Advances in Intelligent Systems and Computing 1038

Yaxin Bi
Rahul Bhatia
Supriya Kapoor *Editors*

Intelligent Systems and Applications

Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2

Springer

Advances in Intelligent Systems and Computing

Volume 1038

Series Editor

Janusz Kacprzyk, Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

Advisory Editors

Nikhil R. Pal, Indian Statistical Institute, Kolkata, India
Rafael Bello Perez, Faculty of Mathematics, Physics and Computing, Universidad Central de Las Villas, Santa Clara, Cuba
Emilio S. Corchado, University of Salamanca, Salamanca, Spain
Hani Hagras, School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK
László T. Kóczy, Department of Automation, Széchenyi István University, Gyor, Hungary
Vladik Kreinovich, Department of Computer Science, University of Texas at El Paso, El Paso, TX, USA
Chin-Teng Lin, Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan
Jie Lu, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia
Patricia Melin, Graduate Program of Computer Science, Tijuana Institute of Technology, Tijuana, Mexico
Nadia Nedjah, Department of Electronics Engineering, University of Rio de Janeiro, Rio de Janeiro, Brazil
Ngoc Thanh Nguyen, Faculty of Computer Science and Management, Wrocław University of Technology, Wrocław, Poland
Jun Wang, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing such as: computational intelligence, soft computing including neural networks, fuzzy systems, evolutionary computing and the fusion of these paradigms, social intelligence, ambient intelligence, computational neuroscience, artificial life, virtual worlds and society, cognitive science and systems, Perception and Vision, DNA and immune based systems, self-organizing and adaptive systems, e-Learning and teaching, human-centered and human-centric computing, recommender systems, intelligent control, robotics and mechatronics including human-machine teaming, knowledge-based paradigms, learning paradigms, machine ethics, intelligent data analysis, knowledge management, intelligent agents, intelligent decision making and support, intelligent network security, trust management, interactive entertainment, Web intelligence and multimedia.

The publications within "Advances in Intelligent Systems and Computing" are primarily proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

**** Indexing: The books of this series are submitted to ISI Proceedings, EI-Compendex, DBLP, SCOPUS, Google Scholar and Springerlink ****

More information about this series at http://www.springer.com/series/11156

Yaxin Bi · Rahul Bhatia · Supriya Kapoor
Editors

Intelligent Systems and Applications

Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2

Springer

Editors
Yaxin Bi
School of Computing, Computer Science Research Institute
Ulster University
Newtownabbey, UK

Rahul Bhatia
The Science and Information (SAI) Organization
Bradford, West Yorkshire, UK

Supriya Kapoor
The Science and Information (SAI) Organization
Bradford, West Yorkshire, UK

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-030-29512-7 ISBN 978-3-030-29513-4 (eBook)
https://doi.org/10.1007/978-3-030-29513-4

© Springer Nature Switzerland AG 2020
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Editor's Preface

The Intelligent Systems Conference (IntelliSys) 2019 was held on September 5 and 6, 2019, in London, UK. The Intelligent Systems Conference is a prestigious annual conference on areas of intelligent systems and artificial intelligence and their applications to the real world, which is built on the success of the IntelliSys conferences in the past five years held at London.

This conference not only presented the state-of-the-art methods and valuable experience from researchers in the related research areas, but also provided the audience with a vision of further development in the fields. The research that comes out of a series of the IntelliSys conferences will provide insights into the complex intelligent systems and pave a way for the future development.

The Program Committee of IntelliSys 2019 represented 25 countries, and the authors submitted 546 papers from 45 countries. This certainly attests to the widespread international importance of the theme of the conference. Each paper was reviewed on the basis of originality, novelty and rigorousness. After the reviews, 223 were accepted for presentation, out of which 189 papers are finally being published in the proceedings.

The event was a two-day program comprised of 24 paper presentation sessions and poster presentations. The themes of the contributions and scientific sessions ranged from theories to applications, reflecting a wide spectrum of artificial intelligence. We are very gratified to have an exciting lineup of featured speakers who are among the leaders in changing the landscape of artificial intelligence and its application areas. Plenary speakers include: Grega Milcinski (CEO, Sinergise), Detlef D Nauck (Chief Research Scientist for Data Science at BT Technology), Giulio Sandini (Director of Research - Italian Institute of Technology) and Iain Brown (Head of Data Science, SAS UK&I).

The conference would truly not function without the contributions and support received from authors, participants, keynote speakers, program committee members, session chairs, organizing committee members, steering committee members and others in their various roles. Their valuable support, suggestions, dedicated commitment and hard work have made the IntelliSys 2019 successful.

It has been a great honor to serve as the general chair for the IntelliSys 2019 and to work with the conference team. We believe this event will certainly help further disseminate new ideas and inspire more international collaborations.

Kind Regards,
Yaxin Bi
Conference Chair

Contents

A New Approach of Service Platform for Water Optimization in Lettuce Crops Using Wireless Sensor Network 1
Edgar Maya-Olalla, Hernán Domínguez-Limaico, Carlos Vásquez-Ayala, Edgar Jaramillo-Vinueza, Marcelo Zambrano V, Alexandra Jácome-Ortega, Paul D. Rosero-Montalvo, and D. H. Peluffo-Ordóñez

A Novel AI Based Optimization of Node Selection and Information Fusion in Cooperative Wireless Networks 14
Yuan Gao, Hong Ao, Weigui Zhou, Su Hu, Haosen Yu, Yang Guo, and Jiang Cao

Using Automated State Space Planning for Effective Management of Visual Information and Learner's Attention in Virtual Reality 24
Opeoluwa Ladeinde, Mohammad Abdur Razzaque, and The Anh Han

Real-Time Lane Detection and Extreme Learning Machine Based Tracking Control for Intelligent Self-driving Vehicle 41
Sabir Hossain, Oualid Doukhi, Inseung Lee, and Deok-jin Lee

Pedestrian Recognition and Obstacle Avoidance for Autonomous Vehicles Using Raspberry Pi 51
Charlie Day, Liam McEachen, Asiya Khan, Sanjay Sharma, and Giovanni Masala

Improving Urban Air Quality Through Long-Term Optimisation of Vehicle Fleets 70
Darren M. Chitty, Rakhi Parmar, and Peter R. Lewis

Analysing Data Set of the Bike–Sharing System Demand with R Scripts: Mexico City Case 90
Luis A. Moncayo–Martínez

A Path Towards Understanding Factors Affecting Crash Severity in Autonomous Vehicles Using Current Naturalistic Driving Data 106
Franco van Wyk, Anahita Khojandi, and Neda Masoud

Automobile Automation and Lifecycle: How Digitalisation and Security Issues Affect the Car as a Product and Service? 121
Antti Hakkala and Olli I. Heimo

Less-than-Truckload Shipper Collaboration in the Physical Internet . . . 138
Minghui Lai and Xiaoqiang Cai

Smartphone-Based Intelligent Driver Assistant: Context Model and Dangerous State Recognition Scheme 152
Igor Lashkov and Alexey Kashevnik

Voice Recognition Based System to Adapt Automatically the Readability Parameters of a User Interface 166
Hélène Soubaras

A Machine-Synesthetic Approach to DDoS Network Attack Detection . . . 179
Yuri Monakhov, Oleg Nikitin, Anna Kuznetsova, Alexey Kharlamov, and Alexandr Amochkin

Novel Synchronous Brain Computer Interface Based on 2-D EEG Local Binary Patterning 192
Daniela De Venuto and Giovanni Mezzina

LSTM-Based Facial Performance Capture Using Embedding Between Expressions 211
Hsien-Yu Meng and Jiangtao Wen

Automatic Curation System Using Multimodal Analysis Approach (MAA) 227
Wei Yuan, Yong Zhang, Xiaojun Hu, and Mei Song

Mixed Reality for Industry? An Empirical User Experience Study 241
Olli I. Heimo, Leo Sakari, Tero Säntti, and Teijo Lehtonen

Person Detection in Thermal Videos Using YOLO 254
Marina Ivasic-Kos, Mate Kristo, and Miran Pobar

Implementation of a Hybridized Machine Learning Framework for Flood Risk Management 268
Oluwole Charles Akinyokun, Udoinyang Godwin Inyang, and Emem Etok Akpan

Word Similarity Computing Based on HowNet and Synonymy Thesaurus 292
Hongmei Nie, Jiaqing Zhou, Hui Wang, and Minshuo Li

A Comparison of fastText Implementations Using Arabic Text Classification 306
Nuha Alghamdi and Fatmah Assiri

The Social Net of Sentiment: Improving the Base Sentiment Analysis of High Impact Events with Lexical Category Exploration . . . 312
Maxwell Fowler, Aleshia Hayes, and Kanika Binzani

Reflective Writing Analysis Approach Based on Semantic Concepts: An Evaluation of WordNet Affect Efficiency 321
Huda Alrashidi and Mike Joy

Semantic-Based Feedback Recommendation for Automatic Essay Evaluation 334
Tsegaye Misikir Tashu and Tomáš Horváth

Figurative Language Grounding in Humanoid Robots 347
Maja Gwóźdź

Masdar: A Novel Sequence-to-Sequence Deep Learning Model for Arabic Stemming 363
Mohammed M. Fouad, Ahmed Mahany, and Iyad Katib

Nomen Meum Earl: Yet Another Route to Intelligent Machine Behavior 374
Chris Lanz

LIT: Rule Based Italian Lemmatizer 395
Simone Molendini, Antonio Guerrieri, and Andrea Filieri

Intelligent Sense-Enabled Lexical Search on Text Documents 405
Anu Thomas and S. Sangeetha

Predicting Sentiment in *Yorùbá* Written Texts: A Comparison of Machine Learning Models 416
Abimbola Rhoda Iyanda and Omolayo Abegunde

An Introductory Survey on Attention Mechanisms in NLP Problems 432
Dichao Hu

Advanced Similarity Measures Using Word Embeddings and Siamese Networks in CBR 449
Kareem Amin, George Lancaster, Stelios Kapetanakis, Klaus-Dieter Althoff, Andreas Dengel, and Miltos Petridis

Intelligent Flower Detection System Using Machine Learning 463
Amna Safar and Maytham Safar

Development of an Interactive Virtual Reality for Medical Skills Training Supervised by Artificial Neural Network 473
Shabnam Sadeghi Esfahlani, Viktor Izsof, Sabrina Minter, Ali Kordzadeh, Hassan Shirvani, and Karim Sadeghi Esfahlani

Toward Robust Image Classification 483
Basemah Alshemali, Alta Graham, and Jugal Kalita

Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis 490
Muh. Ibnu Choldun R., Judhi Santoso, and Kridanto Surendro

Symmetry Constrained Machine Learning 501
Doron L. Bergman

An Integrated SEM-Neural Network for Predicting and Understanding the Determining Factor for Institutional Repositories Adoption 513
Shahla Asadi, Rusli Abdullah, and Yusmadi Yah Jusoh

Load Balancing Using Neural Networks Approach for Assisted Content Delivery in Heterogeneous Network 533
Raid Sakat, Raed Saadoon, and Maysam Abbod

Combining Diffusion Processes for Semi-supervised Learning on Graph Structured Data 548
Abdullah Al-Gafri, Muhammed Moinuddin, and Ubaid M. Al-Saggaf

Smart Energy Usage and Visualization Based on Micro-moments 557
Abdullah Alsalemi, Faycal Bensaali, Abbes Amira, Noora Fetais, Christos Sardianos, and Iraklis Varlamis

The Functional Design Method for Public Buildings Together with Gamification of Information Models Enables Smart Planning by Crowdsourcing and Simulation and Learning of Rescue Environments 567
Jukka Selin and Markku Rossi

Deep Learning Based Pedestrian Detection at Distance in Smart Cities 588
Ranjith K Dinakaran, Philip Easom, Ahmed Bouridane, Li Zhang, Richard Jiang, Fozia Mehboob, and Abdul Rauf

Infrastructural Models of Intermediary Service Providers in Digital Economy 594
Anton Ivaschenko, Stanislav Korchivoy, and Michail Spodobaev

"Smart City" Governance Technologies Development in the Era of the 4th Industrial Revolution 606
Mikhail Kuznetsov, Maria Nikishova, and Anna Belova

D2C-DM: Distributed-to-Centralized Data Management for Smart Cities Based on Two Ongoing Case Studies 619
Amir Sinaeepourfard, John Krogstie, and Sobah Abbas Petersen

Information and Thermic Control System for Water Contaminant Charge Removal of the City of Villavicencio, Colombia 633
Obeth Romero

Improving Traffic Safety Through Video Analysis in Jakarta, Indonesia 642
João Caldeira, Alex Fout, Aniket Kesari, Raesetje Sefala, Joseph Walsh, Katy Dupre, Muhammad Rizal Khaefi, Setiaji, George Hodge, Zakiya Aryana Pramestri, and Muhammad Adib Imtiyazi

Dynamic Scaling of EEG Fluctuations of Patients with Learning Disorders Based on Artificial Intelligence 650
Oswaldo Morales Matamoros, Jesús Jaime Moreno Escobar, Ixchel Lina Reyes, Teresa Ivonne Contreras Troya, and Ricardo Tejeida Padilla

Blood Glucose Level Prediction Using Optimized Neural Network for Virtual Patients 671
Muhammad Asad, Younas Khan, Usman Qamar, and Saba Bashir

A Review of Continuous Blood Glucose Monitoring and Prediction of Blood Glucose Level for Diabetes Type 1 Patient in Different Prediction Horizons (PH) Using Artificial Neural Network (ANN) 684
Muhammad Asad and Usman Qamar

ED Revisits Forecasting: Utilizing Latent Models 696
Ofir Ben-Assuli and Joshua R. Vest

Color Signal Processing Methods for Webcam-Based Heart Rate Evaluation 703
Mikhail Kopeliovich and Mikhail Petrushan

Genetic Algorithm Based Selection of Appropriate Biomarkers for Improved Breast Cancer Prediction 724
Arnab Kumar Mishra, Pinki Roy, and Sivaji Bandyopadhyay

Electronic Prosthetics for the Rehabilitation of the Carpal Tunnel Syndrome 733
Santiago Núñez, Patricio Encalada, Santiago Manzano, Juan P. Pallo, Dennis Chicaiza, and Carlos Gordón

Artificial Intelligent (AI) Clinical Edge for Voice Disorder Detection . . . 750
Jaya Shankar Vuppalapati, Santosh Kedaru, Sharat Kedari, Anitha Ilapakurti, and Chandrasekar Vuppalapati

Virtual Therapy System in a Multisensory Environment for Patients with Alzheimer's 767
Patricio Encalada, Johana Medina, Santiago Manzano, Juan P. Pallo, Dennis Chicaiza, Carlos Gordón, Carlos Núñez, and Diego F. Andaluz

Using Local Binary Patterns and Convolutional Neural Networks for Melanoma Detection 782
Saeed Iqbal, Adnan N. Qureshi, and Mukti Akter

A Hybrid Model to Predict Glucose Oscillation for Patients with Type 1 Diabetes and Suggest Customized Recommendations 790
João Paulo Aragão Pereira, Anarosa Alves Franco Brandão, Joyce da Silva Bevilacqua, and Maria Lúcia Cardillo Correa Giannella

Modeling a Vulnerability Index for Leprosy Using Spatial Analysis and Artificial Intelligence Techniques in a Hyperendemic Municipality in the Amazon 802
Rafael Eich da Silva, Valney Mara Gomes Conde, Marcos José da Silva Baia, Cláudio Guedes Salgado, and Guilherme Augusto Barros Conde

Ensemble Approach for Left Ventricle Segmentation 824
Chen Avni and Maya Herman

Escaping Diagnosability and Entering Uncertainty in Temporal Diagnosis of Discrete-Event Systems 835
Nicola Bertoglio, Gianfranco Lamperti, Marina Zanella, and Xiangfu Zhao

A Robotic Hand for Arabic Sign Language Teaching and Translation . . . 853
Maha Alrabiah, Hissah AlMuneef, Sadeem AlMarri, Ebtisam AlShammari, and Faten Alsunaid

Can Machine Learning Be Used to Discriminate Between Burns and Pressure Ulcer? 870
Aliyu Abubakar, Hassan Ugail, and Ali Maina Bukar

Genetic Algorithm Based Optimal Feature Selection Extracted by Time-Frequency Analysis for Enhanced Sleep Disorder Diagnosis Using EEG Signal 881
Md. Rashedul Islam, Md. Abdur Rahim, Md. Rajibul Islam, and Jungpil Shin

On Applying Ambient Intelligence to Assist People with Profound Intellectual and Multiple Disabilities 895
Michal Kosiedowski, Arkadiusz Radziuk, Piotr Szymaniak, Wojciech Kapsa, Tomasz Rajtar, Maciej Stroinski, Carmen Campomanes-Alvarez, B. Rosario Campomanes-Alvarez, Mitja Lustrek, Matej Cigale, Erik Dovgan, and Gasper Slapnicar

Style Transfer for Dermatological Data Augmentation 915
Tamás Nyíri and Attila Kiss

Analysis of the Stability, Control and Implementation of Real Parameters of the Robot Walking 924
Arbnor Pajaziti, Xhevahir Bajrami, Ahmet Shala, Ramë Likaj, Lum Rexha, Astrit Zekaj, and Dibran Hoxha

Autonomous Robot Navigation with Signaling Based on Objects Detection Techniques and Deep Learning Networks 940
Carlos Gordón, Patricio Encalada, Henry Lema, Diego León, Cristhian Castro, and Dennis Chicaiza

Intelligent Autonomous Navigation of Robot KUKA YouBot 954
Carlos Gordón, Patricio Encalada, Henry Lema, Diego León, Cristhian Castro, and Dennis Chicaiza

Intrusion Detection in Robotic Swarms 968
Ian Sargeant and Allan Tomlinson

Human Digital Twin: Enabling Human-Multi Smart Machines Collaboration 981
Wael Hafez

'If You Agree with Me, Do I Trust You?': An Examination of Human-Agent Trust from a Psychological Perspective 994
Hsiao-Ying Huang, Michael Twidale, and Masooda Bashir

Using Local Objects to Improve Estimation of Mobile Object Coordinates and Smoothing Trajectory of Movement by Autoregression with Multiple Roots 1014
Nikita Andriyanov and Konstantin Vasiliev

An Empirical Review of Calibration Techniques for the Pepper Humanoid Robot's RGB and Depth Camera 1026
Avinash Kumar Singh, Neha Baranwal, and Kai-Florian Richter

Convolutional Neural Network Applied to the Gesticulation Control of an Interactive Social Robot with Humanoid Aspect 1039
Edisson Arias, Patricio Encalada, Franklin Tigre, Cesar Granizo, Carlos Gordon, and Marcelo V. Garcia

Automation of Synthesized Optimal Control Problem Solution for Mobile Robot by Genetic Programming 1054
Askhat Diveev and Elena Sofronova

Towards Partner-Aware Humanoid Robot Control Under Physical Interactions 1073
Yeshasvi Tirupachuri, Gabriele Nava, Claudia Latella, Diego Ferigo, Lorenzo Rapetti, Luca Tagliapietra, Francesco Nori, and Daniele Pucci

Momentum-Based Topology Estimation of Articulated Objects 1093
Yeshasvi Tirupachuri, Silvio Traversaro, Francesco Nori, and Daniele Pucci

Telexistence and Teleoperation for Walking Humanoid Robots 1106
Mohamed Elobaid, Yue Hu, Giulio Romualdi, Stefano Dafarra, Jan Babic, and Daniele Pucci

Terrain Classification Using W-K Filter and 3D Navigation with Static Collision Avoidance 1122
J. P. Matos-Carvalho, Dário Pedro, Luís Miguel Campos, José Manuel Fonseca, and André Mora

Robot Navigation with PolyMap, a Polygon-Based Map Format 1138
Johann Dichtl, Xuan S. Le, Guillaume Lozenguez, Luc Fabresse, and Noury Bouraqadi

Identification of Motor Parameters on Coupled Joints 1153
Nuno Guedelha, Silvio Traversaro, and Daniele Pucci

Improving Human-Machine Interaction for a Powered Wheelchair Driver by Using Variable-Switches and Sensors that Reduce Wheelchair-Veer 1173
David Sanders, Martin Langner, Nils Bausch, Ya Huang, Sergey Khaustov, and Sarinova Simandjunta

De-Noising Signals Using Wavelet Transform in Internet of Underwater Things 1192
Asiya Khan, Richard Pemberton, Abdul Momen, and Daniel Bristow

Using Experts' Perceptual Skill for Dermatological Image Segmentation 1199
Qiao Li and Wanju Hou

Applications of Gaussian Process Latent Variable Models in Finance . . . 1209
Rajbir S. Nirwan and Nils Bertschinger

Channel-Wise Reconstruction-Based Anomaly Detection Framework for Multi-channel Sensor Data 1222
Mingu Kwak and Seoung Bum Kim

Designing an Artefact for Sharing and Reusing Teaching Practices in Higher Education Institutions: An Exploratory Study 1234
Nouf Almujally and Mike Joy

Intelligent Method for 3D Image Display with Semitransparent Object Representations 1243
Kohei Arai

Estimation of Average Information Content: Comparison of Impact of Contexts 1251
Michael Richter, Yuki Kyogoku, and Max Kölbl

Fuzzy Controller for Sun Tracking (Using Image Processing) 1258
Ali Hamouda, Mutaz Ababneh, Mohamed Al Zahrani, and Abdelkader Chabchoub

GMC: Grid Based Motion Clustering in Dynamic Environment 1267
Handuo Zhang, Karunasekera Hasith, Hui Zhou, and Han Wang

Rising from Systemic to Industrial Artificial Intelligence Applications (AIA) for Predictive Decision Making (PDM) - Four Examples 1281
Bernhard Heiden, Bianca Tonino-Heiden, Tanja Obermüller, Christian Loipold, and Wolfgang Wissounig

A Machine Learning Approach for Classification of Tremor - A Neurological Movement Disorder 1289
Rajesh Ranjan, Marimuthu Palaniswami, and Braj Bhushan

Author Index 1309

Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis

Muh. Ibnu Choldun R.(✉), Judhi Santoso, and Kridanto Surendro

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganecha 10, Bandung, Indonesia
ibnucholdun@poltekpos.ac.id, judhi@stei.itb.ac.id, endro@informatika.org

Abstract. One of the challenges faced in the success of Deep Neural Network (DNN) implementation is setting the values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in DNN learning performance. However, up to now, there has been no guidance on it. Determining these two numbers manually (usually through trial and error methods) to find fairly optimal arrangement is a time-consuming process. In this study, we propose the method used for determining the number of hidden layers was through the number of components formed on the principal component analysis (PCA). By using Forest Type Mapping Data Set, based on PCA analysis, it was found out that the number of hidden layers that provide the best accuracy was three. This is in accordance with the number of components formed in the principal component analysis which gave a cumulative variance of around 70%.

Keywords: Neural network · Hidden layer · PCA

1 Introduction

Neural network has been successfully applied in various fields such as computer science, finance, medicine, engineering, physics, and so forth. The main reason is that neural network could approach arbitrary function. Over the past 30 years, a number of researches showed that artificial neural network called as feedforward network with one hidden layer can approach all functions arbitrarily [1–6]. One of important aspects for designing neural network is architecture or its topology that is closely related to the capability of generalization [7, 8]. Since the era of deep learning, the use of neural network that has more than one hidden layer is a research topic that caught researchers' interest. One of the challenges in the success of deep neural network implementation is setting values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in the performance of deep neural network [7, 8]. Therefore research to determine both of them is very necessary.

© Springer Nature Switzerland AG 2020
Y. Bi et al. (Eds.): IntelliSys 2019, AISC 1038, pp. 490–500, 2020.
https://doi.org/10.1007/978-3-030-29513-4_36

According to some literatures aforementioned, determining the number of hidden layers or optimal number of neurons still has no clear guideline. Moreover, the roles and functions of both are explained minimally. Some literatures propose methods or ways to determine the number of hidden layers or neurons, but they are not generally applied due to the type of input and output data [9, 10]. Studies in this area still leave difficult research tasks [11]. Some of the methods mentioned are apparently not applicable for different types of data. The earlier researchers determined the number of hidden layers or neurons based on their experience. However, less experienced researchers even did it based on 'trial and error'. One of the methods that can be applied practically is through data mining techniques, clustering and regression [12–14]. In this study, the researcher did not explain the relationship between the number of clusters formed by feature extraction that is going to be used in machine learning process.

Although it is a difficult area of research, determining the number of hidden layers and neurons should be carried out. This is because they greatly determine the deep neural network learning performance. As a first step, a method or technique is needed to determine the number of hidden layers. Hidden layers in the neural network in stages represent increasingly complex features, while the main components of the PCA represent the information content that exists. By assuming that the complexity of features is in line with the size of the feature information, then in this study we propose a method specifically for determining the number of hidden layers in a neural network based on the number of main components formed on principal component analysis (PCA).

2 Related Work

Studies compared the use of one or two hidden layers focused on univariate and multivariate functions [4–6, 15]. Thomas [4, 5] got different result that the use of two hidden layers applied to predictive functions showed better performance. Guliyev and Ismailov [6] concluded that the use of one hidden layer was less capable of approaching the multivariate function, so the use of two hidden layers showed better performance.

To determine the number of hidden layers and neurons at once, the method is divided into two: automatically and manually [15]. Automatic determination uses models, but it also can be done without models. Manual determination uses certain formulas, some are done by trial and error. Automatic determination without a model consists of grid search (GS) and random search (RS). GS is generally used to optimize DNN parameters only if the number is very low. First, users choose a range of values to explore. Then, DNN is set for each specification along with other parameters. Usually, GS is carried out for logarithmic scale steps that the best combination is estimated progressively to search [16]. The obvious disadvantage of GS is its time complexity – parameter k takes different values (by assuming that n is the same for all parameters), the complexity grows exponentially at level $O(n^k)$. Random search (RS) is an alternative to GS. It integrates faster into parametricization. RS is not adaptive that it is not dynamically updated during the experiment (the solutions found did not do anything to

the search). Furthermore, there is also hybrid algorithm that combines RS with other techniques to improve its performance, for instance, manual updates provided by experts [16, 17]. The technique that has been done turns out to be not generally accepted.

Model-based algorithm builds replacement models for hyper-parameter space. Then, hyper-parameters are set by using this model. Most of these techniques use Bayesian regression model that turns this problem into a trade-off between exploration (traversing unknown areas of space that classification of the performance is unknown), and exploitation (analyzing hyper-parameters that are likely to perform well and closed to "good" hyper-parameters that have been investigated). One of model-based algorithms is the use Particle Swarm Optimization (PSO) [3, 15]. PSO is an optimization technique done by continuously calculating potential solutions by using a quality reference. This algorithm optimizes problems by moving particles or prospective solutions for problems through certain functions for the position and velocity of the particles. The particles' movement is influenced by the best solution for the particles and the best solution in general that is obtained from other particles. This set of particles is called as a swarm that in the end it will move towards the best solution.

Although some researchers have proposed methods to calculate the number of hidden layers and neurons at once, but the methods were rarely used by other researchers, because they are difficult to apply or do not match the type of dataset used, therefore some researchers decided to use trial and error methods [8]. Therefore, a method or technique is needed to calculate the number of hidden layers or hidden neurons that can be generally accepted.

3 Neural Network

Architecture of back propagation neural network consists of several layers: input layer, hidden layer and output layer. Each neuron in each layer is connected to the next layer. The neurons in the input layer are connected to neurons in the hidden layer through synapses called as weight. In addition, hidden layers of neurons connected through synapses with neurons in the output layer. One of the neural network architectures is called as Multilayer Perceptron (MLP) [18]. MLP consists of: input layers where the neurons are fully connected to each neuron in the first hidden layer, each neuron in the hidden layer is connected to each neuron in the next hidden layer, and each neuron in the last hidden layer is fully connected with each neuron in the output layer. MLP, or often called as feedforward deep network, is a classic example of deep learning model.

Figure 1 is an example of a neural network with an MLP architecture consisting of input layers, two hidden layers, and an output layer. The closer the hidden layer to the output layer the better it identifies the complex features. For instance, MLP architecture that uses three hidden layers for face identification: the first hidden layer is used to identify the geometric shape that can distinguish the facial parts, the second hidden layer is used to identify eyes, nose, mouth, etc., and the third hidden layer is used to identify the face.

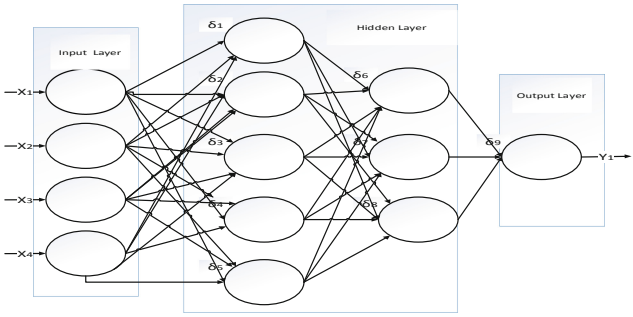


Fig. 1. Neural network

The training process for neural network is done by finding the value of each connection weight, which is done iteratively. Therefore, the output can correctly predict the target value of the training data examples. Backpropagation is an algorithm that is widely used for neural network learning, especially in MLP architecture. Each learning algorithm developed is based on inductive bias, a collection of assumptions that underlie the selection criteria of learning algorithm model. According to Mitchell [19], a learner's inductive bias is a set of additional assumptions that is sufficient to justify an inductive conclusion as a deductive conclusion. An example of an inductive bias that can be applied to MLP learning is to obtain the desired accuracy that the correct architecture must be chosen (the number of hidden layers and neurons for each hidden layer).

After obtaining the output from the perceptron based on the input given, at the error stage the perceptron calculation will be evaluated "whether the perceptron output matches the expected output or not". To calculate the difference between actual output and desired output can be used various functions of errors such as Squared Error (Eq. 1), Root Mean Squared Error (RMSE), or Cross Entropy (Eq. 2).

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{1}$$

for t = actual output, y = desired output, and N = number of data, then squared errors from t and y can be calculated using Eq. 1. The use of squared error often has problems if it is implemented in the case of classification with logistical output. Unlike the squared error, cross entropy does not experience problems in this case. In cross entropy, the greater the prediction error of the neural network, the greater the update value for the weight. The error cross entropy function can be formulated as follows:

$$E = -\frac{1}{N}\sum_{i=1}^{N} a_i \log(y_i) + (1 - a_i)\log(1 - y_i) \tag{2}$$

where y_i is the output of the neuron to i, a_i is the *actual output* to i, and N the total number of datasets.

4 Principal Component Analysis

Principal Component Analysis (PCA) is a multivariate analysis that transforms correlated origin variables into new variables that do not correlate with each other by reducing a number of the variables mentioned, so that the variables have smaller dimension but can explain most of the diversity of the origin variables [20]. The PCA steps are as follows.

1. There is a data matrix ($X = [x_1, x_2, \ldots, x_N]$), N is the total number of samples and xi represents the sample to i.
2. Calculate the average of all samples, as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3}$$

3. Subtract all data samples with the average, as follows:

$$D = \{d_1, d_2, \ldots, d_N\} = \sum_{i=1}^{N} x_i - \mu \tag{4}$$

4. Calculate the covariance matrix, as follows:

$$\sum = \frac{1}{N-1} D \times D^T \tag{5}$$

5. Calculate the eigenvectors V and eigenvalues λ from the covariance matrix (Σ)
6. Sort the eigenvectors based on the eigenvalues that correspond
7. Select eigenvectors which have the largest eigenvalues $W = \{v_1, \ldots, v_k\}$. The selected Eigenvectors (W) represent the PCA projection space.
8. All samples are projected on the dimensional space lower than PCA (W), as follows:

$$Y = W^T D \tag{6}$$

The number of the main components formed is equal to the number of original variables. The selection of the main components used was based on the eigenvalue value that the main components will be used if the eigenvalue is greater than one. However, the reduction (simplification) of the dimension can also be done by finding the percentage criteria of the data diversity explained by the first few main components. For example, the number of the main components used was if the cumulative variance of the components was at least 70%.

5 Research Method

5.1 Research Framework

The more hidden layers in a neural network the more they can represent the increasingly complex features [19]. Deep architecture in neural network can detect features

arranged in each layer. The lower layers (that are closer to the input layer) could detect simpler features, while the layers closer to the ouput layer could detect more complex features [21]. Increasingly complex feature represents higher information content received. The high information in PCA is represented in the main component that has high variance. In other words, increasingly complex feature in hidden layers of the neural network is in accordance with the main components that have higher variance. As a result, the number of hidden layers in the required neural network will be in accordance with the number of the main components in the principal component analysis. Therefore, in this study, the number of hidden layers in the neural network could be determined based on the number of the main components formed through PCA.

5.2 Preparing the Dataset

The data set used in this study was Forest Type Mapping Data Set that was converted into numeric data. The data sets could be downloaded on UCI Machine Learning Repository website[1]. The number of input attribute was 27 and the output category was 4 classes: 's' ('Sugi' forest), 'h' ('Hinoki' forest), 'd' ('Mixed deciduous' forest), 'o' ('Other' non-forest land). The number of dataset used was 325 records, 70% of which was used as training data and the remaining is for testing. The output function used in the data set was classification.

5.3 Standardization and Normalization

Standardization was done by deleting data that had empty attributes and also eliminating extreme values. Normalization was done with the aim of getting data with attributes that are scaled accordingly. The normalization done was Min-Maks with a value between 0 and 1. Min-Maks normalization formula is as follows:

$$v' = \frac{v - min_A}{max_A_min_A}(new_max_A - new_min_A) + new_min_A \tag{7}$$

v′ = new data, v = old data, $\min_A$ = the smallest value of an attribute, $\max_A$ = the biggest value of an attribute, new_min_A = the smallest value of a new attribute (=0), new_max_A = the biggest value of a new attribute (=1).

5.4 Determining Neural Network Topology

Changing network topology was done by varying the number of hidden layers by referring to the number of the components formed in PCA. The number of hidden neurons in each hidden layer is changed by trial and error.

[1] UCI Machine Learning Repository Homepage, https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping, last accessed 2019/01/10.

5.5 Training and Testing

Training was carried out with 70% data from the dataset, while testing was carried out with 30% data from the dataset. The training and testing process would stop when the iteration had reached 500 times.

6 Result and Discussion

6.1 Principal Component Anaysis Result

By analyzing the main components of the input attributes on the dataset that was going to be used for neural network training, the following results were obtained (see Table 1):

Table 1. Principal component analysis processing results

Component	Total	% of variance	Cumulative %
1	10.146	37.578	37.578
2	5.014	18.571	56.149
3	3.772	13.971	70.120
4	1.980	7.333	77.453
5	1.501	5.560	83.013
6	1.328	4.919	87.932
7	.875	3.240	91.172
8	.621	2.301	93.473
9	.511	1.893	95.366
10	.368	1.365	96.731
11	.318	1.179	97.910
12	.176	.652	98.561
13	.112	.416	98.977
14	.069	.256	99.233
15	.066	.244	99.477
16	.036	.132	99.609
17	.030	.112	99.721
18	.028	.104	99.826
19	.022	.081	99.907
20	.011	.042	99.948
21	.007	.026	99.974
22	.003	.013	99.987
23	.002	.009	99.996
24	.001	.003	99.999
25	.000	.001	100.000
26	8.995E−5	.000	100.000
27	2.354E−5	8.718E−5	100.000

If a minimum total diversity of 70% is taken, the number of hidden layers is three, if 80% of the total diversity of hidden layers is five, and if the total diversity is 90%, the number of hidden layers is seven. The number of components that will be used as the basis for determining the number of hidden layers in a neural network is chosen from one to four.

6.2 Neural Network Training Result

Neural network training was done by using multilayer perceptron architecture with the number of hidden layers chosen from one to four, while the number of neurons for each hidden layer was determined by the researcher. With the number of hidden layers selected and the number of neurons in each hidden layer determined, the neural network architecture to be trained can be seen in the following Table 2.

Table 2. The architecture of neural network

Architecture	Number of hidden layer	Number of neuron
I	1	36
II	1	100
III	2	15, 8
IV	3	15, 10, 5
V	3	20, 10, 5
VI	4	20, 15, 10, 5

The researcher gave an example as an explanation of the architecture in the above table that if architecture IV is a multilayer perceptron with three hidden layers, the first hidden layer has 15 neurons, the second hidden layer has 10 neurons, and the third hidden layer has 5 neurons. The training and testing were carried out on each of the above architectures, and the following results were obtained (see Table 3):

Table 3. The accuracy of neural network

Architecture	Accuracy
I	90.45%
II	89.17%
III	87.90%
IV	92.36%
V	92.36%
VI	27.39%

For architecture IV, the accuracy obtained was 92.36% which means that from the test data used, about 7.64% of the output was classified as incorrect.

6.3 Discussion

The experiments carried out used several different network topologies, both in terms of the number of hidden layers and neurons in each hidden layer, each of which has a certain level of accuracy by using the Forest Type Mapping dataset. From the experiments performed, it was shown that by using one hidden layer by varying the number of neurons 36(I) and 100(II), the accuracy were 90.44% and 89.18%, respectively. In the architecture of one hidden layer, the addition of the number of neurons did not increase the accuracy. For the use of two hidden layers, architecture III had an accuracy rate of 87.89% that also did not improve the accuracy compared to the use of one hidden layer. By using three hidden layers, architecture IV and V had an accuracy rate of 92.36%, which means that the use of three hidden layers was more accurate than the use of one or two hidden layers. The use of four hidden layers, in architecture VI, decreased the accuracy.

The experiments showed that the use of three hidden layers provided the best level of accuracy. If it is associated with PCA result, where it is assumed that the number of hidden layers selected was in accordance with the number of the components selected in PCA, the cumulative variance of the selected component was around 70%. This result indicated that the cumulative variance classification needed to determine the number of components does not have to be close to 100%. The classification of a group of different outputs, but still within a certain range, will be grouped in the same classification. It would be different if the regression objective function of each different output was considered as a different value.

7 Conclusion and Future Work

The right number of hidden layers chosen in neural network would provide high learning accuracy. Increasing the number of hidden layers does not guarantee an increase in their accuracy. The determination of the number of hidden layers could be done by using the number of components produced on the PCA principal component analysis by considering the cumulative variance. For the classification function, the number of cumulative variances could be done with a cut off value of 70%. For future works, the researchers need to consider the way to determine the number of neurons in each hidden layer besides determining the number of hidden layers. In addition, determining the number of hidden layers by using the number of PCA components also needs to be applied for the regression objective function. In the research that has been done, each neural network topology chosen is only conducted one-time experiments, in future studies for each topology the experiment repetition must be performed for example at least ten times, then each topology is compared. Experiments also need to be done for different types of datasets.

References

1. Madhiarasan, M., Deepa, S.N.: A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting. Appl. Intell. **44**(4), 878–893 (2016). https://doi.org/10.1007/s10489-015-0737-z
2. Madhiarasan, M., Deepa, S.N.: Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting. Artif. Intell. Rev. **48**(4), 449–471 (2017). https://doi.org/10.1007/s10462-016-9506-6
3. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., Benhaddou, D.: Parameters optimization of deep learning models using particle swarm optimization. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1285–1290 (2017). https://doi.org/10.1109/IWCMC.2017.7986470
4. Thomas, A.J., Petridis, M., Walters, S.D., Gheytassi, S.M., Morgan, R.E.: Eng. Appl. Neural Netw. **744**, 279–290 (2017). https://doi.org/10.1007/978-3-319-65172-9
5. Thomas, A.J., Walters, S.D., Gheytassi, S.M., Morgan, R.E., Petridis, M.: On the optimal node ratio between hidden layers: a probabilistic study. Int. J. Mach. Learn. Comput. **6**(5), 241–247 (2016). https://doi.org/10.18178/ijmlc.2016.6.5.605
6. Guliyev, N.J., Ismailov, V.E.: On the approximation by single hidden layer feedforward neural networks with fixed weights. Neural Netw. **98**, 296–304 (2018). https://doi.org/10.1016/j.neunet.2017.12.007
7. Nitta, T.: Resolution of singularities introduced by hierarchical structure in deep neural networks. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2282–2293 (2017). https://doi.org/10.1109/TNNLS.2016.2580741
8. Koutsoukas, A., Monaghan, K.J., Li, X., Huan, J.: Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. J. Cheminform. **9**(1), 1–13 (2017). https://doi.org/10.1186/s13321-017-0226-y
9. Bunjongjit, S., Ngaopitakkul, A., Pothisarn, C., Jettanasen, C.: Improvement to reduce training time of back-propagation neural networks for discrimination between external short circuit and internal winding fault. In: International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, pp. 614–618 (2014)
10. Chhachhiya, D., Sharma, A., Gupta, M.: Designing optimal architecture of neural network with particle swarm optimization techniques specifically for educational dataset. In: 7th International Conference on Cloud Computing, Data Science and Engineering - Confluence, Noida, pp. 52–57 (2017)
11. Lee, S., Ha, J., Zokhirova, M., Moon, H., Lee, J.: Background information of deep learning for structural engineering. Arch. Comput. Methods Eng. **25**(1), 121–129 (2018). https://doi.org/10.1007/s11831-017-9237-0
12. Tej, M.L., Holban, S.: Comparative study of clustering distance measures to determine neural network architectures. In: IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 189–194 (2018)
13. Tej, M.L., Holban, S.: Determining optimal neural network architecture using regression methods. In: International Conference on Development and Application Systems (DAS), pp. 180–189 (2018)
14. Tej, M.L.: Neural network architecture through data mining techniques. In: Proceedings of the IIER International Conference, Bucharest, Romania (2017)

15. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proceedings of the 2017 on Genetic and Evolutionary Computation Conference - GECCO 2017, vol. 8, pp. 481–488 (2017). https://doi.org/10.1145/3071178.3071208
16. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
17. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: ICML (2007)
18. Bengio, Y., Goodfellow, I., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
19. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
20. Jollife, I.T.: Principal Component Analysis. Springer, New York (2002)
21. Di, W., Bhardwaj, A., Wei, J.: Deep Learning Essentials. Packt Publishing, Birmingham (2018)

Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis

Muh. Ibnu Choldun R.(✉), Judhi Santoso, and Kridanto Surendro

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganecha 10, Bandung, Indonesia
ibnucholdun@poltekpos.ac.id, judhi@stei.itb.ac.id, endro@informatika.org

Abstract. One of the challenges faced in the success of Deep Neural Network (DNN) implementation is setting the values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in DNN learning performance. However, up to now, there has been no guidance on it. Determining these two numbers manually (usually through trial and error methods) to find fairly optimal arrangement is a time-consuming process. In this study, we propose the method used for determining the number of hidden layers was through the number of components formed on the principal component analysis (PCA). By using Forest Type Mapping Data Set, based on PCA analysis, it was found out that the number of hidden layers that provide the best accuracy was three. This is in accordance with the number of components formed in the principal component analysis which gave a cumulative variance of around 70%.

Keywords: Neural network · Hidden layer · PCA

1 Introduction

Neural network has been successfully applied in various fields such as computer science, finance, medicine, engineering, physics, and so forth. The main reason is that neural network could approach arbitrary function. Over the past 30 years, a number of researches showed that artificial neural network called as feedforward network with one hidden layer can approach all functions arbitrarily [1–6]. One of important aspects for designing neural network is architecture or its topology that is closely related to the capability of generalization [7, 8]. Since the era of deep learning, the use of neural network that has more than one hidden layer is a research topic that caught researchers' interest. One of the challenges in the success of deep neural network implementation is setting values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in the performance of deep neural network [7, 8]. Therefore research to determine both of them is very necessary.

© Springer Nature Switzerland AG 2020
Y. Bi et al. (Eds.): IntelliSys 2019, AISC 1038, pp. 490–500, 2020.
https://doi.org/10.1007/978-3-030-29513-4_36

According to some literatures aforementioned, determining the number of hidden layers or optimal number of neurons still has no clear guideline. Moreover, the roles and functions of both are explained minimally. Some literatures propose methods or ways to determine the number of hidden layers or neurons, but they are not generally applied due to the type of input and output data [9, 10]. Studies in this area still leave difficult research tasks [11]. Some of the methods mentioned are apparently not applicable for different types of data. The earlier researchers determined the number of hidden layers or neurons based on their experience. However, less experienced researchers even did it based on 'trial and error'. One of the methods that can be applied practically is through data mining techniques, clustering and regression [12–14]. In this study, the researcher did not explain the relationship between the number of clusters formed by feature extraction that is going to be used in machine learning process.

Although it is a difficult area of research, determining the number of hidden layers and neurons should be carried out. This is because they greatly determine the deep neural network learning performance. As a first step, a method or technique is needed to determine the number of hidden layers. Hidden layers in the neural network in stages represent increasingly complex features, while the main components of the PCA represent the information content that exists. By assuming that the complexity of features is in line with the size of the feature information, then in this study we propose a method specifically for determining the number of hidden layers in a neural network based on the number of main components formed on principal component analysis (PCA).

2 Related Work

Studies compared the use of one or two hidden layers focused on univariate and multivariate functions [4–6, 15]. Thomas [4, 5] got different result that the use of two hidden layers applied to predictive functions showed better performance. Guliyev and Ismailov [6] concluded that the use of one hidden layer was less capable of approaching the multivariate function, so the use of two hidden layers showed better performance.

To determine the number of hidden layers and neurons at once, the method is divided into two: automatically and manually [15]. Automatic determination uses models, but it also can be done without models. Manual determination uses certain formulas, some are done by trial and error. Automatic determination without a model consists of grid search (GS) and random search (RS). GS is generally used to optimize DNN parameters only if the number is very low. First, users choose a range of values to explore. Then, DNN is set for each specification along with other parameters. Usually, GS is carried out for logarithmic scale steps that the best combination is estimated progressively to search [16]. The obvious disadvantage of GS is its time complexity – parameter k takes different values (by assuming that n is the same for all parameters), the complexity grows exponentially at level $O(n^k)$. Random search (RS) is an alternative to GS. It integrates faster into parametricization. RS is not adaptive that it is not dynamically updated during the experiment (the solutions found did not do anything to

the search). Furthermore, there is also hybrid algorithm that combines RS with other techniques to improve its performance, for instance, manual updates provided by experts [16, 17]. The technique that has been done turns out to be not generally accepted.

Model-based algorithm builds replacement models for hyper-parameter space. Then, hyper-parameters are set by using this model. Most of these techniques use Bayesian regression model that turns this problem into a trade-off between exploration (traversing unknown areas of space that classification of the performance is unknown), and exploitation (analyzing hyper-parameters that are likely to perform well and closed to "good" hyper-parameters that have been investigated). One of model-based algorithms is the use Particle Swarm Optimization (PSO) [3, 15]. PSO is an optimization technique done by continuously calculating potential solutions by using a quality reference. This algorithm optimizes problems by moving particles or prospective solutions for problems through certain functions for the position and velocity of the particles. The particles' movement is influenced by the best solution for the particles and the best solution in general that is obtained from other particles. This set of particles is called as a swarm that in the end it will move towards the best solution.

Although some researchers have proposed methods to calculate the number of hidden layers and neurons at once, but the methods were rarely used by other researchers, because they are difficult to apply or do not match the type of dataset used, therefore some researchers decided to use trial and error methods [8]. Therefore, a method or technique is needed to calculate the number of hidden layers or hidden neurons that can be generally accepted.

3 Neural Network

Architecture of back propagation neural network consists of several layers: input layer, hidden layer and output layer. Each neuron in each layer is connected to the next layer. The neurons in the input layer are connected to neurons in the hidden layer through synapses called as weight. In addition, hidden layers of neurons connected through synapses with neurons in the output layer. One of the neural network architectures is called as Multilayer Perceptron (MLP) [18]. MLP consists of: input layers where the neurons are fully connected to each neuron in the first hidden layer, each neuron in the hidden layer is connected to each neuron in the next hidden layer, and each neuron in the last hidden layer is fully connected with each neuron in the output layer. MLP, or often called as feedforward deep network, is a classic example of deep learning model.

Figure 1 is an example of a neural network with an MLP architecture consisting of input layers, two hidden layers, and an output layer. The closer the hidden layer to the output layer the better it identifies the complex features. For instance, MLP architecture that uses three hidden layers for face identification: the first hidden layer is used to identify the geometric shape that can distinguish the facial parts, the second hidden layer is used to identify eyes, nose, mouth, etc., and the third hidden layer is used to identify the face.

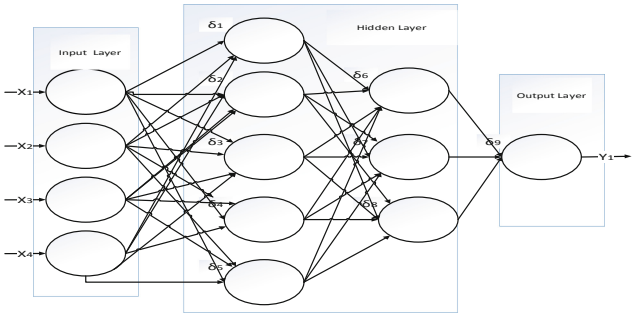


Fig. 1. Neural network

The training process for neural network is done by finding the value of each connection weight, which is done iteratively. Therefore, the output can correctly predict the target value of the training data examples. Backpropagation is an algorithm that is widely used for neural network learning, especially in MLP architecture. Each learning algorithm developed is based on inductive bias, a collection of assumptions that underlie the selection criteria of learning algorithm model. According to Mitchell [19], a learner's inductive bias is a set of additional assumptions that is sufficient to justify an inductive conclusion as a deductive conclusion. An example of an inductive bias that can be applied to MLP learning is to obtain the desired accuracy that the correct architecture must be chosen (the number of hidden layers and neurons for each hidden layer).

After obtaining the output from the perceptron based on the input given, at the error stage the perceptron calculation will be evaluated "whether the perceptron output matches the expected output or not". To calculate the difference between actual output and desired output can be used various functions of errors such as Squared Error (Eq. 1), Root Mean Squared Error (RMSE), or Cross Entropy (Eq. 2).

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{1}$$

for t = actual output, y = desired output, and N = number of data, then squared errors from t and y can be calculated using Eq. 1. The use of squared error often has problems if it is implemented in the case of classification with logistical output. Unlike the squared error, cross entropy does not experience problems in this case. In cross entropy, the greater the prediction error of the neural network, the greater the update value for the weight. The error cross entropy function can be formulated as follows:

$$E = -\frac{1}{N}\sum_{i=1}^{N} a_i \log(y_i) + (1 - a_i)\log(1 - y_i) \tag{2}$$

where y_i is the output of the neuron to i, a_i is the *actual output* to i, and N the total number of datasets.

4 Principal Component Analysis

Principal Component Analysis (PCA) is a multivariate analysis that transforms correlated origin variables into new variables that do not correlate with each other by reducing a number of the variables mentioned, so that the variables have smaller dimension but can explain most of the diversity of the origin variables [20]. The PCA steps are as follows.

1. There is a data matrix ($X = [x_1, x_2, \ldots, x_N]$), N is the total number of samples and xi represents the sample to i.
2. Calculate the average of all samples, as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3}$$

3. Subtract all data samples with the average, as follows:

$$D = \{d_1, d_2, \ldots, d_N\} = \sum_{i=1}^{N} x_i - \mu \tag{4}$$

4. Calculate the covariance matrix, as follows:

$$\sum = \frac{1}{N-1} D \times D^T \tag{5}$$

5. Calculate the eigenvectors V and eigenvalues λ from the covariance matrix (Σ)
6. Sort the eigenvectors based on the eigenvalues that correspond
7. Select eigenvectors which have the largest eigenvalues $W = \{v_1, \ldots, v_k\}$. The selected Eigenvectors (W) represent the PCA projection space.
8. All samples are projected on the dimensional space lower than PCA (W), as follows:

$$Y = W^T D \tag{6}$$

The number of the main components formed is equal to the number of original variables. The selection of the main components used was based on the eigenvalue value that the main components will be used if the eigenvalue is greater than one. However, the reduction (simplification) of the dimension can also be done by finding the percentage criteria of the data diversity explained by the first few main components. For example, the number of the main components used was if the cumulative variance of the components was at least 70%.

5 Research Method

5.1 Research Framework

The more hidden layers in a neural network the more they can represent the increasingly complex features [19]. Deep architecture in neural network can detect features

arranged in each layer. The lower layers (that are closer to the input layer) could detect simpler features, while the layers closer to the ouput layer could detect more complex features [21]. Increasingly complex feature represents higher information content received. The high information in PCA is represented in the main component that has high variance. In other words, increasingly complex feature in hidden layers of the neural network is in accordance with the main components that have higher variance. As a result, the number of hidden layers in the required neural network will be in accordance with the number of the main components in the principal component analysis. Therefore, in this study, the number of hidden layers in the neural network could be determined based on the number of the main components formed through PCA.

5.2 Preparing the Dataset

The data set used in this study was Forest Type Mapping Data Set that was converted into numeric data. The data sets could be downloaded on UCI Machine Learning Repository website[1]. The number of input attribute was 27 and the output category was 4 classes: 's' ('Sugi' forest), 'h' ('Hinoki' forest), 'd' ('Mixed deciduous' forest), 'o' ('Other' non-forest land). The number of dataset used was 325 records, 70% of which was used as training data and the remaining is for testing. The output function used in the data set was classification.

5.3 Standardization and Normalization

Standardization was done by deleting data that had empty attributes and also eliminating extreme values. Normalization was done with the aim of getting data with attributes that are scaled accordingly. The normalization done was Min-Maks with a value between 0 and 1. Min-Maks normalization formula is as follows:

$$v' = \frac{v - min_A}{max_A_min_A}(new_max_A - new_min_A) + new_min_A \tag{7}$$

v′ = new data, v = old data, $\min_A$ = the smallest value of an attribute, $\max_A$ = the biggest value of an attribute, new_min_A = the smallest value of a new attribute (=0), new_max_A = the biggest value of a new attribute (=1).

5.4 Determining Neural Network Topology

Changing network topology was done by varying the number of hidden layers by referring to the number of the components formed in PCA. The number of hidden neurons in each hidden layer is changed by trial and error.

[1] UCI Machine Learning Repository Homepage, https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping, last accessed 2019/01/10.

5.5 Training and Testing

Training was carried out with 70% data from the dataset, while testing was carried out with 30% data from the dataset. The training and testing process would stop when the iteration had reached 500 times.

6 Result and Discussion

6.1 Principal Component Anaysis Result

By analyzing the main components of the input attributes on the dataset that was going to be used for neural network training, the following results were obtained (see Table 1):

Table 1. Principal component analysis processing results

Component	Total	% of variance	Cumulative %
1	10.146	37.578	37.578
2	5.014	18.571	56.149
3	3.772	13.971	70.120
4	1.980	7.333	77.453
5	1.501	5.560	83.013
6	1.328	4.919	87.932
7	.875	3.240	91.172
8	.621	2.301	93.473
9	.511	1.893	95.366
10	.368	1.365	96.731
11	.318	1.179	97.910
12	.176	.652	98.561
13	.112	.416	98.977
14	.069	.256	99.233
15	.066	.244	99.477
16	.036	.132	99.609
17	.030	.112	99.721
18	.028	.104	99.826
19	.022	.081	99.907
20	.011	.042	99.948
21	.007	.026	99.974
22	.003	.013	99.987
23	.002	.009	99.996
24	.001	.003	99.999
25	.000	.001	100.000
26	8.995E−5	.000	100.000
27	2.354E−5	8.718E−5	100.000

If a minimum total diversity of 70% is taken, the number of hidden layers is three, if 80% of the total diversity of hidden layers is five, and if the total diversity is 90%, the number of hidden layers is seven. The number of components that will be used as the basis for determining the number of hidden layers in a neural network is chosen from one to four.

6.2 Neural Network Training Result

Neural network training was done by using multilayer perceptron architecture with the number of hidden layers chosen from one to four, while the number of neurons for each hidden layer was determined by the researcher. With the number of hidden layers selected and the number of neurons in each hidden layer determined, the neural network architecture to be trained can be seen in the following Table 2.

Table 2. The architecture of neural network

Architecture	Number of hidden layer	Number of neuron
I	1	36
II	1	100
III	2	15, 8
IV	3	15, 10, 5
V	3	20, 10, 5
VI	4	20, 15, 10, 5

The researcher gave an example as an explanation of the architecture in the above table that if architecture IV is a multilayer perceptron with three hidden layers, the first hidden layer has 15 neurons, the second hidden layer has 10 neurons, and the third hidden layer has 5 neurons. The training and testing were carried out on each of the above architectures, and the following results were obtained (see Table 3):

Table 3. The accuracy of neural network

Architecture	Accuracy
I	90.45%
II	89.17%
III	87.90%
IV	92.36%
V	92.36%
VI	27.39%

For architecture IV, the accuracy obtained was 92.36% which means that from the test data used, about 7.64% of the output was classified as incorrect.

6.3 Discussion

The experiments carried out used several different network topologies, both in terms of the number of hidden layers and neurons in each hidden layer, each of which has a certain level of accuracy by using the Forest Type Mapping dataset. From the experiments performed, it was shown that by using one hidden layer by varying the number of neurons 36(I) and 100(II), the accuracy were 90.44% and 89.18%, respectively. In the architecture of one hidden layer, the addition of the number of neurons did not increase the accuracy. For the use of two hidden layers, architecture III had an accuracy rate of 87.89% that also did not improve the accuracy compared to the use of one hidden layer. By using three hidden layers, architecture IV and V had an accuracy rate of 92.36%, which means that the use of three hidden layers was more accurate than the use of one or two hidden layers. The use of four hidden layers, in architecture VI, decreased the accuracy.

The experiments showed that the use of three hidden layers provided the best level of accuracy. If it is associated with PCA result, where it is assumed that the number of hidden layers selected was in accordance with the number of the components selected in PCA, the cumulative variance of the selected component was around 70%. This result indicated that the cumulative variance classification needed to determine the number of components does not have to be close to 100%. The classification of a group of different outputs, but still within a certain range, will be grouped in the same classification. It would be different if the regression objective function of each different output was considered as a different value.

7 Conclusion and Future Work

The right number of hidden layers chosen in neural network would provide high learning accuracy. Increasing the number of hidden layers does not guarantee an increase in their accuracy. The determination of the number of hidden layers could be done by using the number of components produced on the PCA principal component analysis by considering the cumulative variance. For the classification function, the number of cumulative variances could be done with a cut off value of 70%. For future works, the researchers need to consider the way to determine the number of neurons in each hidden layer besides determining the number of hidden layers. In addition, determining the number of hidden layers by using the number of PCA components also needs to be applied for the regression objective function. In the research that has been done, each neural network topology chosen is only conducted one-time experiments, in future studies for each topology the experiment repetition must be performed for example at least ten times, then each topology is compared. Experiments also need to be done for different types of datasets.

References

1. Madhiarasan, M., Deepa, S.N.: A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting. Appl. Intell. **44**(4), 878–893 (2016). https://doi.org/10.1007/s10489-015-0737-z
2. Madhiarasan, M., Deepa, S.N.: Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting. Artif. Intell. Rev. **48**(4), 449–471 (2017). https://doi.org/10.1007/s10462-016-9506-6
3. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., Benhaddou, D.: Parameters optimization of deep learning models using particle swarm optimization. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1285–1290 (2017). https://doi.org/10.1109/IWCMC.2017.7986470
4. Thomas, A.J., Petridis, M., Walters, S.D., Gheytassi, S.M., Morgan, R.E.: Eng. Appl. Neural Netw. **744**, 279–290 (2017). https://doi.org/10.1007/978-3-319-65172-9
5. Thomas, A.J., Walters, S.D., Gheytassi, S.M., Morgan, R.E., Petridis, M.: On the optimal node ratio between hidden layers: a probabilistic study. Int. J. Mach. Learn. Comput. **6**(5), 241–247 (2016). https://doi.org/10.18178/ijmlc.2016.6.5.605
6. Guliyev, N.J., Ismailov, V.E.: On the approximation by single hidden layer feedforward neural networks with fixed weights. Neural Netw. **98**, 296–304 (2018). https://doi.org/10.1016/j.neunet.2017.12.007
7. Nitta, T.: Resolution of singularities introduced by hierarchical structure in deep neural networks. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2282–2293 (2017). https://doi.org/10.1109/TNNLS.2016.2580741
8. Koutsoukas, A., Monaghan, K.J., Li, X., Huan, J.: Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. J. Cheminform. **9**(1), 1–13 (2017). https://doi.org/10.1186/s13321-017-0226-y
9. Bunjongjit, S., Ngaopitakkul, A., Pothisarn, C., Jettanasen, C.: Improvement to reduce training time of back-propagation neural networks for discrimination between external short circuit and internal winding fault. In: International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, pp. 614–618 (2014)
10. Chhachhiya, D., Sharma, A., Gupta, M.: Designing optimal architecture of neural network with particle swarm optimization techniques specifically for educational dataset. In: 7th International Conference on Cloud Computing, Data Science and Engineering - Confluence, Noida, pp. 52–57 (2017)
11. Lee, S., Ha, J., Zokhirova, M., Moon, H., Lee, J.: Background information of deep learning for structural engineering. Arch. Comput. Methods Eng. **25**(1), 121–129 (2018). https://doi.org/10.1007/s11831-017-9237-0
12. Tej, M.L., Holban, S.: Comparative study of clustering distance measures to determine neural network architectures. In: IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 189–194 (2018)
13. Tej, M.L., Holban, S.: Determining optimal neural network architecture using regression methods. In: International Conference on Development and Application Systems (DAS), pp. 180–189 (2018)
14. Tej, M.L.: Neural network architecture through data mining techniques. In: Proceedings of the IIER International Conference, Bucharest, Romania (2017)

15. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proceedings of the 2017 on Genetic and Evolutionary Computation Conference - GECCO 2017, vol. 8, pp. 481–488 (2017). https://doi.org/10.1145/3071178.3071208
16. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
17. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: ICML (2007)
18. Bengio, Y., Goodfellow, I., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
19. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
20. Jollife, I.T.: Principal Component Analysis. Springer, New York (2002)
21. Di, W., Bhardwaj, A., Wei, J.: Deep Learning Essentials. Packt Publishing, Birmingham (2018)

This author profile is generated by Scopus. Learn more

Rachmatullah, Muhammad Ibnu Choldun

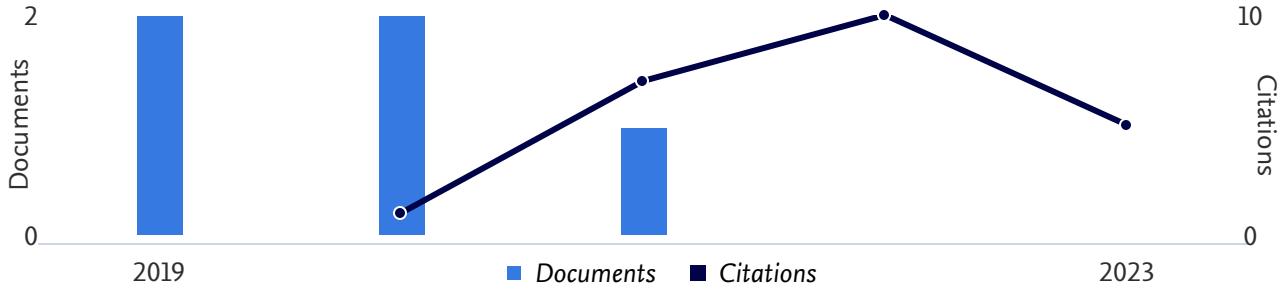
Institut Teknologi Bandung, Bandung, Indonesia 57224844807

https://orcid.org/0000-0003-4071-887X

23	5	3
Citations by ***20*** documents	Documents	*h*-index View *h*-graph

Set alert Edit profile More

Document & citation trends



Scopus Preview

Scopus Preview users can only view a limited set of features. Check your institution's access to view all documents and features.

Check access

Documents 1 Preprint 4 Co-Authors Topics Beta 0 Awarded Grants

Documents (5) Cited by (20)

Note:

Scopus Preview users can only view an author's last 10 documents, while most other features are disabled. Do you have access through your institution? Check your institution's access to view all documents and features.

5 documents

Export all Add all to list Sort by Date (...

Article • *Open access*

Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction

2 Citations

Rachmatullah, M.I.C., Santoso, J., Surendro, K.

PeerJ Computer Science, 2021, 7, pp. 1–19

Show abstract Related documents

Article • *Open access*

A novel approach in determining neural networks architecture to classify data with large number of attributes

3 Citations

Rachmatullah, M.I.C., Santoso, J., Surendro, K.

IEEE Access, 2020, 8, pp. 204728–204743

Show abstract Related documents

Conference Paper

Determining the number of hidden layers in neural network by using principal component analysis

13 Citations

Ibnu Choldun R., M., Santoso, J., Surendro, K.

Advances in Intelligent Systems and Computing, 2020, 1038, pp. 490–500

Show abstract Related documents

Conference Paper

Determining the Neural Network Topology from the Viewpoint of Kuhn's Philosophy and Popper's Philosophy

0 Citations

Rachmatullah, M.I.C., Surendro, K., Santoso, J., Mahayana, D.

Proceeding - 2019 International Conference of Artificial Intelligence and Information Technology, ICAIIT 2019, 2019, pp. 115–118, 8834664

Show abstract Related documents

Conference Paper

Determining the neural network topology: A review

5 Citations

Muh Ibnu Choldun, R., Santoso, J., Surendro, K.

ACM International Conference Proceeding Series, 2019, Part F147956, pp. 357–362

Show abstract Related documents

Back to top

View list in search results format

View references

Set document alert

About Scopus

What is Scopus

Content coverage

Scopus blog

Scopus API

Privacy matters

Language

日本語版を表示する

查看简体中文版本

查看繁體中文版本

Просмотр версии на русском языке

Customer Service

Help

Tutorials

Contact us

ELSEVIER

Terms and conditions Privacy policy

Copyright © Elsevier B.V. All rights reserved. Scopus® is a registered trademark of Elsevier B.V.
We use cookies to help provide and enhance our service and tailor content. By continuing, you agree to the use of cookies.

RELX