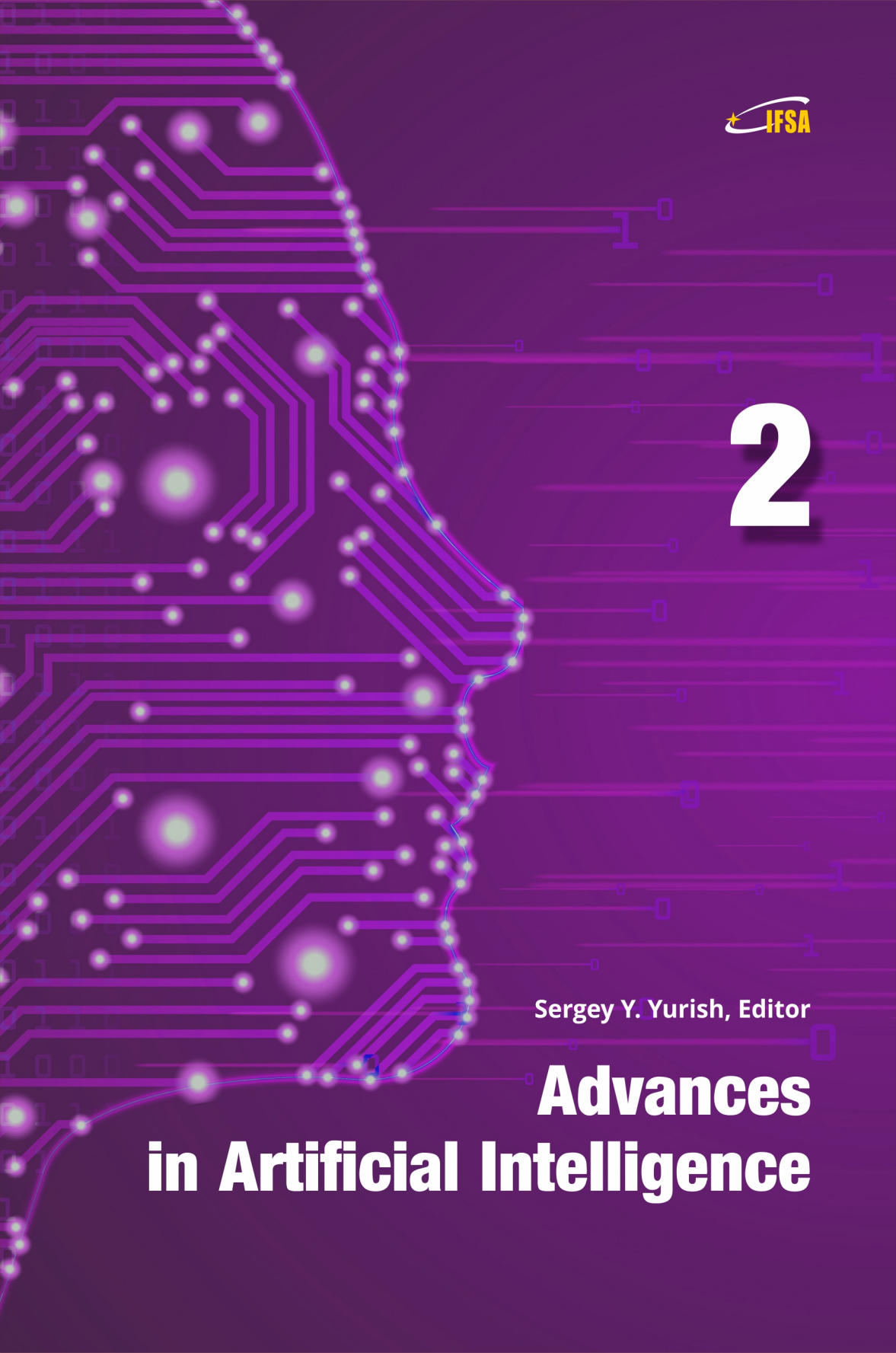
IFSA
2
Sergey Y. Yurish, Editor
Advances in Artificial Intelligence

Advances in Artificial Intelligence

Book Series, Volume 2

S. Y.Yurish
Editor

Advances in Artificial Intelligence

Book Series, Volume 2

International Frequency Sensor Association Publishing

S. Y. Yurish, *Editor*
Advances in Artificial Intelligence,
Book Series, Vol. 2

Published by IFSA Publishing, S. L., 2023
E-mail (for print book orders and customer service enquires):
ifsa.books@sensorsportal.com

Visit our Home Page on http://www.sensorsportal.com

e-ISBN: 978-84-09-47562-9
BN-20230211-XX
BIC: UYQ

Contents

Preface

It is my great pleasure to present the 2nd volume of the '*Advances in Artificial Intelligence*' Book Series, started by the IFSA Publishing in 2017.

Artificial intelligence has is one of the fastest-growing technologies in recent years. The market growth is mainly driven by factors such as the increasing adoption of cloud-based applications and services, growing big data, and increasing demand for intelligent virtual assistants. Various end-use industries have also employed artificial intelligence such as retail and business analysis that has also boosted the demand in this market. The major restraint for the market is the limited number of artificial intelligence technology experts. The Book Series on 'Advances in Artificial Intelligence' has been launched with the aim to fill-in this gap.

The book volume contains four chapters written by 13 contributors from three countries: France, Iran and Slovenia.

Chapter 1 describes the fuzzy-based optimisation of unit selection algorithm for corpus-based TTS systems.

Chapter 2 reviews the recent works that aim at using graphs as tools to improve deep learning methods. These graphs are defined considering a specific layer in a deep learning architecture.

Chapter 3 is devoted to the machine learning in peer-to-peer lending in financial risk modelling.

Chapter 4 reports the early detection of mild Alzheimer's diseases with combine MRI and EEG signals.

All chapters have the same structure: first an introduction to specific topic under study; second particular field description including appropriate applications. Each of chapter is ending by well selected list of references with books, journals, conference proceedings and web sites.

This book ensures that our readers will stay at the cutting edge of the field and get the right and effective start point and road map for the further researches and developments.

With this unique combination of information in each volume, the '*Advances in Artificial Intelligence*' book Series will be of value for scientists and engineers in industry and at universities, to developers and users.

Sergey Y. Yurish

Editor
IFSA Publishing *Barcelona, Spain*

Contributors

Mahdi Azarnoosh
Department of Biomedical Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran
E-mail: M_Azarnoosh@mshdiau.ac.ir

Myriam Bontonou
ENS de Lyon, CNRS, LBMC, Lyon, France

Lucas Drumetz
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France

Vincent Gripon
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France

Mounia Hamidouche
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France

Timotej Jagrič
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Vita Jagrič
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

David Jesenko
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Carlos Lassance
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France

Elias Mazrooei Rad
Biomedical Engineering Department, Khavaran Institute of Higher Education, Mashhad, Iran, E-mail: Elias_Mazrooei@yahoo.com

Bastien Pasdeloup
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France

Matej Rojc
Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia

Borut Žalik
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Chapter 1

Fuzzy-based Optimisation of Unit Selection Algorithm for Corpus-based TTS Systems

Matej Rojc

1.1. Introduction

For corpus-based text-to-speech synthesis systems, the ultimate goal is to select the most natural-sounding sequence of acoustic units from a huge acoustic inventory. This can be achieved only, when there are no unnatural acoustic transitions between selected acoustic units, and all acoustic mismatches at their concatenation points are minimized. For this goal is directly responsible a unit selection algorithm that is based on unit selection cost function. However, unit selection algorithm operates over large acoustic inventories with huge number of acoustic units in a broad spectrum of linguistic, prosodic and acoustic contexts, and therefore, the issue represents a huge number of concatenation possibilities. The unit selection cost functions, which evaluate the overall cost of concatenating two subsequent acoustic units, is modelled in general manually and optimized then in a time-consuming and laborious iterative process and is often based on subjective evaluation procedures. The issue is also that this process has to be repeated for e.g., new acoustic inventory, after changes in an acoustic inventory etc. [1].

The issue is not only the number of concatenation possibilities in each sentence, but also how to design a unit selection cost function for the unit selection algorithm, which will be consistent in finding the best sequence of acoustic units in a huge set of acoustic units with several linguistic,

Matej Rojc
Faculty of Electrical Engineering and Computer Science,
University of Maribor, Slovenia

acoustic and prosodic Although heuristics and perception tests could be utilized successfully, in order to design such unit selection cost functions that would behave close to human-like judgements, these methods in the end cannot answer the fundamental questions, such as: '*How optimal is the shape of the unit selection cost function*? *How consistent the unit selection cost function will be*? *What could be an effect on its performance, when using lossy data compression techniques?*' etc. As a result, this in general ends with some specific unit selection cost function's shape, covering consistently some part of the overall feature space, for those sentences utilized in experiments, and without any feedback for the feature space that relates to sentences not used in the listening tests.

A strongly related issue with unit selection algorithm for corpus-based TTS systems is then the use of large acoustic inventories, and optimal representation of concatenation costs associated with acoustic units in the acoustic inventory. Namely, concatenation costs are needed for unit selection cost function, in order to evaluate spectral mismatches between two subsequent acoustic units to be concatenated. The problem is that the combinatorics of concatenation costs grows exponentially with the number of acoustic units and can in this way result in hundreds of millions or even billions of possible concatenation costs, when calculated off-line [2]. This represents a performance issue regarding time, or memory demands. Therefore, corpus-based TTS systems often tend to limit the size of acoustic inventories and perform on-line concatenation cost calculations, when synthesis units are known for specific sentence. In general, pruning [12-14] and compression based [15-16] approaches are proposed to minimise this memory footprint in corpus-based TTS systems, and is done by reduction of corpus size. On the other hand, most of compression-based approaches are motivated by linear prediction-based speech coding methods [17]. However, the minimisation of acoustic inventories always has a significant impact on the speech quality and lead to the degradation in speech quality [18].

1.2. The Unit Selection Algorithm in Corpus-based TTS Systems

The unit selection algorithm in corpus-based TTS systems is based on unit selection cost function, which has to evaluate mismatches between any two acoustics units to be concatenated. In general, the following

costs are used for the evaluation of the mismatch between two acoustic units [4], as shown in Fig. 1.1:

- *Local cost* $cost_{lc} = C^l(u_i, t_i)$: this cost represents a distance between unit candidate u_i and the target unit specification t_i by considering the linguistic context;
- *Target cost* $cost_{tc} = C^t(u_i, t_i)$: this cost evaluates a distance between unit candidate u_i and the target unit specification t_i, by considering acoustic prosody information, after the target prosody is specified;
- *Concatenation cost* $cost_{cc} = C^c(u_{i-1}, u_i)$: this cost describes acoustic mismatches between successive unit candidates u_{i-1} and u_i, often by considering their energy and spectral information.

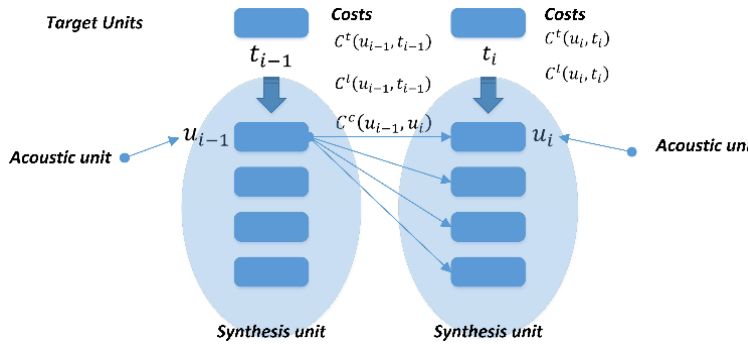

Fig. 1.1. Costs used by unit selection cost function, where C^t is a target cost, C^l a local cost, and C^c a concatenation cost.

1.3. A Lossy and Non-lossy Compression of Concatenation Costs

The main idea behind unit selection algorithm in corpus-based TTS systems is to select the most spectrally suitable sequences of acoustic units that are available in acoustic inventory [3]. Further, for a high-quality speech, a good coverage with acoustic units in several acoustic realisations must be ensured. Only then we are able to achieve desired speech quality, since spectral mismatches at the concatenation points between acoustic units in a sentence would be minimal.

Nevertheless, these spectral mismatches are judged automatically by using the concatenation costs, while the acoustic mismatch between two acoustic units found in a specific linguistic and prosodic context and to be concatenated is evaluated by using a unit selection cost function in the unit selection algorithm. And this process utilises *local cost*, *target cost* and *concatenation cost*. Since in on-line TTS system, lots of concatenation costs should be calculated per each sentence (and we have to load corresponding speech data into memory), the next logic step is that concatenation costs are calculated offline. Nevertheless, in this case synthesis units are not known yet, therefore, concatenations costs have to be calculated and stored for all any two possible subsequent acoustic units in acoustic inventory, although the off-line calculation of $\boldsymbol{cost_{cc}}$ costs guarantee better run-time and real-time performance of the unit selection algorithm in the corpus-based TTS systems. Thus, in on-line case, a unit selection algorithm can use already available concatenation costs. The main issue is then that we are dealing with a huge and sparse concatenation cost matrix consisting of floating-point numbers (Fig. 1.2) [9]. Therefore, their efficient representation and optimal lookup time is a necessity if the unit selection algorithm is to retain its real-time capability.

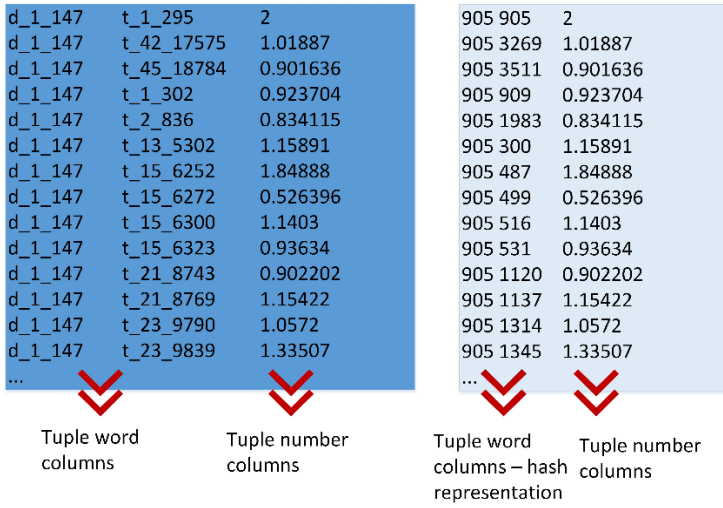

Fig. 1.2. Representation of the concatenation costs.

Namely, the memory footprint of the concatenation costs is:

$$M_f = N \cdot N \cdot sizeof(float), \quad (1.1)$$

where N represents the number of acoustic units in the acoustic inventory, and size of (float) is equal to 4 bytes. In [2] for lossless data compression that can give us time-and-space efficient data representation, tuple structures are proposed. Namely, a tuple structure $\boldsymbol{T^{i,j}}$ as a finite function $(\boldsymbol{W_1} \times ... \times \boldsymbol{W_i}) \rightarrow \boldsymbol{Z^j}$, where $(\boldsymbol{W_1} \times ... \times \boldsymbol{W_i})$ is used, where sets are acoustic units' IDs, while *Z* are corresponding concatenation costs. In this way, the finite function maps pairs of acoustic units into concatenation costs. In the tuple for each word column perfect hash finite automata are used; therefore, a word column *W* is represented by a minimal deterministic acyclic finite automaton *N* that accepts each acoustic unit's ID in a word column *W*. For the minimisation of the tuple memory footprint, each hash-key is then represented with as few bytes as are required by the largest integer number used in each word column. The only number column in tuple *T* (i.e., the third column in Fig. 1.2) keeps the concatenation costs $\boldsymbol{cost_{cc}}$. Since these are floating point numbers, their compact representation can be obtained, when each number is decomposed into a normalised mantissa *m* and an exponent *t*, which are then both stored with the minimal number of bytes. Nevertheless, such tuple structure still requires significant memory resources to be allocated, when acoustic inventories are huge in corpus-based TTS systems. In order to solve this issue, a VQ algorithm can be used as proposed in [2], in order to compress the concatenation costs, and in this way further reduce the tuple memory footprint.

VQ algorithms can represent an efficient technique for compression of concatenation costs [8]. Nevertheless, since VQ represents lossy-data compression technique it is important, which approach to use, how to perform it, and especially in regard to TTS systems, how much we gain, and how much we lose in speech quality. VQ is often implemented with *k*-means clustering, or Linde-Buzo-Gray (LBG) algorithm. However, in the case of large vector dimensions and large codebooks as is in our case, the use of these methods can suffer from complexity [5]. On the other hand, there are constrained VQ methods (partitioned VQ), that can be used to reduce storage and computation complexity, while they can severely increase the coding error [6]. Since in corpus-based TTS systems we are usually dealing with a huge number of concatenation costs, the Radial Basis Function Network (RBFN) approach can be used, as proposed in [2]. By applying the RBF network over the concatenation

costs space, we are able to further reduce the memory footprint of the tuple used in the unit selection algorithm, although VQ induces errors. This represents a new issue, since in case of TTS systems, it is also important that we are able to evaluate the effect of VQ errors on speech quality. In order to evaluate this, a series of perceptive tests in general can be performed. However, such a task would be highly subjective, very costly and labour intensive, and hard to perform it consistently. Namely, the solution regarding optimising the size of compressed concatenation costs and tuple, and to allow for the best possible performance of the unit selection cost function, is practically impossible to find in this way.

1.4. Fuzzy-based Unit Selection Algorithm Optimization

The unit selection algorithm in corpus-based TTS systems utilizes unit selection cost function for searching the best path between many acoustic realizations of acoustic units from a huge acoustic inventory (Fig. 1.3). The transition cost between any two acoustic units in a huge finite state machine is then determined in general by a weighted unit selection cost function that relies on several partial costs that evaluate linguistic, acoustic and prosodic contexts.

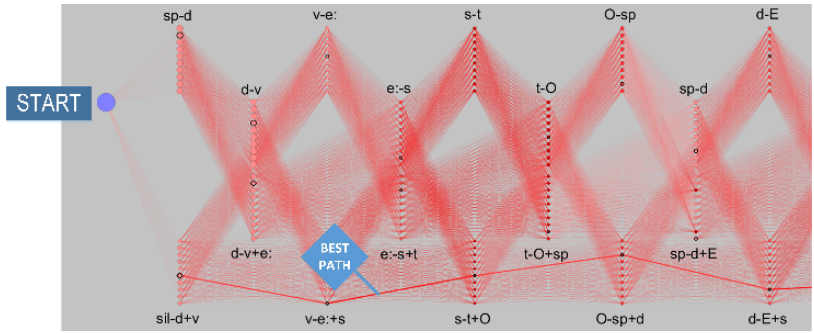

Fig. 1.3. A unit selection algorithm task - searching the best path between many acoustic realizations of acoustic units.

The unit selection algorithm and unit selection cost function have to find such a sequence of acoustic units that will result in minimal overall cost, since only minimal cost in the end will result also in minimal acoustic mismatches at concatenation points between adjacent acoustic units [10]. It is evident that it is very important that unit selection cost function is

able to consider the relative importance of *local*, *target*, and *concatenation costs*, and also their internal importance of linguistic and acoustic features. For simulating the relative importance of all these features, in general weights are used. Then for the optimal unit selection cost function, optimized values for these weights have to be found. Heuristics and perception tests could be utilized for this task, in order to design unit selection cost functions that would behave close to human-like judgements, but this is very time consuming, and at the end of the process, we cannot answer to: how optimal is the shape of the unit selection cost function, or how consistent is the unit selection cost function for specific huge multidimensional feature space. In order to tackle these issues better, in [1] it is proposed the following unit selection cost function to be used in corpus-based TTS systems:

$$C(u_i) = \left(\left(S_{cc}(cost_{cc})\right)^{w_{cc}} \cdot \left(\prod_{j=1}^{3} \left(S_{tc_j}\left(cost_{tc_j}\right)\right)^{w_j} \right)^{w_{tc}} \cdot \left(S_{lc}(cost_{lc})\right)^{w_{lc}} \right)^{w_{unittype}}, \quad (1.2)$$

where $C(u_i)$ is the overall cost of selecting unit u_i in the context of preceding unit u_{i-1}, while the unit selection cost function fuses three partial fuzzy-based cost functions: *local cost*, *target cost* and *concatenation cost* for estimating the mismatches between acoustic units. Namely, fuzzy cost functions' shapes can give to the unit selection cost function a lot of flexibility, in order to describe well the relative importance of their parameters for a specific acoustic inventory, after the optimization algorithm is performed. Therefore, in Eq. (1.2) S_{cc} represents the partial fuzzy cost function evaluating the *concatenation cost* and evaluates the spectral mismatch at the concatenation points between successive acoustic units u_{i-1} and u_i. The S_{tc} represents the partial fuzzy cost function evaluating the *target cost* $cost_{tc} = C^t(u_i, t_i)$, and evaluates the distance between acoustic unit u_i and the target unit t_i, by considering acoustic prosody information. And the S_{lc} represents the partial fuzzy cost function evaluating *local cost*: $cost_{lc} = C^l(u_i, t_i)$ and evaluates the distance between acoustic unit u_i and the target unit specification t_i, by considering linguistic context. Contributions from each partial fuzzy cost function are then additionally weighted by using: $w_{cc}, w_{tc}, w_j, w_{lc,} w_{unittype}$, in order to incorporate the relative importance criteria, including the type of the acoustic unit in a given context. In this way in Eq. (2) there is included also individual contribution of each partial fuzzy cost function, and their relative importance in the overall fuzzy unit selection cost function. In [1] it is

proposed that partial fuzzy cost functions (or fuzzy membership function) are defined by Eq. (1.3) and (1.4).

$$f(cost_*; a, b) = \begin{cases} 1, cost_* \le a \\ 1 - 2 \cdot \left(\frac{cost_* - a}{b - a}\right)^2, a \le cost_* \le \frac{a+b}{2} \\ 2 \cdot \left(\frac{cost_* - b}{b - a}\right)^2, \frac{a+b}{2} \le cost_* \le b \\ 0, cost_* \ge b \end{cases}, \quad (1.3)$$

where $cost_*$ are C^c or C^l costs, while a and b parameters that define some shape of the partial fuzzy cost function. Further, in Eq. (1.4), then $cost_*$ represents C^t costs that represent a match between acoustic unit duration and target unit duration, while a, b, c, d parameters define its shape.

$$f(cost_*; a, b, c, d) = \begin{cases} 0, cost_* \le a \\ 2 \cdot \left(\frac{cost_* - a}{b - a}\right)^2, a \le cost_* \le \frac{a+b}{2} \\ 1 - 2 \cdot \left(\frac{cost_* - b}{b - a}\right)^2, \frac{a+b}{2} \le cost_* \le b \\ 1, b \le cost_* \le c \\ 1 - 2 \cdot \left(\frac{cost_* - c}{d - c}\right)^2, c \le cost_* \le \frac{c+d}{2} \\ 2 \cdot \left(\frac{cost_* - d}{d - c}\right)^2, \frac{c+d}{2} \le cost_* \le d \\ 0, cost_* \ge d \end{cases} \quad (1.4)$$

The main challenge for unit selection cost function is then to find such weights and shapes of partial fuzzy cost functions that overall fuzzy unit selection cost function will operate optimally and also consistently over the entire acoustic units' feature space. Clear message, but difficult task. Use of fuzzy logic as proposed in [1] offers lots of flexibility, when searching this by optimisation algorithm and not subjectively. Namely, the multiplication of fuzzy cost functions guarantees that even small deviations for any acoustic unit's feature can be noticeable in the overall cost. We can say that the significance of a particular acoustic unit's features is reflected through the shape of a partial fuzzy cost function. Further, also the relative importance of each criterion is expressed through the shape of individual partial fuzzy cost function. The relative surface of partial fuzzy cost functions (Z-Shaped, PI-Shaped etc.) defines the weight of the search criteria, regarding the acoustic unit's suitability to be used. In this way, the fuzzy method represents a methodology ideal

for self-adaptation of fuzzy representations [11], which can adjust the initial shapes to the nature of the datasets very efficiently. Nevertheless, in the next step all the parameters in Eqs. (1.2)-(1.4) have to be defined, by utilizing knowledge, or experience, obtained while testing the unit selection algorithm. A subjective process is sub-optimal, rigid, laborious and time-consuming, especially when dealing with huge acoustic inventories and several voices, since the search space is multi-dimensional. In this case it is also hard to know, whether we found a global optimum for unit selection cost function in the end. Furthermore, relative importance of costs and partial fuzzy based functions is database specific, therefore, the process has to be repeated for new voice again. Therefore, in [1] a fully automated optimization process for unit selection algorithm and unit selection cost function, has been proposed and is outlined in Fig. 1.4.

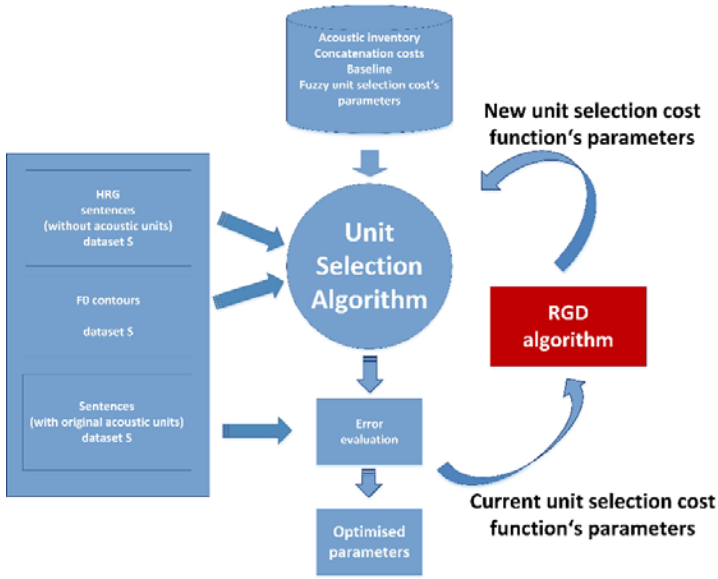

Fig. 1.4. The RGD-based unit selection algorithm optimization.

As seen in Eq. (1.2), the fuzzy unit selection cost function in unit selection algorithm needs comprehensive linguistic, prosody and acoustic information in order that the costs, such as $cost_{cc}$, $cost_{tc_j}$ and $cost_{lc}$, for all acoustic units in the sentence can be calculated.

In general, all this information is provided in the TTS databases for each particular sentence. Therefore, for general corpus-based TTS system, we already have all information together with pitch (F0) contour, and we are able to operate with all linguistic, prosody information, and acoustic units from the acoustic inventory that are used in the sentence. Each sentence in the dataset can then be represented e.g., as shown in Fig. 1.5, where linguistic and prosodic data are visualized in several dimensions. The data are organized in the form of several circuits, where each circle belong to a specific information layer. Namely, the *RawDBUnits* layer contains acoustic units as found in the original database sentence, and with corresponding IDs in the acoustic inventory. Additionally, each acoustic unit in this layer is also assigned with phonetic representation. At the *Syllable* layer are then stored related syllables, with their prominence level (e.g., primary accent – PA, secondary accent – NA, or no accent) represented in figure by the node's shape, while their target pitch values are represented by the color. In the *Segment* layer we also have related phonemes, and their durations that are represented by the diameter of the nodes, while their target pitch value is again represented by using a color.

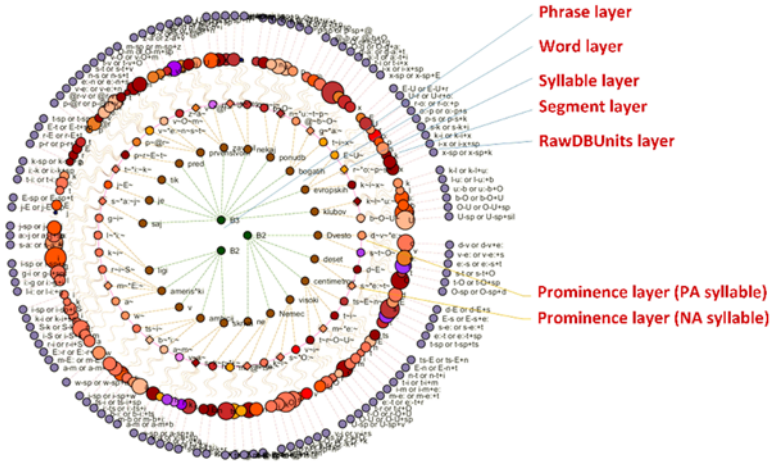

Fig. 1.5. A visual representation of the TTS database sentence.

The unit selection algorithm optimization performs RGD algorithm as proposed in [7]. The RGD-based optimization goal is in our case: for each database input sentence select with baseline unit selection cost

function as many as possible original acoustic units in the same positions by considering the corresponding linguistic and prosodic context. An objective measure for measuring the performance of the unit selection algorithm is then *Error* calculated as follows:

$$Error = \frac{N = \left(\sum_{j=1}^{S}\sum_{i=1}^{length(Sentence(j))}(SelectedCandidate_i! = OriginalCandidate_i)\right)}{M = \left(\sum_{j=1}^{S}\sum_{i=1}^{length(Sentence(j))}SelectedCandidate_i\right)}[\%] \tag{1.5}$$

In Eq. (1.5) the *Error* is calculated as a ratio of *N* acoustic units selected by the fuzzy unit selection cost function incorrectly, and *M* that represents all the selected acoustic units in the database subset used for optimization. Namely, in [1] it is argued that it is not needed to consider the complete database in the optimization process. This is also very important for time performance of the optimization process. Then the RGD-based unit selection optimization algorithm continues by using database subset *S* in the following steps:

1. Unit Selection Algorithm acts on a subset *S* by using fuzzy unit selection cost function with baseline parameters in Eq. (1.2), and by using a complete acoustic inventory.
2. For each sentence in subset *S*, partial derivations on all parameters in the fuzzy unit selection cost function are calculated, in order to compute a search direction in the multidimensional parameter search space, as follows:

$$d_k = -\nabla f(x_k) \tag{1.6}$$

3. Then step t_k via *backtracking line search* algorithm for each parameter *k* is defined,
4. Then parameters in Eq. (1.2) are updated, while $\theta_k \in (0,1)$ is randomly defined, as follows:

$$x_{k+1} = x_k + \theta_k \cdot t_k \cdot d_k \tag{1.7}$$

Finally, a test is performed regarding the end criterion – for stopping the optimization algorithm. Namely, the *Error* has to be calculated between the number of correctly selected unit candidates *N*, and the number of all selected unit candidates *M*. When the criterion is already fulfilled, the optimization algorithm stops; otherwise, the next iteration is performed by using updated parameters in Eq. (1.2).

1.5. Discussion

The RGD-based unit selection optimization algorithm has been performed in [1] by using dataset *S* with only 10 sentences, a fuzzy unit selection cost function with predefined parameters, and by a complete acoustic inventory of acoustic units (from 300 sentences). In this case it was shown that the final shape of the partial fuzzy cost function S_{lc} remains practically unchanged. Therefore, all those unit candidates with local cost in the interval [0.1-1.0] are considered acceptable, while units with $cost_{lc}$ lower than 0.1 rapidly become unacceptable. Shapes related to the partial fuzzy cost function S_{tc}, are expected to be very selective in order to categorize acoustic units strictly regarding duration, pitch match, and pitch differences against the target unit specification defined by the desired prosody information. We could also assume that only those acoustic units with just slight differences in duration and pitch when compared with the target unit would be acceptable, and acoustic units with already slight differences would rapidly become less acceptable. Nevertheless, after the optimisation, the final shapes for S_{tc} suggested that slopes of fuzzy shapes are less steep and, therefore, a larger set of acoustic units, having even more noticeable differences when compared against the target unit specification, are also taken into account (are still acceptable). Namely, the partial fuzzy cost function S_{tc_1} that evaluates candidates by considering their correlation values that evaluate matching of f0 contours between acoustic unit and target unit specification, considers all acoustic units within [-0.8, 1.0] as acceptable, and as rapidly less acceptable those with correlation values below -0.8. Further, the partial fuzzy cost function S_{tc_2} that evaluates acoustic units by considering the pitch differences between acoustic units and the target unit specification based on RMSE measure, larger RMSE costs are severely punished, since only costs around 0.0 are totally acceptable, while others with larger values are considered less acceptable, and all costs above 100.0 as unacceptable. The partial fuzzy cost function S_{tc_3} that evaluates acoustic units by considering duration mismatches between each acoustic unit and the target unit specification, the absolute differences in duration are, in both cases (shorter, or larger durations), very costly. After the RGD-based optimization, in general the slopes of the fuzzy cost function's shape are less steep and less strict when considering duration mismatches, nevertheless, only for smaller duration mismatches the acoustic units are considered as totally acceptable. In the shape of the partial fuzzy cost function S_{cc}, also less steep slope is suggested after the optimisation process. Namely, the S_{cc} evaluates

concatenation costs between two acoustic units, with values in the range [0.0-2.0], where the higher value also represents a better match between candidates (regarding energy and spectrum). The final shape of the unit selection cost function, in this way, the costs in the interval [0.1-2.0], are considered acceptable, while those with costs below 0.1 rapidly become less acceptable. After the RGD-based unit-selection optimisation algorithm, we obtain an optimized set of fuzzy unit selection cost function parameters, and therefore, new shapes of partial fuzzy cost functions.

The results in Fig. 1.6 (a) and (b) show that after using RGD-based unit selection optimisation algorithm proposed in [2], the *Error* regarding unit selection algorithm is reduced from 36 % to roughly 11 % after the optimisation. And when comparing results in (a) and (b), namely, whether we are using lossless or VQ-based lossy data compression (2000 clusters) for concatenation costs $cost_{cc}$, induced VQ errors have negligible impact on the unit selection cost function performance on unseen sentences. Therefore, advanced RBFK clustering guarantees very good representations for the concatenation costs, and on the other hand significant reduction in the size of the tuple structure.

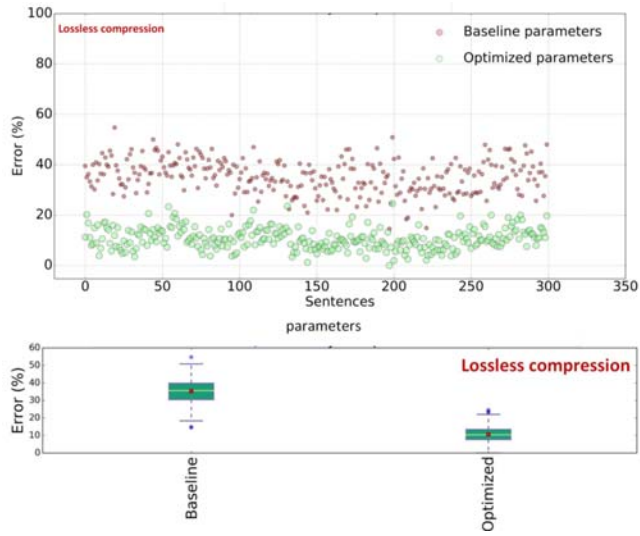

Fig. 1.6 (a). Error, when using baseline non-/optimised unit selection cost function lossless compression (a), and non-/optimised unit selection cost function with RBFK compression with 2000 clusters (b).

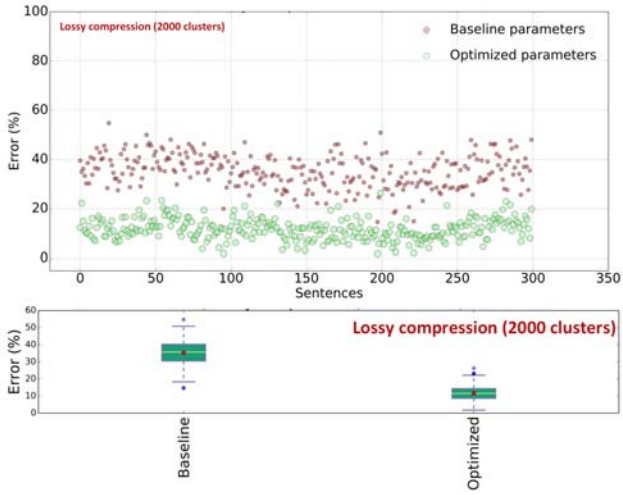

Fig. 1.6 (b). Error, when using baseline non-/optimised unit selection cost function lossless compression (a), and non-/optimised unit selection cost function with RBFK compression with 2000 clusters (b).

1.6. Conclusion

An important issue of the corpus-based TTS systems is that the speech quality and unit selection algorithm performance greatly depend on the size and richness of the acoustic inventory. Although it is clear what we need and is also possible to do this, it is then hard or even impossible to process huge inventories that covers lots of possible contexts and guarantee to synthesise speech with high quality, close to natural speech. The memory demands become very quickly simply too huge. A reduction of acoustic units and, therefore, also the diversity of acoustic realisations will unpredictably increase spectral mismatches in the synthesised speech.

Therefore, in [2] it was proposed that rather than targeting reduction of the acoustic inventory, lossy compression techniques for the concatenation costs could be better approach. When using VQ-based compression implemented via an RBF network, with a *k*-means algorithm used for clustering, the experiments show that tuple memory footprints for the same acoustic inventories as used prior to compression, can be reduced by up to 50 %. The RBFK-based lossy compression can significantly reduce the memory footprint with a proven minimal degrade in the quality of the synthesised speech. Nevertheless, the size

of the final tuple still represents a significant computational problem for larger acoustic inventories, mostly because of hash IDs in a tuple, needed for indexation. Therefore, data compression techniques have to be further investigated in order to use even larger acoustic inventories, and to expect better and more consistent speech quality of the corpus-based TTS systems.

Next important issue of the corpus-based TTS systems and unit selection cost functions is their performance and generated speech quality that greatly depends on the available acoustic inventory, where only huge and high-quality acoustic inventory that covers lots of different linguistic, prosodic, and acoustic contexts, results also in high quality of generated speech. This represents high demands for available hardware resources, and it is also impossible to consider and optimize the unit selection algorithm consistently via perceptive tests. The lossy compression techniques for concatenation costs can help, but then the side effect represent induced errors. A fuzzy unit selection cost function, as proposed in [1], can help to reduce these issues. In this case the unit selection algorithm performance is defined by its adaptable shape, described by several fuzzy parameters. Nevertheless, the baseline fuzzy unit selection cost function is sub-optimal, and parameters defined heuristically. It was shown that the unit selection cost function can be efficiently optimized by using RGD-based algorithm on only a small subset of database sentences (i. e. 3 % of the data) and using objective measure. The main limitation of this optimization lies in the costs' estimation. Namely, the algorithm operates on acoustic units, where linguistic and prosody information has been semi-automatic annotated, therefore, may contain annotation errors. Further, the fundamental frequency contours are obtained automatically by algorithms, which also may induce errors regarding pitch estimation on units. Therefore, the partial fuzzy cost functions' calculations might be suboptimal, when differentiation in acoustic units' features may not be correctly considered. Additionally, acoustic units also with noticeably different spectral information may have too similar, or equal cost. Nevertheless, after the optimization, the unit selection *Error* remains consistent over the whole dataset. In this way, even if the unit selection cost function has to evaluate a completely unseen context (i.e., linguistic and prosody context not captured by the database), the unit selection algorithm will select acoustic units that are close to the best match for a given acoustic inventory. The consistency achieved through the RGD-based optimization and the final performance of the unit selection cost function

affects the naturalness of synthesized speech. Namely, RGD-based optimization significantly reduces the manual effort, inconsistency, and uncertainty, due to the perceptive experiments, while the unit selection algorithm is capable of selecting appropriate sequences of unit candidates for given sentence, even when dealing with unknown contexts, not involved in the optimization process. This represents also fully automated process for general corpus-based TTS systems, in order to consistently improve the performance of the unit selection algorithm. The optimization method also allows developers to assess the consistency of the performance based on an objective measure.

The benefit of the RGD-based optimization algorithm is obvious also in regard to lossy compression techniques used for concatenation costs. Namely, an objective measure expressed as *Error*, can be used to estimate automatically the degradation in speech quality due to lossy compression method used. Further, also very importantly, through this algorithm we are also able to minimize effect of induced VQ errors on the unit selection algorithm performance.

Acknowledgements

This paper is funded by the Slovenian Research Agency, project HUMANIPA (research core funding No. J2-1737 (B)).

References

[1]. M. Rojc, I. Mlakar, A new fuzzy unit selection cost function optimized by relaxed gradient descent algorithm, *Expert Systems with Applications*, Vol. 159, Nov. 2020, 113552.
[2]. M. Rojc, I. Mlakar, A new unit selection optimisation algorithm for corpus-based TTS systems using the RBF-based data compression technique, *IEEE Access*, Vol. 7, 02 August 2019, pp. 108035-108048.
[3]. M. Rojc, I. Mlakar, Z. Kačič, The TTS-driven affective embodied conversational agent EVA, based on a novel conversational-behavior generation algorithm, *Engineering Applications of Artificial Intelligence*, Vol. 57, 2017, pp. 80-104.
[4]. M. Rojc, K. Zdravko, Time and space-efficient architecture for a corpus-based text-to-speech synthesis system, *Speech Communication*, 2007, Vol. 49, Issue 3, pp. 230-249.
[5]. W. Jiang, P. Liu, F. Wen, An improved vector quantization method using deep neural network, *AEU-International Journal of Electronics and Communications*, Vol. 72, 2017, pp. 178-183.

[6]. A. Vasuki, P. T. Vanathi, A review of vector quantization techniques, *IEEE Potentials*, Vol. 25, Issue 4, 2006, pp. 39-47.
[7]. N. Andrei, A New Gradient Descent Method for Unconstrained Optimization, ICI Technical Report, *ICI*, 2004.
[8]. A. K. Jain, Data clustering: 50 years beyond K-means, *Pattern Recognition Letters*, Vol. 31, Issue 8, 2010, pp. 651-666.
[9]. P. Sharma, V. Abrol, A. K. Sao, Reducing footprint of unit selection-based text-to-speech system using compressed sensing and sparse representation, *Computer Speech & Language*, Vol. 52, 2018, pp. 191-208.
[10]. F. Hinterleitner, B. Weiss, S. Möller, Influence of corpus size and content on the perceptual quality of a unit selection MaryTTS voice, in *Proceedings of the IEEE Spoken Language Technology Workshop (SLT'16)*, 2016, pp. 680-685.
[11]. C. F. Wu, C. J. Lin, C. Y. Lee, A functional neural fuzzy network for classification applications, *Expert Systems with Applications*, Vol. 38, Issue 5, 2011, pp. 6202-6208.
[12]. T. Capes, P. Coles, A. Conkie, L. Golipour, A. Hadjitarkhani, Q. Hu, K. Prahallad, Siri on-device deep learning-guided unit selection text-to-speech system, in *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH'17)*, 2017, pp. 4011-4015.
[13]. M. Gruber, J. Matousek, D. Tihelka, Z. Hanzlicek, Reducing footprint of unit selection TTS system by removing linguistic segments with rarely selected units, in *Proceedings of the 12th IEEE International Conference on Signal Processing (ICSP'14)*, 2014, pp. 494-499.
[14]. H. Lu, W. Zhang, X. Shao, Q. Zhou, W. Lei, H. Zhou, A. Breen, Pruning redundant synthesis units based on static and delta unit appearance frequency, in *Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH'15)*, 2015, pp. 269-273.
[15]. J. Nurminen, H. Silén, M. Gabbouj, Speaker-specific retraining for enhanced compression of unit selection text-to-speech databases, in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH'13)*, 2013, pp. 388-391.
[16]. P. Sharma, V. Abrol, A. K. Sao, Reducing footprint of unit selection-based text-to-speech system using compressed sensing and sparse representation, *Computer Speech & Language*, Vol. 52, 2018, pp. 191-208.
[17]. M. K. Jayesh, C. S. Ramalingam, A one-dimensional search method with stable 1-norm solution for linear prediction, *The Journal of the Acoustical Society of America*, Vol. 142, Issue 2, 2017, pp. 170-176.
[18]. D. Giacobello, M. G. Christensen, T. L. Jensen, M. N. Murthi, S. H. Jensen, M. Moonen, Stable 1-norm error minimization based linear predictors for speech modeling, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 22, Issue 5, 2014, pp. 912-922.

Chapter 2

Graphs as Tools to Improve Deep Learning Methods

Carlos Lassance, Myriam Bontonou, Mounia Hamidouche, Bastien Pasdeloup, Lucas Drumetz and Vincent Gripon

2.1. Introduction

In recent years, Deep Neural Networks (DNN) have known an important rise in popularity. A milestone was reached in 2012, when for the first time, a DNN called AlexNet [29] won the CVPR LSRV challenge [48] by a large margin. Since then, outstanding performance on many important problems have been achieved with DNN, particularly in the fields of computer vision [17, 29], natural language processing [57] and gaming [51]. Other domains such as text translation [24], speech to text [60], person identification and verification [16], help in screening and diagnosis in medicine [7], object detection [62], user behavior study [42] or even art restoration [12] have also gained momentum in the past few years. This performance of DNN systems in a large number of application domains is largely due to the accessibility of large training datasets and to large computational processing resources as with GPUs [13].

DNN models are thus state-of-the-art in many machine learning challenges, even if they have limitations. For example, DNN require a lot of training data, which might not be available in some practical applications. In addition, even though trained models seemingly perform well, if small perturbations are added to their inputs, they are prone to misclassification errors [53]. This susceptibility can be critical in some applications such as self-driving vehicles. Furthermore, DNN models are often viewed as “black-boxes” and as such their decisions are often

Myriam Bontonou,
ENS de Lyon, CNRS, LBMC, Lyon, France

criticized for their lack of *interpretability* [55]. Indeed, there are a lot of domains that would benefit from explainable models, among which product-recommendation, targeted advertising, medical diagnosis…

There is a clear explanation to the above-mentioned shortcomings. Indeed, DNN were introduced as a mean to replace "hand-crafted" features with learned ones. This is performed thanks to the use of automatic differentiation to train a model end-to-end, meaning that only the output is explicitly constrained during the learning process. And yet DNN are compositional models obtained by assembling elementary functions called *layers*. As such, they produce multiple intermediate representations when processing an input element. The intermediate representations are rarely explicitly constrained during training with the intention of removing any arbitrary human intervention. However, giving more latitude to the training process brings the disadvantage of harder interpretability, unwanted behaviors and more data requirements.

In recent years, authors have shown that processing multiple inputs at once can help in designing more efficient models. This is the key ingredient behind the popular *batch-normalization* [20] layer. Batch-normalization introduced the promising idea of constraining the intermediate representations of a batch of inputs relatively to each other. As such, there is no explicit constraint on a single representation, that is still freely optimized during the training phase. But multiple representations are constrained to exhibit certain behaviors, such as competition in the case of batch-normalization. More generally, exploiting the relations between different samples of the training set is a direction of research to address the above-mentioned problems.

In this chapter, we review recent works that aim at using graphs as tools to improve deep learning methods. These graphs are defined considering a specific layer in a deep learning architecture. Their vertices represent distinct samples, and their edges depend on the similarity of the corresponding intermediate representations. The same principle can be deployed to define graphs at the input, or at the output of deep learning architectures. These graphs can then be leveraged using various methodologies, many of which built on top of graph signal processing [50].

The outline of this chapter is as follows. We first introduce the needed concepts from DNN and GSP frameworks in Section 2.2. We then review the recent contributions aiming at using graphs to improve deep

learning methods in Section 2.3. This section is composed of four main parts: tools for visualizing intermediate layers in a DNN, denoising data representations, optimizing graph objective functions and regularizing the learning process. Section 2.4 provides some concluding remarks.

2.2. Background

2.2.1. Deep Neural Networks for Computer Vision

In this section, we introduce the fundamental aspects about DNN that will be of use in the remaining of the document.

A DNN is a mathematical function f that associates an input $\boldsymbol{x}$, typically in a multidimensional array, with a corresponding output $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$. It is obtained by assembling elementary functions that are called *layers* in the literature. In the simplest case, this assembly consists in a simple composition of the layer functions, so that the DNN can be mathematically described as

$$f = f^{\ell_{\max}} \circ f^{\ell_{\max}-1} \circ \ldots \circ f^{1}, \tag{2.1}$$

where each f^{i} is a layer function. In more complex cases, it is often the case that a DNN function can be decomposed as $f = g \circ h$, where g and h are possibly complex assemblies themselves.

When processing an input element $\boldsymbol{x}$, it is therefore possible to decompose two parts $\boldsymbol{z} = h(\boldsymbol{x})$ and $\hat{\boldsymbol{y}} = g(\boldsymbol{z})$. In such a case, we call $\boldsymbol{z}$ an *intermediate representation* associated with $\boldsymbol{x}$. In practice there are multiple intermediate representations associated with an input $\boldsymbol{x}$ for multiple layers in the considered DNN.

Most layers contain trainable *parameters*. Their values are initialized at random and updated during a training phase. This phase usually consists in minimizing a loss function, which is a function of the discrepancy between the actual output $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$ of the DNN and the expected one $\boldsymbol{y}$. This loss is minimized on a *training set*. In the literature, many architecture designs f can be found, some of which are particularly popular. One example is the Residual Network (ResNet) [17], that typically achieves good performance with a limited design complexity.

As various designs exist, it is common to rely on cross validation to choose the one that yields the best performance. In that case, the training

set is split into two parts prior to training. The first part is used as a new training set for the DNN. The second part, called the *validation set* is used to pre-evaluate the ability of the trained model to generalize to new data. More precisely, at the end of the training, we obtain the final architecture with the corresponding parameters Then, one evaluates the accuracy of the DNN on the validation set. The accuracy is the number of correctly predicted data points out of all the validation set The chosen design is the one that generalizes better. When the error on the training set is small, whereas the error on previously unseen inputs is high, the DNN is *overfitting*. To assert that the chosen DNN does not overfit and can be used to process previously unseen inputs, it is evaluated on a third dataset, called *test set*.

Limitations of DNNs

Considering a large amount of labeled data, and extensive computational power, DNNs are in many cases the golden standard of machine learning problems. However, major limitations refrain the adoption of these methods in specific domains. This chapter is built around four limitations that are all tackled using graph-based representations.

The first limitation is the **lack of interpretability** of the decisions taken by DNNs. This is problematic for applications that aim at assisting human experts (e.g. assisted medical diagnosis). Since it is difficult to understand how the DNN processes information, its ability to take a good decision on new inputs is assessed via the test set. However, a DNN performing well on a test set is not guaranteed to perform well on slightly different or unexpected inputs. Various methods have been proposed to visualize the features that matter in the decision process, both in the inputs and in the learned intermediate representations [47, 1, 53], and Fergus]. To directly measure the ability of generalization, without necessarily needing a validation set, some metrics have also been proposed [11, 4]. In Subsection 2.3.1, graph-based measures are introduced to assess the ability of generalization of DNNs.

The second limitation is the difficulty to **train models with little data**. In the literature, learning on very few labeled examples is called *few-shot learning*. More generally, we a few-label is a task that is described as N-way K-shot U-unlabeled, where N is the number of classes to distinguish within data, K the number of labeled images available per class during training, and U unlabeled images per class may be available for training. In the case of a few-shot task U is always equal to 0.

Many popular solutions for few-shot problems rely on *knowledge transfer* [54, 27]. The core idea is to exploit a DNN $f' = g' \circ h'$ trained on a distinct, yet similar and larger dataset, for the newly considered problem. More precisely, instead of training a new DNN f directly on raw inputs $\boldsymbol{x}$, a function g, taking $h'(\boldsymbol{x})$ as its input, is trained. In such a case, h' is called a *feature extractor* and g a *classifier*, $f = g \circ h'$ is therefore the new DNN function. The presented approach is preferred when the small dataset is very limited [3, 40]. If more data is available, it is also common to *fine tune* h' as well by training f for a few epochs [27]. An epoch refers to a pass of the gradient descent algorithm through the entire training set. Subsection 2.3.2 introduces ways of using graphs to help denoising the representations learned by the feature extractors, in order to better handle the underlying problem.

The third limitation lies on the fact that the output of the DNN is most of the time independent of the distribution of the input and of the initialization of the network parameters, which can slow down and complicate the training process. To understand this limitation, consider the cross-entropy loss function, the most popular loss function for classification problems in computer vision. In this case, the dimension of the output vectors has to be equal to the number of classes, preventing an easy adaptation to the introduction of new classes. In scenarios where the number of classes is large, this also causes the last layer of the network to contain a lot of parameters. Moreover, the cross-entropy loss considers that it is possible to completely separate the classes but does not take into account a possible hierarchy or similarity between classes. To overcome these drawbacks, one alternative to this is to optimize the embeddings directly, without forcing arbitrary separation such as the case of the siamese [26], triplet [16] and smoothness losses [5]. This limitation is addressed in Subsection 2.3.3.

The last considered limitation is the **lack of robustness** that occurs when DNN architectures are susceptible to deviations of their inputs and thus perform poorly [53, 10]. The lack of robustness may cause errors in decisions in various practical settings, in which data is prone to noise or manipulations. When discussing robustness of DNNs, it is common to separate two sources of deviations. The first source of deviation is the one that is not performed intentionally to fool the DNN [15][3]. Examples of such deviations range from defects in the data capture (e.g. Gaussian noise) to different luminosity conditions (e.g. brightness and contrast).

[3] Such deviations are called *corruptions* in this chapter.

The second deviation is the one that is created explicitly to fool the network, also called adversarial attacks. In this case, a metric is used to measure the distance between the original image and the adversarial one [10]. A successful adversary is the one that fools the DNN, without trespassing a perceptibility threshold on the metric (e.g. the adversary does not change any individual pixel color of more than 8/255 [38]). This limitation is addressed in Subsection 2.3.4.

2.2.2. Graph Signal Processing

Graph Signal Processing (GSP) offers a convenient framework for studying data while taking into account the complex domain on which they are defined. This section introduces the main tools that are of use in GSP. In the remainder of this chapter, we will use them to express desirable constraints on DNNs.

Graphs and Graph Signals

Consider a dataset comprising n samples, and suppose that there exist specific relations between them. For example, samples can be individuals in a social network and relations are given by their friendship connections. A concise and convenient way to model samples and their relations is to use the formalism of graphs.

A *graph* $\mathcal{G}$ is a tuple of sets $\langle \mathbb{V}, \mathbb{E} \rangle$, where $\mathbb{V}$ is composed of *vertices* indexed from 1 to n: $\mathbb{V} = \{v_1, \dots, v_n\}$, and $\mathbb{E}$ is composed of pairs of vertices of the form (v_i, v_j) called *edges*. These edges are unordered, such that (v_i, v_j) refers to the same pair as (v_j, v_i).

It is common to represent the set $\mathbb{E}$ using an edge-indicator symmetric *adjacency matrix* $\boldsymbol{A} \in \mathbb{R}^{|\mathbb{V}| \times |\mathbb{V}|}$. When relations are weighted, this matrix can take values other than 0 and 1.

In a graph, some vertices might have more importance than others. A common way to measure importance of vertices is to define their *degree* (or *strength* when graphs are weighted). The degree of a vertex v is simply obtained by summing the weights of all edges v is part of. These quantities can be assembled in the diagonal *degree matrix* $\boldsymbol{D}$:

$$D_{i,j} = \begin{cases} \sum_{j' \in \mathbb{V}} A_{i,j'} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Now let us suppose that each of the n samples is associated with a vector of dimension d. This can be represented as a $n \times d$ matrix. Columns of this matrix are called graph signals. For example, the age of each individual in a social network can be viewed as a graph signal.

As a natural representation of complex data structures, graphs and graph signals are ubiquitous, in particular in the field of machine learning. In the following paragraphs we introduce operations on graphs that are going to be useful in the rest of this chapter.

Graph Fourier Transform

In the domain of signal processing, *frequencies* are one of the most important concepts to analyze a signal and a starting point to introduce many useful tools such as filtering. The frequency of a signal can be simplified as its rate of change, or in other words how it varies from one sample to another. Generally speaking, any signal can be decomposed into a (possibly continuous) sum of sines/cosines of various frequencies, by performing the celebrated Fourier transform. This decomposition can be understood as expressing the signal in the frequency domain, providing a dual representation of the original signal in the time domain. In other words, the Fourier transform of a signal is a simple change of basis, that transports a signal to a convenient domain for many signal processing operations.

GSP arose as a graph-centric generalization of the classical Fourier analysis [50]. In this framework, a ring or a path-shaped graph is used to model the time domain, in which each vertex models an instant in time where the intensity of the signal is sampled. One therefore manipulates two separate objects: a graph $\mathcal{G}$ modeling time – the support of information – and a graph signal $\boldsymbol{s}$ – the sampled information. In a path graph, each vertex is connected with at most one subsequent vertex (in this case the next sample in the time domain) and at most one previous vertex (in this case the previous sample in the time domain). This means that in a path graph there exist two vertices with only one edge, called “first” and “last”. A regular ring graph is then defined as a path graph where the “first” vertex is connected to the “last” (one-dimensional lattice), forming a cycle with periodic boundary conditions, that is very much a natural subject for Fourier analysis.

The core observation of GSP is that Fourier transform is closely tied to the domain over which signals evolve, and not to the signals themselves. More precisely, consider a graph whose adjacency matrix only contains nonnegative values. Its combinatorial Laplacian $\boldsymbol{L}$ is defined as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2.3}$$

As the Laplacian matrix is both real and symmetric it can be eigendecomposed into:

$$\boldsymbol{L} = \mathbf{F}\boldsymbol{\Lambda}\boldsymbol{F}^{T}, \tag{2.4}$$

where the columns of $\boldsymbol{F}$ are the eigenvectors of $\boldsymbol{L}$ and the diagonal coefficients of $\boldsymbol{\Lambda}$ are the eigenvalues in increasing order of magnitude.

If we consider the regular ring graph mentioned above that has a periodic property to model time as a support of information, one can show that the eigenvectors in $\boldsymbol{F}$ are described as sine/cosine functions (Fourier series), with increasing frequency as associated eigenvalues increase. The Fourier transform of a graph signal $\boldsymbol{s}$ is thus simply a change of basis defined by $\boldsymbol{F}$, and can be written as:

$$\hat{\boldsymbol{s}} = \boldsymbol{F}^{T}\boldsymbol{s} \tag{2.5}$$

This projection of $\boldsymbol{s}$ in the eigenbasis $\boldsymbol{F}$ of $\boldsymbol{L}$ is what we call the *Graph Fourier Transform (GFT)* of $\boldsymbol{s}$. Note that the inverse GFT can be similarly defined as:

$$\boldsymbol{s} = \boldsymbol{F}\hat{\boldsymbol{s}} \tag{2.6}$$

GFT obviously depends on the graph that is used to model the signals under study. As a consequence, the choice of the underlying graph is of paramount importance. A path (or ring) graph provides a comprehensive domain to study (periodic) time signals, but it is often not clear which graph should be used for more complex signals. However, it still holds that eigenvectors $\boldsymbol{F}$ of the Laplacian matrix of a graph characterize the variations over that graph, with increased local variations as the associated eigenvalues increase. The interested reader can refer to [50] and [44] for illustrative examples.

Exploiting this property, authors have defined many tools generalizing signal processing operations on temporal signals, such as convolution,

filtering, translation or modulation [50]. We present in the next sections some of the uses of the GFT relevant to the works introduced in this chapter.

Smoothness of graph signals. A measure that will be at the core of many of the tools we introduce in the next section is the *graph smoothness* Let $\mathcal{G}$ be a graph of adjacency matrix $\boldsymbol{A}$, and let $\boldsymbol{L}$ be its combinatorial Laplacian, with eigenvalues $\boldsymbol{\Lambda}$. The smoothness $\sigma(\boldsymbol{s})$ of a graph signal $\boldsymbol{s}$ supported on $\mathcal{G}$ is defined as:

$$\sigma(\boldsymbol{s}) = \boldsymbol{s}^T \boldsymbol{L}\boldsymbol{s} = \sum_{i=1}^{|\mathbb{V}|} \Lambda_{i,i} \hat{s}_i^2 = \sum_{i=1}^{|\mathbb{V}|} \sum_{j=1}^{|\mathbb{V}|} A_{i,j}(s_i - s_j)^2 \quad (2.7)$$

Lower values of the smoothness metric are said to be very smooth with respect to the graph, and higher values of the smoothness metric are said to be very rough (or unsmooth) with respect to the graph. Equation (2.7) shows that a smooth signal is a signal aligned with the first eigenvectors of the Laplacian. In addition, we observe that the smoothness is strongly related to the rate of change of the signal values from one vertex to its neighbors. Considering the smoothness as the rate of change in a signal $\boldsymbol{s}$ across a graph, it is a very useful abstraction, even more when $\boldsymbol{s}$ is binary. In this case, the smoothness can be simplified as the sum of the weights between neighboring nodes with different values in $\boldsymbol{s}$.

Consider an illustrative case where vertices represent samples in a classification dataset. An interesting signal is the label indicator vector of a chosen class, which is a binary vector that contains a 1 at coordinate i if i is of the considered class and 0 otherwise. In such a case, the smoothness boils down to the sum of the weights of edges connecting pairs where one element is of the considered class and the other is not. Therefore, the only graphs that would nullify smoothness of this signal are those that contain no edge between samples of the class and samples of a distinct class.

Graph filters. In classical discrete signal processing, filters are tools meant to act on signals by removing specific ranges of frequencies. In the following, we develop the equivalent concept of graph filters for graph signals in GSP.

Here, we introduce three possible representations of these filters. For each representation, we provide its advantages/drawbacks and detail some of their applications. We refer the reader to [14, 56] for a more in depth discussion on graph filters.

1) Design of filters in the spectral domain. The simplest and most general way to define a filter is to describe its response for each frequency. In the case of graph signals, frequencies are defined by the diagonal values of the eigenvalue matrix $\boldsymbol{\Lambda}$. For notation simplicity we consider the frequency vector $\boldsymbol{\lambda}$ where $\lambda_i = \Lambda_{i,i}$, sorted in increasing order of magnitude.

We define a filter on a graph $\mathcal{G}$ using a diagonal matrix $\boldsymbol{H}_{\mathcal{G}} \in \mathbb{R}^{|\mathbb{V}|\times|\mathbb{V}|}$, where we call each element $H_{i,i}$ the response of the filter to frequency λ_i. The filter can then be convoluted with a graph signal $\boldsymbol{s}$ as in classical signal processing, by performing a multiplication in the spectral domain [50]:

$$\tilde{\boldsymbol{s}} = \boldsymbol{F}\boldsymbol{H}_{\mathcal{G}}\boldsymbol{F}^T\boldsymbol{s} \tag{2.8}$$

Defining filters directly in the spectral domain generates filters that are graph-specific. Indeed, as the frequencies λ_i are discrete, the same filter $\boldsymbol{H}$ will be applied to very different frequencies for two distinct graphs $\mathcal{G}$ and $\mathcal{G}'$. Therefore this type of filter tends to be mostly used to remove the lowest or highest frequencies of the graph, without considering their "true" value. Another drawback is that it is unlikely that this type of filter may be represented with a low order polynomial filter which impacts the complexity (i.e. the possibility of scaling to larger graphs) of applying the filter.

2) Design of filters using their spectral response. Another straightforward way to define a filter is by its spectral response, i.e. a function of the frequency. In this way the filter becomes less graph-dependent and more general. One such design is the Simoncelli filter, depicted in Fig. 2.1, where $\tau \in [0,1]$ is a user-defined threshold and λ_i the i -th Laplacian eigenvalue.

Defining a filter by its spectral response allows for more universal filters (i.e. filters that do not heavily depend on the graph support) and also to easily represent the filter by a low order polynomial function. In this work, we use the PyGSP [9] toolbox to implement this type of graph filters, which uses the Chebyshev polynomial approximation in order to apply the filters.

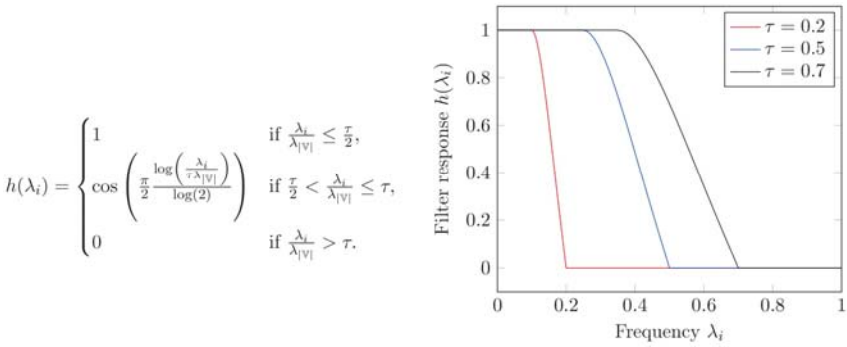

Fig. 2.1. Equation (left) and depiction (right) of the Simoncelli filter spectral response for various values of τ.

3) Design of filters using diffusion operators. It is also possible to define a graph filter using a *diffusion operator* $\boldsymbol{S} \in \mathbb{R}^{|\mathbb{V}|\times|\mathbb{V}|}$, sometimes also called *graph shift operator*. A diffusion operator on a graph $\mathcal{G}$ is a matrix $\boldsymbol{S}$ describing the same edge set as the adjacency matrix $\boldsymbol{A}$ of $\mathcal{G}$, i.e. such that $S_{i,j} = A_{i,j}$ where $A_{i,j} = 0$. It allows to define the notion of information flow over a graph, as it represents one elementary discrete step on how the information propagates (shifts) from one node to its neighbors. In classical signal processing – where the graph structure can be modeled as a directed cycle graph – the information flow is unidirectional, and goes from each node to its only neighbor. In a more complicated graph structure, the information flow is neither restricted to unidirectional structure nor to a limited number of physical neighbors but depends on the graph adjacency matrix. When information flows to a vertex from all the vertices it is neighbor of, incoming signal is combined through a weighted sum as follows [49]:

$$\tilde{\boldsymbol{s}} = \boldsymbol{S}\boldsymbol{s} \tag{2.9}$$

Note that by being applied with just a simple matrix multiplication, this type of filter is easily integrated in deep learning scenarios, where matrix multiplication is prevalent. Indeed, most of the recent developments in graph convolutional layers use this design.

Example of Application of GSP

Here is an illustrative example showing the usefulness of the notion of graph filters and smoothness. Fig. 2.2 depicts a graph signal on a path graph, a noisy version of it, and a low-pass filtered version of the noisy

graph signal. Note that both the original graph signal and the filtered version are smoother than the noisy version of the signal. Thus, the smoothness of a graph signal can also be used to detect the presence of noise in the signal.

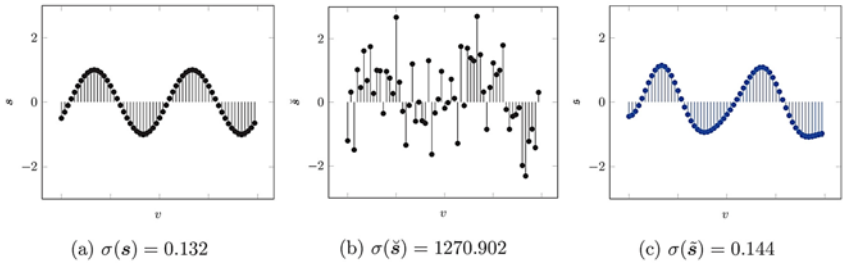

Fig. 2.2. Depiction of a sampled signal $\boldsymbol{s}$, its noisy version $\check{\boldsymbol{s}}$ and its filtered version $\tilde{\boldsymbol{s}}$, with their respective smoothness values.

In addition, here is an example of a picture, seen as a signal of dimension 3 (RGB) over a grid graph. Using a low-pass graph filter, it is possible to exploit the structure of the pixels to remove noise from the image as shown in Fig. 2.3. By removing the high frequencies of the graph signal, the resultant image is more in line with the original image. Note also that the smoothness value of the recovered graph signal is the same as the original image, while the images are very different. Indeed as the smoothness of a graph signal is a global measure and not a localized one it is easy to see that multiple image configurations are possible for the same value of smoothness.

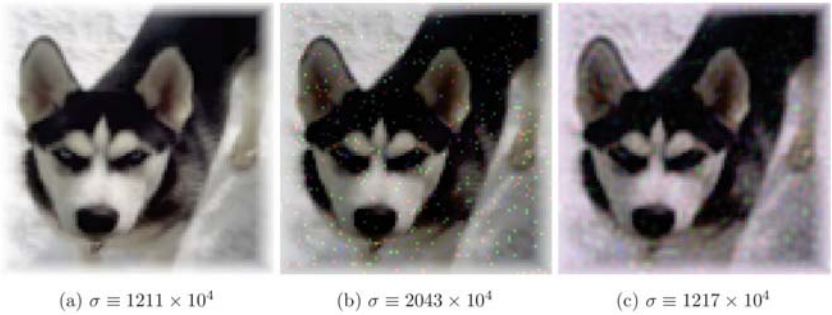

Fig. 2.3. Depiction of a dog (left), a noisy realization of the image (center) and the graph filtered image (right).

2.3. Graphs as Tools to Improve Deep Learning Methods

Now that we have introduced DNNs, some of their limitations and presented the framework of GSP, we delve in more details into how to improve deep learning methods using graphs.

2.3.1. Graphs to Interpret Intermediate Representations

Using Smoothness to Measure the State of DNNs

Throughout the training process of a DNN, an input vector is transformed into multiple intermediate representations of various dimensions and shapes. Understanding how these intermediate representations evolve during the training process is a major open question.

In particular, it is difficult to know whether a trained architecture is able to solve the problem it was trained for or not. To address this limitation, cross validation is often performed to measure the state of a DNN: underfitted, overfitted or correctly optimized. However, in many practical cases, it is problematic to rely on cross validation. Indeed, the training set is reduced in size, requiring to tune parameters with a limited number of examples. Also, the model generalization abilities are assessed using a fraction of the training set, giving a possibly biased judgement on their real performance on unseen data.

A more principled approach was proposed in [11] where the authors propose to use GSP to monitor the training of DNNs. Their method is based on the notion of smoothness of graph signals. Namely, a graph $\mathcal{G}$ is constructed at different layers of the DNN. Its vertices are the intermediate representations of the training samples and its edges are the similarities between these samples. The graph signal $\boldsymbol{s}^c \in \mathbb{R}^{|\mathcal{V}|}$ indicates whether the vertices belong to the class c. More precisely, given a vertex i, $s_i^c = 1$ if the class of the i -th sample is c and 0 otherwise.

Let M be the number of examples in each class c, C be the number of classes and $\boldsymbol{L}$ the Laplacian of $\mathcal{G}$. For a given graph signal $\boldsymbol{s}^c$, the label smoothness is then defined as:

$$Smooth = \frac{\sum_{c=1}^{C} \boldsymbol{s}_c^T \boldsymbol{L} \boldsymbol{s}_c}{M^2 C(C-1)}$$

The term $M^2(C-1)$ is a normalization factor. The label smoothness can be rewritten as the sum of the similarities between samples of distinct classes. Thus, the label smoothness is 0 if and only if the similarity between samples of distinct classes is 0.

The authors of [11] were able to show that the label smoothness can clearly differentiate different states of trained DNNs. More precisely, they observed that when the considered architecture is correctly optimized, the values of label smoothness on the last layers of the trained architecture are similar, whereas in the other conditions, they are separated by important gaps.

They also observed that during training, the label smoothness continues to evolve when the training accuracy stabilizes around 100 %. Indeed, this motivates the idea that even if the training accuracy is maximal, the intermediate representations are still changing. Therefore, the label smoothness could be used as a measure to stop or continue the DNN training when the training accuracy has reached its maximal value.

Detecting the Influential Instances

To better interpret a DNN, one can look at many different variables, e.g. the training examples that influence the most the predictions or the most influential pixels inside an image. In this subsection, a method called *model analysis and reasoning using graph-based interpretability* (MARGIN) [1] is presented. In this method, the studied influential variables are called *influential instances*. The method is divided into four steps:

1. *Influential instances*: First, the domain on which the analysis is performed is described. It can be the set of training samples, where each instance is an image, or even a single image, where a group of neighboring pixels is an instance.
2. *Graph construction*: A graph $\mathcal{G}$ modeling the relationships between the instances is constructed. The vertices are the instances and the edges are the similarities between the instances. The similarities can be computed from the raw instances or, when applicable, from intermediate representations of the instances within the DNN.
3. *Graph signal*: The graph signal represents the source of variations on which the influence of each instance is studied. For example, considering that an instance is a training example, the graph signal

could be a measure indicating how much the neighbors of the example in $\mathcal{G}$ disagree with its label.

4. *Sample-wise influence estimation*: MARGIN aims at finding the vertices of the graph $\mathcal{G}$ that characterize the most the variations of the signal with respect to the structure of the graph $\mathcal{G}$. Using the formalism of GSP, a high-pass filter is applied over the graph signal. The influence of each vertex is proportional to the amplitude of the filtered signal.

In [2], the MARGIN protocol is evaluated on three tasks: showing which parts of an image influence the most the prediction made by a DNN, pointing out to the hardest training examples to classify and detecting adversarial attacks. Experiments are performed using a common dataset of image classification and the Alexnet [29] DNN architecture.

MARGIN enables to obtain sparse saliency maps highlighting the importance of some group of pixels in the classification decision. MARGIN also enables to find hard examples. In a two-way classification task, trying to distinguish tabby cat and great dane, the images with the highest influence values (harder to classify) are the ones where the animal face is not visible. MARGIN is also useful to detect adversarial examples. An experiment shows that MARGIN induces a very sharp difference between the distribution of the adversarial examples and the one of the training examples, with only a small overlap in the distributions. These overlaps have been investigated and it was shown that they correspond to samples that are similar to the confusing examples from the previous experiment.

2.3.2. Graphs to Denoise Intermediate Representations

In this section, we introduce a few examples of methods based on GSP and aiming at "denoising" the features extracted from DNNs.

Graph Filtering for Localizing Images

Visual-based localization (VBL) is the problem of retrieving the location and orientation (pose) of the camera which generated a given query image. For example, from one photograph of a place, the goal is to retrieve its GPS coordinates. Many different photographs can be taken from one place, with different angles of view, various weathers, different people on them. Thus, the VBL problem becomes challenging because

of the difficulty to learn a representation which is resilient this amount of appearance variations.

Some researchers [6, 25] proposed to directly train a DNN to map images to GPS positions [6, 25]. However, such method comes with serious limitations. For example, they are unable to generalize to GPS positions unseen during training. Also, appending new GPS positions to the dataset requires retraining the whole DNN. Furthermore, small differences between the training images and a new image can cause significant localization errors.

In this subsection, another method addressing both limitations is presented [32]. It is based on Graph Signal Processing, and aims at improving the performance of VBL by incorporating additional available information. The additional information acts on the representations of the data, making them smoother on a graph designed using all available information. The consequence is a boost in localization performance.

The method requires both a support set of images from a predefined set of places and the access to a DNN trained to map images to representations that are more resilient to appearance changes [3]. The method does not require additional learning, allowing it to be possibly executed on a resource constrained system. Interestingly, it can be extended to new places without needing to retrain as well.

In details, the proposed method consists in denoising the representations exploiting a low-pass graph filter [3, 46]. In the graph $\mathcal{G}$, each vertex is associated with an image. The edges model relations between images and are derived from the additional source of information (e.g. duration between two consecutive photographs, GPS positions, similarities of the representations). In the following, the inference of the graph $\mathcal{G}$ and the graph filter are defined more precisely.

The graph $\mathcal{G}$ is described by its adjacency matrix $\boldsymbol{A}$ built from three different sources of information:

$$\boldsymbol{A} = \boldsymbol{A}_{dist} + \boldsymbol{A}_{seq} + \boldsymbol{A}_{sim}, \tag{2.10}$$

where $\boldsymbol{A}_{dist}$ represents the distance measured by the GPS coordinates between two vertices, $\boldsymbol{A}_{seq}$ represents the delay in time acquisition between two images (acquired as frames in videos), $\boldsymbol{A}_{sim}$ represents the cosine similarity between the representations of the two images.

The matrix $\boldsymbol{A}_{dist}$ is parametrized by a scalar γ and a threshold parameter $dist_{\text{max}}$ cutting edges between distant vertices:

$$A_{dist}[i,j] = \begin{cases} e^{-\gamma dist_{i,j}} & \text{if } \text{dist}_{i,j} < \text{dist}_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

To exploit the information of time acquisition of frames, the function $seq(k,i,j)$, returns 1 if the frame distance between i and j is exactly k and 0 otherwise. The matrix $\boldsymbol{A}_{seq}$ is parametrized by scalars β_k and k_{max}:

$$A_{seq}[i,j] = \sum_{k=1}^{k_{max}} \beta_k \text{seq}(k,i,j) \tag{2.12}$$

The matrix $\boldsymbol{A}_{sim}$ depends on the cosine similarity sim_{cos} between the representations of the images. It is parametrized by a scalar α, which controls the importance of the similarity with respect to the two other sources of information. The cosine similarity between two images is only considered if their GPS positions are close or if the photographs have been taken successively:

$$A_{sim}[i,j] = \begin{cases} \alpha sim_{\text{cos}}(i,j) & \text{if } A_{dist}[i,j] > 0 \\ & \text{or } A_{seq}[i,j] > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

Given the signal $\boldsymbol{s}$, the normalized Laplacian matrix $\boldsymbol{L}_{\text{norm}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$ and two hyperparameters a and m, the filtered signal is defined by the equation:

$$\tilde{\boldsymbol{s}} = (\boldsymbol{I} - a\boldsymbol{L})^m \boldsymbol{s} \tag{2.14}$$

Note that when $m = 0$ no filtering is performed.

The authors stressed their method using two datasets collected in Australian cities (Adelaide and Sidney). The datasets are composed of videos taken by vehicles while they were moving around the cities. The goal is to learn to retrieve the GPS position of each frame.

The experiments show that on the Adelaide dataset, denoising increases performance, even when applied only on the test set, and as expected, using graph filters on both test and support gives the best results. Second, on the Sydney dataset, while using the parameters optimized for the

Adelaide dataset, the graph filter gets better performance in both median distance and accuracy. This shows that it is not needed to re-estimate parameters for a new dataset.

Manifold Denoising for few-label Classification

In classification, training a DNN requires a huge number of labeled data samples. When a DNN is trained on a small set of samples, it is likely to learn many irrelevant details of the data. It will not infer well the class of new samples coming to the classifier. This is a well known problem called *overfitting* (see Section 2.2.1). To handle overfitting, one can take advantage of the fact that DNNs are really good at extracting relevant information from data. Indeed, as a reminder a DNN can often be split into two functions $f = g \circ h$. The first part h learns to represent the data samples in a space where a simple classifier g can separate the data samples according to their classes. For example, when dealing with images, it is common to choose h as a convolutional neural network and g as a logistic regression.

The features of the data samples at the output space of the function h are often assumed to be only in a smooth subspace (manifold) of the entire output space. Sometimes, because of noise within the representations, features may appear outside this manifold. Removing the noise from the representations could help in making the features closer to their underlying structure, and thus, improve the performance.

The denoising method presented in this subsection is used in a few-label setting in which few labeled samples $\boldsymbol{X}_S$ and many unlabeled samples $\boldsymbol{X}_U$ are given. Consider a DNN, trained on a generic dataset, used to extract some features for a new few-shot classification task. A classifier will be trained on top of the extracted features to distinguish the new classes. However, the features may not be completely relevant for the new classification task. One may act directly on the training of the DNN using self-supervision, manifold-mixup or even training on a variety of side tasks [41, 43]. However, here, it is assumed that the features of all data samples are already given. In this case, one can exploit the additional information given by the representations of the unlabeled samples (more representative feature space). That is the case of a method based on GSP (Transfer+SGC [19]) that adds a simplified graph convolution (SGC) [59] on top of the extracted features. Note that SGC can be seen as a graph filter (previously defined in Section 2.2.2).

The authors use the notion of cosine similarity between the representations of two data samples in the feature space. Note that in the experiments, the features are always extracted after a ReLU function, so that they contain non-negative values. Thus, the output of the cosine similarity ranges from 0 to 1. The idea is to build a k -nearest neighbor similarity graph $\mathcal{G} = \langle \mathbb{V}, \mathbb{E} \rangle$, where each node represents a data sample. The connections between the nodes are weighted by the cosine similarity of their associated samples. After removing self-loops, only the k most similar neighbors are kept for each vertex. Then, the resulting adjacency matrix $\boldsymbol{A}$ is normalized using the inverse square root of the degree matrix: $\boldsymbol{E} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. Given $\boldsymbol{F}$ the matrix containing the features of all data samples and $\boldsymbol{I}$ the identity matrix, the new features are obtained by propagating the extracted features as follows:

$$\boldsymbol{F}_{\text{diffused}} = (\alpha \boldsymbol{I} + \boldsymbol{E})^m \boldsymbol{F} \ , \tag{2.15}$$

where α, m and k are the hyperparameters. On top of the diffused features of the labeled samples, a logistic regression is trained to distinguish
the labels.

The authors of [19] observed that their method enabled a significant gain in the 1-shot case and a smaller one in 5-shot with respect to existing alternatives. As a matter of fact, the added information coming from the unlabelled samples is relatively more important in the 1-shot case than it is for the 5-shot case.

Manifold Denoising for Semi-supervised Classification

Here, a different scenario is considered in which a DNN $f = g \circ h$ is learned from scratch. It takes into input an image and outputs its class without overfitting on the training samples $\boldsymbol{X}_S$. However, in many cases, the training data are still not representative of all the possible data in the real world. In a semi-supervised setting, one solution to better fit the real data distribution is to use the unlabeled samples $\boldsymbol{X}_U$ as additional training data for the DNN. In the following paragraphs, an approach described in [21] will be presented. This approach exploits the so-called “manifold assumption”: similar examples in the input space of the classifier g should have the same labels. This assumption is applied to label the unlabelled examples, so that they can be used as additional training data.

The method consists in merging the label propagation method, which is a classical semi-supervised learning method, with a DNN. First, a DNN is trained on the labeled examples to minimize a classification loss defined on the labeled images as:

$$L_s(X_S, Y_S, \theta) = \sum_{i=1}^{l} l_s(h(\boldsymbol{x}_i, y_i)) \ , \tag{2.16}$$

where l_s is the cross-entropy loss.

A nearest-neighbor similarity graph is built at the input space of the classifier g. More precisely, each node of the graph corresponds to an image (labeled or unlabeled). The adjacency matrix $\boldsymbol{A}$ of the graph is defined so that $A_{i,j}$ measures the similarity between $\boldsymbol{f}_i = h(\boldsymbol{x}_i)$ and $\boldsymbol{f}_j = h(\boldsymbol{x}_j)$. After removing self-loops ($A_{i,i} = 0$), only the k -th largest values on each row of $\boldsymbol{A}$ are kept. To make $\boldsymbol{A}$ symmetric, its transposed version is added to it. As in the previous paragraphs, a normalized version of the adjacency matrix is used.

The core idea of the approach is to propagate the known labels on the similarity graph to define pseudo-labels for the unlabeled images. Indeed, a label matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$ is defined as:

$$Y_{i,j} = \begin{cases} 1 & \text{if } 1 \leq i \leq l \text{ and } y_i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

The pseudo-labels are obtained by diffusing the label matrix $\boldsymbol{Y}$ according to $\boldsymbol{A}$ until convergence (theoretical details in [64] and practical implementation in [21]):

$$\boldsymbol{Z} = (\boldsymbol{I} - \alpha \boldsymbol{A})^{-1} \boldsymbol{Y} \ , \tag{2.18}$$

where α is a hyperparameter. The pseudo-label $\hat{y}_i$ of an unlabeled sample x_i is finally defined as:

$$\hat{y}_i = \arg\max_j z_{i,j} \tag{2.19}$$

The set of pseudo-labels attributed to the unlabelled samples is noted $\hat{Y}_U$. A weight is associated with each pseudo-label to measure its uncertainty. Finally, the DNN is fine-tuned for a few epochs to minimize a new weighted loss that considers the pseudo-labels. The new loss is defined as:

$$L_w(X_S, Y_S, X_U, \hat{Y}_U, \theta) = \sum_{i=1}^{l} \zeta_{y_i} l_s(h(\boldsymbol{x}_i, y_i) + \sum_{i=l+1}^{n} \omega_i \zeta_{\hat{y}_i} l_s(h(\boldsymbol{x}_i, \hat{y}_i)) \ , \quad (2.20)$$

where ζ_j is the reciprocal of the number of samples assigned to class j (known labels and pseudo labels).

The authors of [21] perform experiments using the CIFAR-10 and mini-Imagenet datasets. Both datasets are split into a training and a test set. In the training set, a number of images are labeled, the others are not.

In the semi-supervised setting, the error is reduced with respect to the fully-supervised setting. Without any surprise, the gain is observed to be decreasing when the number of labeled data points increases. To summarize, in this part it was described how to denoise the features of the data samples in three applications, where the graph filtering of either features or labels is considered. In the features case, it is assumed that the representations of all data samples should live on a manifold, and that the extracted features are noisy versions of these "ideal" representations. So, using a graph framework, it is possible to average the representation of the images with the most similar representations obtained with other images. In the second case, images having similar representations are assumed to have similar labels. Thus, the main goal is to smooth the label signal on the similarity graph whose vertices are the images and the edge weights are the similarity between the representations of two images.

2.3.3. Graphs as Losses

Training a DNN for classification often requires to optimize it in order to minimize a cross-entropy loss. As stated in Section 2.2.1, the cross-entropy loss introduces several biases in the learning process. This section gathers several alternative graph-based losses showing interesting properties.

Smoothness-based Loss

In this subsection, a graph-based loss handling the limitations of the very popular cross-entropy loss is presented. Recall that the cross-entropy loss requires the outputs of the network to converge to label indicator vectors, which comes with several shortcomings described in Section 2.2.1. Authors have proposed solutions to overcome these drawbacks. For

example, in [16], the authors use triplets as loss instead of the cross-entropy. In a triplet, the first element is the example to train, the second belongs to the same class and the last to another class. They update the parameters of the DNN so that the first example is closer to the second example than to the last one. Here, a similar loss based on GSP is presented. Contrary to the previous loss, the so-called graph smoothness loss only aims at maximizing the distances between outputs of different classes [5]. To avoid the DNN converging to a trivial solution that would scatter the outputs far away from each other regardless of their class, the outputs are constrained to remain in a compact subset of the output space.

Consider the problem of training a DNN f to classify examples. The input of the ith example is denoted $\boldsymbol{x}_i$ and the output of the DNN $\widehat{\boldsymbol{y}}_i$, such that $\widehat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i)$. Given a distance metric $\|\cdot\|$ and a parameter α, a graph $\mathcal{G}$, whose vertices are examples, is generated. Its edges correspond to the similarity between the outputs of the examples. Thus, the adjacency matrix $\boldsymbol{A}$ of $\mathcal{G}$ is given by:

$$A_{i,j} \neq 0 \Rightarrow A_{i,j} = exp\big(-\alpha \parallel f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) \parallel\big), \forall i, \forall j \quad (2.21)$$

The matrix $\boldsymbol{A}$ is then thresholded in order to only keep the k biggest edges between the vertices. Finally, to store the information about a label c, a signal on $\mathcal{G}$ is defined as a binary indicator vector $\boldsymbol{s}_c \in \mathbb{R}^{|\mathbb{V}|}$. Hence, $s_c[i] = 1$ if and only if $\boldsymbol{x}_i$ has c as label.

According to the definition of the smoothness in Section 2.2.2, the graph smoothness loss over the graph $\mathcal{G}$ is given by:

$$\mathcal{L}_{\mathcal{G}} = \sum_{\forall c} \boldsymbol{s}_c{}^T \boldsymbol{L} \boldsymbol{s}_c = \underbrace{\sum_{\substack{\boldsymbol{x}_i, \boldsymbol{x}_j, A_{i,j} \neq 0 \\ s_c[i] s_c[j] = 0, \forall c}} exp\big(-\alpha \parallel f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) \parallel\big)}_{\text{sum over inputs of distinct classes}}.$$

The graph smoothness loss addresses the following drawbacks of the cross-entropy loss:

- The dimension of the DNN output is less tightly tied to the number of classes with the proposed loss than with the cross-entropy one;
- Keeping only k edges per vertex gives more flexibility to the proposed loss: using a small value of k, it is possible to minimize the graph smoothness loss with multiple clusters of points for each class;

- The graph-smoothness loss is only interested in relative positioning of outputs with regards to one another, and is therefore built upon the initial distribution yielded by the network.

In [5], the performance of the graph-smoothness loss is evaluated using three common datasets of image classification. For each dataset, a Resnet-18 [17] is trained with the cross-entropy loss and another one with the graph smoothness one. To perform classification with the graph smoothness loss, an additional classifier (in the paper a K-nearest neighbors classifier with $K = 10$) is also trained on top of the DNN. The authors show that the graph smoothness loss is able to compete in terms of raw performance with the cross-entropy one. They also perform robustness experiments where they observe that the networks trained with the graph smoothness loss are able to better handle additive noise to the processed inputs.

Graph Knowledge Distillation

To achieve state of the art results, DNNs often rely on a large number of trainable parameters, and considerable computational complexity. This is why there has been a lot of interest in the past few years towards their compression, so that they can be deployed onto embedded systems or in real-time settings. The purpose of this subsection is to analyze a graph-based technique, called Graph Knowledge Distillation (GKD), that allows for an efficient compression of the DNN architecture, while maintaining a high level of accuracy. This framework has been already used with success in the literature in [37, 32].

Distillation-based approaches aim at distilling knowledge from a pre-trained larger network that is called teacher to a smaller yet to be trained network called student. For simplicity, assume that both architectures always generate the same number of intermediate representations, even if they do not have the same depth. The loss function of the student network trained with knowledge distillation is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{KD}} \cdot \mathcal{L}_{\text{KD}} \ , \tag{2.22}$$

where $\mathcal{L}_{\text{task}}$ is the same loss as the one used to train the teacher (e.g. cross-entropy), $\mathcal{L}_{\text{KD}}$ is the distillation loss and λ_{KD} is a scaling parameter to control the importance of the distillation with respect to that of the task.

Given an architecture A, a batch of inputs $\boldsymbol{X}$ and a subset of layers, the set $\mathcal{X}'$ contains the intermediate representations of $\boldsymbol{X}$ after all considered layers. For each layer, the representations after the layer $\boldsymbol{X}' \in \mathcal{X}'$ are used to define a similarity graph $\mathcal{G}^A(\boldsymbol{X}')$. These graphs contain a vertex for each input in the batch, and the weight of the edge between two vertices is inferred using a measure of similarity between the representations of the associated inputs. Finally, in order to control the importance of outliers, the adjacency matrices of the graphs are normalized ($\boldsymbol{A} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$).

While training the student, the training batch of inputs goes through both the student architecture and the (now fixed) previously trained teacher architecture. The loss to minimize combines the task loss, as expressed in Equation (2.22), with the following GKD loss:

$$\mathcal{L}_{\text{KD}} = \sum_{\boldsymbol{X}' \in \mathbb{X}'} \mathcal{L}_d(\mathcal{G}^S(\boldsymbol{X}'), \mathcal{G}^T(\boldsymbol{X}')) \ , \tag{2.23}$$

where $\mathcal{L}_d$ is the squared Frobenius norm between the adjacency matrices. The GKD loss measures the discrepancy between the adjacency matrices of teacher and student graphs. In this way, the geometry of the intermediate representations of the student is forced to converge to the one of the teacher. The intuition is that since the teacher network is expected to generalize well to the test, mimicking the geometry of its intermediate representations should allow the student network to generalize better. An equivalent definition of the proposed loss is:

$$\mathcal{L}_{\text{KD}} = \sum_{\boldsymbol{X}' \in \mathbb{X}'} \| \boldsymbol{A}^S(\boldsymbol{X}') - \boldsymbol{A}^T(\boldsymbol{X}') \|_2^2 \tag{2.24}$$

In [33], the advantage of using GKD compared with a baseline method (RKD-D [45]) is evaluated using CIFAR-10 and CIFAR-100 datasets [28]. The obtained results are analyzed on their: accuracies decision consistency spectral representations.

In terms of accuracy, the authors compare student sized networks trained without distillation (Baseline), with GKD and RKD-D GKD methods. The comparison shows that RKD-D by itself provides a small gain in error rate with respect to the Baseline approach, while GKD outperforms RKD-D by almost a similar amount. Furthermore, the performance of GKD are shown to be better when using task specific graphs. In other words, by removing the edges between elements of the same class. GKD with task specific graphs provides a gain over GKD of the same magnitude as GKD over RKD-D. Thus, by going from RKD-D to

GKD-task specific it was possible obtain a twofold gain over the baseline.

In terms of decision consistency, the authors evaluate the consistency by taking the trained students and comparing their outputs to the trained teacher's outputs. It was reported that the ideal scenario that greatly improves the classification performance occurs when the student is 100 % consistent with the teacher's decision on the test set. The experiment shows that the GKD is more consistent with the teacher than the RKD-D.

In terms of spectral representations, it is quite natural to analyze performance from a GSP perspective [50]. Considering specific graph signals, the respective smoothness on each of the two graphs (RKD-D and GKD) are compared. Two signals are considered: the label binary indicator signal the Fiedler eigenvectors from each intermediate representation in the teacher. Experiments show that both signals are smoother in the networks trained with GKD. This means that the geometry of the intermediate spaces from GKD are more aligned to those of the teacher.

Affinity-based Loss

In this subsection, we present another way of defining the loss of a DNN using graphs [58]. During training, a DNN is evaluated on a whole batch of data samples before being updated. The idea described in [58] is to ease the training by exploiting meaningful relationships between these data samples. In short, the idea amounts to add a regularization term to the usual loss of the DNN. While the usual loss is only focused on the task to achieve, the regularization term is more representative of the relationships between the data samples within the DNN.

Let us consider that a DNN is trained on a batch of N examples. The input of the ith sample is denoted $\boldsymbol{x}_i$. The representation of $\boldsymbol{x}_i$ obtained after a given layer of the DNN is denoted $f(\boldsymbol{x}_i)$. Depending on the task, f can represent a hidden layer or the output of the DNN.

The affinity between two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ within the DNN can be computed with any similarity function $\mathcal{W}$, as

$$A_{i,j} = \mathcal{W}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) \tag{2.25}$$

Then, to model the relationships between the examples within the DNN, a graph $\mathcal{G}$ is defined. Its vertices are the examples. The connection between two examples i and j is weighted by their affinity $A_{i,j}$. The adjacency matrix of this affinity graph is denoted by $\boldsymbol{A}$.

Now, the idea in [58] is to select a meaningful set of connections that we will aim at maximizing during training. For instance, in a classification task, we want to maximize the similarity between the outputs of data samples of the same class. So, the set of meaningful connections amounts to the set of connections between the examples of the same class. Formally, to select the relevant connections, we need to define a target matrix $\mathcal{T} \in \mathbb{R}^{N\times N}$. Given S the set of meaningful pairwise connections, the target $\mathbb{T}$ is defined as

$$\mathcal{T}_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in S, \\ 0 & \text{otherwise} \end{cases}$$

Finally, given $\widehat{\boldsymbol{A}} = \text{softmax}(\mathbf{A})$ a matrix-wise softmax operation, the quantity we aim at maximizing is:

$$\mathcal{M} = \sum \widehat{\boldsymbol{A}} \odot \mathcal{T}$$

In [58], the maximization of $\mathcal{M}$ is expressed as the minimization of a loss $\mathcal{L}_{\mathcal{G}}$. A discussion on the choice of the loss form is made in the original paper [58]. Finally, the total loss of the DNN is given by

$$\mathcal{L} = \mathcal{L}_{task} + \lambda \mathcal{L}_{\mathcal{G}},$$

where $\mathcal{L}_{task}$ could be the cross-entropy loss, and λ is a parameter that takes values in $[0,1]$. The minimization of the affinity loss $\mathcal{L}_{\mathcal{G}}$ places greater emphasis on the connections in S during training.

In [58], experiments are made on CIFAR-10, CIFAR-100 and Tiny-ImageNet are summarized. The experiments show that there is an improvement in terms of classification accuracy over a baseline, when using the affinity-based loss. In particular, for datasets with a large number of categories, such as CIFAR-100 (100 classes) and Tini-ImageNet (200 classes), the performance gain is above 1 %.

2.3.4. Graphs as Regularizers

In this section, a graph-based technique is used to improve the robustness of DNNs to deviations of the inputs. In other words, the goal is to ensure that a small difference in the input does not lead to a large difference in the output (decision) of the DNNs.

Bounding Smoothness Evolution Between Layers

In this subsection, a *regularizer* that penalizes large deformations of the class boundaries (i.e. distances between the closest elements of distinct classes) throughout the network architecture is presented [35]. The regularizer is defined independently of the types of perturbations that is expected to face when the system is deployed.

The regularizer is based on a series of graphs, one for each layer of the DNN. Each graph captures the similarity between the intermediate representations of training examples at that layer.

Given a batch of b inputs $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_b\}$ and a similarity measure sim, the adjacency matrix of the graph generated by intermediate representations at layer ℓ is defined as:

$$A^{\ell}_{i,j} = \text{sim}(\mathbf{x}^{\ell+1}_i, \boldsymbol{x}^{\ell+1}_j), \forall 1 \leq i, j \leq b \tag{2.26}$$

In the experiments, the cosine similarity is mainly used[4].

To measure the deformation induced by a given layer ℓ, a regularizer δ^{ℓ}_{σ} is computed as the difference between label signal smoothness before and after the layer for all labels. More precisely,

$$\delta^{\ell}_{\sigma} = \sum_c \left| \sigma^{\ell}(\boldsymbol{s}_c) - \sigma^{\ell-1}(\boldsymbol{s}_c) \right| , \tag{2.27}$$

where σ^{ℓ} is the smoothness (Section 2.2.2) of the binary label indicator vector on the intermediate representation graph from layer ℓ. These quantities are then used to regularize modifications made by each of the layers during the learning process.

The regularizer penalizes large changes in the smoothness (computed using the Laplacian quadratic form) of the class indicator vectors

[4] It is often the case that the output $\mathbf{x}^{\ell+1}$ is obtained right after using a ReLU function, that forces all its values to be nonnegative, so that all values in $\mathbf{A}^{\ell}$ computed with the cosine similarity are between 0 and 1.

(viewed as "graph signals") between layers. As a result, the margin is kept almost constant across layers, and the deformations of space are controlled at the boundary regions.More details can be found in the article [34].

Since only label signals are considered, only the similarities between examples of distinct classes are of interest. As such, the regularizer only focuses on the boundary, and does not vary if the distance between examples of the same label grows or shrinks.

The experiments show that the regularizer is able to increase robustness to random perturbations and to weak adversarial attacks on the CIFAR-10 dataset. To validate the generality of the method, the reader is referred to [31, 34], where the analysis is extended to two other datasets (CIFAR-100 [28] and Imagenet 32×32 [8]). In the following, the regularizer robustness to some common perturbations is discussed: noise [15, 39] adversarial attacks [53, 10] implementation defects.

The robustness of the network to noise is evaluated. The benchmark from [15] is used. The benchmark consists in 15 different perturbations, with 5 levels of severity each. it is observed that the regularizer is able to increase the relative performance under perturbations by a significant amount (using the mREI measure from [34] an improvement of 0.29 is noticed).

The robustness to adversarial inputs is evaluated, which are specifically built to fool the DNN. The network performance is evaluated by adding the scaled gradient sign (FGSM attack) to the input [30], so that a target SNR of 33 is obtained.The experiments show that the regularizer increases robustness against adversarial attacks as the regularizer improves the network's error rate from 66 % to 49 %.

The robustness of the architecture to noise on parameters and activations is evaluated. Two types of noises are considered: erasures of the memory (dropout) quantization of the weights [18].

Both experiments show that the regularizer is able to improve the robustness. In the case of the dropout experiment, the regularizer improves the error rate from 55 % to 34 % and in the case of quantization the error rate improves from 76 % to 50 %.

Peernets: Mixing Graph and Image Filters to Improve Robustness

In this subsection a second graph-based method which specifically addresses the problem of adversarial attacks [52] is presented. Small perturbations on an input image, often invisible for the human eye, can mislead a DNN and completely distort its prediction. The authors of [52] propose to modify the architecture of standard DNNs to make DNNs more robust to adversarial attacks. The intuition was that smoothing some fragments of the intermediate representations of the images with a linear combination of fragments from intermediate representations of other similar images would improve the robustness without damaging too much the accuracy. In this way, this should allow one to "filter away" (with a graph filter) the adversarial noise.

The authors of [52] propose to implement this concept as a new layer, called the *Peer Regularization layer*, to intertwine within standard layers of existing DNN architectures. They have been inspired from various domains including Deep learning on graphs, low-dimensional regularization and non-local image filtering. In the following, a detailed description of the Peer Regularization layer is given.

Consider N images (the peers) and their intermediate representations at the output of a given layer. The representation of an image is a $n \times d$ matrix, where n is the number of pixels and d is the number of features in each pixel.

The idea of the Peer Regularization layer is to leverage the structure of the feature space of the intermediate representations obtained after a conventional layer (e.g. after a CNN). Each pixel of each intermediate representation is replaced by a smoothed version of itself. The smoothed version is computed from the K most similar pixels among all the pixels belonging to the intermediate representations of all the other images (the peers). More precisely, for each pixel of each intermediate representation, a K nearest neighbor graph is computed. This means that the similarities (for instance, cosine similarities) are computed between the d-dimensional vector representing the pixel and all d-dimensional vectors representing pixels of all the peer images. Then, only the K pixels showing the biggest similarities are kept to compute the smoothed version of the initial pixel.

The d-dimensional vectors of the K pixels are combined using an attention mechanism. Let us denote the pth pixel of image i x_p^i. Its kth

neighbor is a pixel q_k from an image j_k written $x_{q_k}^{j_k}$. The smoothed version of the pixel $\tilde{x}_p^i$ is given by the equation:

$$\tilde{x}_p^i = \sum_{k=1}^{K} \alpha_{ij_kpq_k} \boldsymbol{x_{q_k}^{j_k}}, \alpha_{ij_kpq_k} = \frac{\text{LeakyReLU}(exp(a(x_p^i, x_{p_k}^{j_k})))}{\sum_{k'=1}^{K'} \text{LeakyReLU}(exp(a(x_p^i, x_{p_{k'}}^{j_{k'}})))} \quad (2.28)$$

The function a is a fully connected layer whose input is a $2d$-dimensional vector and outputs a scalar. The α represents the relative importance of each neighboring pixel.

In the original paper, the authors report experiments on CIFAR-10 with a Resnet-32 containing intertwined Peer Regularization layers (PR-Resnet-32). The test images are disrupted with three white-box adversarial attacks (gradient descent, fast-gradient sign method, projected gradient descent) of varying amplitude (measured by the variable ε). It was shown that PR-Resnet-32 is more robust under these adversarial attacks than the standard Resnet-32 (more details in the original paper [52]).

While this kind of methods is not perfect yet, it shows an interesting example of how graph methods can help to handle a Deep Learning problem. Here, the graph enables to take advantage of the relationships between data at any point within a DNN, and to use this knowledge to make it more robust to small perturbations. Note however, that it requires additional training time and arbitrary choices about the training images to keep during testing. More information on the implementation can be found in the original paper.

2.4. Conclusion

DNNs achieve state of the art performance in many machine learning tasks. Unfortunately, their practical uses in real-world applications are still limited due to some major limitations. In this chapter, we have presented an overview of graph-based tools that aim at solving some of these limitations. Namely, interpreting the intermediate representations of DNNs train models with few data learning meaningful embeddings for classification improving the robustness of DNNs compressing the architectures.

Approaching these limitations with graphs enables to view them from a new angle. In many methods, the graphs are used to model the intermediate spaces of the DNNs. In general, their vertices represent distinct data samples and their edges depend on the similarity of the intermediate representations of the data samples within the DNNs. Such graphs are called *latent geometry graphs*. Given the ubiquity of graphs in representing relations between data samples, it is quite natural to imagine that GSP, a branch of the signal processing field that handles data supported on graphs, is a valuable path to address these challenges.

This area of research is in its infancy. We hope that such an overview will inspire researchers to work in this line of research that we believe could help improving the applications of deep learning that we see in day-to-day life.

References

[1]. R. Anirudh, P. Bremer, R. Sridhar, J. J. Thiagarajan, Influential Sample Selection: A Graph Signal Processing Approach. Technical Report, *Lawrence Livermore National Lab*., Livermore, CA, United States, 2017.

[2]. R. Anirudh, J. J. Thiagarajan, R. Sridhar, T. Bremer, Margin: Uncovering deep neural networks using graph signal analysis, *arXiv Preprint*, 2017, arXiv:1711.05407.

[3]. R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, Netvlad: CNN architecture for weakly supervised place recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, 2016, pp. 5297-5307.

[4]. M. Bontonou, L. Béthune, V. Gripon, Predicting the accuracy of a few-shot classifier, *arXiv Preprint*, 2020, arXiv:2007.04238.

[5]. M. Bontonou, C. Lassance, G. B. Hacene, V. Gripon, J. Tang, A. Ortega, Introducing graph smoothness loss for training deep learning architectures, in *Proceedings of the IEEE Data Science Workshop (DSW'19)*, 2019, pp. 160-164.

[6]. S. Brahmbhatt, J. Gu, K. Kim, J. Hays, J. Kautz, Geometry-aware learning of maps for camera localization, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 2616-2625.

[7]. J. R. Burt, N. Torosdagli, N. Khosravan, H. RaviPrakash, A. Mortazi, F. Tissavirasingham, S. Hussein, U. Bagci, Deep learning beyond cats and dogs: Recent advances in diagnosing breast cancer with deep neural networks, *The British Journal of Radiology*, Vol. 91, Issue 1089, 2018, 20170545.

[8]. P. Chrabaszcz, I. Loshchilov, F. Hutter, A downsampled variant of imagenet as an alternative to the cifar datasets, *arXiv Preprint*, 2017, arXiv:1707.08819.
[9]. M. Defferrard, L. Martin, R. Pena, N. Perraudin, PYGSP: Graph signal processing in Python, Zenodo, October 2017, https://doi.org/10.5281/zenodo.1003158
[10]. I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, *arXiv Preprint*, 2014, arXiv:1412.6572.
[11]. V. Gripon, A. Ortega, B. Girault, An inside look at deep neural networks using graph signal processing, in *Proceedings of the Information Theory and Applications Workshop (ITA'18)*, February 2018, pp. 1-9.
[12]. V. Gupta, N. Sambyal, A. Sharma, P. Kumar, Restoration of artwork using deep neural networks, *Evolving Systems*, Vol. 12, 2021, pp. 439-446.
[13]. G. B. Hacene, Processing and learning deep neural networks on chip, PhD Thesis, *Ecole Nationale Supérieure Mines-Télécom Atlantique*, 2019.
[14]. D. K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, *Applied and Computational Harmonic Analysis*, Vol. 30, Issue 2, 2011, pp. 129-150.
[15]. D. Hendrycks, T. Dietterich, Benchmarking neural network robustness to common corruptions and perturbations, in *Proceedings of the International Conference on Learning Representations (ICLR'19)*, 2019.
[16]. A. Hermans, L. Beyer, B. Leibe, In defense of the triplet loss for person re-identification, *arXiv Preprint*, 2017, arXiv:1703.07737.
[17]. K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, 2016, pp. 770-778.
[18]. I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, *The Journal of Machine Learning Research*, Vol. 18, Issue 1, 2017, pp. 6869-6898.
[19]. Y. Hu, V. Gripon, S. Pateux, Exploiting unsupervised inputs for accurate few-shot classification, *arXiv Preprint*, 2020, arXiv:2001.09849.
[20]. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning (ICML'15)*, 2015, pp. 448-456.
[21]. A. Iscen, G. Tolias, Y. Avrithis, O. Chum, Label propagation for deep semi-supervised learning, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, 2019, pp. 5065-5074.
[22]. Y. Jiang, D. Krishnan, H. Mobahi, S. Bengio, Predicting the generalization gap in deep networks with margin distributions, https://openreview.net/forum?id=HJlQfnCqKX
[23]. Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, S. Bengio, Fantastic generalization measures and where to find them, in *Proceedings of the*

International Conference on Learning Representations, 2020, https://openreview.net/forum?id=SJgIPJBFvH

[24]. M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al., Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, Vol. 5, 2017, pp. 339-351.

[25]. A. Kendall, M. Grimes, R. Cipolla, Posenet: A convolutional network for real-time 6-dof camera relocalization, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV'15)*, 2015, pp. 2938-2946.

[26]. G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in *Proceedings of the ICML Deep Learning Workshop*, Vol. 2, Lille, 2015.

[27]. A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, N. Houlsby, Large scale learning of general visual representations for transfer, *arXiv Preprint*, 2019, abs/1912.11370.

[28]. A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf

[29]. A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[30]. A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial machine learning at scale, https://arxiv.org/abs/1611.01236

[31]. C. Lassance, V. Gripon, A. Ortega, Laplacian networks: Bounding indicator function smoothness for neural network robustness, *arXiv Preprint*, 2018, arXiv:1805.10133.

[32]. C. Lassance, Y. Latif, R. Garg, V. Gripon, I. Reid, Improved visual localization via graph smoothing, *arXiv Preprint*, 2019, arXiv:1911.02961.

[33]. C. Lassance, M. Bontonou, G. B. Hacene, V. Gripon, J. Tang, A. Ortega, Deep geometric knowledge distillation with graphs, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'20)*, 2020, pp. 8484-8488.

[34]. C. Lassance, V. Gripon, A. Ortega, Laplacian networks: Bounding indicator function smoothness for neural networks robustness, *APSIPA Transactions on Signal and Information Processing*, Vol. 10, 2021, E2.

[35]. C. Lassance, V. Gripon, J. Tang, A. Ortega, Structural robustness for deep learning architectures, in *Proceedings of the IEEE Data Science Workshop (DSW'19)*, June 2019, pp. 125-129.

[36]. S. Lee, B Song, Graph-based knowledge distillation by multi-head attention network, *arXiv Preprint*, 2019, arXiv:1907.02226.

[37]. Y. Liu, J. Cao, B. Li, C. Yuan, W. Hu, Y. Li, Y. Duan, Knowledge distillation via instance relationship graph, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, 2019, pp. 7096-7104.

[38]. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, https://openreview.net/forum?id=rJzIBfZAb

[39]. S. Mallat, Understanding deep convolutional networks, *Phil. Trans. R. Soc. A*, Vol. 374, Issue 2065, 2016, 20150203.

[40]. P. Mangla, M. Singh, A. Sinha, N. Kumari, V. N. Balasubramanian, B. Krishnamurthy, Charting the right manifold: Manifold mixup for few-shot learning, http://arxiv.org/abs/1907.12087

[41]. P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, V. N. Balasubramanian, Charting the right manifold: Manifold mixup for few-shot learning, in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV'20)*, 2020, pp. 2218-2227.

[42]. J. Ma, C. Zhou, P. Cui, H. Yang, W. Zhu, Learning disentangled representations for recommendation, in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'19)*, 2019.

[43]. T. Milbich, K. Roth, H. Bharadhwaj, S. Sinha, Y. Bengio, B. Ommer, J. P. Cohen, Diva: Diverse visual feature aggregation fordeep metric learning, *arXiv Preprint*, 2020, arXiv:2004.13458.

[44]. A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, Vol. 106, Issue 5, 2018, pp. 808-828.

[45]. W. Park, D. Kim, Y. Lu, M. Cho, Relational knowledge distillation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*, 2019, pp. 3967-3976.

[46]. F. Radenovi, G. Tolias, O. Chum, Fine-tuning cnn image retrieval with no human annotation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 41, Issue 7, July 2019, pp. 1655-1668.

[47]. M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* San Francisco, CA, USA, August 13-17, 2016, pp. 1135-1144.

[48]. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol. 115, Issue 3, 2015, pp. 211-252.

[49]. A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs: Frequency analysis, *IEEE Transactions on Signal Processing*, Vol. 62, Issue 12, 2014, pp. 3042-3054.

[50]. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, *IEEE Signal Processing Magazine*, Vol. 30, Issue 3, 2013, pp. 83-98.

[51]. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, *Nature*, Vol. 550, 2017, pp. 354-359.

[52]. J. Svoboda, J. Masci, F. Monti, M. Bronstein, L. Guibas. PeerNets: Exploiting peer wisdom against adversarial attacks, in *Proceedings of the International Conference on Learning Representations (ICLR'19)*, 2019.

[53]. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, http://arxiv.org/abs/1312.6199

[54]. C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'18)*, 2018, pp. 270-279.

[55]. B. J. Taylor, Methods and Procedures for the Verification and Validation of Artificial Neural Networks, *Springer Science & Business Media*, 2006.

[56]. N. Tremblay, P. Gonçalves, P. Borgnat, Design of graph filters and filterbanks, in Cooperative and Graph Signal Processing, *Elsevier*, 2018, pp. 299-324.

[57]. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in *Proceedings of the Conference on Neural Information Processing Systems (NIPS'17)*, 2017.

[58]. C. Wang, B. Samari, V. G. Kim, S. Chaudhuri, K. Siddiqi, Affinity graph supervision for visual recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*, 2020, pp. 8247-8255.

[59]. F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in *Proceedings of the International Conference on Machine Learning*, 2019.

[60]. W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, A. Stolcke, The microsoft 2017 conversational speech recognition system, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'18)*, 2018, pp. 5934-5938.

[61]. X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, N. Houlsby, A large-scale study of representation learning with the visual task adaptation benchmark, in *Proceedings of the International Conference on Learning Representations (ICLR'19)*, 2019.

[62]. Z.-Q. Zhao, P. Zheng, S.-t. Xu, X. Wu. Object detection with deep learning: A review, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 300, Issue 11, 2019, pp. 3212-3232.

[63]. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in Advances in Neural Information Processing Systems, *MIT Press*, 2003, pp. 321-328.

Chapter 3

Machine Learning in Peer-to-peer Lending – The New Theme Park of Financial Risk Modelling

David Jesenko, Timotej Jagrič, Borut Žalik and Vita Jagrič

3.1. Introduction

The Peer-to-Peer (P2P) lending market has become a kind of a new form of private lending, which is an important market in the Internet financial industry. However, because of the complex correlations between independent factors, the comprehensive approach to predictive analytics of a credit risk for potential investors is a challenging task. Among many changes in the consumer credit market, there are computer-based lending platforms, called Peer-to-Peer lending (P2P lending), where borrowers demand a loan and investors lend money. Lenders select the borrowers according to their Probability of Default (PD) and investors' risk appetite. If the P2P market is efficient, the investors would not be able to gain abnormal returns since all available information is already included in the interest rate [1].

The lending platforms have experienced an enormous and fast growth; however, the total volume for now remains at a low level when compared to total consumer lending activities, according to the data for the UK market by Zhang et al. [2]. The sudden popularity among investors is reflected in the media, and academic research among several disciplines [3]. Various studies have explored the characteristics of lending platforms and their portfolios [4-10].

David Jesenko
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Mainly, the lending platforms do not act as creditors themselves, but bring creditors together with the borrowers. The main benefit for the borrowers is a lower interest rate as compared to a bank loan, while the main benefit for investors would be a higher return on their investments. This alternative is attractive to investors, especially in times of historically low interest rates. Among investors there are also institutional investors, like Hedge Funds [3].

Also, non-professional investors are active in lending platforms in contrast to the banking lending market. They have to be fully aware of the credit risk level they are exposed to [11]. Yum et al. [3] report on the insufficient capacity of an individual lender to evaluate borrowers' information in order to estimate the borrowers' trustworthiness. Several studies also indicate herding behavior [3]. In order that investors are able to estimate the riskiness of their investment, lending platforms already disclose a high level of information on historical loan defaults and provide investors with their estimations on future performance [12-14]. The future growth of the sector depends on the future improvement in its services to investors in order to manage the credit risk better. As reported in [12, 15], some platforms already provide Loan Loss Reserve Funds to offer investors compensation for defaulted loans. The same authors estimate that still insufficiently resolved risk issues in the P2P industry are: Quantifying the risks of loss in a business downturn, operational processes in relation to recovery of lost loans, the risks of platform failure (bankruptcy, operational failure), risk of falling P2P loan prices resulting from a self-reinforcing withdrawal of funds by institutional investors, many operational risks (e.g. fraud, cybercrime, and operational outages).

The credit risk of consumer lending by banks has been studied intensively for decades (a review was made by Lessmann, et al. [16]). However, literature's findings are founded mainly on bank-consumer lending portfolios. The credit risk characteristics of a new business model of lending platforms, might, however, differ significantly from the banks' ones. While banks manage credit risk in a complex risk management process, which is the subject of detailed regulation, the P2P industry suffered from its shortfall. Lending platforms' portfolios had very high default rates in the early stages (as reported by Klafft [17] for 2008). To manage investors' credit risk, researchers started to develop credit scoring for P2P lending platforms [1]. Among promising ones, the hybrid model (integration of an iterative computation model and a logistic classification model) is presented by Zeng et al. [18].

In this study, we propose an enhancement of the credit risk management for the creditor in the P2P lending portfolio when compared to a commonly used classification model, which has long been the logistic regression model. Findings of studies using different methodologies in credit scoring in bank portfolios were studied carefully [16, 19]. However, also other methods were considered and alternative classification algorithms in the Data Mining field. As stated by Beque and Lessmann [19], the choice of a method for credit scoring is usually based on three criteria: The comprehensibility, resource efficiency and predictive accuracy. According to the methodologies' characteristics, an evolutionary algorithm may be a promising way for the research goal of improving the debtors' default rate forecasting when taking into account the P2P lending portfolio features. The obtained results were compared to those obtained from the logit model [20].

3.2. Methodology

3.2.1. Data

The database from the lending platform Bondora in Estonia was used in this study. It includes 11678 consumer loans originated in the years from 2009 to 2015. The model was built on the reduced database, where the reduction followed several steps. First, short-term loans were excluded with a maturity less than 1 year. Next, outliers, loans with missing data and undefined gender data were also excluded. The final database set-up included 1823 loans. The number of used explanatory variables is equal to 13, which is the same reduced number, as in the original database. Variables that were not available for the whole period, were removed. The loans are categorised as defaulted according to the default definition used by the Bondora platform, where a loan is labelled as defaulted when it is 74 Days Past Due (DPD). In order to avoid the so-called dummy variable trap [21], we added an explanatory variable with a constant value equal to one, labelled as CONST. The regression coefficient associated with this variable is regression constant. Statistics of variables are presented in Table 3.1.

The average age of borrowers is 36 years, while the youngest borrower is 18 years old and the oldest one is 77 years old. There are approximately the same number of men and women borrowers. JB statistics show that none of the variables follow the normal distribution. This is, however, a

typical characteristic of financial data and limits the chances of using classical econometric methods.

Table 3.1. Statistical analysis of variables.

Variable	Label	Avg.	Median	Max.	Min.	Std. Dev.	JB	p
Age	AGE	35.88	33.00	77.00	18.00	11.20	178.57	0.00
Country	COUNTRY	1.03	1.00	3.00	1.00	0.21	298348.20	0.00
Status of employment	DZ	3.09	3.00	6.00	1.00	0.75	2653.62	0.00
Education	DE	3.91	4.00	5.00	1.00	0.92	272.90	0.00
Sex	GENDER	0.50	1.00	1.00	0.00	0.50	303.83	0.00
Residence status	DB	1.94	1.00	9.00	-1.00	3.08	248.10	0.00
Interest rate	INTEREST	0.27	0.28	0.49	0.03	0.07	55.31	0.00
Legal status	DM	2.15	2.00	5.00	1.00	0.98	99.21	0.00
Number of previous loans	DPP	1.53	0.00	22.00	0.00	3.05	13920.00	0.00
Number of dependent members	DVZC	0.88	1.00	6.00	0.00	107.00	587.55	0.00
Work area	DP	7.78	7.00	19.00	1.00	5.72	135.25	0.00
The purpose of the loan	DUOL	3.60	3.00	8.00	0.00	2.93	201.07	0.00
Work experience	DIZ	3.62	4.00	6.00	1.00	1.56	91.38	0.00

3.2.2. Definition of Prediction Model

This section describes a new approach for predicting the default rate of P2P lending from the credit risk perspective. Given a vector of explanatory variables $V[n_i]$ associated with each loan n_i and the state of credit risk $C[n_i]$, the objective is to find mapping function $R: V \rightarrow \mathbb{R}$ that gives an accurate estimation of the credit risk, and threshold th. The threshold defines whether the considered loan is risky or not. A definition of all credit risks L^R can then be given as

$$L^R = \{V[n_i] | R(V[n_i]) \geq th\}$$

To permit straightforward interpretation, R has the following polynomial form

$$R(V)[n_i] = \sum_j w_j * V[n_i]_j,$$

where w_j, ($\sum_j |w_j| = 1$), are are coefficients (weights) determining the degree of impact that the related explanatory variable $V[n_i]_j$ has on the credit risk.

An Evolutionary Algorithm (EA) [22, 23] is proposed in order to obtain suitable w_j. EA finds a definition of R during the following steps (see Fig. 3.1):

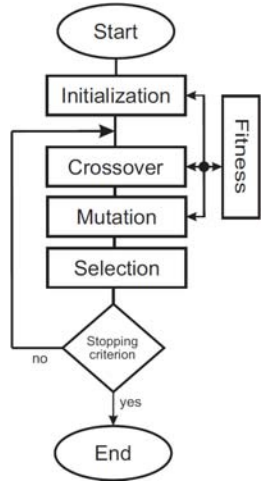

Fig. 3.1. The used EA described by the flowchart.

Initial population was generated was generated at first by defining a set of individuals $P = \{r_s\}$ randomly. Each individual r_s represents an estimation of the searched mapping function R, and is defined by a series of randomly generated coefficients w_j. Each w_j is assigned as a chromosome j from individual r_s. A graphic representation of an individual is shown in Fig. 3.2.

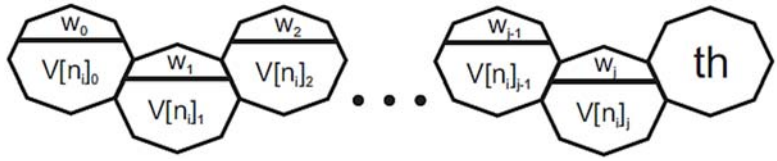

Fig. 3.2. Graphic representation of an individual.

Evaluation of individuals is performed by measuring the efficiency of the forecast made by each $r_s \in P$. This, so-called individual's fitness, is defined by an Area Under the Curve (AUC) of r_s. The AUC of an individual is equal to the probability that the individual ranks a randomly chosen positive example higher than a randomly chosen negative example [24].

Stopping criterion terminates the algorithm when the maximal number of function evaluations is achieved (in our case, when 1000 individuals were evaluated), or when maximal accuracy was reached (i.e. when AUC = 1). If the stopping criterion is not satisfied, the algorithm repeats the basic steps of the EA (crossover, selection, and mutation).

Crossover is used to generate new individuals and to develop the population by combining the fittest chromosomes from the previous generation. The first parent was chosen from the population in sequence, while the second one was selected from the rest of the population at random. A new individual was generated by copying chromosomes randomly from the first or the second parent, as shown in Fig. 3.3.

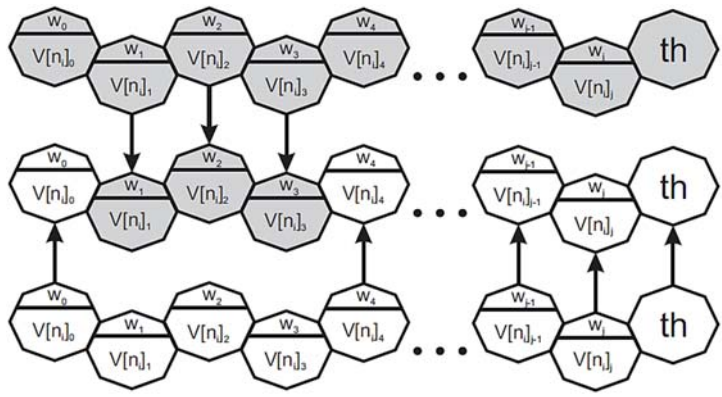

Fig. 3.3. Two-point crossover of individuals.

Selection of the individuals is very important for developing the population towards the optimal solution. The tournament selection of size two was applied in our case. To ensure that each individual r_s is used in the process of selection at least once, the first competitor is selected in a consecutive order, while its challenger is selected randomly from the population. The selected individual with highest AUC continues into the

next generation. That kind of an approach secures the fittest individual (i.e. r_s with the highest AUC) and ensures that the overall population fitness never declines.

Mutation is performed over the population in order to allow a new genetic material to enter and, thus, keeping the population diverse. This was achieved by modifying selected chromosomes w_j randomly. A mutated coefficient w_j is defined by $w_j = w_j + rand()$, where *rand()* returns a value from the range [-0.1, 0.1]. The same as the coefficients, threshold *th* is also changed with *rand()*, such as $th = th+rand()$. After mutation, newly generated individuals were evaluated and the stopping criterion was checked.

The result of the proposed EA is function *R(V)*, capable of predicting the value of a credit risk with the minimal cumulative error on training data.

3.2.3. Validation Procedure

In order to validate the introduced methodology, the prediction accuracies achieved when determining the value of the credit risk were checked. For this intention, the prediction function was learned and used for predicting the value of the credit risk. The prediction errors were assessed in terms of AUC. The procedure used for the validation was a cross-validation. The cross-validation is an approach for estimating how the results of a statistical analysis behaves on an independent data set [25]. It is used primarily in machine learning. In a *k*-fold cross-validation, sample data are divided into *k* complementary subsets, where a single subset is kept as the validation data set (also called a testing set), while the remaining $k - 1$ subsets are used for the learning (called a learning set). The cross-validation process is repeated *k* times, with each of the *k* subsets used exactly once as the testing data set. The *k* results from the folds are then averaged to produce a single estimation. 10-fold cross-validation was used in our study [26].

3.3. Results

3.3.1. Parameter Sensitivity

The proposed evolutionary algorithm for prediction relies on five parameters. These parameters are the size of the population, the probability of crossover and mutation, the coefficients' change range,

and the number of individual evaluations. In order to examine their influences on the method performance and estimate their optimal values, a systematic sensitivity analysis was performed, as proposed by Chang and Delleur [27, 28]. The well-known dataset Wine Quality [29] in machine learning was used for this purpose. The ranges of the input parameter values were first defined during the sensitivity analysis. The size of the population was from the range [10, 100], while the coefficients' change range was on the interval [-1, 1]. The probability of mutation and of crossover were from the range [0.1, 1], and, finally, the number of individual evaluations from the interval [200, 5000]. Taking these restrictions into account, 5000 sets of parameter values were generated randomly, and the proposed evolutionary algorithm was evaluated accordingly. The obtained results were classified as acceptable if the run finished with a score of correctly predicted samples of more than 80 %; otherwise the results were regarded as unacceptable. The cumulative frequencies were plotted of acceptable and unacceptable results. Insensitive parameters are characterised by similar cumulative frequencies of acceptable and unacceptable results throughout the whole range of parameter values. The following parameters were classified as insensitive: The probability of crossover, the probability of mutation, the coefficients' change range and the size of population. On the other hand, the number of individual evaluations was recognised as sensitive. In that case, a significantly larger number (approximately 18.7 %) of acceptable results was achieved when its value was equal to 1000 than in any other case. While influences of the sensitive parameters on the method's performance could be recognised straightforwardly, the optimal values of the insensitive parameters have to be tuned accordingly. In our case, a model-based approach with the linear regression was applied, as proposed in [30]. The selected optimal values for the evolutionary algorithm's parameters can be seen in Table 3.2.

3.3.2. Number of Terms in Function

The proposed method was implemented in C++ under the Microsoft® Windows 10 operating system and tested on an Intel® Core™ i5 CPU with 32 GB of main memory. Before the actual validation, the experiment was done using different numbers of terms in the learned prediction model (function). The number of terms was selected in the range from 10 to 30 terms. The obtained results can be seen in Fig. 3.4. The AUC grew till 20 terms in the learned function, and achieved the peak at the same number. From this point on, the curve stagnates and there is no improvement with more terms in the function. This could

mean two things, first, that the optimization of a higher number of terms is tougher, and, secondly, more likely, 20 terms in a function involve enough information to classify loans successfully. In the continuation, all results are obtained with the number of terms in a function equal to 20.

Table 3.2. Parameters and their optimal values.

Parameter	**Value**
Size of population	60
Number of individual evaluations	1000
Probability of crossover	50 %
Probability of mutation	50 %
Coefficients' change range	[-0.1, 0.1]

3.3.3. Validation of Prediction Model

1000 independent runs of algorithm were made in order to validate proposed EA. The difference between the least and most accurate runs was 2.36 %. In Table 3.3, the prediction accuracy is presented of the most accurate run. The accuracy in that case is 82.78 %. Also, the difference between prediction accuracy for good and bad borrowers is small, and it is equal to 0.17 %. This shows the reliability of predicting both classes with the same precision, which, in our case, is very important. In the continuation, the equation acquired for the most accurate run in Table 3.3 is presented and variables are explained with belonging coefficients.

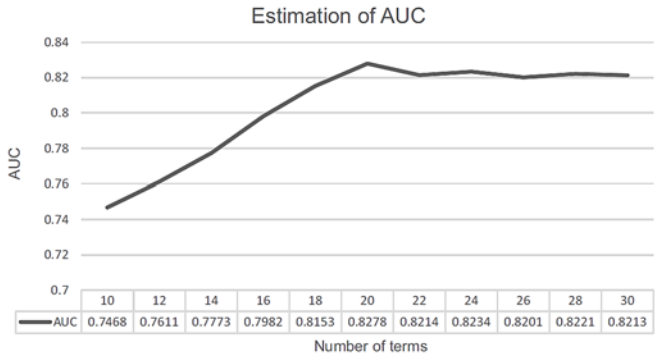

Fig. 3.4. Estimation of AUC at different numbers of terms in the learned prediction model.

Table 3.3. Assessment of prediction accuracies achieved by the proposed method.

	Dep = 0	**Dep = 1**	**Total**
P(Dep = 1) ≤ C	1213	61	1274
P(Dep = 1) > C	253	296	549
Total	1466	357	1823
Correct	1213	296	1509
Correct [%]	82.74	82.91	82.78
Incorrect [%]	17.26	17.09	17.22

3.3.4. Comparison with Machine Learning Algorithms

In order to demonstrate the value of the proposed method, a comparison was performed with several well-known machine learning algorithms (decision tree [31, 32], random forest [33, 34], support vector machine [35, 36], and neural network [37, 38]). For this purpose, OpenCV [39] implementations were used of common classification algorithms. Sensitivity analysis as described in Section 3.3.1 has been performed for all tested machine learning algorithms in order to tune their parameters. In Table 3.4 the acquired results for all the previously mentioned machine learning algorithms can be seen.

From the Table 3.4 can be seen, that not all the standard machine learning algorithms performed well. The decision tree and the support vector machine had a lot of troubles, the reason is probably the high number of different variables. Random forest and neural network performed much better and acquired AUC higher than 70 %.

Table 3.4. Machine learning algorithms and their acquired accuracies.

Machine learning algorithm	**AUC [%]**
Decision tree	48.23
Random forest	71.42
Support vector machine	23.78
Neural network	70.93

3.3.5. Interpretation of the Prediction Model

The best values in AUC was achieved by the model with 20 terms, as seen from Fig. 3.4. This model was, therefore, taken into further

examination as the most appropriate. In Table 3.3 we can see that 82.74 % of goods (good borrowers) and 82.91 % of the bads (bad borrowers) were also classified correctly.

The most accurate run resulted in the prediction model presented in Fig. 3.5. The lowest coefficient (-0.157) has been given to the attribute DB8 coming from the variable residence status, attribute with mortgage. The attribute INTEREST4 has the highest positive value of the coefficient (0.083), which comes from the variable interest rates. Other variables' attributes that have been selected into the model, which turned out to be the most accurate, were from variables: Residence status (DB), age, work area (DP), status of employment (DZ), work experience (DIZ), gender, country and number of dependent members in the household (DVZC).

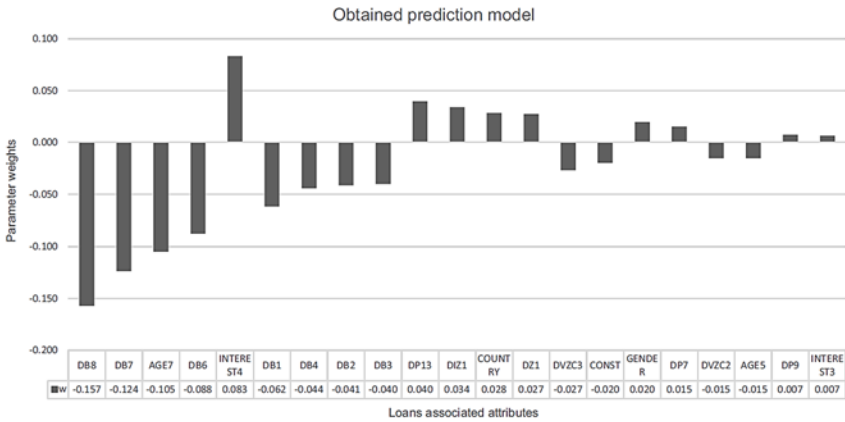

Fig. 3.5. Coefficients and loan attributes associated with the obtained prediction model.

The used methodology enables us to start the learning with a broader range of variables. At the end, however, the number of variables (and attributes) can be significantly lower. The final model of this study includes some of the variables with multiple attributes (e.g. residence status), some of the used variables were not included at all, and, in some cases, only some attributes of a certain variable were included. This way of modelling appears as a major advantage of the presented methodology in credit risk modelling. As it turns out, in credit risk management,

irrespective of the chosen methodology, models have to be renewed frequently due to the rapidly changing environment and, consequently, loan portfolio characteristics. The remodelling process should be started with all available variables each time, and not only with those variables that were included into the previous model. As the number of all available variables is huge and new ones are made available, the methodology should not be time consuming. The information hidden in any of the available variables would not be lost in this way. This advantage is offered by our methodology, as shown in this study.

3.4. Discussion

We compared the obtained results to those of the benchmark model presented in [20]. It was built on the same database and uses logistic regression. The logit model obtained an AUC of 0.76. Since no data on costs for model misclassification are known, the optimal cut-off value for the logit model was set according to the balanced classification accuracy, and was 0.2. At this cut-off value, the share of misclassification of good and bad borrowers is very close, and equals the overall accuracy. The percentage of correctly classified good borrowers is 74.97 %, for the bad borrowers it is 75.91 %, and the overall accuracy is 75.15 %. The results from this study were obtained with machine learning, and are presented in Table 3.3. The best model classified 82.74 % of good and 82.91 % of bad borrowers correctly. The overall accuracy was 82.78 %. The latter model clearly outperforms the classical logit-model. On this particular data set of P2P loans it turned out that credit risk estimation could be more precise with machine learning than with logistic regression.

When examining the variables of the model, the comparability to the credit risk models of the banks' portfolios can be found. Typically, banking retail credit models use data from application forms (e.g. interest rate), customer demographics, and transactional data from the customer history [16]. The variables used in this study have the same structure and come from the same groups of variables. The variables used with this study, like age, interest rate, residence status, employment status, number of dependent members in a household, are the same as those used in common modelling of banking portfolios. The credit risk of banks' portfolios seems to have similar risk factors when compared to the credit risk factors of the P2P lender portfolio. The knowledge on banks' retail portfolio risk management can be used to manage P2P portfolios. The

first years in P2P lending started with high default and low recovery rates, indicating the need for standard well-known banking style risk management. Since P2P platforms are subject to less regulation when compared to banks, they can design their risk management processes with more flexibility, use the best risk management practices in favourable and effective ways. Due to flexibility, P2P platforms allow faster adaptation to market changes. The difference between banks' and P2P portfolios is, therefore, not that big. The P2P lending business apparently substitutes the banking consumer credit business, and is expected to do so in future.

Based on the good results of this study, another benefit of the Evolutionary Algorithm can be found. If the model is integrated into a concrete P2P platform, or made available to investors in another way, they can provide a major benefit to small individual investors. In this way, frontiers between large professional investors (e.g. banks) and small individual investors would be reduced. Banks have a major advantage due to concentrated knowledge in the field of Credit (and other) Risk Management, while small investors do not.

Acknowledgements

This research was joint funded by Slovenian Research Agency and Slovenian Ministry of the Interior, grant number V2-2260. This research was also funded by Slovenian Research Agency, grant number P2-0041.

References

[1]. C. Serrano-Cinca, B. Gutierrez-Nieto, The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending, *Decision Support Systems*, Vol. 89, Issue 1, 2016, pp. 113-122.

[2]. B. Zhang, P. Baeck, T. Ziegler, J. Bone, K. Garvey, Pushing boundaries: The 2015 UK alternative finance industry report, *Cambridge Centre for Alternative Finance*, Vol. 1, Issue 1, 2016, pp. 1-56.

[3]. H. Yum, B. Lee, M. Chae, From the wisdom of crowds to my own judgment in microfinance through online peer-to-peer lending platforms, *Electronic Commerce Research and Applications*, Vol. 11, Issue 5, 2012, pp. 469-483.

[4]. J. Zhang, P. Liu, Rational herding in microloan markets, *Management Science*, Vol. 58, Issue 5, 2012, pp. 892-912.

[5]. D. G. Pope, J. R. Sydnor, What's in a picture? Evidence of discrimination from prosper.com, *Journal of Human Resources*, Vol. 46, Issue 1, 2011, pp. 53-92.
[6]. J. Duarte, S. Siegel, L. Young, Trust and credit: The role of appearance in peer-to-peer lending, *The Review of Financial Studies*, Vol. 25, Issue 8, 2012, pp. 2455-2484.
[7]. A. Agrawal, C. Catalini, A. Goldfarb, Some simple economics of crowd funding, *Innovation Policy and the Economy*, Vol. 14, Issue 1, 2014, pp. 63-97.
[8]. R. Emekter, Y. Tu, B. Jirasakuldech, M. Lu, Evaluating credit risk and loan performance in online peer-to-peer (P2P) lending, *Applied Economics*, Vol. 47, Issue 1, 2015, pp. 54-70.
[9]. L. Gonzalez, Y. K. Loureiro, When can a photo increase credit? The impact of lender and borrower profiles on online peer-to-peer loans, *Journal of Behavioral and Experimental Finance*, Vol. 2, Issue 1, 2014, pp. 44-58.
[10]. Y. Li, A. Hao, X. Zhang, X. Xiong, Network topology and systemic risk in peer-to-peer lending market, *Physica A: Statistical Mechanics and Its Applications*, Vol. 508, Issue 1, 2018, pp. 118-130.
[11]. L. Tang, F. Cai, Y. Ouyang, Applying a nonparametric random forest algorithm to assess the credit risk of the energy industry in China, *Technological Forecasting and Social Change*, Vol. 144, Issue 1, 2019, pp. 563-572.
[12]. A. Milne, P. Parboteeah, The business models and economics of peer-to-peer lending, *ECRI Research Report*, Vol. 16, Issue 1, 2016, pp. 1-37.
[13]. P. Giudici, B. Hadji-Misheva, A. Spelta, Network based credit risk models, *Quality Engineering*, Vol. 32, Issue 2, 2020, pp. 199-211.
[14]. S. Pang, X. Hou, L. Xia, Borrowers' credit quality scoring model and applications, with default discriminant analysis based on the extreme learning machine, *Technological Forecasting and Social Change*, Vol. 165, Issue 1, 2021, 120462.
[15]. P. Damayanty, E. Murwaningsari, The role analysis of accrual management on loss-loan provision factor and fair value accounting to earnings volatility, *Research Journal of Finance and Accounting*, Vol. 11, Issue 2, 2020, pp. 155-162.
[16]. S. Lessmann, B. Baesens, H. V. Seow, L. C. Thomas, Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research, *European Journal of Operational Research*, Vol. 247, Issue 1, 2015, pp. 124-136.
[17]. M. Klafft, Online peer-to-peer lending: A lenders' perspective, in *Proceedings of the International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government (EEE'08)*, Vol. 1, Issue 1, 2008, pp. 371-375.

[18]. X. Zeng, L. Liu, S. Leung, J. Du, X. Wang, T. Li, A decision support model for investment on P2P lending platform, *PloS ONE*, Vol. 12, Issue 9, 2017, e0184242.
[19]. A. Beque, S. Lessmann, Extreme learning machines for credit scoring: An empirical evaluation, *Expert Systems with Applications*, Vol. 86, Issue 1, 2017, pp. 42-53.
[20]. T. Jagrič, A. Blagotinšek, V. Jagrič, Modelling probability of default for P2P lending – Would such portfolio be comparable to bank consumer loans?, *Bančni Vestnik: Revija za Denarništvo in Bančništvo*, Vol. 12, Issue 1, 2017, pp. 29-36 (in Slovene).
[21]. D. Gujarati, Use of dummy variables in testing for equality between sets of coefficients in linear regressions: A generalization, *The American Statistician*, Vol. 24, Issue 5, 1970, pp. 18-22.
[22]. D. Mongus, U. Vilhar, M. Skudnik, B. Žalik, D. Jesenko, Predictive analytics of tree growth based on complex networks of tree competition, *Forest Ecology and Management*, Vol. 425, Issue 1, 2018, pp. 164-176.
[23]. U. Lešnik, D. Mongus, D. Jesenko, Predictive analytics of PM10 concentration levels using detailed traffic data, *Transportation Research Part D: Transport and Environment*, Vol. 67, Issue 1, 2019, pp. 131-141.
[24]. J. A. Hanley, B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, *Radiology*, Vol. 143, Issue 1, 1982, pp. 29-36.
[25]. B. Efron, R. Tibshirani, Improvements on cross-validation: the 632+ bootstrap method, *Journal of the American Statistical Association*, Vol. 92, Issue 438, 1997, pp. 548-560.
[26]. R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, Canada, 1995, pp. 1137-1145.
[27]. F. J. Chang, J. W. Delleur, Systematic parameter estimation of watershed acidification model, *Hydrological Processes*, Vol. 6, Issue 1, 1992, pp. 29-44.
[28]. D. Jesenko, M. Mernik, B. Žalik, D. Mongus, Two-level evolutionary algorithm for discovering relations between nodes' features in a complex network, *Applied Soft Computing*, Vol. 56, Issue 1, 2017, pp. 82-93.
[29]. P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, *Decision Support Systems*, Vol. 47, Issue 4, 2009, pp. 547-553.
[30]. A. Czarn, C. MacNish, K. Vijayan, B. Turlach, R. Gupta, Statistical exploratory analysis of genetic algorithms, IEEE Transactions on Evolutionary Computation, Vol. 8, Issue 4, 2004, pp. 405-421.
[31]. A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, A. Nathaniel, S. D. Brown, An introduction to decision tree modeling, *Journal of Chemometrics: A Journal of the Chemometrics Society*, Vol. 18, Issue 6, 2004, pp. 275-285.

[32]. S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, Issue 3, 1991, pp. 660-674.
[33]. M. Belgiu, L. Dragut, Random forest in remote sensing: A review of applications and future directions, *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 114, Issue 1, 2016, pp. 24-31.
[34]. G. Biau, E. Scornet, A random forest guided tour, *Test*, Vol. 25, Issue 2, 2016, pp. 197-227.
[35]. A. Widodo, B. S. Yang, Support vector machine in machine condition monitoring and fault diagnosis, *Mechanical Systems and Signal Processing*, Vol. 21, Issue 6, 2007, pp. 2560-2574.
[36]. W. S. Noble, What is a support vector machine?, *Nature Biotechnology*, Vol. 24, Issue 12, 2006, pp. 1565-1567.
[37]. S. C. Wang, Artificial neural network, *Interdisciplinary Computing in Java Programming*, Vol. 1, Issue 1, 2003, pp. 81-100.
[38]. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: A survey, *Heliyon*, Vol. 4, Issue 11, 2018, e00938.
[39]. G. Bradski, The OpenCV library, *Dr. Dobb's J. Software Tools*, Vol. 25, Issue 11, 2000, pp. 120-126.

Chapter 4

Early Detection of Mild Alzheimer's Diseases with Combine MRI and EEG Signals

Elias Mazrooei Rad and Mahdi Azarnoosh

4.1. State the Problem

Alzheimer's disease is a progressive disease of the mental faculties commonly seen in the elderly. Significant symptoms of this disease are memory loss, judgment and important behavioral changes in the person. The disease results in the loss of synapses of neurons in some areas of the brain, the necrosis of brain cells in different areas of the nervous system, the formation of spherical protein structures called aging plaques outside neurons in some areas of the brain, and fibrous protein structures called coils. The spiral is characterized in the cell body of neurons [1]. There is currently no definitive diagnosis or treatment for this disease. The prevalence of Alzheimer's disease is increasing rapidly. The costs of treatment as well as care and nursing of these patients are very high and difficult [2]. The main purpose and motivation of this research was to design and present a system for diagnosing mild Alzheimer's disease. In recent methods for diagnosing this disease, the inability to extract the appropriate features in the use of brain signals or medical images, has led to increased results of combined methods. With the help of medical image processing, severe Alzheimer's disease is determined and the similarity of the characteristics of brain signals with medical images is determined. Then, by presenting the characteristics of effective brain signals, the mild Alzheimer's group is determined. The combination of

Elias Mazrooei Rad
Biomedical Engineering Department, Khavaran Institute of Higher Education, Mashhad, Iran

brain signal recording methods and medical imaging increases the accuracy of the results in distinguishing patients from healthy individuals. According to the relationship between this disease and various features in the brain signal and medical images, the level of this disease is diagnosed.

4.2. Necessity of Doing Research

If this disease is not identified in time, new and up-to-date treatment methods will not work. The solution is to accurately identify the mechanism of this disease and its effect on brain signals, which is a very difficult task due to the dynamic nature of brain signals and the complex nature of this disease. As a result, we must determine the best and most effective indicator to identify this disease in a mild state and how this indicator relates to the characteristics of brain signals and images. Also, extracting appropriate characteristics and deciding on classification in this field are among the issues to be considered. There are several methods available to identify this disease, and it is important to examine two issues in these methods. The high costs of these methods along with the acceptable inaccuracy are the points under consideration, so it seems necessary to identify a low-cost method with appropriate accuracy and precision.

4.3. Methods of Diagnosing Alzheimer's disease

A) Memory and Psychology Test: Some of the symptoms of Alzheimer's disease such as forgetfulness and lack of concentration and lack of motivation are also seen in depression. Psychology test can evaluate reasoning, writing ability, judgment, etc. and determine whether the patient has is depressed or has Alzheimer's [3].

B) Blood tests: Some endocrine and internal diseases cause conditions similar to Alzheimer's disease, such as hypothyroidism, infections (AIDS, syphilis), so in a patient with memory problems, thyroid function tests, AIDS virus, etc. should be requested. [4].

C) Brain CT and MRI: Since the symptoms of some brain space lesions and brain structure defects such as tumors are similar to Alzheimer's disease, the treatable causes of dementia should be ruled out by performing computed tomography and magnetic resonance imaging.

Long-term use of sedatives and hypnotics can cause drug interactions and impair memory. Alcohol consumption also causes dementia or dementia [5].

Medical diagnosis: There are currently no tests that can diagnose Alzheimer's disease alone. Diagnosis of Alzheimer's disease is made by measuring and evaluating memory, and eliminating other possible causes of dementia [6].

4.4. EEG Signal Preprocessing

In order to properly process the signal, the first step is to properly process the EEG signal in order to properly filter the signal from interfering factors and all kinds of disturbances and noise. For this reason, 5 steps have been considered for this purpose: elimination of deviation from the baseline, elimination of high and low frequency artifacts, elimination of electrical noise, reduction of sampling rate and segmentation [7]:

- Remove deviations from the baseline;
- Remove high and low frequency artifacts;
- Eliminate city electricity noise;
- Reduce sampling rates;
- Segmentation.

It is clear that by removing redundant information from the EEG signal, it improves the quality and accuracy of the work in subsequent processes. Eliminating rapid signal changes plus base line fluctuations greatly increases processing quality. The recorded EEG signal has a set of artifacts and noises. Although the variety of artifacts in the EEG signal is large, it can be divided into two main parts, namely, high-frequency artifacts due to EMG of the head and neck muscles and low-frequency artifacts due to electrode movement and transpiration. To eliminate these artifacts and the noise of the city's electricity, a transient filter with cut-off frequencies of 0.5 to 45 Hz has been used. Reducing the sampling rate is optional, and the reason for this is that low-grade filters are usually used in signal recording equipment instead of precision anti-aliasing filters, which are large and expensive, but instead reduce the signal sampling rate. Select multiple times the cut-off frequency of the filter. Signal segmentation is done in two ways. One method is to divide the

signal into pieces of the same length so that the signal can be assumed to be static along it, and another method is to use adaptive segmentation methods so that the signal pieces are divided into pieces of unknown length based on a static criterion. Although the accuracy of the second method is higher, but due to the ease of working with the same parts, the first method has been used. The length of the pieces is 2 seconds. The length of the piece is the stationary of the signal in that part, and what is certain is that the brain signal is not static in general and can only be assumed to be static in certain conditions. The maximum length of the stationary part varies depending on the application and type of EEG signal, in other words, the condition of the brain along with the type of processing and subsequent analysis are effective in static truth. Fig. 4.1 shows the three brain channels after the necessary preprocessing.

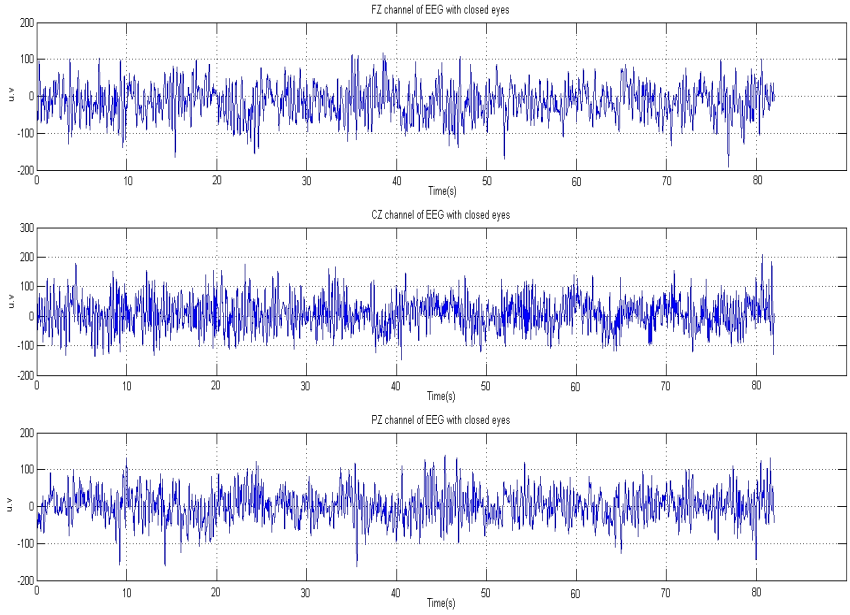

Fig. 4.1. Display of three channels after preprocessing (Application of filter – removal of deviation from the baseline – reduction of sampling rate).

4.5. Power Spectrum

Quantitative analysis of EEG signal rhythms is very inexpensive and contains appropriate information for the study of neurological disorders.

This type of analysis involves estimating the power intensity in the frequency bands. By comparing the EEG signal of healthy people and Alzheimer's patients, we see a slowing of the EEG signal. Increased bandwidth of delta (4-0.5 Hz) and theta (4-8 Hz) is seen along with decreased bandwidth of alpha (13-8 Hz). The slowness of the EEG signal rhythms is related to the upper stages of the disease and the progression of the disease from the early stages. The neural mechanism of high alpha rhythm is the result of a series of excitatory and inhibitory postsynaptic potentials in the dendrites of brain pyramidal neurons. This potential is mainly related to the distance and proximity of cortical modules and, like inhibitory thalamic reticular fibers and thalamic cortical fibers, is excitatory cortex. Simultaneous activity of a large number of neurons increases the alpha rhythm amplitude and decreases the peak frequency of the alpha power amplitude. The alpha band is considered from the frequency of 8 to 13 Hz, which changes in the frequency window due to age-related diseases. EEG frequency evaluation actually shows that due to various pathologies, an increase or decrease in the alpha band frequency has been seen and the result is an alpha peak shift towards higher or lower frequencies. One of the important results of EEG signal analysis is the identification of alpha band frequency bands, which indicate different cognitive performance. The two categories of frequency indicators studied are theta-alpha transmission frequency (TF) and separate alpha frequency (IAF). The separate alpha frequency indicates the power peak in the frequency range of 5 to 14 Hz, and the theta-alpha transmission frequency is the frequency in the EEG signal, which is less than the separate alpha frequency and has the lowest power. The frequency range of the EEG signal is determined by TF and IAF, and the frequency bands are: Delta, Theta, Alpha 1, Alpha 2, and Alpha 3. Figs. 4.3-4.4 show a model of a typical EEG signal power range that averages all electrodes [8]. As a result, using the processing method of power spectrum and analysis of individual frequency bands, changes such as reduced EEG signal rhythm in Alzheimer's patients, correlation between EEG signal and MMSE test, a special reduction in alpha bandwidth 2 and 3 with no slow frequency of alpha frequency band. And we will increase the power of the relative delta. The important point is that according to the diagnosis of mild Alzheimer's patients, characteristics can be considered to differentiate this group of healthy people and severe patients. According to the power spectrum and frequency analysis, we will see how the power spectrum changes. In the early stages of the disease as an increase in theta power and a decrease in beta power with a decrease in alpha frequency and in the more severe

stages of the disease, we see a decrease in alpha activity and an increase in delta power.

4.6. Time-frequency Analysis

For more useful and better display of the signal, simultaneous display in both time and frequency dimensions has been proposed. Time-frequency analysis is more important for non-static signals because different parts of the signal have different frequency content. Different time-frequency analyzes have been used in the field of EEG signal, which can be referred to as short-term instantaneous STFT conversion (Wigner distribution, Violet conversion, etc. [9].

4.6.1. Time-frequency Analysis with Short-term Instantaneous Conversion

Using this method, by calculating the Fourier transform of the signal, the signal strength is displayed as color intensity in terms of time and frequency. Warmer color intensity indicates more power and colder color intensity indicates less power. Fig. 4.2 shows the time-frequency curve of the EEG signal corresponding to the 3 channels Fz, Cz, Pz after preprocessing in the closed state for a healthy subject.

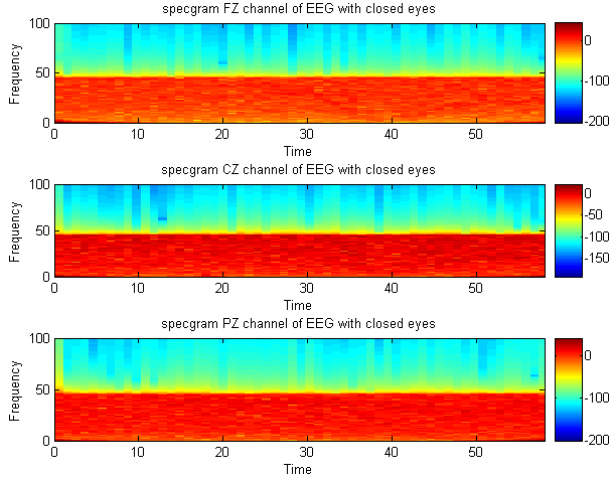

Fig. 4.2. EEG signal power spectrum related to 3 channels Fz, Cz, Pz after preprocessing in the closed mode.

4.7. Wavelet Properties

The number of decomposition levels is selected based on the dominant frequency components in the signal, those parts of the signal that are highly correlated with the frequencies required for signal classification are kept in violet coefficients Given that the EEG signal usually does not have usable components at frequencies above 64 Hz, our select level was 5. Thus, 5 sets of coefficients have been obtained that indicate the amount of signal energy in the frequency bands of 32-34 Hz (gamma band), 16-32 Hz (beta band), 8-16 Hz (alpha band), 8-4 Hz (theta band), and 0-4 Hz (Delta band). Each of these bands, due to the reduction of the sampling rate in the calculation process, contains only the minimum number of coefficients required to store the information required in that band. This property is not limited to a particular type of violet and is in fact an intrinsic property of all violets, but the better the violet used is in accordance with the signal, the better the quality of this decomposition. In order to select the most suitable wavelet for analysis and analysis of EEG and ERP signals for Alzheimer's disease, a discrete one-dimensional violet has been used. DWTs contain fast and simple computational algorithms and because of the signal analysis in different frequency bands with different resolutions, hence, they are called multi-resolution analyzers.

He used db4, db8 and db20 to diagnose Alzheimer's disease. So the signal is broken down into details D1-D4 and the last approximation, A4. This approximation and the details recorded using violet coefficients have been reconstructed in such a way that, for example, the A4 approximation is obtained by placing the D4 component on the A4 approximation. In fact, here the Violet transform acts like a mathematical microscope, focusing on small scales to detect them compactly in the space of time events, and to direct large scales, Displays the overall wave pattern shrinking [10].

4.8. Function and Correlation Coefficient

Correlation indicates the degree of similarity or similarity between two variables. Correlation function analysis is mainly presented in two ways: 1- Mutual correlation. 2- Self-solidarity. Determining signal components from the autocorrelation function is difficult when the signal contains several dominant rhythms, while it can be easily determined by the

power spectrum. Sequential values of the EEG signal, which are the result of a statistical process, are not necessarily independent [11]. Discrete and consecutive values of a signal have a certain dependence on each other and in fact the correlation of each signal at moment n indicates the degree of similarity of the signal to m sampling period before the current time. If the signal changes drastically, its resemblance to the previous moments will be small and it is a sign for deformation and static detection of the signal.

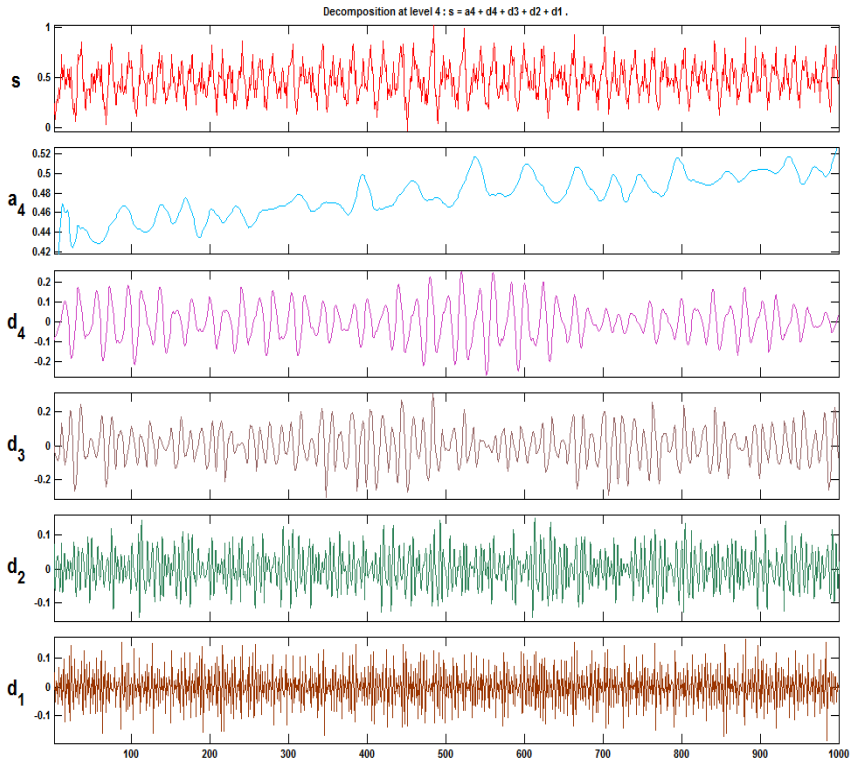

Fig. 4.3. Fz channel EEG signal broken down into decomposition surfaces by violet conversion for a healthy subject with eyes closed.

4.9. Coherency

Coherency and correlation dimension indicate the amount of interactions and functional connections in different areas of the brain, and in Alzheimer's disease, by measuring the amount of interactions, we

conclude that not all frequencies are equally degraded. Decreased functional interactions in the cerebral cortex are a possibility for cognitive activity abnormalities in Alzheimer's disease. Decreased coherence indicates a decrease in interactions, and in Alzheimer's disease, due to the characteristics at the neuronal level, where at this level there is damage to their connections and communications, we see this decrease in interactions. This is important in different frequency bands. For example, a decrease in coherence is compatible for the alpha band, but in the theta band we see an increase in compliance. Fig. 4.5 shows the alignment between different channels with a bar chart in the closed eye mode for a healthy subject.

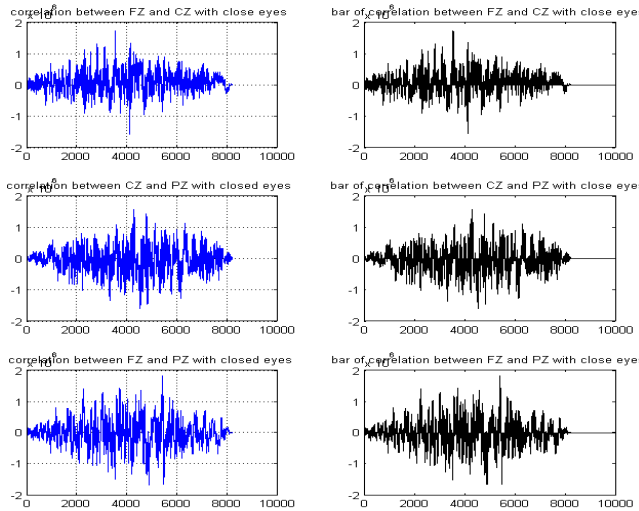

Fig. 4.4. Correlation function with bar chart between Fz, Cz, Pz channels in the blindfold mode for a healthy subject.

4.10. Nonlinear Features

Another approach to the EEG signal is the nonlinear perspective and the chaos of the signal type. With this in mind, tools describing the dynamics and absorption substrate for a chaotic nonlinear system and signal system should be used. The parameters that express chaotic behavior are twofold. The first group is those who emphasize the dynamics of chaotic behaviors, such as Lyapunov's exponent and entropy. These parameters

describe how the system behaves over time and in close paths as time goes on. The second category emphasizes the geometric nature of motion paths in state space, such as the correlation dimension. In the last two decades, nonlinear dynamic analysis for physiological signals has been widely used and the interesting point is the analysis and analysis of EEG signal of Alzheimer's patient by this method [12]. The relationship between nonlinear EEG signal characteristics of Alzheimer's patient is associated with a decrease in the complexity of the EEG pattern and a decrease in functional connections in different areas of the cerebral cortex in these patients. The basic premise of this processing method is that the EEG signal itself has nonlinear properties and the interaction of neurons is nonlinear and chaotic. In recent years, with the use of nonlinear dynamics, it has become increasingly clear that the brain has nonlinear dynamics and a nonlinear system, and the EEG signal, which is one of the outputs of this system, also has nonlinear dynamics.

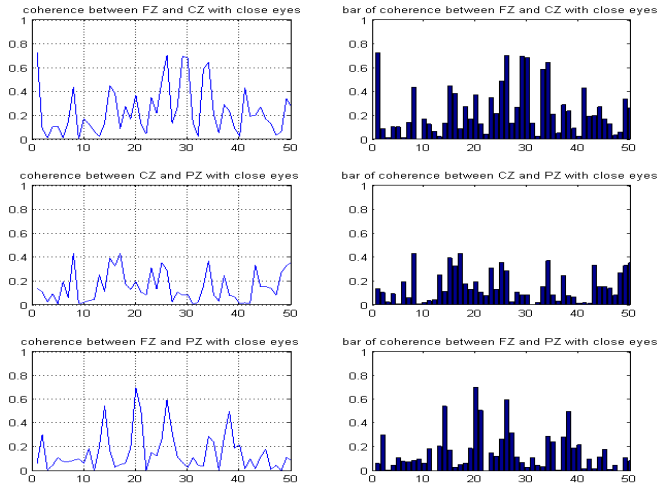

Fig. 4.5. Coherency between Fz, Cz, Pz channels with bar chart in closed eye for healthy subject.

4.11. Lyapunov's Exponent

Lyapunov's exponent shows the average view of the convergence or divergence of the neighboring trajectory in the phase space, which also shows the sensitivity of the dependence on the initial conditions. Lyapunov exponent criterion is used according to the assumption that the

difference in distance between trajectories in the adsorption bed space has an exponential growth [13].

4.12. Correlation Dimension

The correlation dimension indicates the number of independent variables needed to describe the dynamics of the system and is another way to check the chaotic signal. For example, when the correlation dimension of a trajectory is zero, it indicates a steady state or steady state of the system, and for a state where this value is equal to one, it indicates periodic behavior. When a chaos occurs in the system, the value of this variable is incorrect. The higher the value of this parameter leads to more complexity of the nonlinear system. Therefore, it can be stated that the correlation dimension is the size of the complexity of the distribution of points in the phase space and then the EEG signal correlation reflects the complexity of brain dynamics [13].

4.13. Entropy

Entropy means randomness and predictability. Higher entropy means more randomness and less order of signal. Entropy is extracted from the signal in two ways, one by time and by histogram and the other by frequency or violet. This feature can be used for signals. Entropy indicates the degree of complexity and irregularity of the signal, in the regular time series it has a small value and in the irregular or chaotic time series it has a higher value.

4.14. Optimal Feature Selection

After selecting the various features, the goal is to calculate and select the optimal feature. There are various methods for examining the features or channels that have the greatest effect on the classification. The most common method is statistical analysis and conclusion on the confidence interval. To identify the features that have the most significant differences, according to the classification and having three classes of severe patient, mild patient and healthy person, all the features calculated by analysis of variance (ANOVA) are compared. This is to be able to identify the best period according to the characteristics extracted in different periods (closed-eye-open-eye-reminder and stimulation period). Another method is to search between features or channels, in the

sense that the features or channels are subtracted from the whole, one by one, and its effect is observed in the classification. This is a general search method that is very time consuming in practice. For the applicability of this method, step-by-step and branch and boundary algorithms have been proposed [14]. But the most common search methods where derivation is not possible are genetic algorithms, bird algorithms (PSO), ant colonies (ACO), and the cooling process of molten metals [15]. In practice, these algorithms provide acceptable solutions with implementations smaller than the main space, but there is no guarantee that their solutions will be optimal. On the other hand, due to the feature space defined on the input signals containing information, it is very large, which has no special effect on helping to categorize the feature. Therefore, by reducing the dimension of feature space, we can reduce the computational load. In this study, to reduce the dimension, the PCA principal component analysis method has been used.

In this project, after extracting the feature matrix in the closed-eye and open-eye mode and the reminder period, which is 40*37; 37 is the number of extracted features and 40 is the number of data (subjects), and in the recording mode in the course. The stimulation of the property matrix is 40×45, of which 45 are the number of extracted properties and 40 are the number of data (subjects). Genetic algorithms are then applied to investigate the optimal feature and principal component analysis to reduce the feature space.

4.15. Classification

The ultimate goal in any pattern identification problem is to separate a set of samples into two or more different categories. In this study, the aim is to differentiate brain signals into three states: healthy, mild and severe. In teacher methods, a tagged data set is used as a training set to set the classifier parameters. In other words, the classification tool learns the input and output space of the problem and the relationship between them from a series of educational data. Therefore, in this study, several classifiers such as linear separator (LDA), support vector machine and Elman neural network have been used, which was the purpose of the contrast between static and dynamic classifiers. With these classifiers considered, it is possible for them to over-learn, in which case the trained classifier becomes overly dependent on the data of the education, and despite achieving the accuracy of the higher education, the ability it has little generalizability and the accuracy of the test results decreases. Since

the advent of deep learning, convolutional neural networks have been the primary basis of ideas in deep learning. The convolution network is abbreviated as CNN or ConvNet. Convolution neural network like other neural networks, for example, the multilayer perceptron neural network is composed of weighted and biased neuronal layers with the ability to learn. You know for sure that the following things happen in every neuron:

1. The neuron receives input sets;
2. Internal multiplication is done between the weights of the neurons and the inputs;
3. The result is added by bias;
4. Finally, a nonlinear function (the same as the activation function) is passed.

The difference between MLP and convolution is the difference in input. Suitable data for CNN include:

1. One-dimensional data: signal and sequence (for example, a sequence of words);
2. Two-dimensional data: image and sound spectrogram;
3. 3D data: video and volumetric images (e.g. MRI images);
4. Four-dimensional data: Volumetric images with time (such as fMRI).

The convolution network usually consists of several blocks. The different layers or blocks on the CNN network are: 1. Input layer; 2. Convolutional layer; 3. Non-linear activation function; 4. Pooling layer; 5. Fully connected layer. The polishing layer is another important layer in the convolutional neural network. The purpose of the polishing layer is to reduce the spatial size of the feature map obtained using the convolution layer. The polishing layer has no teachable parameter. Performs only a simple and effective sampling [16].

4.16. Methods for Dividing Data into Training and Test Categories and Validation Methods

In many cases, the amount of data is usually not large enough to be classified accurately, so methods have been proposed to develop

methods for splitting the dataset. In the first division, the existing data set is divided into two groups: model design and model test. Test data is used to measure the quality of modeling. In the second stage, the design data is divided into two groups: training and validation. Training data is used to build train, model, or classifier, and validation data is used to validate training, preventing over-learning. It should be noted that the model should not be used in the test until the validity of the model is verified. Different data sharing methods have been studied and compared to evaluate the efficiency of processing methods in the reference [17]. There are several different ways to share data, each with its own strengths and weaknesses, the most common of which are:

HCV: In this method, the data are divided into two groups of training and testing with a fixed ratio (for example, 80 % and 20 %). In this method, depending on the amount of data, which is high or low, the percentage of division is different. Usually, if the data is large, 50 % is considered for the test and 50 % for the test. On the other hand, the accuracy obtained with this procedure depends to some extent on the data selected for the test, that is, depending on whether the test data is data close to the category boundaries or data far from the border, the accuracy of the method is changing.

KCV: This method, which is proposed to solve HCV problems, the whole data set is randomly divided into K groups, which do not overlap. Then, for each desired combination, training and test combinations are made from this K group. Then, during the K stage, each time one of the categories is used as a test and the other K-1 category is used as a training, and the validity of the method is measured on the test category. Finally, K value is obtained for the accuracy of the method, the average of which can be considered as the accuracy of the method. The lower the diffraction of the values obtained from step K, the greater the validity of the result. Most articles on the division of the EEG signal into test and experimental groups use more K = 10 and K = 5. In this project, the KCV method with a value of k equal to 10 is considered.

PCV: In this method, K loads of test and design sets are formed relatively randomly and they are used to build and test the model. The value of K should be such as to ensure that all data are used uniformly for testing and design. If N is the total number of data and NT is the number of data in the test group, it must be K≤ (N¦N_T). For example, 50 times the data is randomly divided into two groups of 20 % test and 80 % design and

they are used to build and test the model and then the average and diffraction of the results as the overall result becomes.

LOO: This method is the limit state of PCV and KCV methods. In this method, K is assumed to be equal to the total number of available samples, so in each step one sample is discarded and then the rest of the data is used to build and teach the model and then the discarded data is discarded. To use for testing this method has a very good reputation, but it is very time consuming because the model has to learn the number of elements of the main set. This method is the best option in cases where the total number of samples is small.

4.17. Labeling Steps

In the first stage of the project, the subjects must first be evaluated by a physician with criteria such as clinical tests. One of the appropriate criteria for labeling subjects is the MMSE memory test. Of course, there is another test such as DRS, which is not needed due to the length of the test and proper evaluation of the memory status test [18].

4.18. Type and Number of Channels to Record the Signal

In this project, 3 channels Fz, Pz, Cz are used unipolar to record the brain signal because the aim is to record the total brain activity and to study the activity of EOG and its effect on the electrical activity of the brain for a single channel bipolar Alzheimer's patient. The electrical activity of the eye has been recorded.

4.19. Signal Recording Protocol

Recording the brain signal from the subject includes the following steps: 1- Teaching the subject; 2- Closing the eyes for 1 minute; 3- Record open eyes for 1 minute; 4- Recording while performing the task assigned to the subject, which includes: a- Remembering the displayed shapes;. B- Counting target and non-target sounds in the audio audible sample [19]. After labeling by the doctor on the subjects (identification of a healthy person and a patient of mild and severe type) in the first part, the

subject will be asked to remain calm during the recording and the recording of the signal will not cause any side effects or effects on him. We address his concern about this issue and then describe to him the 4 steps and explain how it works and how the target and non-target sounds. After preparing the subject, we perform the second step. In the closed eye mode, we record the signal for 1 minute. Then in the third step, the subject will be asked to open his eyes and record the signal for 1 minute. At the end of the third stage, the images according to Fig. 4.6 will be displayed for 1 minute for the subject, and after this time, the subject will be asked to close his eyes and recall the images (review the images) in his mind. Meanwhile, his brain signal will be recorded for 1 minute. He will then be asked to open his eyes and express the shapes one by one aloud. In the last part, two sounds with different frequencies of 1 and 1.5 kHz will be played, which have been taught to the subject before recording, one of which is the target sound and the other is the non-target sound. In the fourth part of section (b) these sounds are played and the subject is asked to press the right key as soon as he hears the target sound and the left key as soon as he hears the non-target sound. The interval between stimuli (sound play) is 2 seconds and the sounds will be played randomly. The only important point is that 75 % of the number of stimuli is non-target sound and 25 % of the number of stimuli is target sound. If we assume the total number of stimuli to be 120, we will have a total of 30 target stimuli and 90 non-target stimuli, which can be randomly distributed between the non-target stimuli at 2-second intervals. The playing time of each sound (target and non-target) will be 300 milliseconds. This recording section is 276 seconds with the assumption of 120 excitations and the total signal recording time for each subject is approximately 10 minutes. This display pattern is shown in Fig. 4.7.

4.20. Research Volunteers

In this study, 40 participants aged 60 to 88 years (mean age 68.43 with standard deviation of 8.86) were used to record brain signals. They were all right-handed and had equal numbers of participants in each gender. There were nineteen healthy participants with MMSE scores from 23 to 30 with a mean of 27.57 and a standard deviation of 2.29. Eleven participants were in the mild patient group with an MMSE score range of 19 to 22 with a mean of 20.71 and a standard deviation of 0.95, and the last ten participants were in the severe patient group with an MMSE score range of 3 to 18 with a mean of 13 and a standard deviation of 6.09.

And 40 female and individual subjects are divided into literate and illiterate individuals according to the following conditions. The classification of research subjects is as shown below:

- 10 literate women with an age range of 60 to 88 years;
- 10 literate men with an age range of 60 to 88 years;
- 10 illiterate women in the age range of 60 to 88 years;
- 10 illiterate men aged 60 to 88 years.

Fig. 4.6. Images displayed for each volunteer.

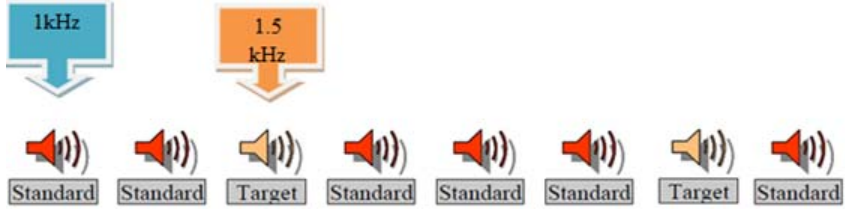

Fig. 4.7. A sample of audible oddball.

4.21. Brain Electrical Signals

Due to the electrical connection between nerve neurons and the transmission of neural information through electrical pulses between

different parts of the central nervous system, this very complex system can be considered as a set of countless electrical dipoles that all processes the human mind is manifested in it. Recording this electrical activity is possible by installing appropriate electrodes on the scalp, but due to the attenuation of signals reaching the electrodes and due to the volumetric conduction of electrical current inside the brain tissue, the signal recorded at each electrode includes information. It corresponds to a large volume of brain tissue and contains a complex combination of information about electrical sources in this area. The general method of recording brain signals is continuous recording, that is, the signal is continuously taken from a person during a certain state, and its changes are examined through ocular observation or processing methods. This type of recording of brain activity is called electroencephalography. But in some cases, the goal is to monitor brain activity during a specific event, such as in response to a stimulus. Although in this case, as in the previous case, the response to that condition appears on the EEG, but since different other parts also affect the brain at the same time, the effect of this particular event cannot be separated and examined separately [20]. The usual method in this case is to repeat the examined state and to average the synchronous (simultaneous) of the brain signals recorded during the repeated orders. In this case, due to the quasi-noise nature of the EEG signal, it is attenuated due to the average effect and the event-dependent signal, all of which are added with the same phase, is amplified. These signals that are generated in connection with an event are called ERP. Major ERPs are the responses that the brain elicits when exposed to a sensory stimulus. If the stimulus entered is a little more complex than a simple stimulus and is conceptual and requires diagnostic activity, other than the primary EPs known as exogenous ERPs, the response involves sensing the stimulus. Another group of responses appear with more delay, called endogenous ERPs, and are due to cognitive activity performed on the stimulus. ERPs are broken down into their constituent components according to various criteria. In the most common method of classifying and naming these components, they show the polarity of the waveform with the letters N (for negative) and P (for positive), followed by the amount of delay of the component or a number (respectively). Example N100 or N1 shows a negative peak with a delay of about 100 milliseconds relative to the excitation time) [21]. Research on the endogenous components of ERP, including P100, N100, P200, N200, P300 and N400, has shown their direct relationship with some cognitive activities, but among all these components, P300 is one of the most authoritative and powerful positions in It is related to the cognitive activity of the brain.

4.22. P300

P300 is a specific type of ERP, or in other words, a component of ERP that appears in specific situations. According to research, a P300 wave appears in the recorded brain signal when the brain encounters a new stimulus (unusual stimulus) while processing a series of normal stimuli. Of course, to produce a P300, it is necessary to define a specific task for the individual to perform only in response to the target stimulus, for example, to ask the subject to count the number of these stimuli. Physically, this component has a positive polarity and a range of about 10 to 15 microvolts. For acoustic stimulation, the average P300 wavelength is about 300 milliseconds, which was chosen because of its positive polarity and its 300 millisecond delay, but for other stimuli, such as visual stimulation, this time may be up to about 1000 milliseconds. Also increase. But in general, the average delay is between 300 and 1000 milliseconds. Spatially, this signal is recorded from three channels on the midline of the head, namely Fz, Cz, Pz. Research has shown that in comparison between these three channels, most cases of P300 have the highest amplitude in the parietal region (Pz) and the lowest amplitude in the frontal region (Fz) [22,23] (Pz> Cz> Fz). Of course, it is necessary to pay attention to the fact that age, gender and many psychological characteristics such as intelligence and personality, etc. affect the amplitude and delay of this wave. In addition, the magnitude of this wave varies with the amount of information provided by the stimulus, meaning that the greater the mental task required of the individual on unusual stimuli, the greater the amplitude of the extracted P300 wave.

4.23. Origin P300

Despite various studies on the P300, a single conclusion has not yet been reached as to where it will be produced. Part of the initial studies on this subject led to the conclusion that part of the P300 is produced in the Medial-Temporal and in the Hippocampus is related to learning and memory. Subsequent research has shown that Hippocampal has an indirect effect on P300, and new findings further emphasize the effect of Temporal-Parietal Junction, indicating that P300 is a cortical process. As for the functional roots of the wave as a whole, studies have shown that changes in the size of the P300 indicate a significant resource allocation for memory processing, and a P300 delay reflects the time required to allocate resources and update memory for a It is a special task and a special subject [24].

4.24. Oddball Pattern for P300 Extraction

According to Sokolov, the P300 appears when special attention needs to be paid to processing a new stimulus that is different from previous stimuli. In the Oddball pattern, two different types of stimuli are applied to a person in a series of stimuli, one of which is relatively unusual. The subject is asked to respond to this unusual stimulus, for example by pressing one key and not respond to another stimulus, or to press another key. Fig. 4.8 shows the different components of the ERP signal during two different forms of stimulation during a healthy subject's reaction.

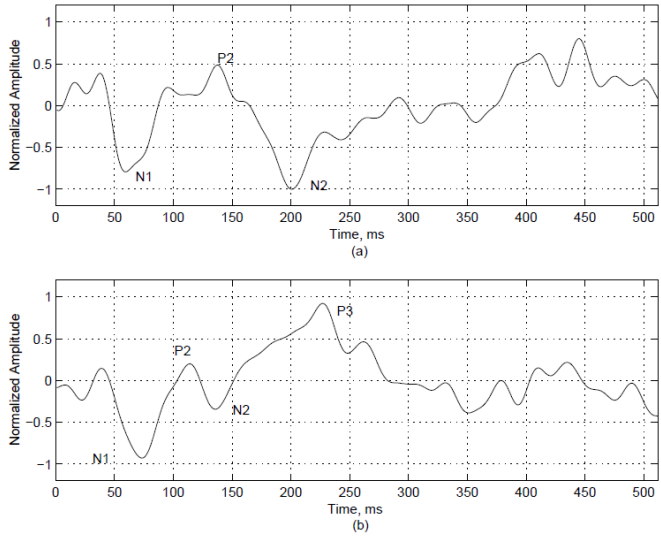

Fig. 4.8. Different components of the ERP signal are stimulated in two different ways during a healthy subject's reaction, a: Demonstration of components N2, P2, N1 of ERP signal after normal stimulation (non-target), b: Demonstration of components P3, N2, P2, N1 of ERP signal after unusual stimulation (target).

Fig. 4.9 shows the results of LDA-assisted brain signal resolution with optimal properties obtained from PCA, GA, ANOVA. The results of separation accuracy in three channels Fz, Cz, Pz in three categories of optimal properties selected by PCA, GA, ANOVA methods are shown in Fig. 4.10.

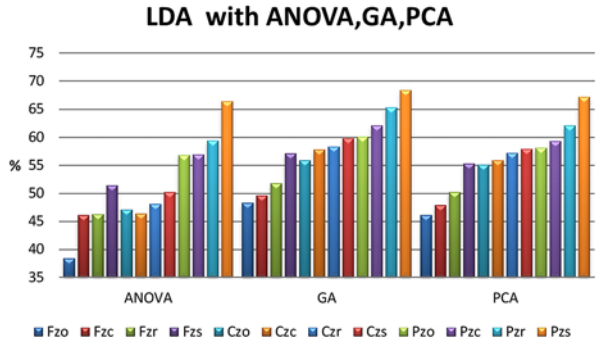

Fig. 4.9. Results of LDA-assisted brain signal resolution accuracy with optimal features from PCA, GA, ANOVA.

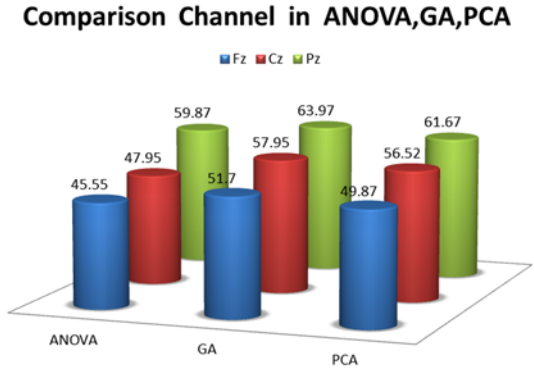

Fig. 4.10. Results of separation accuracy in three channels Fz, Cz, Pz in three categories of optimal properties selected by PCA, GA, ANOVA methods.

In this section, after identifying the optimal extraction properties by analysis of variance, genetic algorithm and reducing the dimensions of the properties by principal component analysis, it is applied to the SVM classifier with different core functions. KCV method is used for validation. The results of SVM signal separation accuracy with different kernel functions with optimal features obtained from ANOVA, GA, PCA are shown in Figs. 4.11, 4.12 and 4.13.

Fig. 4.14 shows the accuracy of the results obtained from all brain signals with the help of Elman with the optimal properties obtained from PCA, GA, ANOVA.

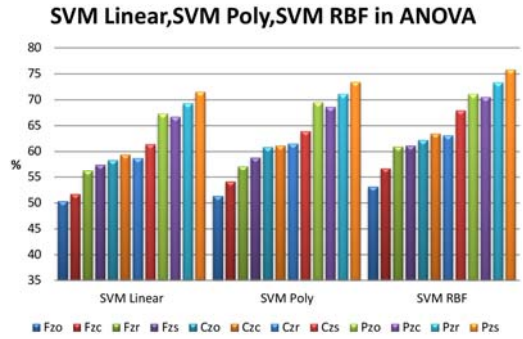

Fig. 4.11. Results of SVM signal separation accuracy with different core functions with optimal characteristics obtained from ANOVA.

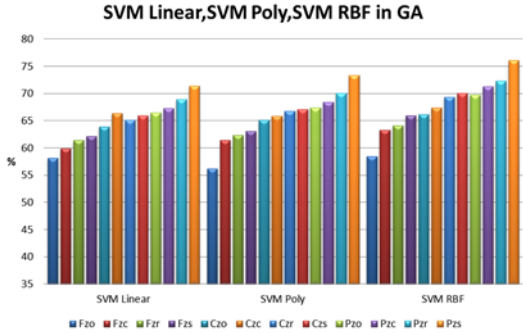

Fig. 4.12. Results of SVM signal separation accuracy with different core functions with optimal GA characteristics.

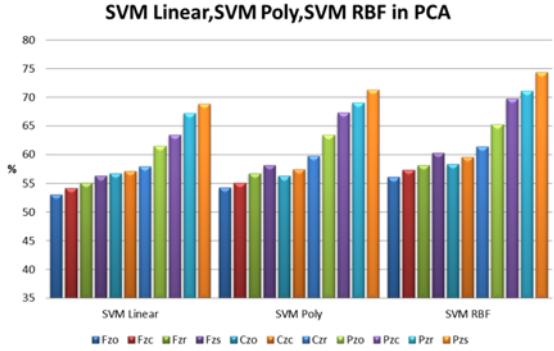

Fig. 4.13. Results of SVM signal separation accuracy with different core functions with optimal properties.

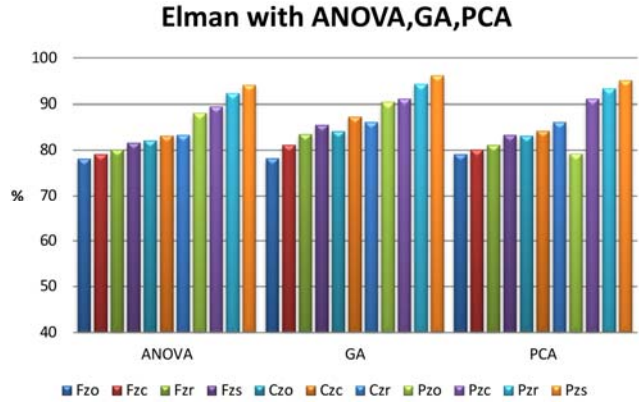

Fig. 4.14. Results of accuracy of Elman-assisted brain signal separation with optimal properties obtained from PCA, GA, ANOVA.

4.25. Results of Two Neural Networks, Canalization and Perceptron Network

The use of classification methods aims to achieve the highest possible accuracy in classifying and identifying categories. In some cases, it is more important for us to correctly identify samples from one of the categories. The quantitative representation of the convolutional neural network disorder matrix in three groups of healthy, mild and severe patients in recall mode is shown in Table 4.1. The concept of confusion matrix divides the accuracy of results into several groups for classification. Initially, a channel neural network is applied to the brain signal, which is the highest accuracy in the reminder mode. Fig. 4.15 shows the perturbation matrix for the canal neural network in the reminder mode.

Table 4.1. Quantitative display of convolutional neural network disruption matrix in three groups of healthy, mild and severe patients in reminder mode.

Class	TP (%)	FN (%)	FP (%)	TN (%)
Class(1):Healthy	**30**	**17.5**	**7.5**	**45**
Class(2):Mild AD	**15**	**12.5**	**15**	**57.5**
Class(3):Severe AD	**22.5**	**2.5**	**10**	**65**

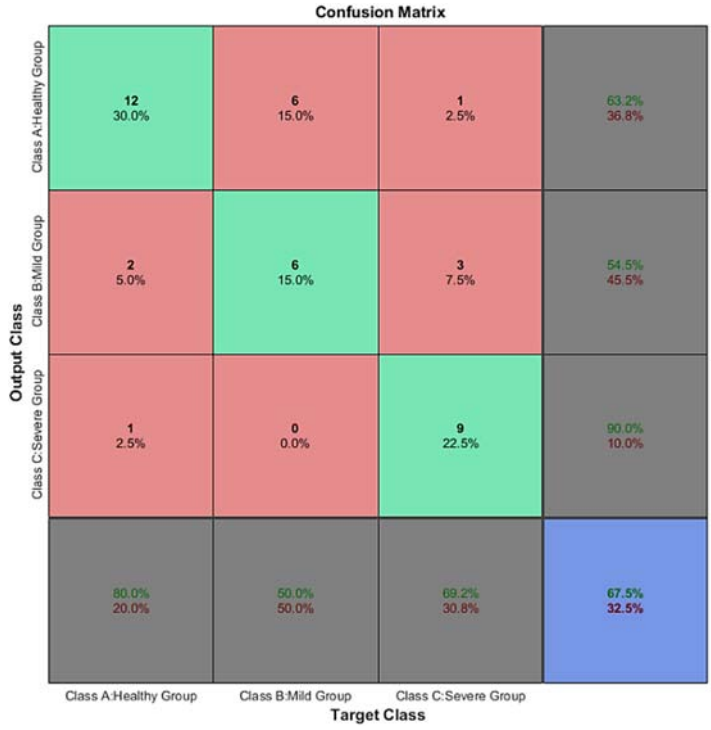

Fig. 4.15. Disruption matrix for canalization neural network in three groups of healthy individuals, mild and severe patients in reminder mode.

In Table 4.2, the parameters of the accuracy of the results of the canal neural network for the three groups of healthy individuals, mild and severe patients in the state of reminder are determined and these results show that accuracy in the group of severe patients is higher than other groups and accuracy in the group of mild patients is higher than others. There have been fewer groups. Disorder matrix of convolutional neural network in three groups Healthy, mildly and severely irritated patients are shown in Fig. 4.16. The quantitative representation of the convolution neural network disorder matrix in three groups of healthy, mildly and severely stimulated patients is shown in Table 4.3.

In Table 4.4, the parameters of accuracy of canalization neural network results for three groups of healthy, mild and severely stimulated patients are determined, and these results show that accuracy in severe patients is higher than other groups and accuracy in mild patients is higher than others. There have been fewer groups.

Table 4.2. Parameters of accuracy of canal neural network results for three groups of healthy, mild and severe patients in reminder mode.

Class	**Precision (%)**	**Accuracy (%)**	**Sensitivity (%)**	**Specificity (%)**	**F-Score (%)**
Class(1): Healthy	80	75	63.1	85.7	70.5
Class(2): Mild AD	50	72.5	54.5	79.3	52.1
Class(3): Severe AD	95.7	87.5	90	86.6	92.7

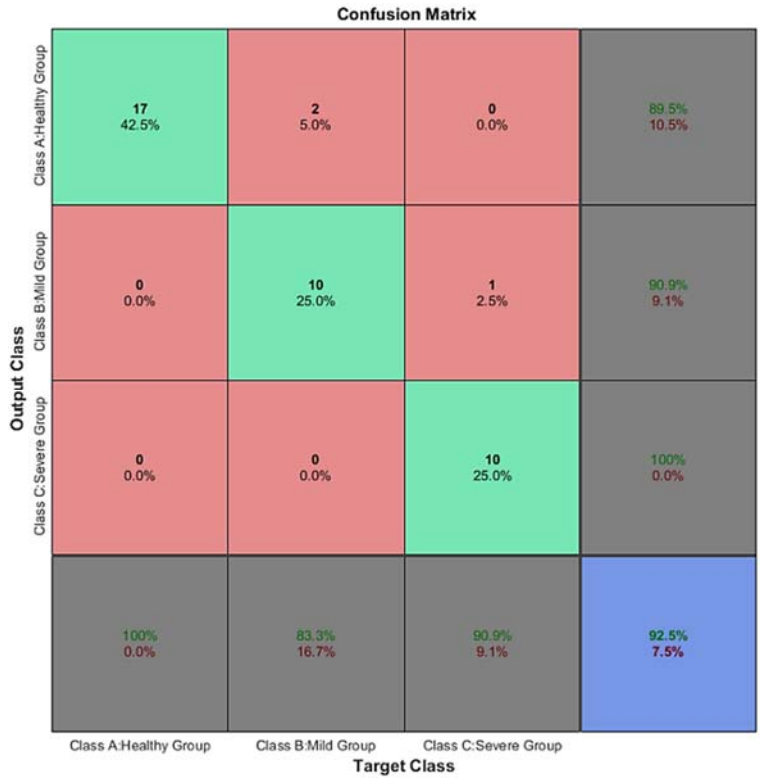

Fig. 4.16. Disorder matrix for convolutional neural network in three groups of healthy, mild, and severely stimulated patients.

Table 4.3. Quantitative representation of convolutional neural network disruption matrix in three groups of healthy, mild and severely stimulated patients.

Class	TP (%)	FN (%)	FP (%)	TN (%)
Class(1):Healthy	42.5	5	0	52.5
Class(2):Mild AD	25	2.5	5	67.5
Class(3):Severe AD	25	0	2.5	72.5

Table 4.4. Parameters of accuracy of channel neural network results for three groups of healthy, mild and severely stimulated patients.

Class	Precision (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)	F-Score (%)
Class(1): Healthy	100	95	89.4	100	94.4
Class(2): Mild AD	83.3	92.5	90.9	93.1	86.9
Class(3): Severe AD	90.9	97.5	100	96.6	95.2

4.26. Discussion and Conclusion

Memory status test is an assessment of a candidate's condition for initial diagnosis and classification. This test was able to assess the volunteer's learning memory and working memory. On the other hand, in order to differentiate Alzheimer's and dementia, the treating physician performs a dementia scale test from the volunteer to prevent dementia patients from entering this study. Lack of hearing, appropriate age range of volunteers and assessment of uniform literacy of volunteers reduce the factors that interfere in the research process.

All volunteers are right-handed so that this factor is not considered as an interference factor in choosing the right and left buttons after hearing the target and non-target sounds, because if the candidates are right-handed or left-handed, they can respond quickly to broadcast sounds. Act differently.

In the study of literacy and the separation of literate and illiterate people, only reading and writing skills in this study is considered literacy and having a degree and ranking does not change the level of literacy in this study. Because increasing the academic level, including the degree, has no effect on changing the ability of memory or cognitive deficits.

In order to better evaluate the working memory along with the learning memory, the candidates have been asked to use drawing them on paper, if necessary, in the memory mode. The displayed images have no color so as not to interfere with any candidate according to the concept of color in his mind. The recall time of the images is 1 minute and the display time of the images is the same. The more images the candidate recalls, the more they are reviewed in the initial category.

The brain signal in Alzheimer's disease has other characteristics such as reduced mean frequencies and reduced complex and correlated activities. Abnormal symptoms of this disease on the brain signal such as increased theta and delta activity and decreased beta and alpha activity and decreased dependence of beta and alpha bands are other symptoms of this disease. Decreased delta frequency band activity indicates a decrease in attention and concentration in individuals. Also, frequency shift in mild and severe Alzheimer's patients in theta and alpha frequency bands indicates that there is not always higher frequency band contents in this group of patients because the reduction Neurons and nervous system interactions occur in severe Alzheimer's patients.

Statistical and temporal characteristics of the EEG signal were not suitable in this study due to the rapid changes in this signal.

The alpha frequency band is a more important band than other frequency bands, so this frequency band is divided into three categories: alpha band 1, alpha band 2 and alpha band 3, and reducing the amplitude and changing the frequency range of this band to frequencies. It occurs lower in mild Alzheimer's patients, indicating a decrease in patients' concentration and attention.

In the preprocessing stages, a mid-pass filter with frequencies of 0.5 to 45 is used, because in order to eliminate city noise and use the notch filter, it is possible not to delete harmonics from city electricity that are combined with the signal, and to remove the component. It is clear that by removing redundant information from the brain signal, it improves the quality and accuracy of the work in subsequent processing. Eliminating rapid signal changes plus base line fluctuations greatly increases processing quality. The recorded signal has a set of artifacts and noises. Although the variety of artifacts in the signal is large, it can be divided into two main parts, namely, high-frequency artifacts due to the muscles of the head and neck, and low-frequency artifacts due to

electrode movement and transpiration. Reducing the sampling rate is optional, and the reason for this is that low-grade filters are usually used in signal recording equipment instead of precision anti-aliasing filters, which are large and expensive, but instead reduce the signal sampling rate. Select multiple times the cut-off frequency of the filter. Signal segmentation is done in two ways. One method is to divide the signal into pieces of the same length so that the signal can be assumed to be static along it, and another method is to use adaptive segmentation methods so that the signal pieces are divided into pieces of unknown length based on a static criterion. Although the accuracy of the second method is higher, but due to the ease of working with the same parts, the first method has been used. The length of the pieces is 2 seconds. The length of the piece is calculated based on the stationary amount of the signal in that part, and what is certain is that the brain signal is not static in general and can only be assumed to be static in certain conditions. The maximum length of the signal part varies depending on the application and the type of signal, in other words, the condition of the brain along with the type of processing and subsequent analysis are effective in static truth. Due to the high sampling frequency of the device, uniform segmentation method has been used and the removal of samples does not cause damage to the main signal.

The use of EEG signal is important for several reasons. The first reason is that Alzheimer's disease is a cortical disease whose abnormalities is well visible in this brain signal and have been comparable to the normal signal. Abnormal states of Alzheimer's disease in the brain signal directly reflect the functional and anatomical defects of the damaged cerebral cortex. Therefore, research on EEG signal dynamics in relation to neuronal injury has been very important. Another reason is that non-invasiveness can be considered a privilege according to the signal analysis and study of the mechanism of the disease. The study of nonlinear dynamics of this signal in Alzheimer's disease indicates nonlinear brain activity in different stages of the disease. Nonlinear dynamic analysis of this signal shows a decrease in the complexity of the brain signal pattern and a decrease in connections due to a decrease in the nonlinear cell dynamics between cortical regions. The next two features are correlation and Lyapunov's appearance, which indicates the feature space and the convergence or divergence of this space is slightly reduced in this disease. Nonlinearity is a prerequisite for chaotic behavior in dynamically known systems in nature.

The first changes in brain signal are an increase in theta activity and a decrease in beta activity, which is accompanied by a decrease in alpha activity. Increased delta activity occurs in later stages of the disease. In patients with higher stages and severe progression of this disease, we see a decrease in alpha activity and an increase in delta activity. In the intermediate stages, a decrease in beta activity and an increase in theta activity in the brain signal can be examined.

Due to the non-static nature of brain signals, Fourier features in this study were not suitable features for the diagnosis of mild Alzheimer's disease, and the conversion of violets to non-static EEG signals resulted in better results.

Among the applications of violet conversion is the separation of non-static signals with different frequency characteristics. While the Fourier transform is used to analyze station signals. Given the EEG signal, which contains static and non-static characteristics, it does not seem appropriate to use only the Fourier transform for analysis. Violet converter is able to show good properties of the signal in the time-frequency domain. Unlike Fourier transforms, where time accuracy and frequency are in conflict, violet transformers use variable-length windows that use a narrow window for high frequencies and a large window for low frequencies. Slowly usually the same violet is used for different frequencies and only its length changes. In this way, changing the length of the window increases the time accuracy for high frequencies and decreases for low frequencies. Conversely, frequency accuracy decreases at short times and at high times. The violet converter is a linear operator that conserves the energy of the signal and is an inverse converter that breaks down the signal into components, each of which appears at a scale or resolution.

Lack and reduction of functional interactions in the cerebral cortex is a possibility for cognitive activity abnormalities in Alzheimer's disease. Conformity indicates the interactions and functional interactions of different parts and the reflection of their interactions. Decreased compliance indicates a decrease in interactions and an increase in independent network agents, an increase in the number of degrees of freedom, and an increase in the correlation dimension.

In Alzheimer's disease, we see this decrease in interactions due to damage to connections and communications at the neuronal level. But

the important point is that the conformity is different in different frequency bands. For example, the decrease in compliance is compatible for the alpha band, but in the theta band we see an increase in compliance.

Contrary to expectations, the correlation decreases later in Alzheimer's disease. The combination of reduced complexity and reduced compliance in Alzheimer's disease is contradictory. One of the reasons for this discrepancy here could be that the change in functional connections may differ in specific frequency bands, and the difference in methods is in the calculation of conformity and correlation dimension. To estimate the correlation dimension, single-channel analysis is performed and to calculate compliance, two-channel calculations are performed. On the other hand, for Alzheimer's disease, a broadband-filtered signal is used to calculate the correlation dimension, and special frequency bands are used to calculate compliance. One of the objectives of this study is to investigate the relationship between correlation and compliance in Alzheimer's disease compared to multichannel methods. On the other hand, there are significant changes in the connections and interactions of different brain signal channels in Alzheimer's patients in higher frequency bands, greater sensitivity to correlation, and decrease in conformity and increase in correlation dimension in beta frequency band, decrease in synchronization in alpha and beta frequency bands. And gamma, increase in compliance and connections in the theta frequency band, decrease in functional connections and decrease in compliance in the alpha and beta frequency bands in the brain signal of Alzheimer's patients.

In this study, with increasing disease severity, EEG decreased entropy properties, indicating the degree of irregularity and signal changes. Because with the increase of the severity of the disease and the destruction of the interactions and neurons of the nervous system, the amount of signal changes decreases.

The courses studied are closed-eyed, open-eyed, reminder and stimulation, and among these 4 periods, the stimulation period was the best period for recording brain signal, because to diagnose Alzheimer's disease, it is more effective to evaluate the speed of stimulus response.

Channel is more suitable for recording brain signal because better information than Pz of Fz, Cz, Pz channels can be found in this channel between three channels to memory.

The number of optimal extraction features of the genetic algorithm is more than the number of optimal features of the principal component analysis method and the number of extractive features of the principal component analysis is more than the extraction features of the analysis of variance method. Among the optimal extraction properties, nonlinear and spectral properties are available as common optimal properties among different modes and it is expected that higher accuracy results will be achieved by these properties.

The principal component analysis method was not useful in this study compared to the analysis of variance or genetic algorithm because this method changes the dimension and vector of features and transfers them from one space to another by mapping, which in the new space has a significant relationship between features do not exist.

In statistically extracting the features, if the data distribution is normal and the features are considered independent of each other, the method of analysis of variance between the three groups can produce good results.

Nonlinear and dynamic optimal feature detection methods with the approach of competitive algorithms such as genetic algorithm due to the non-static and nonlinear nature of EEG signal have better results than analysis of variance and principal component analysis.

In this study, the ERP signal was formed by appropriate and accurate averaging of the EEG when creating sound stimulation with the target sound, and the maximum amount of the signal was named as component P3 or P300.

The two indicators of amplitude and latency of the P3 component are very important in the process of diagnosing Alzheimer's disease, but on the other hand, it is important to note that these two indicators are closely related to various parameters of the Oddball protocol and if these parameters are not examined, false results The amplitude and latency index of the P3 component of the ERP signal depends on the amplitude of the excitation signal, the frequency of the excitation signal, the excitation intervals and especially the probability of occurrence of two excitation classes. For example, if the frequency of target stimuli decreases or the intervals between stimuli increase, the amplitude of the P3 component increases in healthy individuals.

Another important point is that the amplitude and delay of the P3 component change with age and the stages of Alzheimer's disease and even forms of dementia. On the other hand, the scope of this component is directly related to the way memory works, that is, the more a person enjoys a more appropriate state of memory, the amplitude of this component increases. The delay of this component has been reported in various sources as approximately 300 milliseconds, which can vary depending on the condition of the subject and the recording of this component, but another important point is that the delay is 300 milliseconds, or in other words delays of less than 300 milliseconds represent better mental performance.

In this study, auditory stimulation was used and the effect of hearing loss was measured at the beginning of the test so that if the candidates have hearing loss, they would be excluded from this study and the response rate of people, including the duration and correct or incorrect number of voice recognition. The purpose of evaluating the candidates is that there are no intervention factors. Auditory stimulation can test a person's learning and working memory, as volunteers pressed the right and left keys to distinguish between target and non-target sounds.

The number of features extracted from ERP signal was 45 features and the number of features extracted from EEG signal was 37 features. It was higher than the EEG signal. Because in the ERP signal, there is a discussion about the speed of response of the volunteer to the sound stimulation, the importance of this signal is more than the EEG for the evaluation of mild Alzheimer's disease.

Due to the fact that the method of support vector machines and linear separators for separating the three classes has low accuracy. Therefore, for these two methods, two-level separation has been done, in which case the studied levels are: healthy-mild, mild-severe, healthy-severe. Between these two levels, the healthy-severe class has a higher resolution than the mild-severe class, and this class has a higher resolution than the mild-healthy class, because the difference between the healthy and severe classes is greater than the difference between the healthy and mild classes and is intense. The accuracy of the results by linear separator with selected features of genetic algorithm for Pz channel in healthy-mild, mild-severe, healthy-severe are 73.1 %, 79.5 % and 86.2 %, respectively. The accuracy of the results by RBF core vector machine and selected features of genetic algorithm for Pz channel in

healthy-mild, mild-severe, healthy-severe are 80.1%, 89.1% and 92.4 %, respectively.

The results of the backup vector machine are more accurate than the linear separator due to the non-stationary nature and dynamics of the EEG signal, provided that the data distribution is normal. This test is one of the most important statistical tests and due to the normal distribution of data, parametric test has been used.

The distinction between healthy and severe patients in this study had better results than the separation of mild and healthy patients because the changes in brain signal characteristics and medical images had a more significant difference between the two groups.

References

[1]. U. R. Acharya, S. Bhat, O. Faust, H. Adeli, E. C.-P. Chua, W. J. E. Lim, et al., Nonlinear dynamics measures for automated EEG-based sleep stage detection, *European Neurology*, Vol. 74, Issues 5-6, 2015, pp. 268-287.

[2]. B. A. Ally, G. E. Jones, J. A. Cole, A. E. Budson, The P300 component in patients with Alzheimer's disease and their biological children, *Biological Psychology*, Vol. 72, Issue 2, 2006, pp. 180-187.

[3]. A. Cichocki, S. L. Shishkin, T. Musha, Z. Leonowicz, T. Asada, T. Kurachi, EEG filtering based on blind source separation (BSS) for early detection of Alzheimer's disease, *Clinical Neurophysiology*, Vol. 116, Issue 3, 2005, pp. 729-737.

[4]. A. Craik, Y. He, J. L. Contreras-Vidal, Deep learning for electroencephalogram (EEG) classification tasks: a review, *Journal of Neural Engineering*, Vol. 16, Issue 3, 2019, 031001.

[5]. B. Czigler, D. Csikós, Z. Hidasi, Z. A. Gaál, É. Csibri, É. Kiss, et al., Quantitative EEG in early Alzheimer's disease patients – Power spectrum and complexity features, *International Journal of Psychophysiology*, Vol. 68, Issue 1, 2008, pp. 75-80.

[6]. C. Duyckaerts, M.-C. Potier, B. Delatour, Alzheimer disease models and human neuropathology: similarities and differences, *Acta Neuropathologica*, Vol. 115, Issue 1, 2008, pp. 5-38.

[7]. R. Ferenets, T. Lipping, A. Anier, V. Jantti, S. Melto, S. Hovilehto, Comparison of entropy and complexity measures for the assessment of depth of sedation, *IEEE Transactions on Biomedical Engineering*, Vol. 53, Issue 6, 2006, pp. 1067-1077.

[8]. H. Hampel, S. Lista, S. J. Teipel, F. Garaci, R. Nistico, K. Blennow, et al., Perspective on future role of biological markers in clinical therapy trials of

Alzheimer's disease: A long-range point of view beyond 2020, *Biochemical Pharmacology*, Vol. 88, Issue 4, 2014, pp. 426-449.

[9]. D. Hedges, R. Janis, S. Mickelson, C. Keith, D. Bennett, B. L. Brown, P300 amplitude in Alzheimer's disease: A meta-analysis and meta-regression, *Clinical EEG and Neuroscience*, Vol. 47, Issue 1, 2016, pp. 48-55.

[10]. R. Hussein, H. Palangi, R. K. Ward, Z. J. Wang, Optimized deep neural network architecture for robust detection of epileptic seizures using EEG signals, *Clinical Neurophysiology*, Vol. 130, Issue 1, 2019, pp. 25-37.

[11]. C. E. Jackson, P. J. Snyder, Electroencephalography and event-related potentials as biomarkers of mild cognitive impairment and mild Alzheimer's disease, *Alzheimer's & Dementia*, Vol. 4, Issue 1, 2008, pp. S137-S143.

[12]. J. Jeong, Nonlinear dynamics of EEG in Alzheimer's disease, *Drug Development Research*, Vol. 56, Issue 2, 2002, pp. 57-66.

[13]. M.-S. Lee, S.-H. Lee, E.-O. Moon, Y.-J. Moon, S. Kim, S.-H. Kim, et al., Neuropsychological correlates of the P300 in patients with Alzheimer's disease, *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, Vol. 40, 2013, pp. 62-69.

[14]. C. Melissant, A. Ypma, E. E. Frietman, C. J. Stam, A method for detection of Alzheimer's disease using ICA-enhanced EEG measurements, *Artificial Intelligence in Medicine*, Vol. 33, Issue 3, 2005, pp. 209-222.

[15]. T. Meuser, Clinical Dementia Rating (CDR) Scale, *Alzheimers's Disease Research Center Washington University*, St. Louis, 2001.

[16]. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV'17)*, 2017, pp. 618-626.

[17]. I. Szirmai, A. Kamondi, EEG investigations in cognitive impairments, *Ideggyogyaszati Szemle*, Vol. 64, Issues 1-2, 2011, pp. 14-23.

[18]. G. Vrbancic, V. Podgorelec, Automatic classification of motor impairment neural disorders from EEG signals using deep convolutional neural networks, *Elektronika ir Elektrotechnika*, Vol. 24, Issue 4, 2018, pp. 3-7.

[19]. X. Zhou, C. Zhou, B. Stewart, Comparisons of discrete wavelet transform, wavelet packet transform and stationary wavelet transform in denoising PD measurement data, in *Proceedings of the IEEE International Symposium on Electrical Insulation (ELINSL'06)*, 2006, pp. 237-240.

[20]. Y. R. Tabar, U. Halici, A novel deep learning approach for classification of EEG motor imagery signals, *Journal of Neural Engineering*, Vol. 14, Issue 1, 2016, 016003.

[21]. M. Torabi, H. Moradzadeh, R. Vaziri, S. M. J. Razavian, R. D. Ardekani, M. Rahmandoust, et al., Development of Alzheimer's disease recognition using semiautomatic analysis of statistical parameters based on frequency characteristics of medical images, in *Proceedings of the IEEE International Conference on Signal Processing and Communications (ICSPC'07)*, 2007, pp. 868-871.

[22]. H. Toyama, D. Ye, M. Ichise, J.-S. Liow, L. Cai, D. Jacobowitz, et al., PET imaging of brain with the β-amyloid probe, [11C]6-OH-BTA-1, in a transgenic mouse model of Alzheimer's disease, *European Journal of Nuclear Medicine and Molecular Imaging*, Vol. 32, Issue 5, 2005, pp. 593-600.

[23]. J. Van Deursen, E. Vuurman, V. van Kranen-Mastenbroek, F. Verhey, W. Riedel, 40-Hz steady state response in Alzheimer's disease and mild cognitive impairment, *Neurobiology of Aging*, Vol. 32, Issue 1, 2011, pp. 24-30.

[24]. M. R. Elias, A. Mahdi, G. Majid, K. Mohammad, Diagnosis of mild Alzheimer's disease by EEG and ERP signals using linear and nonlinear classifiers, *Biomedical Signal Processing and Control*, Vol. 70, 2021, 103049.

Index

IFSA

Advances in Artificial Intelligence
Volume 2

Sergey Y. Yurish, Editor

Artificial intelligence has is one of the fastest-growing technologies in recent years. The market growth is mainly driven by factors such as the increasing adoption of cloud-based applications and services, growing big data, and increasing demand for intelligent virtual assistants. Various end-use industries have also employed artificial intelligence such as retail and business analysis that has also boosted the demand in this market. The major restraint for the market is the limited number of artificial intelligence technology experts. The Book Series on *'Advances in Artificial Intelligence'* has been launched with the aim to fill-in this gap.

This book ensures that our readers will stay at the cutting edge of the field and get the right and effective start point and road map for the further researches and developments.

With this unique combination of information in each volume, the *'Advances in Artificial Intelligence'* book Series will be of value for scientists and engineers in industry and at universities, to developers and users.

ISBN 978-84-09-47562-9