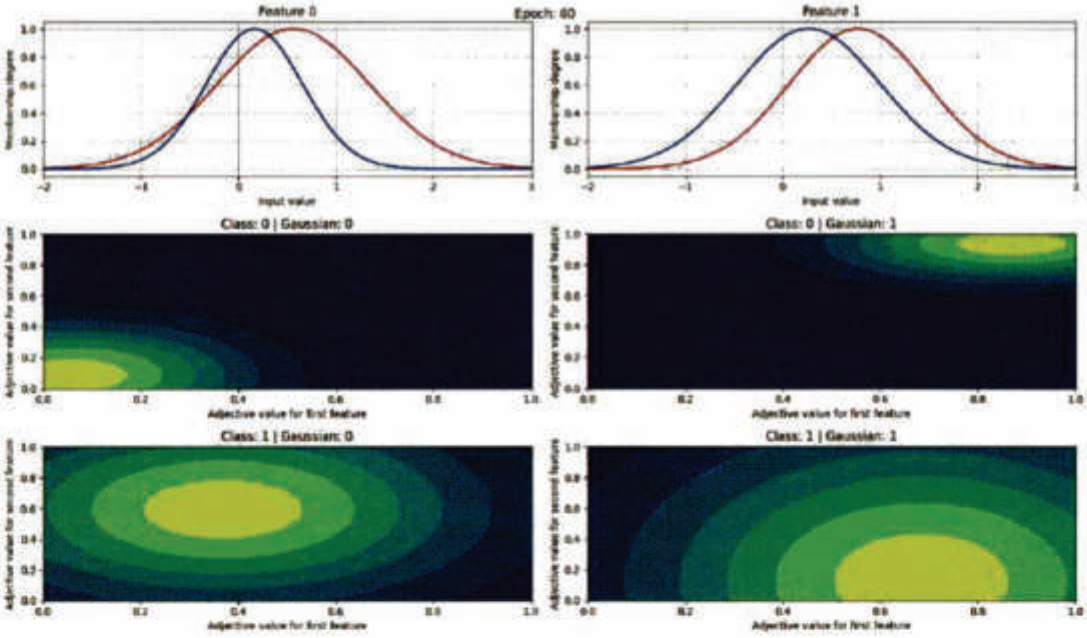


Figure 4.12: Fuzzy representation at 60-th epoch.

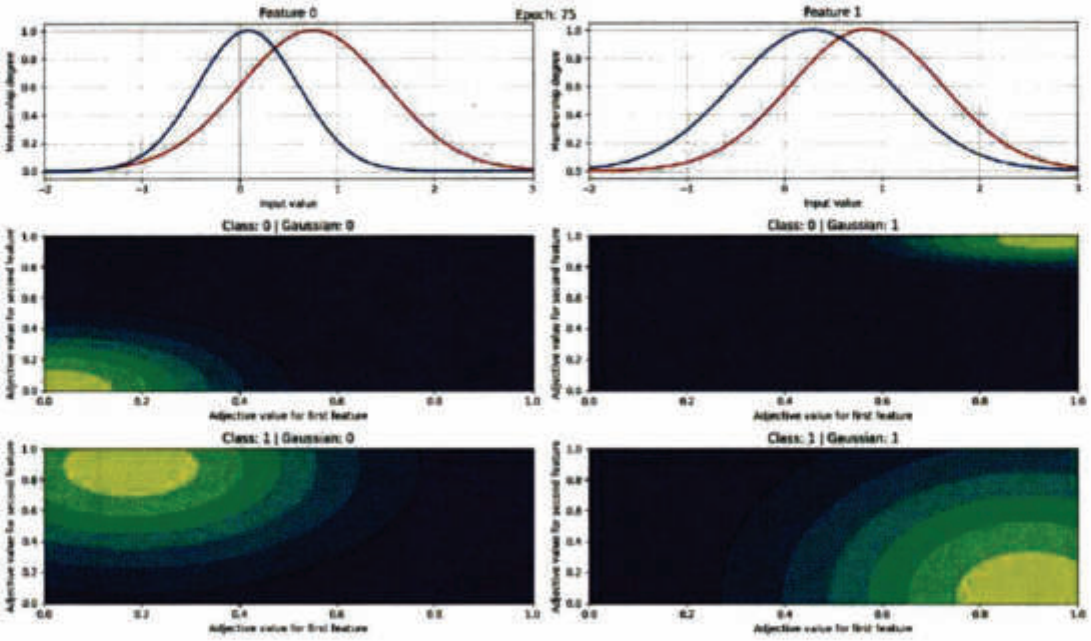


Figure 4.13: Fuzzy representation at 75-th epoch.

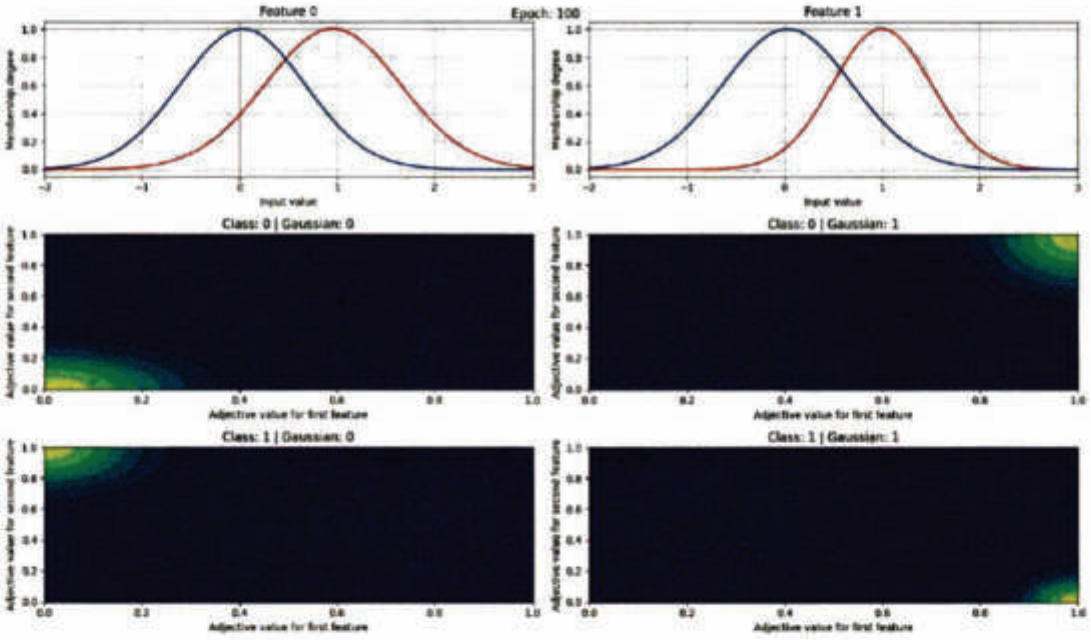


Figure 4.14: Fuzzy representation at 100-th epoch.

The 4.14 describes the final fuzzy inference system. The first row presents how the neural network assigns the "adjectivity" score to the input variables. The blue line indicates when the network "thinks" of its input as being "small", and the red one denotes when the neural network thinks its input as "big". The next two rows

describe how the algorithm assigns the class memberships for an adjective vector. The second row and first column image describes the membership to the low-state class (0) by considering when the adjectives for both features are "small". The same logic is applied to every other class membership presented in 4.14.

From the results presented above, it is clear that the deep neural network solved the XOR problem using the fuzzy logic system. However, the solution is not perfect and will be discussed in the next section.

4.4.4 Summary

Although it may seem otherwise, this work is essentially different from the approaches described in chapter 4.3.

The proposed solution may resemble the third type of cooperative fuzzy inference neural network. The main counterargument to classifying this solution as the third type of cooperative fuzzy inference neural network is that, in the existing literature, this setting of the fuzzy inference system is described as working independently of the neural network [54]. In the paper [64] authors described the cooperative fuzzy inference system in a more precise way, but they also added that the structure is not totally interpretable, which is also not the case for the proposed solution. The clear example of a cooperative neural network and a fuzzy logic system was described in [65] where the neural network was used to correct the output of the fuzzy system in inference or uses them to predict another parameter. In that case, the proposed solution does not fit the cooperative type.

The presented algorithm may also seem like a concurrent type at first glance, because there is a direct link between the neural network's output and the fuzzy inference system, but it cannot be because the neural network does not serve as a pre-processing step in the proposed solution.

This paper describes a system that combines both a neural network and a fuzzy logic inference system into a single structure, optimized by a single objective function. Therefore, this method can be classified as a hybrid one, although unlike the solutions presented in chapter 4.3 in this setting, the fuzzy inference system is not built into the structure of the neural network, rather the neural network proposes a fuzzy inference system.

Taking into account all the arguments presented above, the proposed method may be classified as a hybrid fuzzy inference neural network, but given its structure, information processing paths, and intuition behind it, the most fitting name would be fuzzy representation learning.

There are a number of things that are important to add or improve with the current solution. For the future work, two variants can be considered, short- and long-term. To enumerate some of them:

1. **Short-term**:

    - Adding an optimizeable constraint on membership function variance, as can be seen in figure 4.14 the memberships are not perfect. Membership functions describing both high and low states of an input feature should approach the delta function for this case. The same goes for class memberships - the variance of the function should be minimized;

- Adding additional membership functions;
- Making the feature and class granularity an optimizable parameter, so the additional work is taken off of the user;
- Adding the mechanism of mixing different membership functions;
- Testing this solution on other, more challenging datasets.

2. **Long-term**

   - Creating a reusable package that can be used and evaluated by other researchers;
   - Extending this idea to work on time-series data, images and text;
   - Evaluate possibility for different learning paradigms (beyond purely supervised).

This chapter presents a new method that combines deep learning methods and fuzzy logic. As stated in numerous papers, the potential that comes from merging these two techniques is enormous, but yet no effective solution has been found. With fuzzy representation learning, there is hope that this type of information processing may be useful in a variety of machine learning problems.

By learning to create fuzzy systems, the model has to conceptualize the problem itself and not simply make a prediction that can be caused by a correlated variable in a data set. Because the output of the model is fully explainable, the knowledge discovered by the model can be accessed directly by a human in an understandable manner.

Likewise, there exists a possibility of going the other way around. Fuzzy systems can be directly understandable by humans and now by neural networks as well. One can imagine a communication mechanism between the user and the neural network by expressing the user's will and knowledge by adjusting the fuzzy system (maybe even via a rule-like mechanism).

This solution may also enable rapid learning. By learning fuzzy representation of one task, the representation may be stored in a form of fuzzy system, function, or even a set of if-else conditions. By finding the similarity measure between the tasks, the model may just used the already learned fuzzy representation and quickly solve the new task at hand, without much computation and without the catastrophic forgetting (because it would not be trained to solve a particular task).

Finally, the biggest advantage of this system is that its task is not to make a prediction (at least not directly), its task is to state a hypothesis. By making it solve only one problem, which is a redundancy, the real power may come from training the same model to produce fuzzy representations for a variety of different problems, scaling it as much as possible.

In summary, this chapter was a proof-of-concept work for a new kind of algorithm. The tests were carried out and the results are satisfactory. Although there is much work to be done to make this solution applicable to standard machine learning data sets and real-world data. The example, toy problem shows that it is actually possible to train such a system, whose benefits may be significant.

Chapter 5

Conclusions

This thesis describes the potential of deep learning applications in the biomedical engineering domain. It starts by outlining the history of artificial intelligence and development of different approaches, such as expert systems, machine learning, and finally deep learning.

Deep learning has great potential in the field of biomedical engineering. It can be used for tasks such as medical image analysis, medical signal processing, drug discovery, molecules classification, and disease diagnosis, potentially improving patient outcomes and accelerating the development of new treatments.

Molecule toxicity classification is the process of determining the potential toxicity of a chemical compound or molecule. It is an extremely important subject for several reasons such as human safety, environmental protection, drug development, and regulatory compliance. Chapter 3.1 presents an example solution in which a deep neural network is applied to perform toxicity estimation of a molecule based on its string-encoded representation.

Medical signal processing is an important field of study that involves the analysis and interpretation of physiological signals generated by the human body. These signals can be used to diagnose, monitor, and treat a wide range of medical conditions or to examine human behavior under different conditions. Chapter 3.2 presents an example application in which deep neural networks can help evaluate differences in driving styles by recognizing the action performed based on the EOG signal.

Extracting specific segments of the electrocardiogram (ECG) signal, such as the P-wave, T-wave, and QRS complex, is important because it can help diagnose cardiac abnormalities, monitor cardiac function, evaluate drug therapies, and minimize risk of potential cardiac failure thanks to constant monitoring. That is why precise and efficient estimation of these segments is extremely important. Chapter 3.4 describes a neural network architecture and training procedure which results in a model capable of accurate segmentation of P-wave, T-wave and QRS complex. This model can be directly deployed on hardware, which makes it extremely useful in the context of real-world applications.

Although segmenting different parts of signals such as the ECG, which is a well-established medical exam, is important, there is also a need to extract precise information from novel medical signals, such as seismocardiograms that are becoming more and more widely adopted nowadays. Chapter 3.3 refers to the paper which employs a similar architecture to the one from 3.4 to estimate signal's interval representing a heartbeat.

It shows the versatility of deep learning architectures which can be applied to different domains by adjusting the data set, instead of developing a whole new algorithm.

Deep learning methods show great potential for applicability in the biomedical engineering field. But it also suffers from many flaws. This technology is often described as not being sufficiently transparent, not being well integrated with prior knowledge, or not generalizing well to out-of-distribution samples, to name a few. All of these reasons constitute a great debate if applying these methods in their current state to perform any kind of medical examination is ethical. On the other hand, fuzzy logic systems, an expert system style algorithm, have proved to be resilient to out-of-distribution samples, representing knowledge in a transparent manner and being easily modifiable. Because of this, chapter 4 outlines different approaches of combining both deep neural networks and fuzzy logic systems. It shows the pros and cons of both approaches and gives a review of the literature on the approaches explored in the past and suggests why they failed. It also proposes a new paradigm, which was not found in any of the previous approaches, called fuzzy representation learning.

To conclude, advances in artificial intelligence field have brought about almost endless ways in which technology can help advance biomedical sciences. Deep learning especially shows a great promise of providing a new set of solutions that can help improve our everyday lives and health. However, these methods need to be modified and improved to be fully appreciated and integrated into biomedical engineering systems. Because of that, a new algorithm was designed and implemented in proof-of-concept form. This algorithm is the starting point for a new research domain which will be explored further in the future.

Bibliography

[1] Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, 61:000812561986492, 07 2019.

[2] S. Kaisler. Expert systems: An overview. *IEEE Journal of Oceanic Engineering*, 11(4):442–448, 1986.

[3] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[4] codecrucks. https://codecrucks.com/, 12 2021. Accessed: 25.02.2023.

[5] Jiří Pospíchal. Fuzzy sets and fuzzy logic: Theory and applications. by george j. klir and bo yuan. prentice hall: Upper saddle river, nj, 1995. 574 pp. $60.00. isbn 0-13-101171-5. sales e-mail: corpsales@prenhall.com. *Journal of Chemical Information and Computer Sciences*, 36(3):619–619, 1996.

[6] Earl Cox. *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*. Academic Press Professional, Inc., USA, 1994.

[7] A. Burkov. *The Houndred Page Machine Learning*. 2019.

[8] Arun Kuchibhotla, Lawrence Brown, Andreas Buja, and Junhui Cai. All of linear regression, 10 2019.

[9] Shai Ben-David Shai Shalev-Shwartz. *Understanding Machine Learning:From Theory to Algorithms*. Cambridge University Press, 2014.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[12] Jeffrey Dean. 1.1 The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 8–14, San Francisco, CA, USA, February 2020. IEEE.

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Big vision. Available online at: `https://github.com/google-research/big_vision`, 2022. Accessed on:.

[16] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. 2022.

[17] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. 2022.

[18] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. *arXiv preprint arXiv:2211.05778*, 2022.

[19] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training, 2022.

[20] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks, 2022.

[21] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[22] Haotian Yan, Chuang Zhang, and Ming Wu. Lawin transformer: Improving semantic segmentation transformer with multi-scale representations via large window attention. *arXiv preprint arXiv:2201.01615*, 2022.

[23] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? 2023.

[24] Myle Ott, Sergey Edunov, Alexei Baevski. Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[25] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

[26] Pengyong Li, Yuquan Li, Chang-Yu Hsieh, Shengyu Zhang, Xianggen Liu, Huanxiang Liu, Sen Song, and Xiaojun Yao. TrimNet: learning molecular representation from triplet messages for biomedicine. *Briefings in Bioinformatics*, 11 2020. bbaa266.

[27] Aditya Khamparia and Karan Singh. A systematic review on deep learning architectures and applications. *Expert Systems*, 36:e12400, 06 2019.

[28] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[30] Max Normalization - an overview | ScienceDirect Topics.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[32] Jian Xiong, Kai Zhang, and Hao Zhang. A Vibrating Mechanism to Prevent Neural Networks from Overfitting. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1737–1742, Tangier, Morocco, June 2019. IEEE.

[33] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

[34] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122, 2018.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[36] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. cite arxiv:1607.06450.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[38] Erdi Çallı, Ecem Sogancioglu, Bram van Ginneken, Kicky G. van Leeuwen, and Keelin Murphy. Deep learning for chest X-ray analysis: A survey. *Medical Image Analysis*, 72:102125, August 2021.

[39] Nathalie Lassau, Samy Ammari, Emilie Chouzenoux, Hugo Gortais, Paul Herent, Matthieu Devilder, Samer Soliman, Olivier Meyrignac, Marie-Pauline Talabard, Jean-Philippe Lamarque, Remy Dubois, Nicolas Loiseau, Paul Trichelair, Etienne Bendjebbar, Gabriel Garcia, Corinne Balleyguier, Mansouria Merad, Annabelle Stoclin, Simon Jegou, Franck Griscelli, Nicolas Tetelboum, Yingping Li, Sagar Verma, Matthieu Terris, Tasnim Dardouri, Kavya Gupta, Ana Neacsu, Frank Chemouni, Meriem Sefta, Paul Jehanno, Imad Bousaid, Yannick Boursin, Emmanuel Planchet, Mikael Azoulay, Jocelyn Dachary, Fabien Brulport, Adrian Gonzalez, Olivier Dehaene, Jean-Baptiste Schiratti, Kathryn Schutte, Jean-Christophe Pesquet, Hugues Talbot, Elodie Pronier, Gilles Wainrib, Thomas Clozel, Fabrice Barlesi, Marie-France Bellin, and Michael G. B. Blum. Integrating deep learning CT-scan model, biological and clinical variables to predict severity of COVID-19 patients. *Nature Communications*, 12(1):634, January 2021.

[40] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift für Medizinische Physik*, 29(2):102–127, May 2019.

[41] J. Teye Brown and Walid Zgallai. Chapter 5 - Deep EEG: Deep learning in biomedical signal processing with EEG applications. In Walid Zgallai, editor, *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, Developments in Biomedical Engineering and Bioelectronics, pages 113–151. Academic Press, January 2020.

[42] Heba Askr, Enas Elgeldawi, Heba Aboul Ella, Yaseen A. M. M. Elshaier, Mamdouh M. Gomaa, and Aboul Ella Hassanien. Deep learning in drug discovery: an integrative review and future challenges. *Artificial Intelligence Review*, November 2022.

[43] Konrad M. Duraj and Natalia J. Piaseczna. Predicting Molecule Toxicity Using Deep Learning. In Natalia Piaseczna, Magdalena Gorczowska, and Agnieszka Łach, editors, *Innovations and Developments of Technologies in Medicine, Biology and Healthcare*, Advances in Intelligent Systems and Computing, pages 18–25, Cham, 2022. Springer International Publishing.

[44] Rafał J. Doniec, Szymon Sieciński, Konrad M. Duraj, Natalia J. Piaseczna, Katarzyna Mocny-Pachońska, and Ewaryst J. Tkacz. Recognition of drivers' activity based on 1d convolutional neural network. *Electronics*, 9(12), 2020.

[45] Konrad M. Duraj, Szymon Siecinski, Rafal J. Doniec, Natalia J. Piaseczna, Pawel S. Kostka, and Ewaryst J. Tkacz. Heartbeat detection in seismocardiograms with semantic segmentation. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 662–665, 2022.

[46] Konrad Duraj, Natalia Piaseczna, Pawel Kostka, and Ewaryst Tkacz. Semantic segmentation of 12-lead ecg using 1d residual u-net with squeeze-excitation blocks. *Applied Sciences*, 12(7), 2022.

[47] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[48] Gary Marcus. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631, 2018.

[49] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[50] Tomi Peltola. Local interpretable model-agnostic explanations of bayesian predictive models via kullback-leibler projections. *CoRR*, abs/1810.02678, 2018.

[51] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[52] Vladimir Cherkassky. Fuzzy inference systems: A critical review. In Okyay Kaynak, Lotfi A. Zadeh, Burhan Türkşen, and Imre J. Rudas, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, pages 177–197, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[53] Huzaimu Lawal Imam, Muhammad Gaya, and Garba Galadanci. Short term load forecast of kano zone using artificial intelligent techniques. 16:562–568, 11 2019.

[54] Elena Vlamou and Basil Papadopoulos. Fuzzy logic systems and medical applications. *AIMS neuroscience*, 6:266–272, 10 2019.

[55] Teuvo Kohonen. *Self-organizing maps*. Springer series in information sciences, 30. Springer, Berlin, 3rd edition, 2001.

[56] Yunis Ahmad Lone Himanshu Singh. *Deep Neuro-Fuzzy Systems with Python With Case Studies and Applications from the Industry*.

[57] Erich Peter Klement, Radko Mesiar, and Endre Pap. 2 - triangular norms: Basic notions and properties. In Erich Peter Klement and Radko Mesiar, editors, *Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms*, pages 17–60. Elsevier Science B.V., Amsterdam, 2005.

[58] Shivali Chopra, Gaurav Dhiman, Ashutosh Sharma, Mohammad Shabaz, Pratyush Shukla, and Mohit Arora. Taxonomy of adaptive neuro-fuzzy inference system in modern engineering sciences", computational intelligence and neuroscience, vol. 2021, article id 6455592, 14 pages, 2021.

[59] Yann LeCun. Available online at: https://twitter.com/ylecun/status/864463163135324165?lang=en. Accessed on: 27 December 2022.

[60] Yann LeCun. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27. 2022.

[61] Diogo C. Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. *CoRR*, abs/1710.02322, 2017.

[62] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[63] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952.

[64] José Vieira, F. Morgado-Dias, and Alexandre Mota. Neuro-fuzzy systems: A survey. 03 2004.

[65] H. Takagi. Cooperative systems of neural networks and fuzzy logic and its applications. In *[1992] Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems & Computers*, pages 771–775 vol.2, 1992.

Symbols and Abbreviations

EEG	Electroencephalography
ECG	Electrocardiography
EOG	Electrooculogram
MRI	Magnetic Resonance Imaging
CT	Computed Tomography
AI	Artificial Intelligence
ML	Machine Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Reccurent Neural Network
ReLU	Rectified Linear Unit
SMILES	Simplified Molecular Input Line Entry Specification
SCG	Seismocardiography
ICT	Information and Communication Technologies
SHAP	SHapley Additive exPlanations
LIME	Local Interpretable Model-Agnostic Explanations
GradCAM	Gradient-weighted Class Activation Mapping
FINN	Fuzzy Inference Neural Network
ANFIS	Adaptive Neuro Fuzzy Inference Systems

Deep learning applications in biomedical engineering - summary

Konrad Duraj

In the early days, artificial intelligence algorithms were mainly constructed using the so-called *expert systems*. These were the algorithms that most often implemented an inference system based on fuzzy logic. Fuzzy logic is a framework in which logic operations are applied to continuous variables that represent the membership of a given feature in a given category. To create a fuzzy inference system for a given problem, the computer scientists must first create a *knowledge base* - its a construct that represents the in-domain knowledge about the phenomenon studied. Based on the knowledge base, scientists can construct a set of rules that describe the input-output relationship of the considered features and their respective classes. Expert systems are often described as transparent, easy to implement, work with, and adjustable. But they are also single purpose, costly to develop, do not provide new knowledge, and require the following steps:

- creation of a knowledge base - collecting the necessary information to solve a problem,
- creation of a feature extraction pipeline - selecting relevant features (from image, audio, etc.) that will be useful for the fuzzy inference system,
- creation of a fuzzy inference system.

Machine learning is an alternative approach to artificial intelligence in comparison to expert systems. It revolves around the idea that the data itself, in its annotated form, provide enough information to create a statistical model capable of mapping the input features to relevant categories through some optimization process. To create a machine learning system, you must do the following steps:

- provide collection of labeled examples to a given problem,
- create a feature extraction pipeline,
- select the algorithm - this step automates the decision making process.

Deep learning goes one step further, instead of creating a feature extraction pipeline, it tries to optimize its set of parameters to actively select the relevant features needed to make an accurate prediction. By allowing an algorithm to choose the relevant features, we abandon the transparency of algorithms such as fuzzy inference expert systems in exchange for automating the feature extraction and the decision-making process. This approach has led to a revolution in many computer science fields and can be also utilized for biomedical and life sciences applications (which this dissertation describes), such as:

1. Predicting Molecule Toxicity Using Deep Learning,
2. Recognition of Drivers' Activity Based on 1D Convolutional Neural Network,
3. Heartbeat Detection in Seismocardiograms with Semantic Segmentation,
4. Semantic Segmentation of 12-Lead ECG Using 1D Residual U-Net with Squeeze-Excitation Blocks.

Furthermore, this dissertation also outlines the main problems that these algorithms face when applied in the medical domain. The biggest being the transparency problem. Although powerful, these deep neural networks do not provide a sufficient explanation for their decision-making process, which in turn is crucial when considering their applicability to any healthcare applications. Because of that, this dissertation introduces a new concept called *fuzzy representation learning* which combines the advantages provided by neural networks and a fuzzy inference system. It is a new type of algorithm, in which the neural network, through the optimization process, tries to create the entire, fuzzy inference system that solves a given problem, just based on the example training set.

Deep learning applications in biomedical engineering - streszczenie

Konrad Duraj

Jednym z pierwszych, powszechnie wykorzystywanych algorytmów sztucznej inteligencji były tzw. "*systemy eksperckie*". Takie systemy bardzo często bazowały na logice romzytej. Aby utworzyć system inferencji logiką rozmytą, naukowcy muszą wpierw stworzyć tzw. "*bazę wiedzy*" - jest to twór, reprezentujący wiedzę domenową, pochodzącą od ekspertów. Korzystając z takiej bazy, naukowcy są w stanie stworzyć zestaw reguł rozmytych, które opisują zależność pomiędzy cechami podawanymi na wejście takiego systemu, a jego wyjściem. Z jednej strony, systemy eksperckie są często opisywane jako transparentne, łatwe do implementacji i użytkowania oraz sterowalne i dostrajalne. Z drugiej strony są one unikatowe dla danego problemu i domeny, kosztowne do wypracowania oraz nie pozwalają na stworzenie nowej wiedzy dotyczącej danego problemu. Aby stworzyć system inferencji rozmytej należy:

- stworzyć bazę wiedzy - akwizycja wszystkich niezbędnych informacji dotyczących danego problemu,
- stworzyć proces ekstrakcji cech - polega na wyznaczeniu cech, z danych w surowej postaci (obrazy, audio, itp.), które mogą okazać się przydatne w procesie budowania systemu inferencji,
- stworzenie systemu inferencji rozmytej.

Uczenie maszynowe jest alternatywną metodologią budowania rozwiązań z zakresu sztucznej inteligencji. Polega ono na zebraniu dużej ilości, oznaczonych danych opisujących dany problem i stworzeniu modelu statystycznego, zdolnego do przetworzenia cech wejściowych i konwersji ich na daną klasę/kategorię. Aby stworzyć model uczenia maszynowego należy podjąć następujące kroki:

- zebranie zbioru danych i jego annotacja,
- utworzenie procesu ekstrakcji cech,
- selekcja algorytmu - zastępuje proces ręcznego tworzenia funkcji mapującej.

Uczenie głębokie jest pójściem o jeden krok naprzód, poprzez pozwolenie algorytmowi dokonania procesu selekcji cech, odrzucamy jakąkolwiek transparentność na rzecz automatyzacji ekstrakcji i klasyfikacji cech. Ten paradygmat doprowadził do rewolucji w dziedzinach takich jak wizja komputerowa czy przetwarzanie języka naturalnego. Algorytmy uczenia głębokiego mogą zostać również użyte do przetwarzania danych biomedycznych (prace wymienione w rozprawie), takich jak:

1. Przewidywanie toksyczności molekuł,
2. Rozpoznawania akcji kierowców na drodze, bazując na sygnale EOG,
3. Detekcja uderzeń serca w sygnałach sejsmokardiograficznych,
4. Segmentacja składowych EKG.

Ponadto, niniejsza rozprawa porusza kwestie związane z problematyką we wdrożeniu i bezpiecznym użytkowaniu tych systemów. Jedną z największych wad, w kontekście aplikacji medycznych jest problem transparentności/wyjaśnialności procesu decyzyjnego tych algorytmów. Aby algorytmy głębokiego uczenia zostały wprowadzone jako standardowe systemy wsparcia opieki zdrowotnej, koniecznym jest wypracowanie metody pozwalającej wyjaśnić oraz prześledzić "tok rozumowania" głębokich sieci neuronowych. W niniejszej rozprawie zaproponowany został nowy rodzaj algorytmu, nazwany "*uczeniem reprezentacji rozmytej*", który łączy ze sobą sieci neuronowe oraz systemy inferencji logiką rozmytą. Ten rodzaj algorytmu, bazuje na założenie, że sieć neuronowa zamiast predykcji powinna na swoim wyjściu podawać wyjaśnienia, które w tym konkretnym przypadku mają formę systemu inferencji logiką rozmytą.

Curriculum Vitae

Personal Information

Name:	Konrad Duraj
Date of Birth:	[redacted]
Email:	konrad.duraj@polsl.pl

Education

Degree:	PhD candidate
Institution:	Silesian University of Technology
Date:	2023

Degree:	Master of Science in Biomedical Engineering
Institution:	Silesian University of Technology
Date:	June 2019

Degree:	Bachelor of Science in Biomedical Engineering
Institution:	Silesian University of Technology
Date:	February 2018

Commercial Experience

Company:	Noctuai
Position:	Deep learning research engineer
Finish date:	-

Company:	Iconity
Position:	Machine learning expert
Finish date:	October 2022

Company:	KP Labs
Position:	Deep learning engineer
Finish date:	April 2020

Company:	Whiteaster
Position:	Machine learning engineer
Finish date:	June 2019

Company:	Kamsoft
Position:	Junior .NET developer
Finish date:	January 2018

Lectures

- Programming in Python,
- Automation and Control Systems,
- Bionics,
- Machine learning