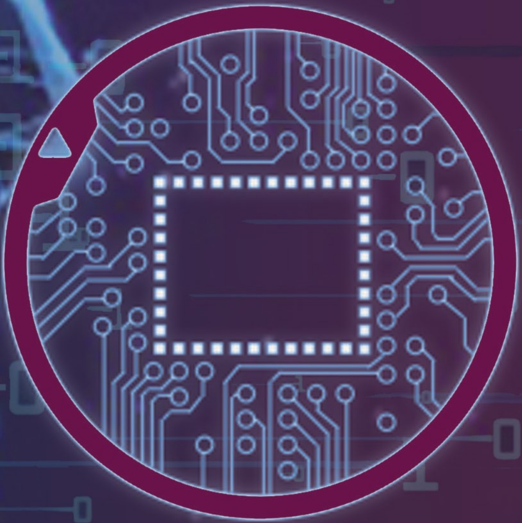
VOLUME-1
ISSUE-2
2021

e-ISSN: 2791-8335

JOURNAL OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



İZMİR KÂTİP ÇELEBİ UNIVERSITY
Artificial Intelligence and Data Science
Research and Application Center

JAIDA

HTTPS://JAIDA.IKCU.EDU.TR/

JOURNAL OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

Volume 1 – Issue 2 – December 2021 **E-ISSN: 2791-8335**

Aim & Scope

The Journal of Artificial Intelligence and Data Science (JAIDA) is an international, scientific, peer-reviewed, and open-access e-journal. It is published twice a year and accepts only manuscripts written in English. The aim of JAIDA is to bring together interdisciplinary research in the fields of artificial intelligence and data science. Both fundamental and applied research are welcome. Besides regular papers, this journal also accepts research field review articles.

Contact

Web site: https://jaida.ikcu.edu.tr/
E-mail: ikcujaida@gmail.com
Phone: +90 (232) 329 35 35/3731/ 3808/3819

Fax: +90 (232) 325 33 60
Mailing address: İzmir Katip Çelebi Üniversitesi, Yapay Zeka ve Veri Bilimi Uygulama ve Araştırma Merkezi, Balatçık Kampüsü, Çiğli Ana Yerleşkesi, 35620, İzmir

ÖNSÖZ

Yapay Zeka ve Veri Bilimi alanındaki teknolojik ve bilimsel gelişmeler; Yapay Zekanın endüstri, sağlık, otomotiv, ekonomi, eğitim gibi bir çok farklı alanda uygulanmasına imkan sağlamıştır. Ülkemiz Ulusal Yapay Zeka Stratejisinde; yeni bir çağın eşiğine gelindiği, yapay zekayla üretim süreçleri, meslekler, gündelik yaşam ve kurumsal yapıların yeni bir dönüşüm sürecine girdiği vurgulanarak, Yapay Zekanın öneminden bahsedilmiştir.

Sayın Cumhurbaşkanımızın da belirttiği gibi ülkemiz adına insan odaklı yeni bir atılım yapmanın zamanının geldiğine inanıyoruz. Yapay zeka çağına geçiş noktasında Türkiye'nin lider ülkelerden biri olması motivasyonu ile üniversitemizde yapay zeka teknolojilerinin kullanıldığı projeler gerçekleştirmekte, kongreler ve bilimsel etkinlikler düzenlemekteyiz.

Günümüz dünyasına rengini veren dijital teknolojilerin odağındaki ana unsurun yapay zeka teknolojilerinin olduğu düşüncesi ile yola çıkarak hazırlamış olduğumuz Yapay Zekâ ve Veri Bilimi Dergisinin, Ülkemiz Ulusal Yapay Zeka Stratejisinde belirtilen "Dijital Türkiye" vizyonu ve "Milli Teknoloji Hamlesi" kalkınma hedefleri doğrultusunda katkı sağlayacağı inancındayız.

Dergimizin hazırlanmasında emeği geçen üniversitemiz Yapay Zekâ ve Veri Bilimi Uygulama ve Araştırma Merkez Müdürü, Baş Editör Prof. Dr. Ayşegül ALAYBEYOĞLU'na, Editör ve Danışma kurulu üyelerine, akademik çalışmaları ile sağladıkları destek için tüm yazarlara, hakem olarak görev alan değerli bilim insanlarına teşekkür eder, dergimizin ilk sayısının ülkemize hayırlı olmasını dilerim.

Prof. Dr. Saffet KÖSE, Rektör

Dergi Sahibi

PREFACE

Technological and scientific developments in Artificial Intelligence and Data Science enabled the application of Artificial Intelligence in many different fields such as industry, health, automotive, economy and education. In our country's National Artificial Intelligence Strategy; the importance of Artificial Intelligence was mentioned by emphasizing the transformation process of production processes, occupations, daily life and corporate structures with artificial intelligence.

As stated by our President, we believe that the time has come to make a new human-oriented breakthrough on behalf of our country. With the motivation of Turkey being one of the leading countries at the point of transition to the age of artificial intelligence, we realize projects in which artificial intelligence technologies are used, and organize congresses and scientific events at our university.

We have prepared the Journal of Artificial Intelligence and Data Science with the idea that the main element in the focus of digital technologies that color today's world is artificial intelligence technologies, and we believe that our journal will contribute to the development goals of the "Digital Turkey" vision and "National Technology Move" stated in the National Artificial Intelligence Strategy of our country.

I would like to thank Prof. Dr. Ayşegül ALAYBEYOĞLU, the Director of Artificial Intelligence and Data Science Application and Research Center of our university. I would also like to thank to Editor and Advisory Board members, to all authors for their supports with their academic studies and to reviewers for their contributions to the preparation of our journal. I wish the first issue of our journal to be beneficial for our country.

Prof. Dr. Saffet KÖSE, Rector

Privilege Owner

BAŞ EDİTÖR'DEN

Değerli Araştırmacılar ve Dergi Okuyucuları;

İzmir Kâtip Çelebi Üniversitesi Yapay Zekâ ve Veri Bilimi Uygulama ve Araştırma Merkezi olarak Rektörümüz Prof. Dr. Saffet Köse sahipliğinde Yapay Zekâ ve Veri Bilimi Dergisinin ikinci sayısını sizlerle buluşturmanın gururunu yaşamaktayız.

İzmir Kâtip Çelebi Üniversitesi Yapay Zekâ ve Veri Bilimi Uygulama ve Araştırma Merkezi olarak hedefimiz; Cumhurbaşkanlığı Dijital Dönüşüm Ofisi Başkanlığı ve Sanayi ve Teknoloji Bakanlığı tarafından hazırlanan "Ulusal Yapay Zekâ Stratejisi" hedefleri doğrultusunda dergi, kongre, eğitim, bilimsel etkinlikler ve proje faaliyetleri gerçekleştirerek ülkemizin yapay zekâ alanındaki gelişim sürecine katkı sağlamaktır.

Farklı üniversitelerden, bilimsel disiplinlerden ve alanlardan değerli araştırmacıların hazırlamış oldukları 7 adet İngilizce araştırma makalesi bu sayı kapsamında sunulmaktadır. Merkez olarak düzenlemiş olduğumuz I. Uluslararası Yapay Zekâ ve Veri Bilimi Kongresi'nde sunulmuş olan çalışmalardan seçilenlerin genişletilmiş versiyonlarının da bulunduğu bu sayımızda makaleler çok titiz bir hakem sürecinden geçirilmiş ve son şekli verilmiştir. Siz değerli araştırmacılarımızın destekleri ile kaliteyi daha da arttırarak en kısa sürede ulusal ve uluslararası indekslerde taranan bir dergi olmayı hedeflemekteyiz.

Dergimizin yayın hayatına başlaması ve tüm merkez faaliyetlerinde büyük desteklerini gördüğümüz başta Rektörümüz Prof. Dr. Saffet KÖSE olmak üzere; dergimize olan destekleri için tüm yazarlara, dergimizin yayına hazırlanmasında heyecanla çalışan ve çok büyük emek harcayan Baş Editör Yardımcılarına, Editör ve Danışma kurulu üyelerimize, hakem olarak görev alan tüm değerli bilim insanlarına en derin şükranlarımı sunarım.

Saygılarımla,

Prof. Dr. Ayşegül ALAYBEYOĞLU

Baş Editör

LETTER FROM THE EDİTOR-IN-CHIEF

Dear Researchers and Readers of the Journal,

As İzmir Katip Çelebi University Artificial Intelligence and Data Science Application and Research Center, we are proud to present you the second issue of the Journal of Artificial Intelligence and Data Science (JAIDA), hosted by our Rector Prof. Dr. Saffet Köse.

As İzmir Katip Çelebi University Artificial Intelligence and Data Science Application and Research Center, our goal is; to contribute to the development process of our country in the field of artificial intelligence by carrying out journals, congresses, education, scientific events and project activities in line with the objectives of the "National Artificial Intelligence Strategy" prepared by the Digital Transformation Office of the Presidency of Türkiye and the Ministry of Industry and Technology.

7 research articles prepared by valuable researchers from different universities, scientific disciplines and fields are presented within the scope of this issue. This issue also includes extended versions of selected studies from the I. International Artificial Intelligence and Data Science Congress, which we have organized as a centre and the articles have been subjected to a rigorous peer-review process and have been finalized. With the support of esteemed researchers, we aim to increase the quality even more and become a journal that is scanned in national and international indexes as soon as possible.

I would like to express my deepest gratitude to Our Rector, Prof. Dr. Saffet KÖSE, who supported the publication of our journal and the center activities; to all the authors for their support to our journal; to our Associate Editors, who worked enthusiastically and put great efforts into the preparation of our journal; to our Editorial and Advisory Board members, and all esteemed scientists who served as reviewer.

Best Regards,

Prof. Dr. Ayşegül ALAYBEYOĞLU

Editor-in-Chief

CONTENTS

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 116-124, 2021
Research Article

Comparison of Multi-Label Learning Approaches for Pie Chart Image Classification Using Deep Learning

Derya BIRANT [*], Cem KOSEMEN

Dokuz Eylul University, Faculty of Engineering, Department of Computer Engineering, Turkey

Abstract
A pie chart is a powerful and circular information graphic used to display numerical proportions to the whole. However, the properties of pie charts cannot be directly noticed by machines since they are usually in an image format. To make a pie chart classifiable by machines, this paper proposes a novel solution using deep learning methods. This study is original in that it automatically and jointly classifies charts in terms of two respects: shape (pie or donut) and dimension (2D or 3D). This is the first study that compares two multi-label learning approaches to classify pie charts: *binary-class-based convolutional neural networks* (BCNN) and *multi-class-based convolutional neural networks* (MCNN). The experimental results showed that the BCNN model achieved 86% accuracy, while the MCNN model reached 85% accuracy on real-world pie chart data.
Keywords: Convolutional neural networks; deep learning; image classification; machine learning; multi-label learning; pie charts.

1. Introduction

A *pie chart* is a useful graph in which a circle container is divided into slices (or sectors) to illustrate proportions based on percentages. The principal purpose of a pie chart is to display the relationship of a part to the whole. Pie charts are widely-used due to their visual representation advantages over textual representation, both when it comes to expressive results and comprehension. They have been widely used in various types of sources such as books, reports, web pages, and newspapers. For instance, in scientific articles, pie charts have been utilized to represent the results of the experiments. Furthermore, in the presentation slides, they are frequently used as a graphical way to illustrate the information.

Currently, pie charts have been created in image format, and therefore, they can be understandable by humans, but not naturally machine-understandable or machine-readable. However, in recent years, there is an increasing need for machines to automatically interpret the properties of pie chart images. In many applications, it is needed to give the information about a pie chart image to use this knowledge for further processes such as for providing recommendations, developing better search engines, automatically tagging the pie chart images, semantically describing charts, segmenting a collection of pie chart images, and redesigning pie charts. Based on these motivations, in this study, we developed convolutional neural network (CNN) models to automatically classify pie charts based on their properties.

Pie charts on use have different characteristic features and so they can be classified with different properties. In this study, the proposed model makes predictions based on whether the charts are 2D or 3D, and whether they are pie or donut. A *donut chart* is a variant of the pie chart with a hole in the center. 3D pie charts could provide advantages over 2D pie charts since they support the comparison of multiple series. By visualizing the multi-variate information together, the 3D setup can mitigate the potential split-attention effects more effectively than the 2D setup. On the other hand, some 3D pie charts have been criticized for not carrying more information than their 2D counterparts, tending to make readers confused.

The main contributions of this study can be summarized as follows. Previous research works on pie chart classification use the standard (single-label) classification. Our study proposes a multi-label learning approach on pie charts. This is the first study that compares two multi-label learning approaches to classify pie charts: *binary-class-based convolutional neural networks* (BCNN) and *multi-class-based convolutional neural networks* (MCNN). This study is original in that it jointly classifies charts in terms of two respects: shape (pie or donut) and dimension (2D or 3D). Furthermore, a training dataset with high variety was generated for classifying pie chart images.

The proposed CNN model was developed from labeled training data; therefore, supervised learning was performed. The pie chart images in the training dataset were generated with random properties using a Python script. However, the constructed model was tested on real-world pie chart images that were collected from the Google search engine. The experimental results showed that the BCNN model achieved 86% accuracy, while the MCNN model reached 85% accuracy on real-world pie chart data.

*Corresponding author e-mail address: derya.birant@deu.edu.tr

Received: 08/12/2021; Accepted: 27/12/2021

2. Literature Review

An amount of information is presented inside chart images and inaccessible by visually impaired people and machines. To overcome this limitation, some studies have been done on information extraction from charts, including chart recognition, chart segmentation, chart classification, and chart interpretation.

Table 1 presents the previous work with regard to chart processing. Most studies on chart classification [1-3] have been concerned with determining the types of charts such as bar chart, scatter plot, area chart, and line chart. On the other hand, some studies especially focused on a single chart type and classify it according to its visual properties such as the classification of control charts [4], line charts [5, 6], and bar charts [7].

In the literature, many different machine learning approaches have been experimented with for classifying real-world chart images by their visual properties such as random forest (RF) [4], support vector machines (SVM) [4, 8], perceptually important points (PIP) [5], time series forest (TSF) [4], and decision tree (DT) [9]. Some studies have been developed neural network-based models for classifying charts such as recurrent neural networks (RNN) [7], convolutional neural networks (CNN) [10-16], and deep belief networks (DBN) [17]. Different deep learning architectures have been tested for chart classification such as residual networks (ResNet) [2, 6, 11, 15], Alex networks (AlexNet) [2, 15, 16], visual geometry group (VGG) [1, 2, 11, 14, 15], and you only look once (YOLO) [12]. For example, Tang et al. [17] developed a model for classifying charts with deep convolutional neural networks combined with deep belief networks. They focused on the following chart categories: bar, flowchart, line, pie, and scatter plot.

Table 1. *Some previous works on chart classification.*

Ref.	Year	Methods	Description	Chart Type	Single-Label	Multi-Label
[1]	2021	CNN, VGG	Chart type classification	Area, bar, box, line, map, pareto, pie, radar, scatter, table, venn charts	√	X
[2]	2021	CNN, VGG, AlexNet, LeNet, ResNet	Chart type classification	Area, bar, bubble, histogram, donut, line, pie, scatter, stacked area charts	√	X
[3]	2021	Data embedding	Reviving chart images	Bar, line, pie, scatter plots	√	X
[4]	2021	RF, TSF, SVM, CNN	Classification of control chart patterns	Control chart	√	X
[5]	2021	PIP	Chart pattern classification in financial time series	Line chart	√	X
[6]	2021	CNN, ResNet	Line chart understanding	Line chart	√	X
[7]	2021	CNN, RNN	Reverse-engineering bar charts	Bar chart	√	X
[8]	2021	SVM	Control chart pattern recognition	Control chart	√	X
[9]	2021	DT	Classification of control chart patterns	Control chart	√	X
[10]	2021	CNN	Classifying price chart images	Line chart	√	X
[11]	2020	Xception, ResNet, VGG, MobileNet	Chart recognition via classification and detection	Arc, area, bar, force-directed graph, line, parallel coordinates, pie, scatter plot, reorderable matrix, sunburst, treemap, word cloud	√	X
[12]	2020	CNN, YOLO	Object detection and classification	Candlestick charts	√	X
[13]	2020	CNN	Chart type classification	Area, bar, line, map, pareto, pie, radar, scatter, table, venn charts	√	X
[14]	2019	CNN, VGG	Chart type classification	Area, bar, line, map, pareto, pie, radar, scatter, table, venn charts	√	X
[15]	2018	AlexNet, GoogleNet, VGG, ResNet	Chart type classification	Bar, line, pie, radar, scatter	√	X
[16]	2017	SVM, CNN, AlexNet	Recovering visual encodings from chart images	Area, bar, line, scatter plots	√	X
[17]	2016	SVM, CNN, DBN	Chart type classification	Bar, flowchart, line, pie, scatter plot	√	X
Proposed Approach		CNN	Multi-label classification of charts Shape (Pie vs Donut) Dimension (2D vs 3D)	Pie and donut charts	√	√

Some previous studies [3, 18] have been conducted to interpret chart images using image processing methods and text recognition techniques such as optical character recognition. They applied some image processing methods to locate the chart elements (i.e., title, legend, and text regions) to extract data including in a chart.

Our study differs from the studies aforementioned here in several respects. This is the first study that applies a multi-label learning approach to automatically and jointly classify charts in terms of two aspects: shape (pie or donut) and dimension (2D and 3D). Our study is also original in that it compares two alternative MLC approaches combined with CNN on classifying pie charts in terms of accuracy: binary-class-based convolutional neural networks (BCNN) and multi-class-based convolutional neural networks (MCNN).

3. Methodology

A pie chart is a circular graphic that shows the relative sizes of elements to one another and to the whole. Currently, pie charts have been created as an image object, and thus, their visual properties can be only understandable by humans, but not by machines. However, recently, there is an increasing need for machines automatically classifying pie chart images. Furthermore, visually impaired people cannot classify these graphics. To overcome these limitations, in this study, we developed convolutional neural network (CNN) models to automatically classify pie charts according to their properties.

There are various representations and shapes of pie charts. This research is mainly focused on the processing of two chart shapes (pie and donut), as well as chart images in 2D and 3D formats. Since multiple class labels are assigned to the same chart image, this task requires a multiple-label learning approach.

3.1. Multi-label learning

Multi-label learning (MLL) is the task of learning from data samples that are represented by a single feature vector and its associated multiple labels. Multi-label classification (MLC) is one of the main multi-label learning paradigms. MLC is concerned with assigning a data instance to a set of categories or classes, meaning that each instance belongs simultaneously to multiple classes. MLC is a more challenging task than the standard single-label classification because output class label spaces are more complex. Moreover, annotating an object with more than one label is more difficult than annotating it with a single label.

The existing multi-label learning approaches can be divided into two main categories [19]: (i) algorithm adaptation approach and (ii) problem transformation approach. In the *algorithm adaptation approach*, an existing machine learning algorithm is adapted, extended, or customized for the task of MLC. In the *problem transformation approach*, a multi-label classification problem is transformed into more than one single-label classification problems. *Label powerset* (LP) and *binary relevance* (BR) are two of the well-known approaches in this type of multi-label classification. LP converts the multi-label data into multi-class form by considering all unique label combinations. BR transforms an MLC problem into a set of binary classification problems, depending on the one-against-all strategy. In this study, these two MLC methods were compared to each other in terms of accuracy to determine the best one for pie chart classification.

3.2. Convolutional neural networks

Our solution involves classifying pie charts by using deep learning, especially convolutional neural networks (CNN). CNN is a kind of deep neural network that is typically formed by multiple layers of convolutional operations with a filtering process followed by fully connected layers. CNN is usually applied to image data for image classification since it is a multi-layer structured deep learning model architecture designed for two-dimensional image analysis that can capture complex relationships among image pixels by reducing the required number of features [20].

The main operation of CNNs is "*convolution*". Convolution is basically mathematical operations, applying a function on the output of the previous layer. In a convolutional layer, a kernel or filter moves over the image and extracts feature maps that contain features of an image. Here, some image processing filters can also be applied to the images with convolution operations, such as edge detection, Sobel filtering, and noise reduction filters. In CNNs, important properties of the given data are extracted with convolution operations and pooling layers. The neural network model learns the appropriate filter matrices. These matrices are later used for extracting the important properties of the input data and making them ready to be processed in a dense layer. After the main features are extracted with convolution and pooling operations, a traditional artificial neural network is used with that data, called a dense layer or fully-connected layer. This final layer can have some hidden layers and has an output layer for giving the prediction result.

3.3. Proposed approach

This study compares two multi-label learning approaches to classify pie charts: *binary-class-based convolutional neural networks* (BCNN) and *multi-class-based convolutional neural networks* (MCNN).

Binary-class-based convolutional neural networks (BCNN): Based on the binary relevance approach, the multi-label classification problem is converted into a series of binary classification problems. A separate classification model is generated with a CNN architecture for each class label. For a given unseen data instance *x*, BCNN predicts its associated label set *y* by querying it on each independent binary classifier and then combing all labels together. Although the BCNN strategy may be a practical approach, a common disadvantage is that it disregards the relationship between target labels since each binary model is independent of the other.

Multi-class-based convolutional neural networks (MCNN): Based on the label powerset approach, the multi-label classification problem is converted into a multi-class classification problem. It considers each combination of class labels as if it was a new label. A multi-class classifier is built with a CNN architecture. For a given unseen data instance *x,* MCNN predicts its associated label set *y* by using this model. Although the MCNN strategy may be a practical approach, it increases computational complexity as the count of labels in the data increases, and the training dataset is large.

3.4. Proposed architecture

In this section, the proposed CNN model and its properties are discussed. Table 2 shows the general layer structure of the proposed CNN model. The architecture of the CNN model consists of convolution, pooling, flatten, dropout, and dense layers. The architecture in this work utilizes convolution layers and max-pooling layers on determining distinctive features of the charts before feeding them to the neural network. Dropout layers are also used in the feature extraction part.

Table 2. *Proposed CNN model.*

Layer Type	Description

Conv2D	Kernel (3 x 3 x 1 x 64)
	Bias (64)
	Activation = ReLu
	Filters = 64
	Kernel_size = 3, 3

MaxPooling2D	

Conv2D	Kernel (3 x 3 x 64 x 128)
	Bias (128)
	Activation = ReLu
	Filters = 128
	Kernel_size = 3, 3

MaxPooling2D	

Conv2D	Kernel (3 x 3 x 128 x 256)
	Bias (256)
	Activation = ReLu
	Filters = 256
	Kernel_size = 3, 3

Flatten	

Dense	Kernel (173056 x 512)
	Bias (512)
	Activation = ReLu
	Units = 512

Dropout	Rate = 0.2

Dense	Kernel (512 x 256)
	Bias (256)
	Activation = ReLu
	Units = 256

Dropout	Rate = 0.2

Dense	Kernel (256 x 4)
	Bias (4)
	Activation = Softmax
	Units = 4

Convolution layers are used to extract the features (characteristics) of the input chart image data such as color, contour, and types. The three convolution layers in the proposed architecture have 64, 128, and 256 filters, respectively. The kernel size (3, 3) is kept the same for all convolutional layers.

Pooling is a part of the feature extraction process. They are generally utilized for shrinking the spatial size of the data while maintaining its main properties. Pooling operation provides prevention of overfitting in CNNs by downsampling the input images. As can be seen in Table 2, we utilized the max-pooling technique in the model.

Flatten is used to convert the matrix values to a one-dimensional array to pass it to the fully-connected layer. In other words, the output of the convolutional layer is converted into a single vector before it is given as input to the dense layer. One key component of CNN is the loss function. The loss method basically calculates the difference between the true label and the label predicted by the trained model. In the proposed CNN model, the sparse categorical cross-entropy function was applied as a loss function. This function is implemented in Keras and it is well-known for image classification problems. Another important parameter of CNN is the optimizer. The Adam optimizer was used for minimizing loss value. This optimizer is a well-known and successful example of a gradient descent algorithm. The learning rate of the optimizer was set to 0.001, which is suggested as a default value.

The *dropout* method randomly eliminates some outputs from a layer in the network during the training phase. Dropout is useful to prevent overfitting like pooling. The proposed architecture utilizes dropouts after hidden layers in the fully-connected (dense) layer. The best dropout rate was found as 20% in this model.

A *dense* (or *fully-connected*) layer runs as a traditional multiple layer perceptron neural network. An important parameter in neural networks is batch size. We did not determine a specific batch size in the proposed architecture, so the batch size is the same as the count of training samples. *Rectified linear unit* (ReLU) is usually utilized in dense layers of neural networks as an activation method. In the proposed architecture, ReLU is utilized in both convolution layers and fully connected layers. ReLU produces zero for negative inputs, and the input value goes to output without any change if it is a positive number. In the last dense layer, the softmax function is commonly used to normalize output scores over multiple categories. This function accepts a list of real numbers as inputs and turns them into probability percentages. Output matrix size of the softmax layer must have the same value as the number of labels in the training data. Each label has a percentage ratio and the sum of all must be 100%. The label with the highest ratio is determined as the prediction output for a given input data instance.

4. Results

The aim of this study is to build a deep learning model that correctly classifies pie charts. A CNN architecture was designed to build a classifier. The Python programming language was used when constructing the model. Some packages, libraries, and tools were used during the data manipulation and training stages such as Numpy, Pandas, Matplotlib, Scikit-learn, Keras, and Tensorflow. Keras is a comprehensive deep learning framework and has various deep learning functionalities. TensorFlow is an open-source machine learning platform that includes many operations for creating many different predictive models.

For experimenting with the model, a simple computer was used for training the CNN model and making benchmark tests. The hardware that was used in this study is a PC with Windows 10 64 bit operating system and with the Intel Core i7-8750H CPU with 2.20 GHz and 16 GB RAM. We also used the CUDA toolkit, which is a GPU accelerated library for increasing the training speed of neural network models with parallel computation and fast matrix multiplication operations.

The performances of the models were assessed in terms of accuracy, recall, precision, f-measure, and confusion matrix. *Accuracy* is the ratio of the instances, which are correctly predicted by the model, to the total size of the test dataset. The *precision* metric states how many of the values estimated as positive are actually positive. The *recall* is a metric that indicates how much of the test instances that are required to estimate as positive, we estimated as positive. *F-measure* is the harmonic mean of the recall and precision values.

4.1. Dataset description

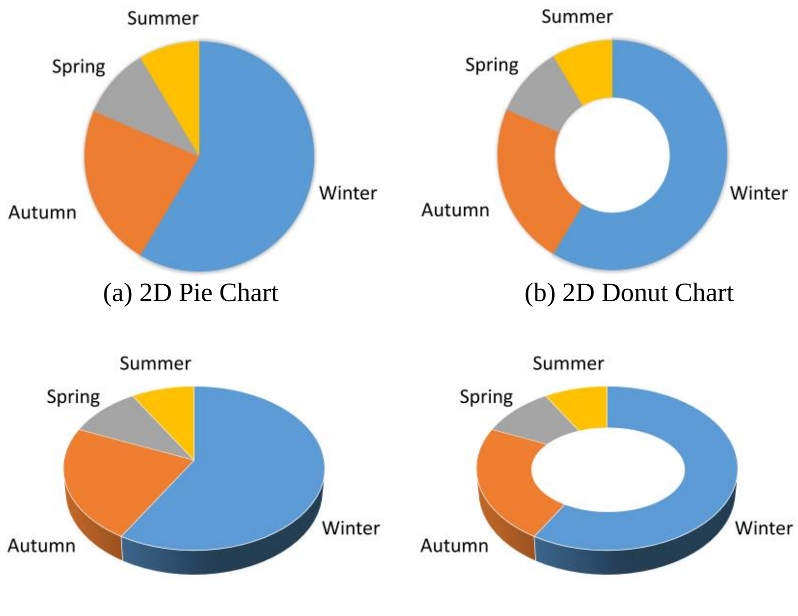
For making accurate predictions on real-world images, training data should have pie charts with different properties and features. Since a specific image repository that is suitable for the purpose of the study is not available, pie chart images in the training dataset were generated by using a computer script. We used the ChartDirector v6 chart drawing library in Python to create the training data. Chart styles (i.e., title, color, legends, and the number of slices) were designed randomly by using different visualization methods available in the library. The sizes of the pie charts in the outlining border were also designed randomly within a specified range.

The size of the image file is an important factor in the classification task. In this study, all the chart image samples were resized to 360 × 280 pixels since neural networks accept fixed inputs. Here, it is important that the converting operation preserves its aspect ratio in a chart. Changing the aspect ratio of an image can cause distortions and noises. Additionally, images with large sizes do not improve the resulting accuracy ratio. Besides, this causes longer training times (more epochs for training the neural network model) because the input vector size would be larger. Very small images can reduce classification accuracies because it causes loss in information.

With the chart generator, we created 400 pie chart image samples for each class that means 1600 images for training in total. Thereby, the training data is well-balanced for all the classes because it includes an equal number of instances from each class. The test dataset used in this study contains real-world pie charts which were collected from the Google Image Search engine. We gathered 50 images for each chart type, so there are 200 images in total in the test dataset. They differ in size and have different styles such as with or without title, legend, and marker.

In the data preprocessing phase, chart images were transformed to a greyscale palette since slices with different colors have no effect on the outcome of the classification. In other words, colored images store redundant data since we do not classify using colors. In addition, converting color images to grayscale images also reduces the training time since each pixel in greyscale images holds a single value, instead of three (RGB).

In this study, two different datasets were generated, one for the BCNN approach and one for the MCNN approach. For the BCNN approach, the training dataset was organized to be involved two target label attributes: one contains the labels of "2D" and "3D", and the other one includes the labels of "Pie" and "Donut". On the other hand, for the MCNN approach, the training dataset was organized to be involved a single target label attribute, which contains the labels of "2D Pie", "2D Donut", "3D Pie", and "3D Donut". Figure 2 shows the types of charts.



(a) 2D Pie Chart (b) 2D Donut Chart

(c) 3D Pie Chart (d) 3D Donut Chart

Figure 1. *Types of charts.*

4.2. Results obtained by the MCNN approach

The MCNN approach achieved 85% of accuracy on average on four class labels: "2D Pie", "2D Donut", "3D Pie", and "3D Donut". During the model is trained, accuracy started to converge to this test score after 25 - 30 epochs and the training loss also converged to near zero. Figure 2 shows the confusion matrix that was generated by the MCNN approach. According to the matrix, it is possible to conclude that the classifier usually had no difficulty in distinguishing chart types. For example, 43 of 50 “2D Donut” charts were correctly predicted; however, only 7 of them were misclassified by the constructed model. As can be seen in the confusion matrix, three test samples were incorrectly classified as a “3D Donut”, instead of “2D Donut”. Although chart types were identified well with high accuracy rates, “3D Pie” and “2D Pie” classes were slightly confused by the algorithm during prediction. As can be seen from the matrix, the best result on the real-world images was obtained for the "2D Pie" label with an accuracy of 98%.

		Predicted Classes			
	Class	**2D Donut**	**2D Pie**	**3D Donut**	**3D Pie**
Actual Classes	**2D Donut**	43	2	3	2
	2D Pie	0	49	0	1
	3D Donut	5	2	35	8
	3D Pie	0	6	1	43

Figure 2. *Confusion matrix that was generated by the MCNN approach.*

For further investigation of the proposed model performance, Table 3 shows the precision, recall, and f-measure values that were observed for each class. While the precision scores are ranging between approximately 0.8 to 0.9 for all classes, the recall scores are distributed between 0.7 and 0.98. Although all the results are very promising, the precision scores for the donut charts are higher than the scores for the pie chart. For instance, the precision value for the "2D Donut" chart is 0.8958, whereas it is the value of 0.8305 for the "2D Pie". On the other hand, the recall scores for the pie charts are higher compared to the donut charts. For instance, the recall value for the "3D Pie" chart is 0.86, whereas it is 0.7 for the "3D Donut". Among all the labels, MCNN achieved the best f-measure value (0.8990) on the "2D Pie" class. When the results given in Table 3 are considered as a whole, it can be deduced that the model built by MCNN has a good generalization ability to classify pie and donut charts in both 2D and 3D formats, so it can be effectively utilized to predict them well.

Table 3. *Performance values that were obtained by the MCNN approach.*

Chart Type	Precision	Recall	F-Measure
2D Donut	0.8958	0.8600	0.8775
2D Pie	0.8305	0.9800	0.8990
3D Donut	0.8974	0.7000	0.7865
3D Pie	0.7962	0.8600	0.8269

4.3. Results obtained by the BCNN approach

In this experiment, the BCNN approach was tested on the same dataset. The BR approach uses two different models at the same time. The first model predicts the dimension of the chart, whether if it is a *2D* or *3D* chart. The second model predicts the shape of the chart, whether it is a *pie* or *donut*. The average accuracy of these two models is considered as the final performance of the approach.

Figure 3 shows the confusion matrices that were generated by the BCNN approach. While the first model achieved 85% of accuracy on the prediction of chart dimension (2D or 3D), the second model reached 87% of accuracy on the prediction of chart shape (pie or donut). Thus, the average accuracy of these two models is 86%. It was observed from the experiment that the BCNN approach could distinguish the shape of the pie chart more correctly compared to the donut. Likewise, it achieved higher accuracy when distinguishing 2D charts compared to 3D charts. For example, 96% of 2D charts were classified correctly; nevertheless, only 2 out of 50 2D charts were predicted incorrectly. It was observed from the matrices that the models usually had no difficulty in identifying the charts.

		Predicted Classes	
	Class	**2D**	**3D**
Actual Classes	**2D**	48	2
	3D	13	37

		Predicted Classes	
	Class	**Pie**	**Donut**
Actual Classes	**Pie**	48	2
	Donut	11	39

Figure 3. *Confusion matrices that were generated by the BCNN approach.*

Table 4 shows the precision, recall, and f-measure values that were obtained by the BCNN approach. While the precision scores are ranging between approximately 0.78 to 0.95 for all classes, the recall scores are distributed between 0.74 and 0.96. Although all the precision scores are very promising, the best precision value (0.9512) was obtained for the pie shape. Among all the labels, BCNN achieved the best f-measure value (0.8807) on the donut-shaped charts. When the results given in Table 4 are considered as a whole, it can be deduced that the models constructed by BCNN have a good generalization ability to classify pie and donut charts in both 2D and 3D formats.

Table 4. *Performance values that were obtained by the BCNN approach.*

Chart Type	Precision	Recall	F-Measure
Donut	0.8135	0.9600	0.8807
Pie	0.9512	0.7800	0.8571
2D	0.7868	0.9600	0.8648
3D	0.9487	0.7400	0.8314

4.4. Comparison of the MCNN and BCNN approaches

Figure 4 shows the comparison of the performances of the MCNN and BCNN approaches. According to the results, BCNN (86%) slightly outperformed MCNN (85%) in terms of classification accuracy. This means that the model constructed by BCNN has a better chance of correctly classifying charts. Nevertheless, since the difference in their performances is small, these two approaches can be alternatively used in real-world applications.

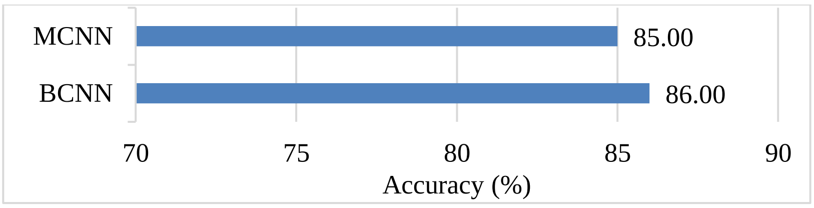


Figure 4. *Comparison of the MCNN and BCNN approaches.*

5. Conclusion

A pie chart is a useful graphic that is divided into slices to illustrate the proportions of elements to the whole. The properties of pie charts cannot be directly noticed by machines since they are usually in an image format. To make a pie chart classifiable by machines, this paper proposes a novel solution using deep learning techniques. A CNN architecture is designed and proposed for the pie chart classification task. While the previous works on pie chart classification use the standard (single-label) classification, our study proposes a multi-label classification approach on pie charts. This study is original in that it jointly classifies charts in terms of two respects: *shape* (pie or donut) and *dimension* (2D or 3D). This is the first study that compares two multi-label learning approaches to classify pie charts: binary-class-based convolutional neural networks (BCNN) and multi-class-based convolutional neural networks (MCNN). In the experimental studies, these two approaches (BCNN and MCNN) were tested and compared on the real-world pie chart images. The results showed that the BCNN model achieved 86% accuracy, while the MCNN model reached 85% accuracy.

As future work, the study can be further improved in several respects. In addition to visual properties, more information about pie chart images can also be given to visually impaired people by obtaining textual descriptions such as the values of the slices, the title of the chart, and the values reported in the legends. Moreover, similar studies can be conducted on other chart types such as bar charts.

Declaration of Interest

The authors declare that there is no conflict of interest.

References

[1] F. Bajic and J. Job, “Chart classification using Siamese CNN,” Journal of Imaging, vol. 7, no. 11, pp. 1-18, 2021.

[2] P. Mishra, S. Kumar, and M. K. Chaube, “ChartFuse: a novel fusion method for chart classification using heterogeneous microstructures,”, Multimedia Tools and Applications, vol. 80, no. 7, pp. 10417-10439, 2021.

[3] J. Fu, B. Zhu, W. Cui, S. Ge, Y. Wang, H. Zhang, H. Huang, Y. Tang, D. Zhang, and X. Ma. “Chartem: reviving chart images with data embedding,” IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 337-346, 2021.

[4] C-S. Cheng, Y. Ho, and T-C. Chiu, “End-to-End control chart pattern classification using a 1D convolutional neural network and transfer learning,” Processes, vol. 9, no. 9, pp. 1-26, 2021.

[5] Y. Zheng, Y-W. Si, and R. Wong, “Feature extraction for chart pattern classification in financial time series,” Knowledge and Information Systems, vol. 63, no. 7, pp. 1807-1848, 2021.

[6] C. Sohn, H. Choi, K. Kim, J. Park, and J. Noh, “Line chart understanding with convolutional neural network,” Electronics, vol. 10, no. 6, pp. 1-17, 2021.

[7] F. Zhou, Y. Zhao, W. Chen, Y. Tan, Y. Xu, Y. Chen, C. Liu, and Y. Zhao, “Reverse-engineering bar charts using neural networks,” Journal of Visualization, vol. 24, no. 2, pp. 419-435, 2021.

[8] R. Ünlü, “A robust data simulation technique to improve early detection performance of a classifier in control chart pattern recognition systems,” Information Sciences, vol. 548, pp. 18-36, 2021.

[9] M. Zaman, and A. Hassan, “Fuzzy heuristics and decision tree for classification of statistical feature-based control chart patterns,” Symmetry, vol. 13, no. 1, pp. 1-12, 2021.

[10] M. Siper, K. Makinen, and R. Kanan, “TABot - a distributed deep learning framework for classifying price chart images,” Advanced Computing, vol. 1367, pp. 465-473, 2021.

[11] T. Araujo, P. Chagas, J. Alves, C. Santos, B. Santos, and B. Meiguins, “A real-world approach on the problem of chart recognition using classification, detection and perspective correction,” Sensors, vol. 20, no. 16, pp. 1-21, 2020.

[12] S. Birogul, G. Temür, and U. Kose, “YOLO object recognition algorithm and buy-sell decision model over 2D candlestick charts,” IEEE Access, vol. 8, pp. 91894-91915, 2020.

[13] F. Bajic, J. Job, and K. Nenadic, “Data visualization classification using simple convolutional neural network model,” International Journal of Electrical and Computer Engineering Systems, vol. 11, no. 1, pp. 43-51, 2020.

[14] F. Bajic, J. Job, and K. Nenadic, “Chart classification using simplified VGG model,” In International Conference on Systems Signals and Image Processing, Osijek, Croatia, 2019, pp. 229-233.

[15] W. Dai, M. Wang, Z. Niu, and J. Zhang, “Chart decoder: generating textual and numeric information from chart images automatically,” Journal of Visual Languages & Computing, vol. 48, pp. 101-109, 2018.

[16] J. Poco and J. Heer, “Reverse-engineering visualizations: recovering visual encodings from chart images,” Computer Graphics, vol. 36, no. 3, pp. 353-363, 2017.

[17] B. Tang, X. Liu, J. Lei, M. Song, D. Tao, S. Sun, and F. Dong, “DeepChart: combining deep convolutional networks and deep belief networks in chart classification,” Signal Process, vol. 124, pp. 156-161, 2016.

[18] J. Shtok, S. Harary, O. Azulai, A. R. Goldfarb, A. Arbelle, and L. Karlinsky, “CHARTER: heatmap-based multi-type chart data extraction,” In Document Intelligence Workshop at KDD, 2021, pp. 1-5.

[19] M-L. Zhang and Z-H. Zhou, “A review on multi-label learning algorithms,” IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 8, pp. 1819-1837, 2014.

[20] S. H. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, “Impact of fully connected layers on performance of convolutional neural networks for image classification,” Neurocomputing, vol. 378, pp. 112-119, 2020.

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 125-135, 2021
Research Article

Snake Detection and Blurring System to Prevent Unexpected Appearance of Snake Images on Visual Media Sources Using Deep Learning

Selay TEKGÜL, Gokce NUR YILMAZ *

TED University, Faculty of Engineering, Computer Engineering, Turkey

Abstract
This paper proposes an object detection and blurring system using deep learning especially for the detection of snake images since while some of the people does not care about the appearance of the snake images on the visual sources, there is a significant portion of them who are uncomfortable of seeing snakes as well as their images. Therefore, with the system that is proposed in this paper, the snake objects from the images would be detected with YOLOv4 trained model and blurred using OpenCV. The detection f1 score of the selected model is 92% on the test set.
Keywords: blur, deep learning, object detection, phobia, snake.

1. Introduction

It is a fact that there is a significant attention that humans pay to the animals because of the effects of the inheritance [1]. Especially for the snakes, not only the effect of live snakes but also their photographs can cause a great deal of fear, i.e., phobia of snakes. Therefore, their unexpected appearance on the visual media sources e.g., TV, movies, photography is a problem for people who are suffering from that fear. As a result of the problem caused by these unfiltered visual media sources, the quality of their life is affected negatively.

Considering this fact, a model including snake detection on images and videos, as well as blurring the snakes is developed. Therefore, the proposed model consists of two stages:

1. Detection of the snakes on the visual source
2. Blurring the detected objects

For the first stage, YOLOv4 [2], which is a state-of-the-art, real-time object detection system, is used. YOLO has several advantages when it is compared to other object detection algorithms such as EfficientDet developed by Google Brain Team[3], CNN, fast R-CNN and faster R-CNN. One of the most significant advantages is that YOLO considers an image completely. Therefore, it results in better and faster detections with YOLO [2]. After the performance evaluation results, it is observed that 92% f1 score is achieved using YOLO for the snake detection. The second stage relying on the blurring operation is implemented considering with another blurring method from OpenCV that is a software library which includes open-source machine learning and computer vision algorithms [4].

The rest of the paper is organized as follows. In the second section, the proposed model is introduced. The results and discussions are presented in Section 3. Finally, Section 4 concludes the paper and discuss about future work.

2. Proposed Model

In the proposed model for the snake detection, the YOLO approach is exploited due to the advantages it provides compared to other object detection algorithms including CNN, fast R-CNN and faster R-CNN in terms of considering the entire image and faster detection [2]. The proposed model of this study is presented in Figure 1.

As it can be seen from the figure, first of all, dataset is fed into the deep learning model for training purpose. As a dataset, the latest version of Open Image Dataset V6 [5] is used. Open Image Dataset V6 is a dataset of 9M image-level annotated images provided by Google [6]. 1000, 52 and 316 images of the "snake" class by Open Image Dataset V6 are used as training, validation, and test images, respectively, in this study.

Since there is only one class in the proposed model, it is decided to train the model for 6000 iterations based on the recommendations on [7] which particularly explains the selection of the number of iterations as 2000 per class but not less than 6000. During the training process, the losses of the models after each iteration are also observed. When the losses are investigated, a decreasing trend is observed as a function of number of iterations. Moreover, the models that is encountered after each 1000 iterations are saved and ready to be used. Therefore, six models are derived due to the fact that 6000 iterations are divided by each 1000 iteration. As mentioned, there are

*Corresponding author e-mail address: gokce.yilmaz@tedu.edu.tr

Received: 23/11/2021; Accepted: 27/12/2021

six different trained models each encountered after 1000 iterations of training out of total 6000 iterations. The models will be stated as M1, M2, M3, M4, M5 and M6 after each 1000 iterations sequentially trained, respectively as they can be observed from Figure 1.

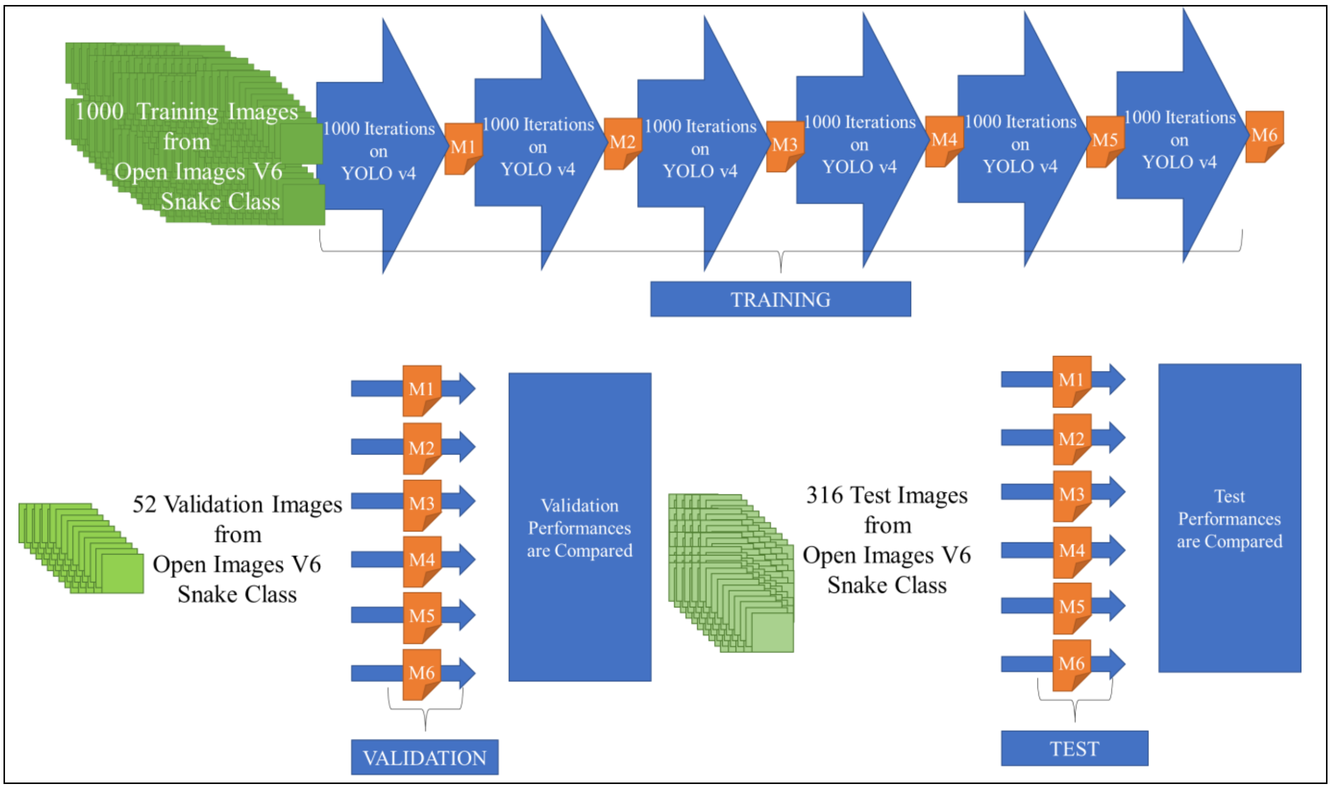
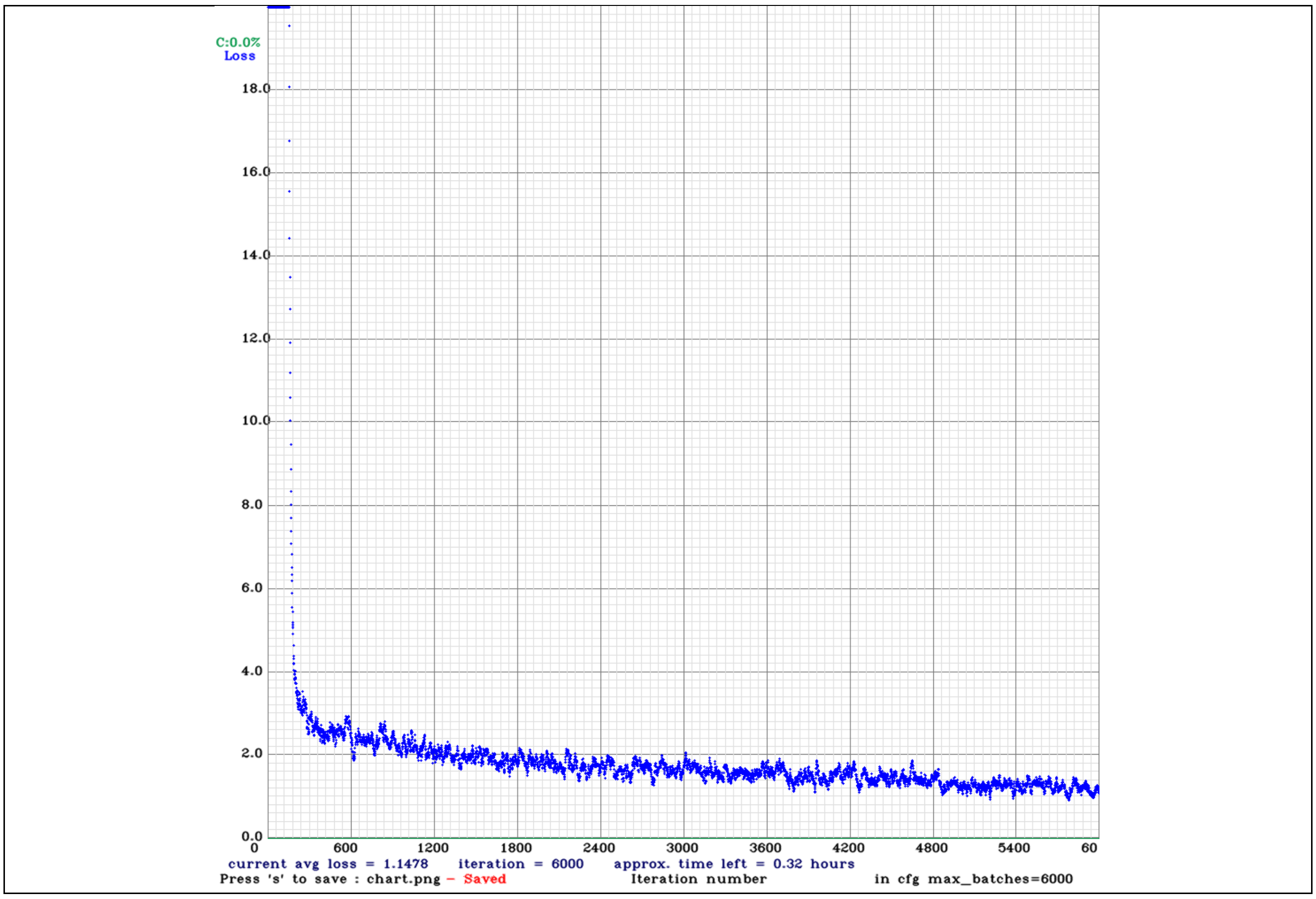

1000 Training Images from Open Images V6 Snake Class
1000 Iterations on YOLO v4
M1
M2
M3
M4
M5
M6
TRAINING
52 Validation Images from Open Images V6 Snake Class
Validation Performances are Compared
VALIDATION
316 Test Images from Open Images V6 Snake Class
Test Performances are Compared
TEST

Figure 1. *Proposed model.*

Additionally, these six different models are used to detect the snakes on the validation and test set images. After models are used for detection on the validation and test sets, the performance values of the models are evaluated.

The losses encountered after 6000 iterations are shown in Figure 2. When the losses are investigated form the figure, a decreasing trend is observed as a function of number of iterations. However, so as to better evaluate the performance of the models after each 1000 iteration, not only the losses but also the true positive (TP), false positive (FP), false negative (FN), precision, recall, F1 score, average intersection over union (IoU), average precision (AP) and mean average precision (mAP) values are calculated and compared. These values will be presented and discussed in Section 4.

Consequently, as a second step of the solution, after determining the best model to be used in the test set, the detected bounded boxes where the snakes are predicted to be on the images are blurred with the blur method from OpenCV. This method uses the normalized box filter and kernel to blur the given image. In this study, the kernel size is selected as 53 to 53.



current avg loss = 1.1478 iteration = 6000 approx. time left = 0.32 hours
Press 's' to save : chart.png - Saved in cfg max_batches=6000

Figure 2. *Losses as a Function of Number of Iterations (Total 6000 iteration).*

3. Results and Discussions

Corresponding performance values related with TP, FP, FN, precision, recall, F1 score, average IoU, AP and mAP of six models on validation set which consists of 52 images are shown in Table 1 by emphasizing the best scores on each column with bold texts. As observed from the bold texts, model M3, has the best TP, FP, FN, precision, recall, F1 score, AP and mAP values among the siz models that are compared. On the other hand, for the average IoU model M5 which is encountered after 5000 iterations gives the best score.

Table 1. *Comparison of the performances of six models on 52 images validation set.*

Models	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**Average IoU**	**AP**	**mAP**
M1	54	3	12	0.95	0.82	0.88	67.88%	94.28%	94.28%
M2	48	12	18	0.80	0.73	0.76	51.83%	68.39%	68.39%
M3	**60**	**1**	**6**	**0.98**	**0.91**	**0.94**	72.13%	**94.94%**	**94.94%**
M4	58	6	8	0.91	0.88	0.89	61.43%	88.36%	88.36%
M5	58	3	8	0.95	0.88	0.91	**79.12%**	88.76%	88.76%
M6	59	3	7	0.95	0.89	0.92	79.05%	93.66%	93.66%

The precision, recall and F1 score values of six models on the validation set consisting of 52 images are compared in Figure 3 with a chart. As can be observed from the chart, the precision, recall and F1 score values of of the models are at the range of 0.80 to 0.98, 0.73 to 0.91 and 0.76 to 0.94 on the validation set respectively. Moreover, the best precision, recall and therefore F1 score values are gathered with the model M3 on the validation set.

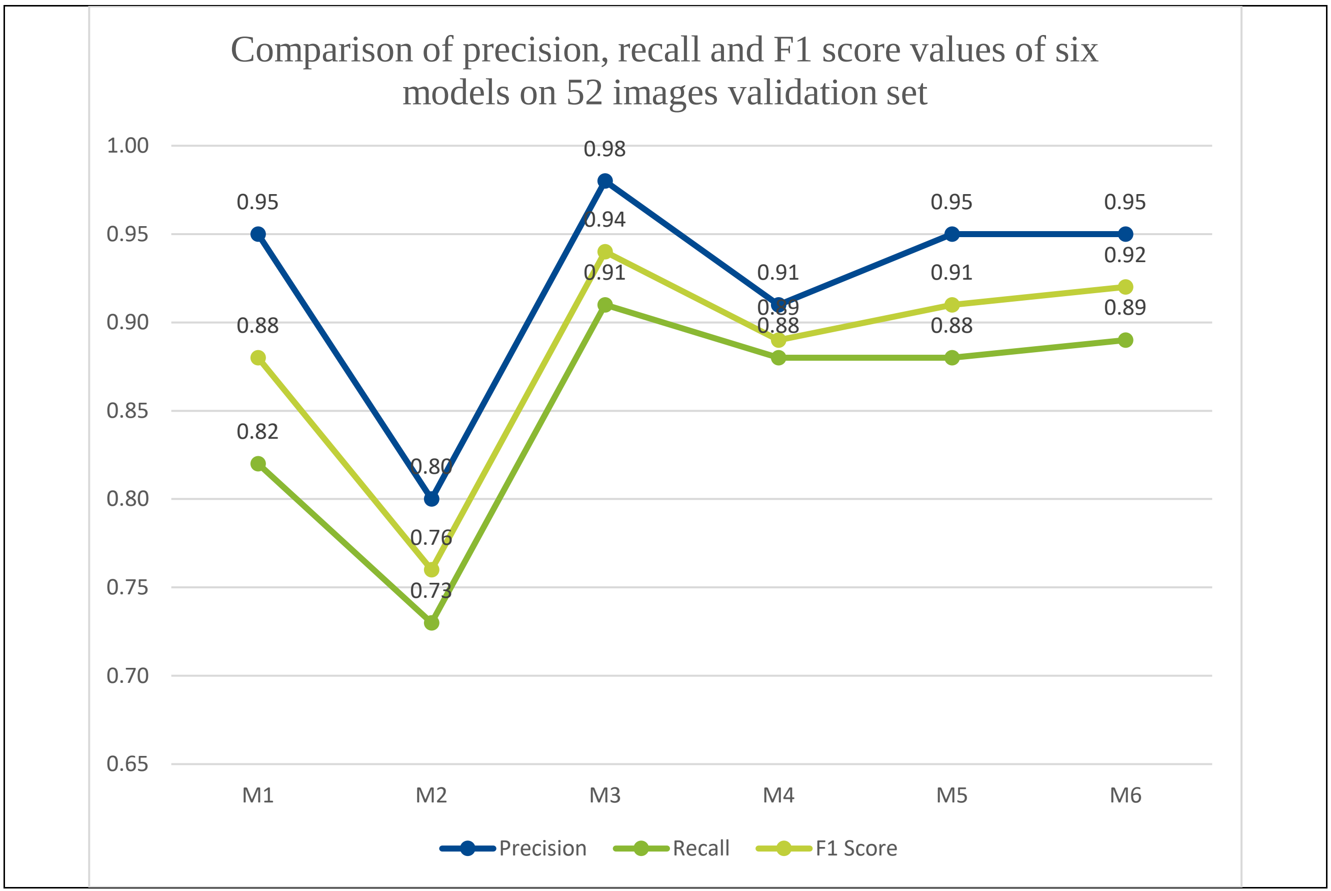


Figure 3. *Chart of comparison of precision, recall and F1 score values of six models on 52 images validation set.*

Moreover, the percentage of average IoU and mAP values of six models on the validation set consisting of 52 images are compared in Figure 4 with a chart. As seen from the chart, the average IoU and mAP values of the six models on the test set are at the range from 51.83% to 79.12% and from 58.39% to 94.94%, respectively. Additionally, the best performance scores are observed with the model M3 for both average IoU and mAP values on the validation set.

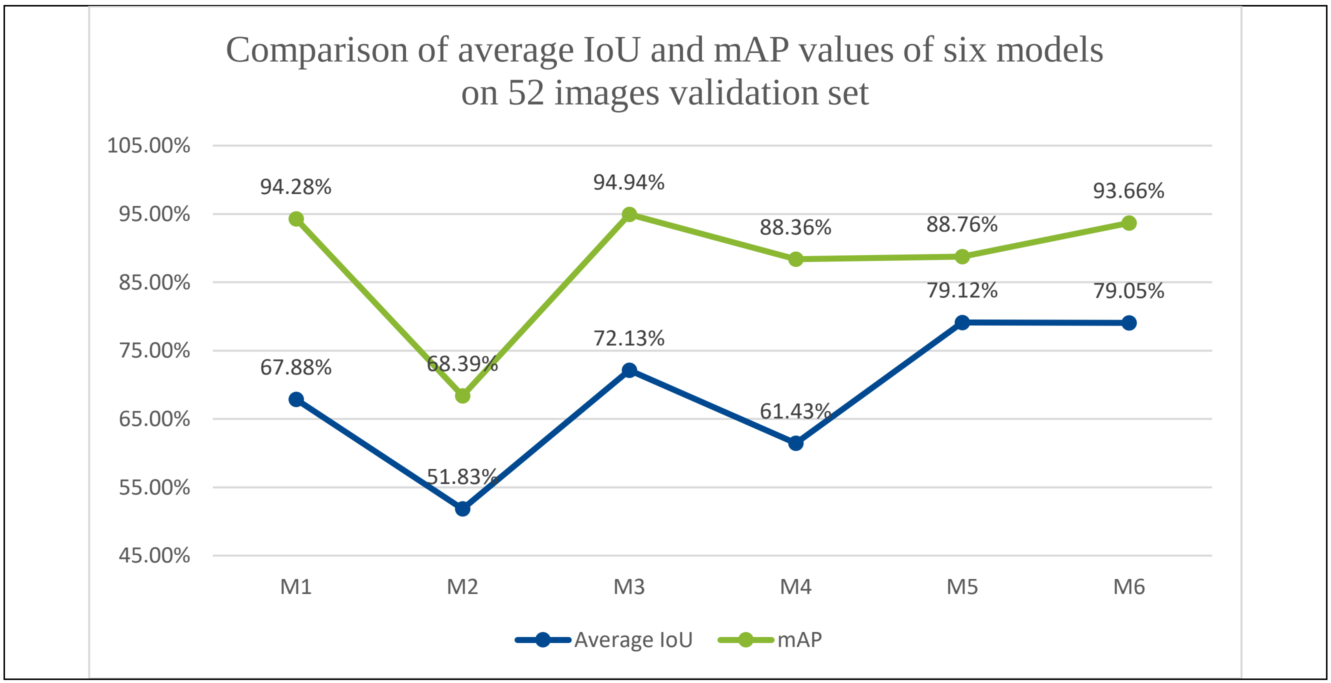


Figure 4. *Chart of comparison of precision, recall and F1 score values of six models on 52 images validation set.*

The training, validation and testing are completed on a laptop computer with a single Nvidia 950M 4GB GPU and 8GB RAM on 64-bit Ubuntu 20.04 operating system using Python version 3.8.5 and OpenCV version 4.5.2. The training is completed within 18 hours and the detection process of the validation set examples took 10-11 seconds for each of the six models which would not affect the comparison of the models drastically. Detection times of the six models on 52 images validation set are compared in Table 2.

Table 2. *Comparison of the detection times of six models on 52 images validation set.*

Models	Total Detection Time (in seconds)
M1	11
M2	11
M3	11
M4	10
M5	10
M6	10

During the evaluation of training and detection times, the given environment characteristics should be taken into consideration. When the usage of GPU changes, the training and detection times would differ.

Predicted best detection performance on the test set is taken with model M3 due to its performance on the validation set examples, however, the best TP, FN and Recall values are encountered with model M6 as 218, 25 and 0.90 on the test set examples, respectively. On the other hand, the best FP, Precision, Average IoU, AP and mAP values are gathered when model M5 is used for testing as 10, 0.96, 80.60%, 92.20% and 92.20%, respectively. Lastly, the best F1 score is the same for these two models, M5 and M6 and is 0.92.

The overall comparison of the performances of the models on the test set is put in Table 3.

Table 3. *Comparison of the performances of six models on 316 images test set.*

Models	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**Average IoU**	**AP**	**mAP**
M1	205	24	38	0.90	0.84	0.87	66.17%	88.45%	88.45%
M2	160	53	83	0.75	0.66	0.70	49.79%	63.95%	63.95%
M3	214	21	29	0.91	0.88	0.90	67.75%	89.35%	89.35%
M4	212	24	31	0.90	0.87	0.89	63.04%	87.32%	87.32%
M5	217	**10**	26	**0.96**	0.89	**0.92**	**80.60%**	**92.20%**	**92.20%**
M6	**218**	12	**25**	0.95	**0.90**	**0.92**	79.78%	91.81%	91.81%

The precision, recall and F1 score values of six models on the test set consisting of 316 images are compared in Figure 5 with a chart. As presented in the chart, the precision, recall and F1 score values of of the models are at the range of 0.75 to 0.96, 0.66 to 0.90 and 0.70 to 0.92 on the test set respectively. Moreover, the best precision value is gathered with the model M5, the best recall value is gathered with the models M5 and M6 and lastly, the best F1 score is gathered with model M6 on the test set.

The percentage of average IoU and mAP values of six models on the test set consisting of 316 images are compared in Figure 6 with a chart. As seen from the chart, the average IoU and mAP values of the six models on the test set are at the range from 49.79% to 80.60% and from 63.95% to 92.20%, respectively. Furthermore, when the chart is investigated, it is observed that the model that gives the worst performance is model M2 for both of the performance values, however, the performance values gathered by using the other models are close to each other. Therefore, with the chart which compares the precision, recall and F1 Score values of the models provided at the Figure 5 is a better discriminator while selecting the best model than Figure 6.

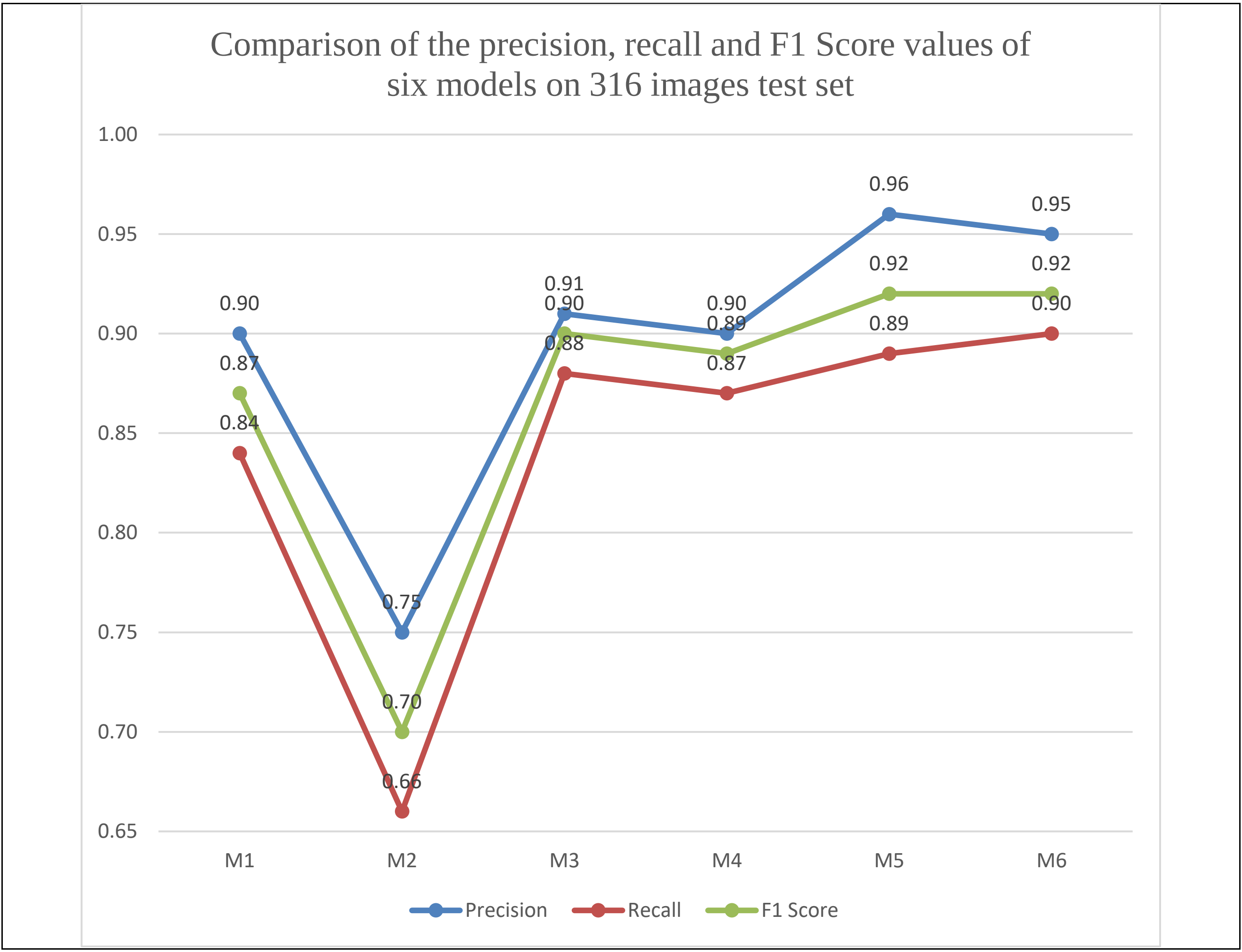


Figure 5. *Chart of comparison of precision, recall and F1 Score values of six models on 316 images test set.*

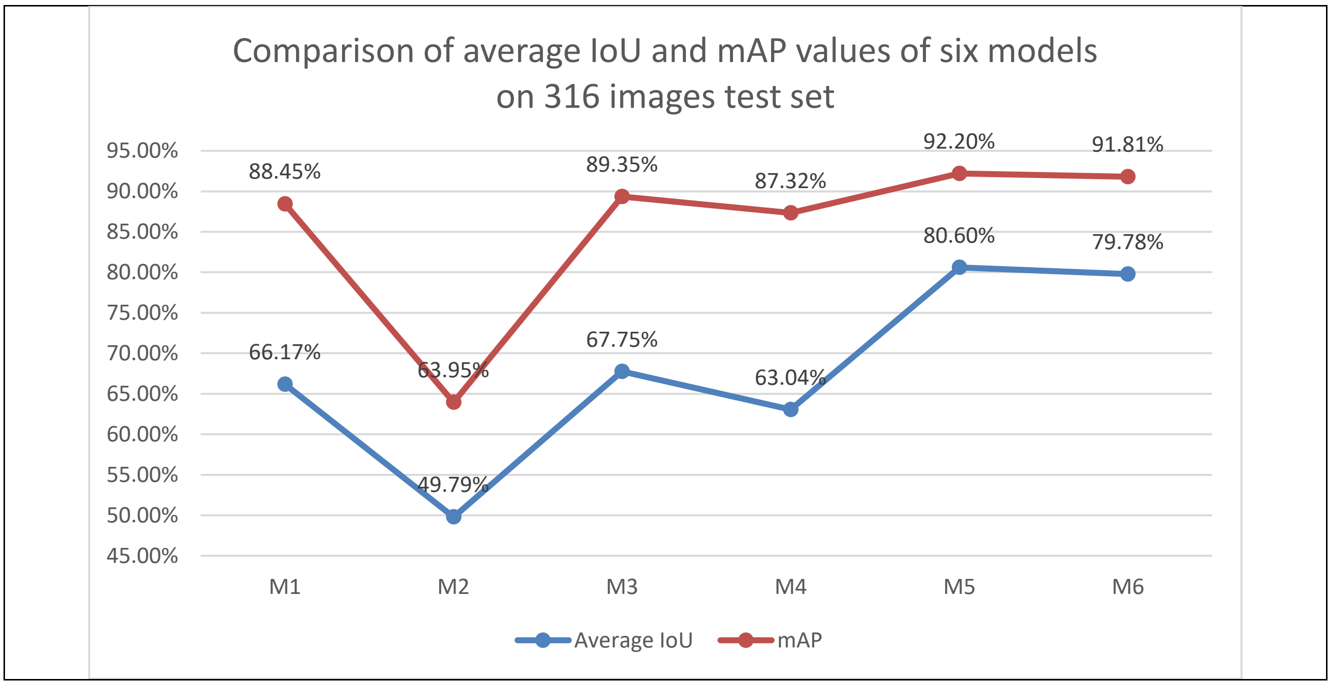


Figure 6. *Chart of comparison of precision, recall and F1 Score values of six models on 316 images test set.*

Problem that is attempted to solve in this paper is to detect as many snake images as precisely as possible when images are tested since when the phobia is considered, it should be guaranteed that none of the snake images will be shown to the viewer. Therefore, when the test results are investigated, since the given values with model M5 and M6 are pretty close to each other, the one whose recall is greater than other should be used before blurring as a final model. Additionally, with YOLO, real-time object detection is possible but requires a better GPU than the one used in the experiment of this study to propose a working model for commercial since the detection of 316 test images took 39-40 seconds to be completed in each of the six models that are shown at Table 4.

Table 4. *Comparison of the detection times of six models on 316 images test set.*

Models	Total Detection Time (in seconds)
M1	40
M2	40
M3	39
M4	40
M5	40
M6	40

In the Figures 7, 8, 9, 10, 11 and 12 the detections on the 12 images selected from the test set that are predicted using M1, M2, M3, M4, M5 and M6 are shown with red bounded boxes representing the predicted snakes, and the green bounded boxes representing the ground truth of the given image.

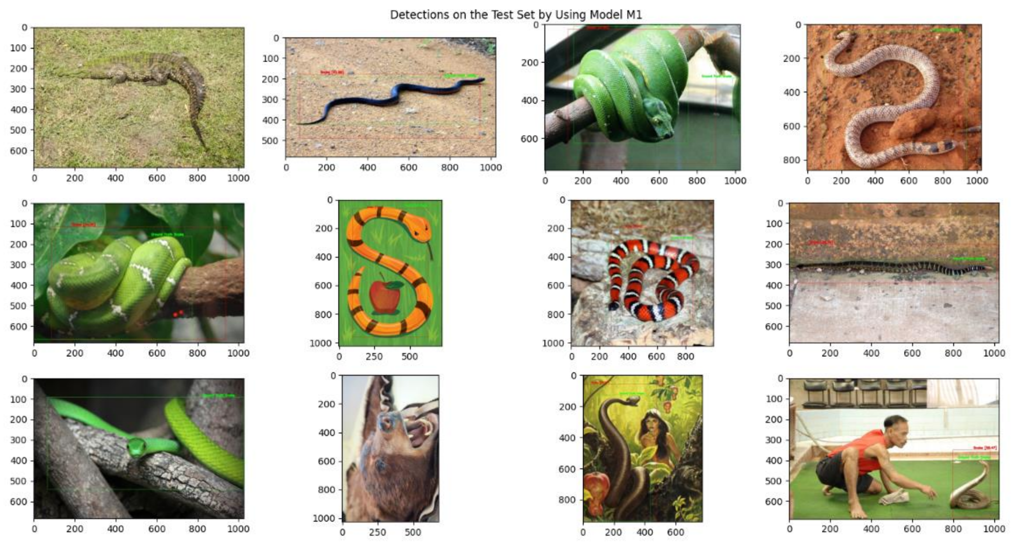


Figure 7. *Predictions of model M1 on the test set.*

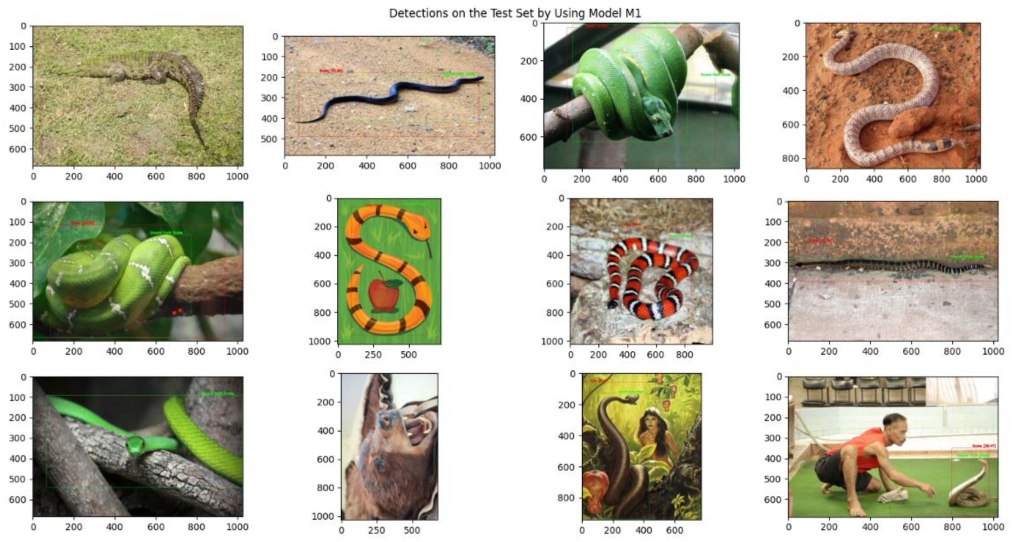


Figure 8. *Predictions of model M2 on the test set.*

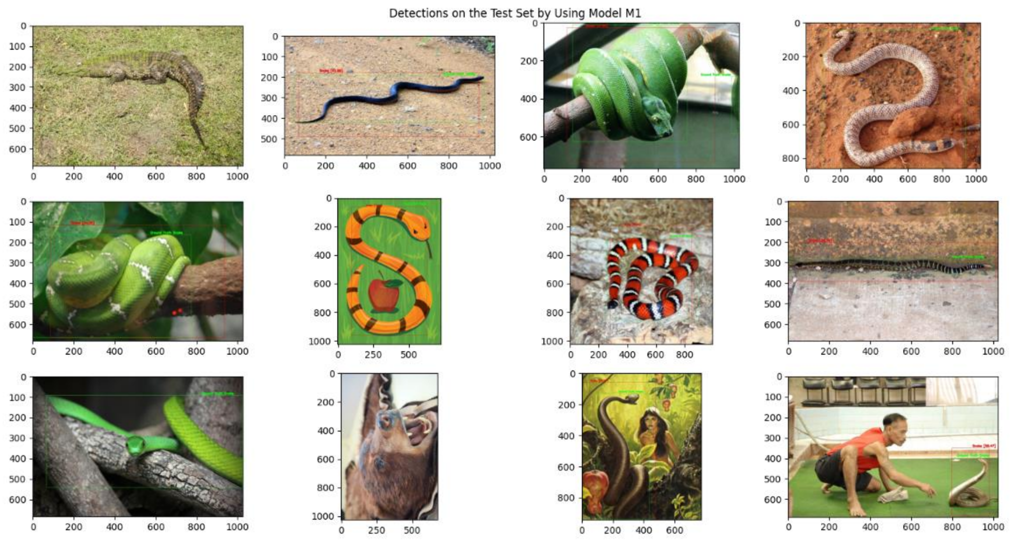


Figure 9. *Predictions of model M3 on the test set.*

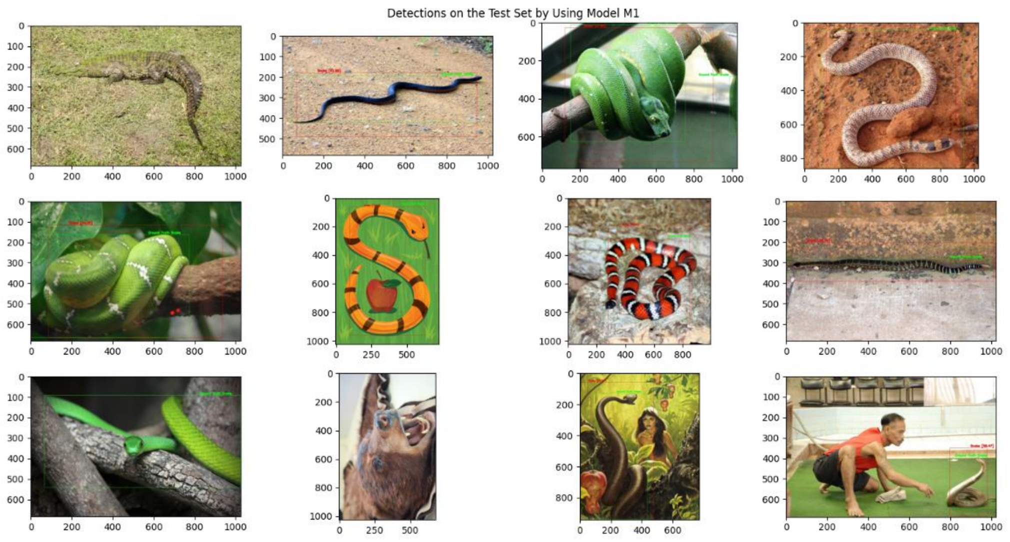


Figure 10. *Predictions of model M4 on the test set.*

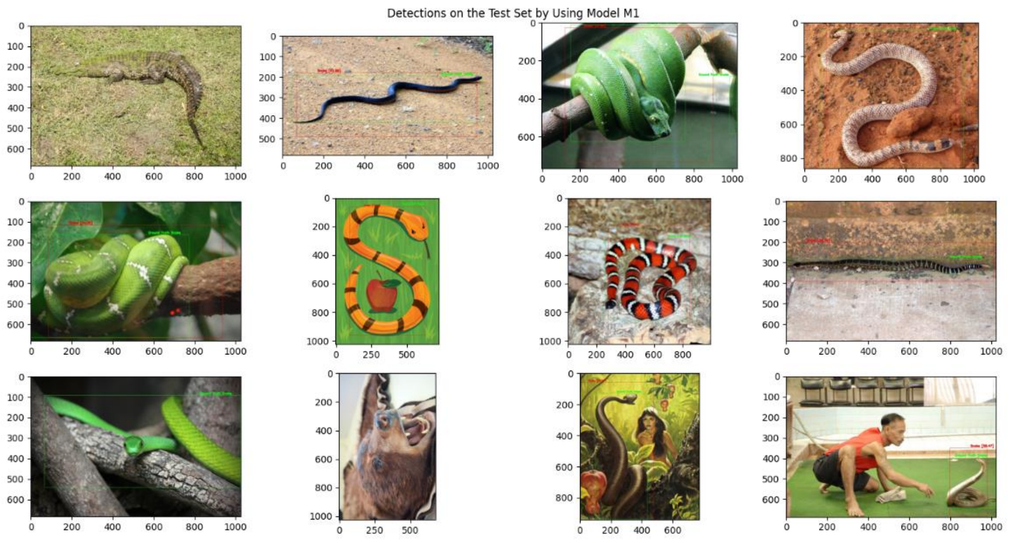


Figure 11. *Predictions of model M5 on the test set.*

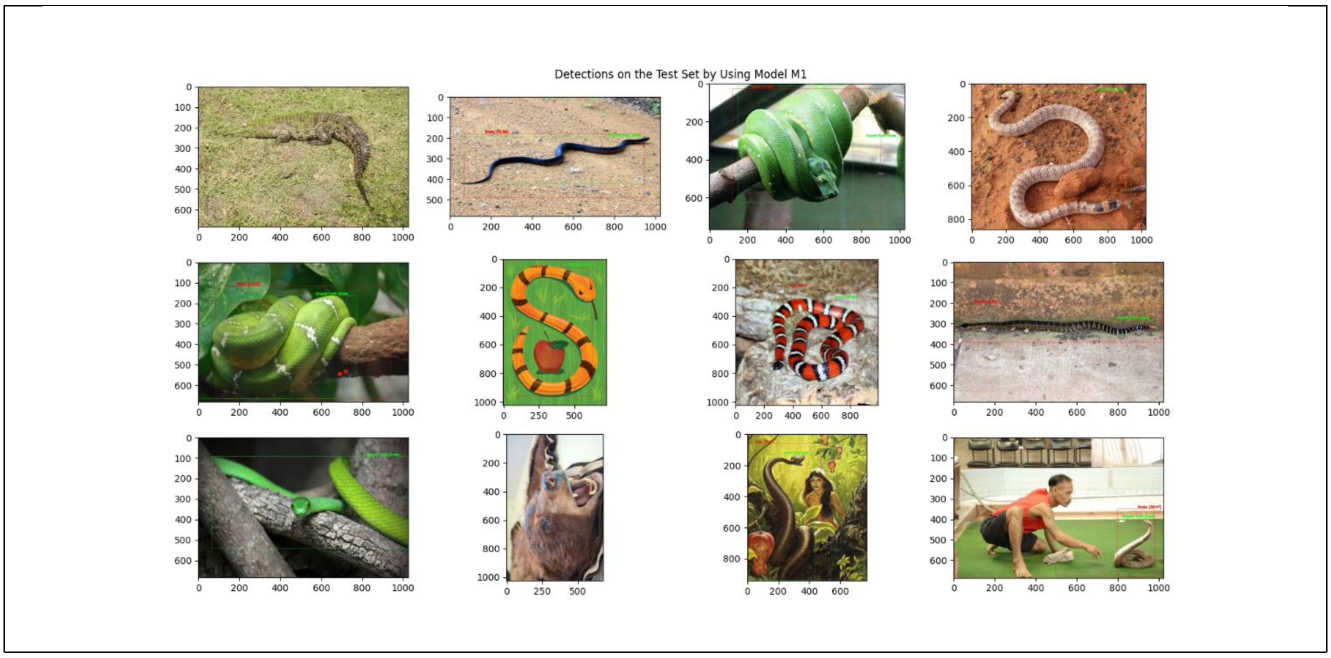


Figure 12. *Predictions of model M6 on the test set.*

With model M1, the non-snake images crocodile (1st row, 1st image on Figure 7) and bear (3rd row, 2nd image on Figure 7) are not predicted as snakes however, there are snake images (1st row, 4th image and 3rd row 1st image on Figure 7) that are not detected as snakes as well. Interestingly, the one and only model who does not predict the crocodile as snake is model M1. Additionally, the snake drawing (2nd row, 2nd image) image, is detected as a snake with only the model M1.

With model M2, bear (3rd row, 2nd image on Figure 8) is not predicted as a snake however, there are snake images (1st row, 4th image and 3rd row 1st image on Figure 8) that are not detected as snakes just like the predictions with model M1. Different than model M1, the image of a crocodile is predicted as snake with model M2.

With model M3, bear (3rd row, 2nd image on Figure 9) is not predicted as a snake however, there are again snake images (1st row, 4th image and 3rd row 1st image on Figure 9) that are not detected as snakes just like the predictions with both models M1 and M2. Also, in the first image the crocodile is predicted as a snake like the prediction of model M2.

With model M4, both the bear and the crocodile are predicted as snakes also, there is a snake image (3rd row 1st image on Figure 10) that is not detected as a snake. In this model M4, as an improvement, the snake in the 4th image at the 1st row of Figure 10 is detected a snake for the first time.

With model M5, there is a snake image (3rd row 1st image on Figure 11) that is not detected as a snake. In this model M5, the snake in the 4th image at the 1st row of Figure 11 is detected a snake like it is detected with model M4. On the other, this model again predicts both the crocodile and bear images as snakes.

With model M6, now there is not a real snake image that is not detected as a snake except the one with the drawing. Also, both the crocodile and bear images are detected as snakes.

In the figures 8, 9, 10, 11, and 12, there is an image of a snake drawing that cannot be detected as a snake with the majority of the models (except the first model M1). Therefore, in order to detect an image of a snake drawing, more snake drawing images should be added to the training set.

4. Conclusions and Future Work

In this paper, a model for snake detection and blurring has been proposed. Six different models using state-of-the-art object detection algorithm YOLOv4 have been explored. The results of this work indicate that different models encountered with different number of iterations on training phase of the deep learning, can vary in terms of detection of different kind of snake images. Generally, it can be said that more iterations would increase the precise detection of the snake images using the dataset that is used in this experiment.

To get the application useful in the real life, there are many possible future directions to improve the work: segmentation and tracking. In the testing of the models, it is observed that there are some body parts of the snakes that are not covered with the bounded box, also there are some areas in the bounded boxes that do not belong to the area where the snake objects reside. So, when it comes to the real-life application, viewer will see some parts of the snake objects occasionally and after blurring operation is completed some unrelated parts of an image will be unseen. Therefore, the quality of a visual experience will end up decreasing for some level. In order to solve this problem, instead of bounding box detection, segmentation could be performed. With segmentation, more precise blurring operation would be completed, and the experience of a viewer would increase with a training using enough number of segmentation annotated images.

Also, if this approach will be adapted to be used on the videos, by implementing snake objects' tracking, the number of the snakes where the detection and blurring applied, on the frames would be increased.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Snake Detection and Blurring System Using Deep Learning”).

References

[1] E. Landova, J. Maresova, O. Simkova, V. Cikanova and D. Frynta, "Human responses to live snakes and their photographs: Evaluation of beauty and fear of the king snakes," Journal of Environmental Psychology, vol. 32, no. 1, pp. 69-77, 3 2012.

[2] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[3] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," 2020.

[4] OpenCV team, "About," OpenCV, 2021. [Online]. Available: https://opencv.org/about/. [Accessed 20 May 2021].

[5] Google Inc., "Open Images V6 Snake Category Detection Type," 2021. [Online]. Available: https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F078jl. [Accessed 20 May 2021].

[6] Google Inc., "Overview of Open Images V6," Google Inc., 2021. [Online]. Available: https://storage.googleapis.com/openimages/web/factsfigures.html. [Accessed 21 May 2021].

[7] AlexeyAB, "Yolo v4, v3 and v2 for Windows and Linux," 15 5 2021. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed 21 May 2021].

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 136-143, 2021
Research Article

Traffic Density Estimation using Machine Learning Methods

Sümeyye AYDIN [1,*], Murat TAŞYÜREK [1], Celal ÖZTÜRK [2]

[1*] Kayseri University, Engineering, Architecture And Design Faculty, Computer Engineering Department, Turkey
[2] Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey

Abstract
In cities where population density is high and transportation systems are widely used, it is necessary to manage traffic systems more effectively not to affect the daily planned works. The Intelligent Transportation System (AUS) is expressed as a system that provides users with better information and safer, more coordinated, and smarter use of transportation networks with different transportation modes and traffic management. One of the most important components of AUS models is the determination of traffic density. The traffic density of intersections is a difficult problem as it affects other interconnected intersections and varies in time. Deep learning method is a widely used method in traffic density estimation in recent years. In this study, the long-term short memory network (LSTM) model, one of the deep learning methods, is proposed to estimate the traffic density of a certain region using open data of Istanbul Metropolitan Municipality. The performance of the proposed LSTM-based model is compared with machine learning methods such as linear regression, decision tree, random forest, and the classical deep learning method (DL). Experimental evaluations show that the proposed LSTM method is more successful in traffic density estimation than the compared methods.
Keywords: Deep learning, Intelligent transportation systems; Istanbul traffic density prediction; LSTM, Machine Learning.

1. Introduction

Intelligent Transportation Systems (ITS) technologies improve transportation safety and mobility and increase productivity by integrating advanced communication technologies into transportation infrastructure and vehicles. One of the most important components of ITS technologies is the estimation of traffic density. Traffic density is expressed as the traffic demand exceeding the transportation capacity. The traffic density of one intersection is a difficult problem as it also affects other interconnected intersections and varies in time. Traffic density is a serious problem that affects the planning of daily life and is becoming increasingly common all over the world. This paper proposes to estimate the intersection density in a particular region by using LSTM-based deep learning method with open data of Istanbul Metropolitan Municipality (IMM).

Deep learning methods have recently been widely used to solve many problems such as natural language processing and real-time object detection [1-3]. There are many different deep learning models in the literature, depending on the data set and the type of problem. Some deep learning methods are insufficient in time-dependent processes. Recurrent Neural Networks (RNN) are successful in datasets containing time-dependent relationships [4]. The RNN model was first introduced in the 1980s [5]. However, traditional RNN methods face the problem of capturing long-term dependencies in the input data. A long short-term memory network (LSTM) has been proposed to solve this problem [6]. Ma et al. have applied the LSTM model for estimating traffic speed with remote micro software sensor data [7]. Tian et al. proposed the LSTM RNN model for traffic flow prediction [8] and found that the LSTM RNN method outperformed most of the non-parametric models [8]. Li et al. evaluated the performance of the LSTM and GRU model for traffic flow prediction [9]. A better performance can be obtained with the LSTM model in traffic flow forecasting, as LSTM can automatically calculate optimal time delays and capture features of longer time interval time series [8].

In this study, it is proposed to estimate traffic density using LSTM-based deep learning method using IMM open transportation data. Istanbul is the city with the largest population density and the most significant traffic problem in Turkey [10,11]. The proposed method is aimed to predict the traffic density of certain intersections during the time periods when the traffic density is high, such as commuting and rush hours. The performance of the proposed LSTM-based model is compared with machine learning methods such as linear regression, decision tree, random forest, and the classical deep learning method (DL) on actual observed data.

*Corresponding author e-mail address: sumeyyeaydin@kayseri.edu.tr

Received: 29/11/2021; Accepted: 21/12/2021

2. Methods

2.1. Linear regression method

Linear regression aims to establish a linear relationship between one or more independent variables and a dependent variable or a numerical result. This method estimates the value according to the independent variables. In simple linear regression, the target value can be estimated with one independent variable, while in multiple regression, the target is estimated by two or more independent variables. A multiple linear regression model was used because the number of independent variables in the traffic dataset was more than one.

2.2. Decision tree

Decision trees are a hierarchical machine learning method consisting of nodes, branches, and leaves. The output of the decision tree is in the form of a tree or rules, so it is easy to understand. The tree expands gradually according to the questions asked to the root node, and the growth is completed when the last leaves are formed. It is divided into regression trees and classification trees. If the dependent variable is categorical, classification trees are used, and if the dependent variable is continuous, regression trees are used. In this article, regression trees were used because it works with constantly changing data.

2.3 Random forest

Random forest algorithm is a machine learning method used to solve regression and classification problems[12]. The algorithm is an ensemble learning algorithm consisting of the output of multiple decision trees. Each node branches by choosing the best among the randomly selected variables in the node [13]. It is not pruned as in decision trees, so it is fast. In this study, a forest was created using 30 decision trees.

2.4. Deep learning

Deep learning is a subset of machine learning. It has many neural nodes. An artificial neural network is formed by the combination of nerve cells. A neural network is called ‘deep’ when it has more than one hidden layer[14].This design was inspired by the biological structure of the human brain. So in deep learning, there can be many layers between the input and output layers. These layers are called hidden layers. In the model consisting of consecutive layers, the previous layer's output is the input of the next layer[15]. Deep learning has been a widely used prediction method in recent years.

2.5. Long short-term memory (LSTM)

The LSTM deep learning algorithm is known as a recurrent neural network introduced by Hochreiter and Schmidhuber in 1997 to eliminate the disadvantages of the RNN architecture [16]. The most important feature of the LSTM network is that it also evaluates the background information about the network. The basic LSTM diagram is given in Figure 1 [17, 18].

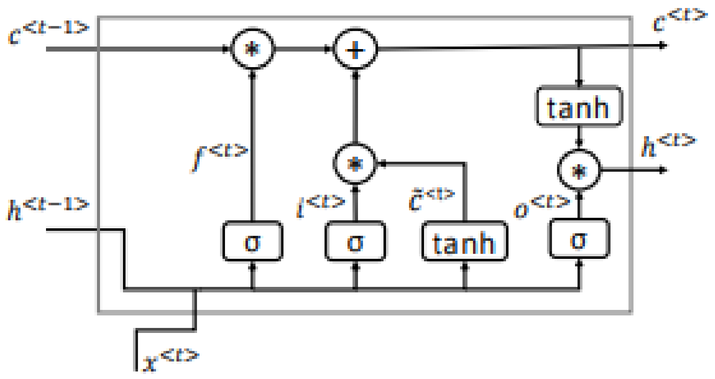


Figure 1. *Basic LSTM block diagram.*

LSTM networks address the problem of vanishing gradients of the RNN model by dividing into three inner cell gates, creating pseudo-memory cells to store information in a long-range context [19]. A typical LSTM neural network cell is basically configured with four gates, namely the input gate, the input modulation gate, the forget gate, and the output gate: The cell is controlled by gates and can keep the value or reset the value according to the state of the gate. In particular, three gates are used to control whether the current cell value will be forgotten (forget gate f_t), whether it will read its input (input gate i_t), and output (output gate o_t) the new cell value; In addition, there is an input modulation gate called $\bar{c}_t$. The LSTM gates and their operations are presented in Eqs. (1)-(6).

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{1}$$

$$\mathrm{i}_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2}$$

$$\bar{\mathrm{c}}_t = tanh(W_{xc}x_t + W_{ch}h_{t-1} + b_c) \tag{3}$$

$$\mathrm{c}_t = f_t c_{t-1} + i_t\bar{\mathrm{c}}_t) \tag{4}$$

$$\mathrm{o}_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{5}$$

$$\mathrm{h}_t = \mathrm{o}_t * \tanh(\mathrm{c}_t) \tag{6}$$

The gateway receives a new entry point from the outside and processes the newly incoming data. The memory cell input gate receives input from the output of the LSTM neural network cell in the last iteration. The forget gate decides when to forget the output results and thus selects the optimal time delay for the input sequence. The output port receives all the calculated results and generates the output for the LSTM neural network cell [6]. LSTM models redesign computing nodes based on the structure of the RNN network.

3. Experimental Evaluation

3.1. Dataset

In this study, traffic density data of August 2020 belonging to IMM were used [20]. IMM open dataset includes time information, locations of intersections, number of vehicles based on intersections, average speed, maximum speed and minimum speed of these vehicles. The dataset used in this study contains 1.048.575 rows of records. The spatial locations of the intersections used in this study are shown in Figure 2 and the summary information of the dataset is presented in Table 1.

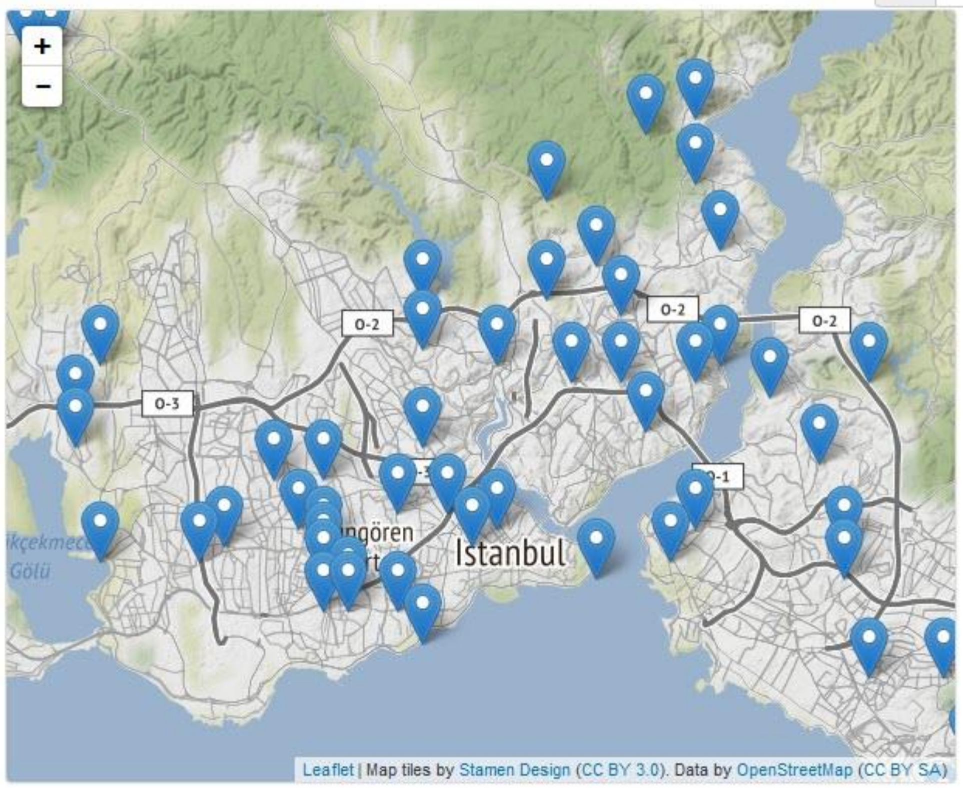

O-2
O-2
O-2
O-3
O-1
İstanbul
Bahçelievler
Küçükçekmece Gölü
Leaflet | Map tiles by Stamen Design (CC BY 3.0). Data by OpenStreetMap (CC BY SA)

Figure 2. *Location of target stations in the used dataset*

When the IMM open dataset, the general information of which is presented in Table 1, is examined in detail, an increase was observed in the number of vehicles, especially during the commuting and exiting hours, while a decrease was observed during the night hours. This paper shows that the number of vehicles in traffic changes over time. Since the traffic density changes according to time, the data were sorted on the basis of date and time before the training process. In addition, the clock data is added to the dataset as an input. In determining the number of vehicles in traffic, it is not sufficient to know only the vehicle number data of a certain time before the relevant station. An intersection is also affected by the density of intersections close to it. An intersection usually consists of 4 branches, so it is important to know the four intersections close to each intersection. Therefore, giving the number of vehicles in four intersections close to the target intersection where the traffic density will be estimated as an input to the LSTM model will strengthen the prediction ability. For this reason, the 4 intersections that are spatially closest to each intersection are determined by using the Euclidean distance as the spatial distance. After determining the four intersections, which are the closest neighbours of each intersection, the number of vehicles in the same date and time zone in these neighbouring intersections is used as input data for the intersection whose density will be estimated.

Table 1. *Data Dictionary.*

Column	Type	Tag	Definition
DATE_TIME	text	Date	It is the field that contains the date and time information. The data format is YYYY-MM-DD HH24:MI:SS. * Date breakdown is hourly.
LONGITUDE	text	Longitude	It is the field that contains the longitude information.
LATITUDE	text	Latitude	It is the field that contains the latitude information.
GEOHASH	text	geohash	Geohash Value of Latitudes and Longitudes
MINIMUM_SPEED	text	Minimum Speed	Minimum speed (km/h) in the geohash area at the relevant hour.
MAXIMUM_SPEED	text	Maximum speed	Maximum speed (km/h) in the geohash area at the relevant hour.
AVERAGE_SPEED	text	Average Speed	Average speed (km/h) in the geohash area at the relevant hour.
NUMBER_OF_VEHICLES	text	Number of Individual Vehicles	Number of different vehicles in the geohash area at the relevant hour.

3.2 Model Settings

By working on the number of layers in the LSTM model, the most suitable LSTM input layer was determined to be one output layer and 20 neurons. The structure of the LSTM model used in the study is given in Figure 3.

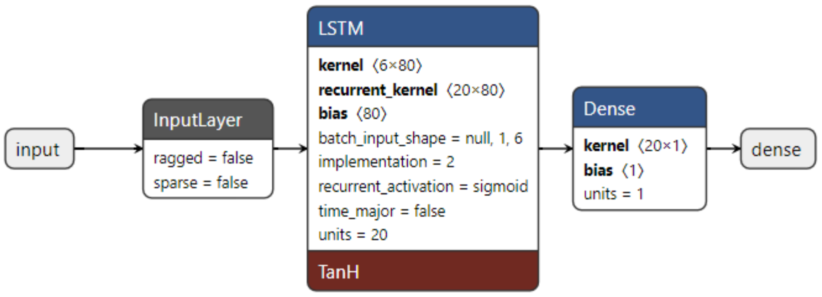


Figure 3. *LSTM model structure in the study.*

In the study with the LSTM method, the Adam algorithm was used as the optimization method. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update iterative Evaluation Metrics mesh weights based on training data.

3.3 Evaluation Metrics

In this study, MAE and RMSE measurements were used as evaluation metrics for performance comparison. The study was carried out by normalizing the data, in order not to make an incorrect comparison, the data was converted to real values by reverse scaling and evaluated.

The MAE is the mean of the absolute error between the predicted value and the true value. It ranges from zero to infinity, it is expected to give a low result. It is a linear value. The MAE formula is shown in Eq. (7). In the formula, p_i represents the predicted value and $o_i$original value.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|o_i - p_i| \tag{7}$$

The RMSE is the root mean square error between the predicted value and the true value. It ranges from zero to infinity, it is expected to give a low result. The RMSE formula is shown in Eq. (8). In the formula, p_i represents the predicted value and $o_i$original value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_i - p_i)^2} \tag{8}$$

3. Experiment Results

Experimental evaluations of linear regression, decision trees, random forest, classical deep learning, and LSTM methods were made using real data from IMM. RMSE and MAE values were used as evaluation criteria to examine the performance of the methods. The station information and the values of the methods that were experimentally evaluated are presented in Table 2 and Table 3.

When the experimental evaluation results (Table 2) created according to the MAE error values are examined, the LSTM method generally gave lower error results than the other compared methods.

Table 2. *MAE Values of the Methods.*

İstasyon Geohash	Decision Tree	Deep Learning	Linear Regression	LSTM	Random Forest
sxk3q9	5,325	3,799	4,237	3,816	**3,773**
sxk3r9	25,678	29,115	19,625	**15,152**	16,536
sxk3rf	12,864	12,387	10,574	9,96	**9,878**
sxk3w2	5,019	4,883	4,315	**4,282**	4,431
sxk3w6	39,309	32,604	25,265	**23,649**	24,429
sxk90y	17,365	16,137	14,353	**13,251**	13,531
sxk91n	5,837	5,311	5,112	4,745	**4,702**
sxk91u	9,289	7,3	7,637	**7,111**	7,493
sxk91w	8,821	6,984	6,386	**5,317**	6,939
sxk92b	10,835	**8,438**	9,205	8,753	8,764
sxk92c	7,1	6,053	5,633	**5,582**	5,998
sxk92e	7,117	**4,547**	4,592	4,606	4,757
sxk92f	8,301	6,155	6,166	**5,71**	6,012
sxk92q	8,926	8,218	8,231	**7,65**	7,739
sxk93s	13,367	13,304	11,497	**8,803**	11,49
sxk966	8,387	5,696	5,752	**5,532**	5,948
sxk96e	6,952	5,193	5,444	**5,118**	5,52
sxk96h	24,372	24,022	19,673	**17,577**	21,22
sxk99b	9,475	**7,279**	8,53	7,291	7,449
sxk99y	10,687	8,745	7,205	**6,225**	7,664

Table 3 shows statistics on the RMSE values of the methods.

Table 3. *RMSE Values of the Methods.*

Station Geohash	Decision Tree	Deep Learning	Linear Regression	LSTM	Random Forest
sxk3q9	7,429	**4,748**	5,135	4,797	4,778
sxk3r9	35,96	37,584	24,058	20,58	**20,449**
sxk3rf	17,084	16,447	13,687	13,353	**13,093**
sxk3w2	6,405	6,165	**5,317**	5,405	5,471
sxk3w6	54,811	41,776	32,909	31,34	**30,639**
sxk90y	21,559	20,193	18,814	**16,812**	17,948
sxk91n	7,658	**5,763**	6,465	6,074	5,978
sxk91u	11,643	**8,658**	9,506	8,872	9,303
sxk91w	10,953	**7,07**	7,897	7,164	8,358
sxk92b	13,44	**10,741**	11,763	11,321	11,339
sxk92c	9,411	7,757	7,387	**7,358**	7,92

sxk92e	8,838	6,276	**5,948**	6,16	5,954
sxk92f	10,008	**7,234**	7,844	7,468	7,643
sxk92q	12,272	**9,66**	10,427	9,777	10,162
sxk93s	17,194	16,429	14,337	**13,007**	14,445
sxk966	11,249	**6,577**	7,471	7,261	7,95
sxk96e	8,927	7,934	7,067	**6,706**	7,309
sxk96h	30,717	31,072	23,533	**21,894**	26,93
sxk99b	12,262	9,885	10,876	9,834	**9,803**
sxk99y	14,047	14,418	8,935	**8,205**	9,734

When the result values were examined (Table 2 and Table 3), the LSTM method made predictions with lower error rates compared to other methods. In addition, comparison of MAE values with statistical data is presented in Table 4. Standard deviation value of the LSTM method gave better results compared to other methods (Table 4). This shows that the LSTM method works more stable.

Table 4. *Comparison of MAE error values.*

Method	Total	Avg	Best	Worst	Std.
Decision Tree	245,03	12,25	5,02	39,31	8,53
Deep Leaning	216,17	10,81	3,80	32,60	8,38
Linear Regression	189,43	9,47	4,24	25,27	5,89
LSTM	170,13	8,51	3,82	23,65	5,16
Random Forest	184,27	9,21	3,77	24,43	5,65

In Figure 4, the success of LSTM is seen with the low total RMSE value in the traffic density of the stations.

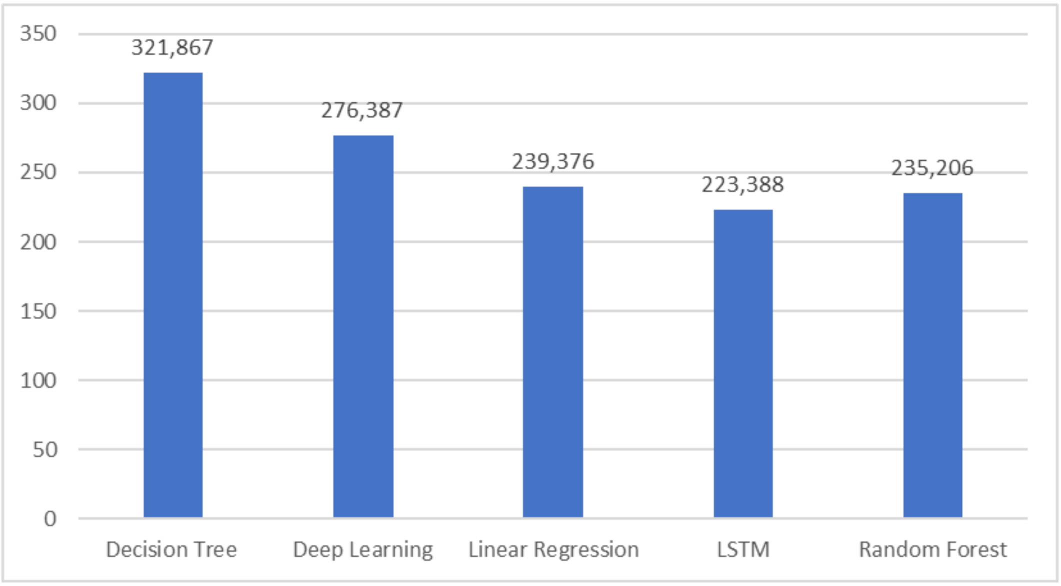


Figure 4. *Total RMSE values of the method.*

Figure 5 shows the actual number of vehicles and the number of vehicles estimated by decision trees, deep learning, linear regression, LSTM and random forest methods in the sxk99y geohash station. Comparison of methods in the sxk99y geohash station is given in figure 5.The graph with the highest overlap between the predicted value and the actual value belongs to the study using the LSTM method.

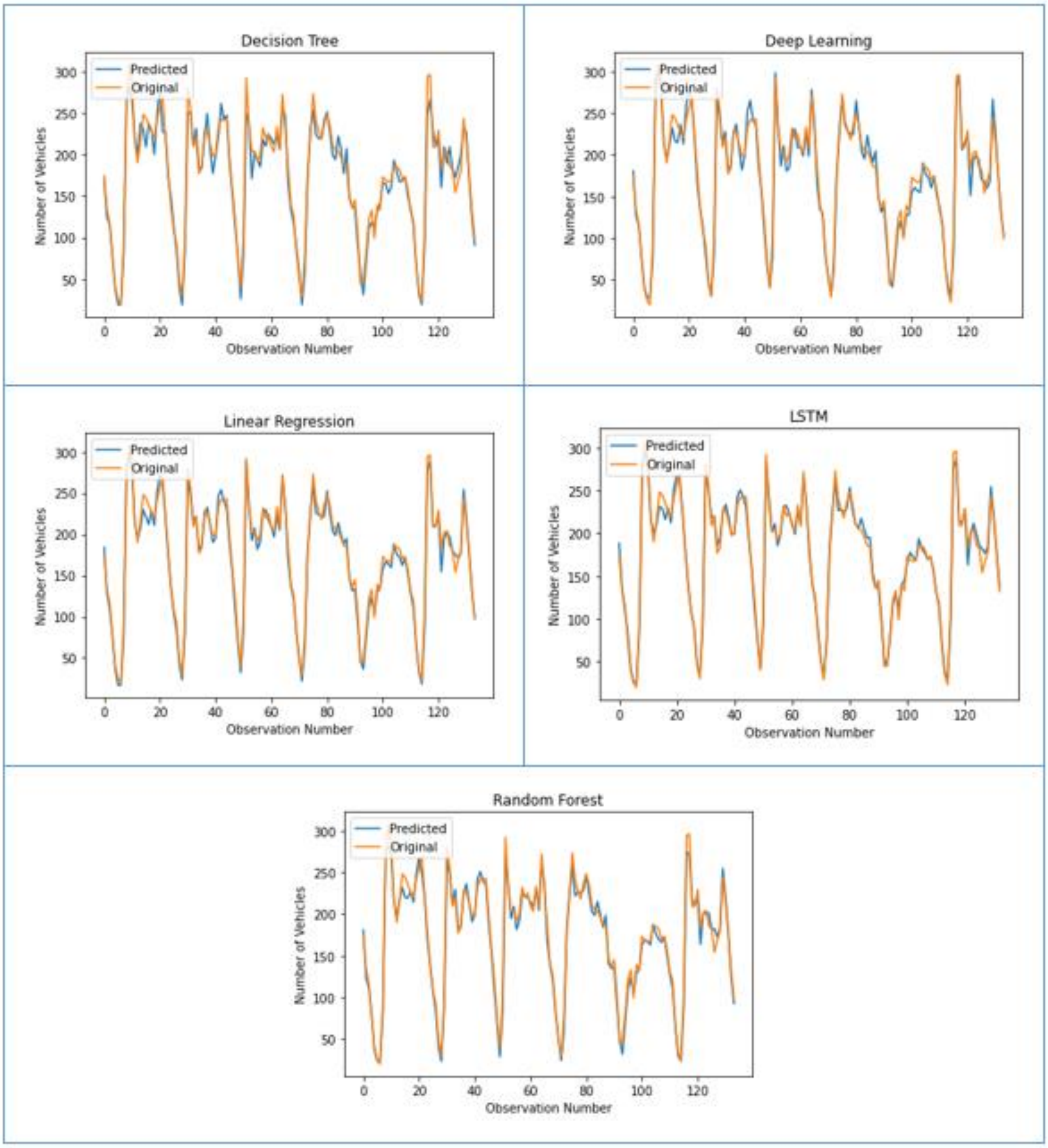


Figure 5. *Comparison of methods test and prediction data for station sxk3r9.*

Conclusion

The traffic problem takes a large part of people's time in last years. In addition, it is very important to manage traffic systems in places with high population density. Intelligent Transportation Systems (ITS) are used to manage traffic systems to overcome the traffic density problem. One of the most important components of AUS models is that traffic density can be determinated in advance. For this purpose, a model has been proposed to predict the number of vehicles in the traffic by working on the data of the city of Istanbul, which is the most crowded and has the most traffic in Turkey.

LSTM model, which is a deep learning approach, is proposed for traffic density estimation. Since LSTM is able to automatically calculate optimal time delays and capture the characteristics of longer time-lapse time series, it has achieved higher performance in traffic flow forecasting. Experimental evaluations have shown that LSTM-based methods are more successful than linear regression, decision trees and random forest, and the classical deep learning methods.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Uzun Süreli Kısa Bellek Ağı Yöntemi ile Trafik Yoğunluk Tahmini”).

We would like to thank IMM for sharing the hourly traffic density data used in this study.

References

[1] D. Ravì, C. Wong, F. Deligianni, M. Berthelot; J. Andreu-Perez, B. Lo and G. Yanget, "Deep Learning for Health Informatics," in IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 1, pp. 4-21, doi: 10.1109/JBHI.2016.2636665, Jan. 2017

[2] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," in IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, doi: 10.1109/MCI.2018.2840738, Aug. 2018.

[3] A. Şeker, B. Diri ve H. H. Balık, "Derin Öğrenme Yöntemleri Ve Uygulamaları Hakkında Bir İnceleme", Gazi Mühendislik Bilimleri Dergisi (GMBD), c. 3, sayı. 3, ss. 47-64, Aralık 2017

[4] A. J. P. Samarawickrama, T. G. I. Fernando, "A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market," 2017 IEEE International Conference on Industrial and Information Systems (ICIIS), pp. 1-6, doi: 10.1109/ICIINFS.2017.8300345, 2017

[5] N. Buduma and N. Locascio, Fundamentals of Deep Learning. Designing Next-Generation Machine Intelligence Algorithms, O'Reilly Media, 172-217, 2017.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[7] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data” Transportation Research Part C: Emerging Technologies, 54, pp. 187–197, 2015.

[8] Y. X. Tian and P. Li, "Predicting Short-term Traffic Flow by Long Short Term Memory Recurrent Neural Network", 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), pp. 153-158, 2015.

[9] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction", Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 324-328, 2016.

[10] TUİK, Adrese Dayalı Nüfus Kayıt Sistemi Sonuçları 2013, Türkiye İstatistik Kurumu Haber Bülteni, Sayı: 37210, 04.Şubat.2021.

[11] TUİK, “İllere göre motorlu kara taşıtları sayısı”, Available: https://data.tuik.gov.tr/Bulten/Index?p=Road-Motor-Vehicles-December-2020-37410, [Accessed Aralık, 2020].

[12] A Liaw, M. Wiener, 2002, Classification And Regression ByRandom Forest, R News, Vol.2/3, December.

[13] L. Breiman, “Random forests”, Machine Learning, Volume 45, pp. 5-32, 2001.

[14] N. Kriegeskorte, T. Golan, Neural network models and deep learning, Current Biology, 29(7), R231–R236, 2019.

[15] L. Deng and D. Yu, “Deep Learning: Methods and Applications,” Found. Trends® Signal Process., vol. 7, no. 3–4, pp. 197–387, 2014.

[16] K. Chakraborty, K. Mehrotra, C. K. Mohan and S. Ranka, “Forecasting The Behavior of Multivariate Time Series Using Neural Networks”, Neural Networks 5(6):961-970, 1992.

[17] Y. Duan, Y. L.V. and F. Wang, "Travel time prediction with LSTM neural network," 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1053-1058, doi: 10.1109/ITSC.2016.7795686, 2016.

[18] S. Samui, I. Chakrabarti, S.K. Ghosh, “Tensor-Train Long Short-Term Memory for Monaural Speech Enhancement” arXiv preprint arXiv:1812.10095, 2018

[19] F. A. Gers, J. Schmidhuber, F. A. Cummins, “Learning to forget: Continual prediction with LSTM” Neural Computation, 12 (10), pp. 2451-2471, 2000.

[20] Ulaşım Daire Başkanlığı, “Saatlik Trafik Yoğunluk Veri Seti, Ağustos 2020 Trafik Yoğunluk Verisi” 13 Aralık, 2020.

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 144-149, 2021
Research Article

Performance Trade-Off for Bert Based Multi-Domain Multilingual Chatbot Architectures

Davut Emre TAŞAR [1,*], Şükrü OZAN [2], Seçilay KUTAL [3], Oğuzhan ÖLMEZ [4], Semih GÜLÜM [5], Fatih AKCA [6], Ceren BELHAN [7]

[1] Dokuz Eylül University, Turkey, davutemre.tasar@deu.edu.tr
[2] AdresGezgini A.S ̧, Ar-Ge Merkezi, Turkey, sukruozan@adresgezgini.com
[3] Areal AI, U.S., secilay.kutal@areal.ai
[4] Manisa Celal Bayar University, Turkey, 172803041@ogr.cbu.edu.tr
[5] AVL Araştırma ve Mühendislik, Turkey, Semih.Gulum@avl.com
[6] Sakarya University, Turkey, mehmet.akca1@ogr.sakarya.edu.tr
[7] Software Engineering, Izmir University of Economics, Turkey, ceren.belhan@std.ieu.edu.tr

Abstract
Text classification is a natural language processing (NLP) problem that aims to classify previously unseen texts. In this study, Bidirectional Encoder Representations for Transformers (BERT) architecture is preferred for text classification. The classification is aimed explicitly at a chatbot that can give automated responses to website visitors' queries. BERT is trained to reduce the need for RAM and storage by replacing multiple separate models for different chatbots on a server with a single model. Moreover, since a pre-trained multilingual BERT model is preferred, the system reduces the need for system resources. It handles multiple chatbots with multiple languages simultaneously. The model mainly determines a class for a given input text. The classes correspond to specific answers from a database, and the bot selects an answer and replies back. For multiple chatbots, a special masking operation is performed to select a response from within the corresponding bank answers of a chatbot. We tested the proposed model for 13 simultaneous classification problems on a data set of three different languages, Turkish, English, and German, with 333 classes. We reported the accuracies for individually trained models and the proposed model together with the savings in the system resources.
Keywords: BERT; classification; chatbot; memory gain; multi-domain; multi-lingual.

1. Introduction

As a sub-discipline of artificial intelligence (AI), Natural Language Processing (NLP) aims to solve text classification, sentiment analysis, and summary information extraction using text data. Today, developments in NLP are also gaining momentum. Live chatbots produced using natural language processing have now reached human-level performance.

In [1], a chatbot model based on Long Short-Term Memory (LSTM) has been created to automatically answer frequently asked questions specific to a related industry with 83% accuracy. In their study, Ozan et al. compared the performances of LSTM, BERT, and Doc2Vec on the categorical classification problem [2]. They concluded that the most successful model was the BERT model with 93% accuracy.

In a previous study, we investigated pre-trained transformer models specific to the Turkish language. We figured out that the most successful model was loodos/bertbase-turkish-cased with a test accuracy score of 89% [3]. This study proposes a system used in a live chatbot previously designed in [4] to select the most appropriate response to a request. The BERT model is preferred as the core language model. Considering the classical understanding that has continued until today, each model trained for live chatbot architectures needs a separate area in memory. We aim to introduce a single model with the proposed model that can simultaneously solve multiple text classification problems. We saw that reducing the solution to a single model can be an effective optimization method for RAM required by numerous simultaneous chatbots.

In [5], Tellez et al. studied the sentiment analysis problem for seven different languages. They used the data collected from Twitter. Baseline for Multilingual Sentiment Analysis model (b4msa) was used as a bootstrapping sentiment classifier by fine-tuning the test results. They achieved a 79.9% accuracy.

We created some of our training data by translating our original Turkish text data to other languages using machine translation as described in [7]. In [7], Xiaochuan et al. successfully classified entry titles of different

*Corresponding author e-mail address: davutemre.tasar@deu.edu.tr

Received: 30/11/2021; Accepted: 24/12/2021

languages in Wikipedia. Even though a given title is in a different language, their proposed system could classify them according to the title subject.

In [8], Schwenk et al. performed sentiment analysis of the data in RCV2 data set using morphologically and grammarly different eight different languages. They trained and tested different language models using a balanced data set, i.e., approximately equal number of data for each language.

Bilingual bi-directional LSTM model is preferred for cross-lingual sentiment classification problems in [8]. Zhou et al. performed sentiment analysis with four different models. They combined the LSTM model with both sentence-based, word-based, and sentence-word-based attention models. They achieved the highest accuracy of 82.4% with the latter model configuration.

2. Methodology

The transformer-based BERT model was preferred to solve the problem of reducing classification problems to a single model. For the developed chatbot to serve more than one company and provide its service in more than one language, data sets belonging to different languages were studied. It is ensured that the model can make company-specific estimations. Also, the performance loss due to a single model used in the structure is minimized using the estimation function.

2.1. Dataset

To increase the complexity, we used multiple data sets for each language. We collected the primary data set of English texts. Then we generated data set for other languages by translating the primary data set using machine translation. We guaranteed our BERT classifier model to learn semantics rather than the language itself by increasing the complexity. We used three languages, English, Turkish and German, and four data sets. Four separate data sets, which will be named as "dataset1 and dataset2", "dataset3", "dataset4" and "dataset5", were included. These data sets were divided among themselves, and the number of data sets was increased to 13.

Table 1. *Example sentences according to the subsets in the dataset and ratios of sub-datasets to the whole dataset (the three dots represent the continuation of the sentence).*

Dataset Name	Sample Text	Ratio of the subdataset to the whole dataset
Dataset1GER	wie kommt ich um mein geld aus dem geldautomaten zu bekommen	%2.88
Dataset2GER	wie kann ich mein konto löschen	%2.75
Dataset3GER	grant morrison, das mit göttern spricht, ist ein dokumentierender merkmalslänge der eine…	%11.07
Dataset4GER	das abingdon und witney college ist ein welteres bildungskollegium in abingdon oxfordshire...	%13.56
Dataset1EN	i want to start using my card how do i active it	%2.88
Dataset2EN	i want to block my card or deactivate it or something it s been stolen and i don t want it misused...	%2.75
Dataset3EN	count the number of food items on list	%11.07
Dataset4EN	the saul river is a tributary of the izvorul river in romania	%13.56
Dataset5EN	i m still feeling pretty low and demotivated including ups	%9.24
Dataset1TR	size birkaç gün önce bir çek gönderdim henüz hesabıma hiçbir şey olmadı bu kabul edilemez param nerede	%2.88
Dataset2TR	sanal kart nasıl alınır	%2.75
Dataset3TR	tom un telefon numarası nedir	%11.07
Dataset4TR	pointe magazine bale dansçılarına yönelik uluslararası bir dergidir ve macfadden sonra sahne sanatları medyası...	%13.56

The data set named "dataset1 and dataset2" has 77 classes and consists of short question sentences belonging to the banking sector and the categories to which the sentences belong [9]. This data set, taken from the Huggingface datasets library, was first translated into Turkish and German with the Translator library, using the

Microsoft Translation API in Python programming language. Afterward, the translation quality of the translated sentences was checked by the article's authors on 500 randomly selected samples, and incorrect translations were removed from the data set. This way, 13083 translated short speech sentences, and their categories were obtained as a usable data set. The dataset holds some similar questions for different use cases. Hence, this data set was divided into 38 and 39 classes according to the number of classes (dataset1 and dataset2) to test the BERT model's performance correctly. We used these sets as data sets belonging to different user groups were used in the model's training.

The "dataset3" was created by selecting 14 tag classes from DBpedia 2014 [10]. Although there were 630K rows of data in the original data set, we only used 31.500 rows, which ensured the data distribution was more balanced.

The data set named "dataset4" has 18 classes. It is categorized as a database of commands for human-robot interaction [12], containing 25.716 labeled commands. The data set named "dataset5" is a data set prepared for sentiment analysis and includes a total of six classes according to the emotional states of the texts in English [11].

It was noticed that there were too many wrong results in translating from Turkish to German. Hence, as shown in Figure 5, only dataset5 was left in its original language and was not translated into Turkish or German by machine translation. In addition, the regular distribution observed among the languages in the data set cannot be seen among the 5 data sets taken as a basis in the translation phase.

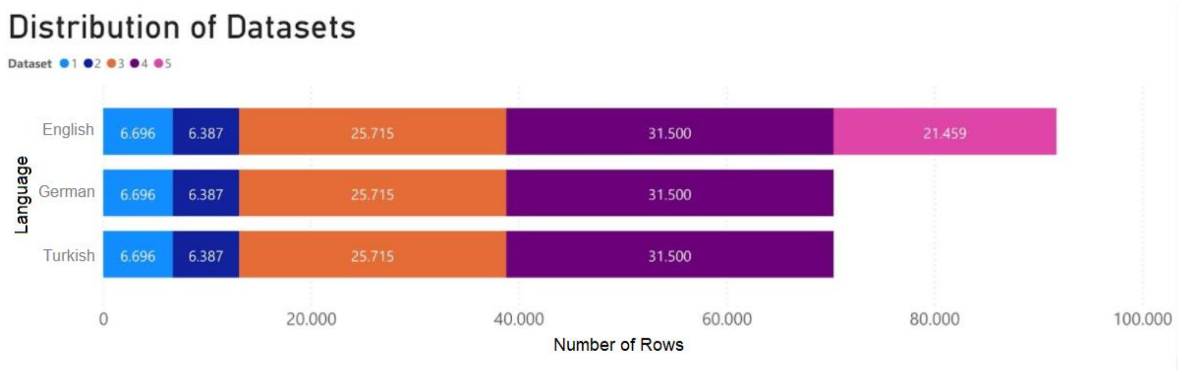


Figure 1. *The number of data contained in each dataset used in training and treated as a separate chatbot company.*

All sub-datasets are split at 70% train - 30% test data ratios. The data sets separated as trains were combined among themselves, and the data sets separated as tests were combined among themselves to form train and test data sets used in model training. In this way, in the model where all data sets are trained together, the distribution of the data sets is ensured to be the same as in the models trained separately for each data set.

2.2. BERT Method

BERT [13], a model developed by Devlin et al., was used to solve the classification problems in the article. The training of the BERT model, which is a bidirectional transformer model consisting of 12 transformer blocks, 12 self-attention heads, and 768 hidden layers, consists of two stages. In the first stage, the pre-training, which is carried out with high-dimensional data, the model is expected to learn the language structure. In the second stage, fine-tuning, the model is retrained with the data set based on the problem. In this process, the model creates a problem-specific attention mechanism and reinforces its predictions about the problem.

We used a pre-trained multilingual BERT model [13] in our study. It can serve more than one language within the chatbot architecture. We previously hit the highest performance in the categorical classification problem in [3] with the corresponding model. This model has been pre-trained with Wikipedia data on 104 languages. We fine-tuned the model with 13 datasets explained in Section 2.1. The models were trained for five epochs, and the training parameters were kept constant for each training.

2.3. Decision Function

Each model takes up approximately 2 GB of memory when transferred to memory. This memory requirement differs with respect to the model architecture. In cases where more than one model needs to be used, as in this study, this number is multiplied by the number of models used to calculate memory usage. For this reason, as the number of models used increases, the area used in memory also increases.

$$Memory\ Space\ Gain = x_{size}(n_{model} - 1) \tag{1}$$

The memory space gain provided by the method proposed in the study is shown in (Equation 1). Memory space gain is calculated by multiplying the number of models (n_{model}) minus one by the size of the fine-tuned BERT model (x_{size}).

Considering a structure that provides chatbot service to more than one user, it is possible to access the information about which problem came from which user. In the study, the classes that the created model will predict are limited with this information. The proposed single model approach finds the correct output class by applying softmax within a corresponding dataset's masked output.

3.Results

At the stage of evaluating the performance of the models whose training has been completed, the complexity matrix, which is formed by binary classification according to the number of correct and incorrect predictions in the test data, is taken as a basis. If the value predicted by the model is correct, the True value increases. If the true value and the predicted value are positive, the number of True Positive (TP) increases, and if both are negative, the number of True Negative (TN) increases. However, if the class predicted by the model is wrong, the value of False increases. While the true value is positive, the False Negative (FN) value increases when the estimated value is negative, while the False Positive (FP) value increases when the true value is negative and the predicted value is positive.

The performance of the models was evaluated considering the accuracy and F1 score metrics calculated over the complexity matrix. Accuracy (Equation 2) expresses what percentage of the predictions produced by the model are correct. F1 score (Equation 3), which is frequently preferred in the evaluation of multiple classification models, is a metric in which not only the correctness of the classification outputs but also the incorrect predictions of the model can be expressed. It is calculated over the harmonic mean of the model's precision (Equation 4) and recall metrics (Equation 5). Instead of calculating the F1-score over all the predictions of the model regardless of the classes of the problem, the averages of the F1-score values calculated for the classes are summed to obtain a macro F1-score (Equation 6). Thus, it can be ensured that each class is evaluated as a separate binary classification problem. In order to make the F1-score better able to express the class distribution, the weighted F1-score (Equation 7) can be calculated by taking into account the distribution ratios of the classes in the dataset.

$$Accuracy = \sum(TP + TN)/\sum(TP + TN + FP + FN) \tag{2}$$

where TP is True Positive count, TN is True Negative count, FP is False Positive count and FN is False Negative count.

$$F1 - score = (2 * Precision * Recall)/(Precision + Recall) \tag{3}$$

where Precision and Recall are defined by,

$$Precision = \sum TP\ /\ \sum(TP + FP) \tag{4}$$

$$Recall = \sum TP\ /\ \sum(TP + FN) \tag{5}$$

$$Macro\ F1 - score = \sum_0{}^i F1_i\ /i \tag{6}$$

$$Weighted\ F1 - score = \sum_0{}^i F1_i * DistributionRatio_i\ /i \tag{7}$$

where i is the count of classes of the classification problem and F1 is F1-score.

14 models, which were trained in 5 terms, were tested on training and test data sets. The mentioned performance metrics were obtained for each period of the models and the results are given in Table 2.

During the estimation of the model, the contribution of the estimation function, which includes preliminary information about which company the data sets belong to, and thus provides the opportunity to make a prediction specific to the data set, to the multilingual model is also examined. The trained model was tested on a fixed test data set with and without using the estimation function. 80% accuracy was obtained with the test performed without utilizing the estimation function. The accuracy increased to 91% when the estimation function was used.

When both accuracy values were compared, it was concluded that the performance of the model increased significantly by the masking method.

Table 3. *Results of trained models by various metrics.*

dataset		epoch	train dataset			test datasetset		
			accuracy	macro f1	weighted f1	accuracy	macro f1	weighted f1
German	dataset1	1	0.81	0.80	0.80	0.78	0.76	0.77
		2	0.94	0.94	0.94	0.88	0.87	0.88
		3	0.97	0.97	0.97	0.89	0.89	0.89
		4	0.99	0.99	0.99	0.91	0.91	0.91
		5	0.99	0.99	0.99	0.91	0.91	0.91
	dataset2	1	0.79	0.76	0.77	0.75	0.74	0.74
		2	0.91	0.9	0.91	0.85	0.84	0.84
		3	0.97	0.96	0.96	0.88	0.88	0.88
		4	0.99	0.99	0.99	0.89	0.89	0.89
		5	0.99	0.99	0.99	0.9	0.9	0.9
	dataset3	1	0.97	0.97	0.97	0.96	0.96	0.96
		2	0.98	0.98	0.98	0.96	0.96	0.96
		3	0.99	0.99	0.99	0.97	0.97	0.97
		4	1	1	1	0.98	0.98	0.98
		5	1	1	1	0.98	0.98	0.98
	dataset4	1	0.87	0.83	0.86	0.84	0.79	0.83
		2	0.91	0.89	0.91	0.86	0.83	0.86
		3	0.95	0.94	0.95	0.88	0.85	0.88
		4	0.97	0.97	0.97	0.88	0.86	0.89
		5	0.98	0.98	0.98	0.89	0.87	0.89
English	dataset1	1	0.87	0.85	0.86	0.84	0.83	0.84
		2	0.96	0.96	0.96	0.92	0.92	0.92
		3	0.97	0.97	0.97	0.92	0.92	0.92
		4	0.99	0.99	0.99	0.94	0.94	0.94
		5	1	1	1	0.94	0.94	0.94
	dataset2	1	0.86	0.85	0.85	0.84	0.83	0.83
		2	0.95	0.94	0.95	0.9	0.9	0.9
		3	0.98	0.98	0.98	0.92	0.92	0.92
		4	0.99	0.99	0.99	0.93	0.93	0.93
		5	0.99	0.99	0.99	0.93	0.93	0.93
	dataset3	1	0.99	0.99	0.99	0.98	0.98	0.98
		2	0.99	0.99	0.99	0.98	0.98	0.98
		3	0.99	0.99	0.99	0.98	0.98	0.98
		4	1	1	1	0.98	0.98	0.98
		5	1	1	1	0.98	0.98	0.98
	dataset4	1	0.87	0.85	0.86	0.9	0.88	0.9
		2	0.96	0.96	0.96	0.93	0.92	0.93
		3	0.97	0.97	0.97	0.95	0.94	0.95
		4	0.99	0.99	0.99	0.96	0.96	0.96
		5	1	1	1	0.97	0.96	0.97
	dataset5	1	0.93	0.9	0.93	0.9	0.87	0.9
		2	0.94	0.91	0.94	0.92	0.88	0.91
		3	0.95	0.92	0.95	0.92	0.89	0.92
		4	0.96	0.95	0.96	0.92	0.88	0.92
		5	0.98	0.96	0.98	0.93	0.9	0.93
Turkish	dataset1	1	0.79	0.77	0.78	0.75	0.74	0.74
		2	0.9	0.89	0.9	0.85	0.84	0.84
		3	0.97	0.97	0.97	0.89	0.89	0.89
		4	0.99	0.99	0.99	0.91	0.91	0.91
		5	0.99	0.99	0.99	0.92	0.92	0.92
	dataset2	1	0.71	0.66	0.68	0.68	0.63	0.65
		2	0.9	0.89	0.9	0.85	0.84	0.85
		3	0.96	0.95	0.96	0.87	0.87	0.87
		4	0.98	0.98	0.98	0.88	0.89	0.88
		5	0.99	0.99	0.99	0.9	0.9	0.9
	dataset3	1	0.83	0.83	0.83	0.81	0.81	0.81
		2	0.85	0.85	0.85	0.82	0.82	0.82
		3	0.86	0.86	0.86	0.83	0.83	0.83
		4	0.87	0.87	0.87	0.84	0.84	0.84
		5	0.87	0.87	0.87	0.84	0.84	0.84
	dataset4	1	0.87	0.85	0.87	0.84	0.82	0.84
		2	0.91	0.89	0.91	0.87	0.85	0.87
		3	0.94	0.93	0.94	0.88	0.86	0.88
		4	0.97	0.96	0.97	0.89	0.87	0.89
		5	0.98	0.98	0.98	0.9	0.88	0.9
Whole Dataset (with the masking method)		1	0.82	0.47	0.81	0.8	0.46	0.79
		2	0.88	0.7	0.87	0.85	0.67	0.85
		3	0.92	0.82	0.91	0.88	0.78	0.88
		4	0.93	0.87	0.93	0.89	0.83	0.89
		5	0.94	0.90	0.94	0.89	0.85	0.89

4. Conclusion

The study aims to develop a solution to reduce the memory space used in a chatbot architecture that can serve more than one company and has more than one language. In this direction, a masking method has been developed with prior knowledge of the class ranges specific to the problem (firm) to be estimated. By including the masking method in the estimation phase of the trained model, a single model to be used in the chatbot architecture with the technique mentioned above consumes less memory space than more than one model, providing similar performance. For this purpose, a single BERT model containing different classification problem data was compared with other BERT models that were fine-tuned for the problem. The selection of classification problems to be used in the study aimed to emphasize the usability of the developed structure in the outside world by selecting problems in similar areas.

We achieved an average accuracy of 92%. The model, which included all data sets and did not use prior knowledge of the problem, performed 80% estimation. However, with the solution found, the model trained with the data sets of all problems was asked to make predictions with the help of prior knowledge instead of making predictions directly.

While five models trained based on the problem take up 10 GB, a single model trained as a result of combining the datasets takes up only 2 GB. While switching to the single model, we compromised only 1.2% of the accuracy value. In comparison, 80% saving in memory space is achieved. Thanks to the structure created, the number of problems handled and the memory gain change can be seen in Equation 1.

To enrich the work and to enlarge its scope, it can be tried to increase the problem's difficulty level by adding different languages and different data sets. In addition, the results obtained with a different architecture to be established can be evaluated by adding a structure consisting of attentions based on words or sentences after the BERT model by going over the 1.2% loss in the accuracy value. By doing translations into different languages, data sets can be added to other languages , and the language diversity in the study can be increased. Instead of the BERT model used in the study, a narrower analysis can be performed with an LSTM structure with smaller model size.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Çok Alanlı Chatbot Mimarilerinde Avantajlı Performans ve Bellek Takası”).

References

[1] P. Muangkammuen, N. Intiruk, and K. R. Saikaew, “Automated Thai-FAQ Chatbot using RNN-LSTM,” 2018 22nd International Computer Science and Engineering Conference (ICSEC), 2018.

[2] S. Ozan and D. E. Tasar, “Auto-tagging of short conversational sentences using natural language processing methods,” 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021.

[3] D.E. Taşar, Ş. Ozan., U. Özdil, M.F.Akca, O. Ölmez, S. Gülüm, S. Kutal, and C. Belhan, “Auto-tagging of Short Conversational Sentences using Transformer Methods”. arXiv preprint arXiv:2106.01735, 2021.

[4] D.E. Taşar, Ş. Ozan., M.F.Akca, O. Ölmez, S. Gülüm, S. Kutal, and C. Belhan,“Çok Alanlı Chatbot Mimarilerinde Avantajlı Performans ve Bellek Takası”, presented at ICADA, Online, 26-28 Nov. 2021.

[5] E. S. Tellez, S. Miranda-Jiménez, M. Graff, D. Moctezuma, R. R. Suárez, and O. S. Siordia, “A simple approach to multilingual polarity classification in Twitter,” Pattern Recognition Letters, vol. 94, pp. 68–74, 2017.

[6] X. Ni, J.-T. Sun, J. Hu, and Z. Chen, “Cross lingual text classification by mining multilingual topics from Wikipedia,” Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11, 2011.

[7] H. Schwenk and X. Li, “A corpus for multilingual document classification in eight languages”, arXiv preprint arXiv:1805.09821, 2018.

[8] X. Zhou, X. Wan, and J. Xiao, “Attention-based LSTM network for cross-lingual sentiment classification,” Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016.

[9] I. Casanueva, T. Temcinas, D. Gerz, M. Henderson ve I. Vulic, "Efficient Intent Detection with Dual Sentence Encoders," 2020.

[10] Lehmann, Jens, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann et al. "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia." Semantic web 6, no. 2 (2015): 167-195.

[11] Xingkun Liu, Pawel Swietojanski, and Verena Rieser. "Benchmarking Natural Language Understanding Services for building Conversational Agents." . In Proceedings of the Tenth International Workshop on Spoken Dialogue Systems Technology (IWSDS) (pp. xxx–xxx). Springer, 2019.

[12] Ishant, “Emotions in text,” Kaggle, 18-Nov-2020. [Online]. Available: https://www.kaggle.com/ishantjuyal/emotions-in-text. [Accessed: 12-July-2021].

[13] J. Devlin, M.-W. Chang, K. Lee ve K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 150-159, 2021
Research Article

Matching Potential Customers and Influencers for Social Media Marketing

Fatih SOYGAZİ [1,*], Muhammet Enes AYDOĞAN [1], Hilmi Can TAŞKIRAN [1], Özgür KAYA [2]

[1] Aydın Adnan Menderes Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Türkiye
[2] Fenomio Influencer Marketing, Türkiye

Abstract
Social media platforms are so important for the advertising industry. Companies have a huge amount of budget for advertisement and try to select an influencer as the face of their brand for these advertisements. Each brand is related to a specific segment of customers. When the true influencer is followed by this segment, advertising companies contact him/her. The objective of this work is to facilitate the job of the advertising company by matching the brand and the influencer to use the budget of the advertising company appropriately. Accordingly, our work makes an analysis of real/fake account detection, gender, and age range prediction of the influencer's followers. In this work, it is focused on the real accounts by eliminating the fake ones and the gender, age-range prediction of these real accounts is considered. The detection of fake accounts is transformed into a binary classification problem by observing the features of real and fake accounts. Another binary classification solution is presented for gender detection by checking the pictures of the account owners and their names together. A pre-trained deep learning model for follower age range prediction is provided based on the pictures of these followers. The accuracy of the predictions is evaluated for each of the three situations and the success of our approach is observed for influencer/follower matching.
Keywords: Deep learning; fake account detection; image processing; machine learning; social media analysis.

1. Introduction

Marketing and sales via social media have been so popular recently. It stands out that social media seriously affects users' product habits considering how advertisements on social media sites change the market shares in e-commerce [1], [2], [3]. Companies are now focusing on social media marketing rather than traditional marketing methods in order to increase their product sales [4], [5], [6]. This method, called Influencer Marketing, is the process of making product promotions and the content marketing strategy by establishing relations with the relevant influencer in order to deliver products or services to more users. It is of great importance to choose the correct advertising face, as determining a customer profile on social media and trying to access customers in this profile with a suitable influencer will increase the sales performance. At this point, the job of advertising companies is to find the most appropriate influencer with the help of applications that analyze social media.

Our work, which aims to make influencer/potential customer matching, is an application that analyzes Instagram followers of the potential influencers. In this study, it is determined how many of the followers of an influencer involve fake accounts firstly. After filtering the fake accounts of the influencer, the gender and the demographic data of the followers are considered. Influencer/potential customer matching is done by using image processing, text processing, and classification algorithms. If a corporation requires advertisement via an influencer on Instagram, it can be predicted whether the sales of its products will be parallel with their targets by using our solution.

The contribution of our work is:

- To match the brands and influencers after filtering the fake accounts and obtaining a demographic view of the influencer's followers
- To propose some metrics for understanding the matching prediction of the influencer and the brand more effectively

2. Related Work

The impact of social media has also caused the increase of the influencers publishing diverse area of content to expand their followers. At that point, e-commerce companies are struggling with finding the proper influencer for their brand and it is the biggest challenge for most of the brands from the view of advertisement [7]. The brands and the especially influencers' accounts must be represented to make this matching with a mathematical model. The social account's history of an influencer must be fetched with his/her information and stored in a social information pool to obtain a general knowledge about that influencer to match with brands. Gan et. al. [8] proposed a multi-modal social account representation involving the history pooling of the account and the social

*Corresponding author e-mail address: fatih.soygazi@adu.edu.tr

Received: 02/12/2021; Accepted: 27/12/2021

account embedding to represent the influencer. Then, they proposed a ranking strategy inspired from engagement metric named as competence score to understand the matching rate between influencers and brands. Wang et. al. [9] proposed two adaptive learned metrics, endorsement effect score and micro-influencer influence score, following the approach in [8]. [9] considered more interpretable concept-based parameters for marketing decisions and applied their experiments on a real-world dataset.

The detection of fake accounts called bots that do not express real identity is important in terms of understanding real behavior in social media. Akyon and Kalfaoğlu [10] created two datasets consisting of real and fake accounts. They detected fake accounts at 86% and 96% accuracy by the traditional machine learning methods they tried on these datasets. They obtained trained models based on different features such as the number of people followed by the user, the number of people following the user, the length of the username, the number of numerical values in the username to detect real and fake accounts.

Jeon et al. [11] aimed to determine the gender of users with deep learning methods in order to be used in the field of advertising. Gender detection was made by analyzing photos and daily activities over 33752 Instagram posts. It was stated that an F1-score of 76% was achieved over the features extracted from the posts.

Han et al. [12] developed an approach that predicts the age range from the posts, based on the behavior of young people and adults on social media. The online behavior information of the users is stored in two datasets obtained by user-profiles and the tags of these profiles. It was stated that estimation was made at an accuracy level of 82% by the training models with these datasets.

Companies interested in influencer marketing benefit from various applications [1,2]. These applications collect various information about profiles by making analyzes with artificial intelligence-based approaches and offer the most suitable influencer alternatives for the demanding companies.

Farseev et al. [13] developed an artificial intelligence-based application about influencer discovery, which they named SoMin. The study focuses on analyzing the user profile in the relevant company's market and finding the near- or long-term influencer matches suitable for this profile. They produced a solution called "Individual User Profiling" in order to find out what the attributes of the people who will follow the product will be. They observed all the multimedia shares made on social media regarding the market where the product is located when offering their solution.

The quality of matching potential customers of a product and the influencer must be evaluated for having an intuition about the future income of the product’s corporation. Hence, there should be some metrics to handle the quality of the matching process. Engagement rate [14], [15], [16], is a metric that is calculated as the number of users that have interacted with a post (whether they liked, shared, commented, or clicked on the photo or link) divided by the number of the influencer’s followers. Sponsored pictures have an impact on the purchase of the customers. Naumanen and Pelkonen [15] propose that the purchase behavior of the followers is affected positively by the sponsored product pictures. They call the influencers micro-influencers having a small number of followers but these influencers have more impact on specific product categories. The engagement rate must be considered to find out the micro-influencers on Instagram.

3. Methodology

Our methodology includes two phases. In the first phase, we detect the fake accounts of the influencer’s followers as some of the influencers pay for fake accounts and increase the follower count to be more popular. We must initially ensure that the influencer is followed by real accounts that may be analyzed. After eliminating the fake accounts, we have two prediction operations for classifying the portfolio related to the influencer: gender and age range. These are the two basic properties of the followers to predict their daily expectations about various products. We measure the quality of these classifications via some metrics explained below.

3.1. Fake account prediction

In the detection of fake accounts, a dataset containing the information of real and fake users is created and model training is carried out with this data set. Our dataset contains the following information:

- Follower: The number of followers of the Instagram user.
- Following: The number of accounts followed by the Instagram user.
- Biography Length: The number of characters of the information written in the biography section of the Instagram user's profile.

[1] https://web.archive.org/web/20211223132633/https://hypeauditor.com/

[2] https://web.archive.org/web/20211223133347/https://analisa.io/

- Total Post: The total number of posts made by the Instagram user.
- Has Profile Picture: The information of whether the Instagram user has a profile photo or not. If there is a profile photo, it says "1", if not, it says "0".
- Has External Url: The information about whether the Instagram user has added url information to his profile. If there is url information in their profile, it says "1", if not, "0" is written.
- Full Name Length: The character length of the Instagram user's name and surname.
- Digit Count: The information of how many numbers the Instagram user has in his username.
- Username Length: The information of how many characters the username of the Instagram user consists of.
- FFR: The most important attribute we use for fake account prediction is the ratio of the number of people following to the number of people following (FFR). If FFR is so high, it is an indicator of a potential fake account. It means that a few users or nearly nobody follow this user whereas that account is following lots of accounts.

FFR = Number of people followed / Number of people following

The model is then trained using a data set that includes both real and false account information. The list of Instagram influencers' followers is then gathered, and a data set comprising the information about their followers is constructed. The accuracy of the obtained users is then assessed using the trained model's prediction. As a result, the accounts that we've identified as fake are found, and the phenomenon's profile performance is assessed. As a result of binary classification with Random Forest Classifier during model training, fake and real accounts are separated. An example of a dataset containing real and fake user information that we have prepared is shown below.

Follower	Following	Biography Length	Total Post	Has Profile Pic	Has Ext Url	Full Name Length	Digit Count	User Name Length	FFR	IsFake
145	142	17	0	1	0	17	0	10	0.97	0
288	279	16	0	1	0	16	0	10	0.96	0
420	435	17	0	1	0	17	0	10	1.03	0
183	201	0	0	1	0	0	0	10	1.09	0
208	252	13	0	1	0	13	1	14	1.21	0
0	115	0	6	1	0	0	4	16	115.0	1
0	120	0	6	1	0	0	4	23	120.0	1

Figure 1. *A fragment of the dataset we construct containing real and fake account information.*

3.2. Gender detection and age-range prediction

In our study, two different methods are used for gender prediction to validate the predicted gender information of each account more accurately. Firstly, we predict the gender of the users with image processing. DeepFace[3] library is used as an image processing algorithm in this paper. It is a hybrid face recognition framework wrapping state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace, and Dlib. Google FaceNet, VGG-Face, ArcFace, and Dlib are overperforming ones based on experiments. Scores of these models are available in both the Labeled Faces in the Wild and YouTube Faces in the Wild datasets as announced by their creators. The default configuration of DeepFace uses the VGG-Face model [17] and we used DeepFace with that configuration. It is stated that this hybrid model overcomes the accuracy of the human recognition of a face with 97.53% accuracy [18]. The gender and age of the account owners are determined through the Instagram profile picture with DeepFace library. While this library works slower and some of the names give nearly exact information about the gender of a user, a name-based predictor is also used for prediction

[3] https://web.archive.org/web/20211223131917/https://github.com/serengil/deepface

to fasten and verify the overall process. Python's gender_guesser[4] module is our auxiliary solution to predict a gender by name. Gender analysis is performed on the name given by the user in the Instagram profile with the gender_guesser module.

After these analyzes are done, the results of two different methods are compared. When the same outputs are obtained, these results are recorded in the database about an account. In this way, our algorithms work faster by not making repeated analyses for the same user. As image processing via DeepFace takes a huge amount of time for analysis of a set of users, we prevent applying the procedure for the same users repeatedly by storing their information. Therefore, each user may be fetched for different matching operations by skipping these phases and accessing their information from the database.

4. Proposed Metrics

The engagement rate [6] is the most commonly used metric to evaluate the matching quality of the influencers with the potential customers. The metrics shown below are novel that we propose to check the success of matching operations in social media. These are;

Alternative engagement rate*:* It calculates the influencer's interaction rate on Instagram.

Alternative engagement rate = (total likes count + total comments count) /
*(total followers count * total post count)* (1)

Average like rate*:* It calculates the average number of likes given to the posts of the influencer.

Average like rate = total likes count / total post count (2)

Average comment rate: It calculates the average number of comments written under the posts of the influencer.

Average comment rate = total comments count / total post count (3)

Like rate: It calculates the percentage of the average like rate with respect to the total number of followers.

Like rate = average like rate / total followers' count (4)

Comment rate: It calculates the percentage of the average comment rate with respect to the total number of followers.

Comment rate = average comment rate / total followers' count (5)

Ghost follower rate: There is another concern that needs to be dealt with besides detecting fake accounts. Some of the accounts might be stolen and managed like it is the real owner of the account. In another case, some commercial accounts are compulsorily followed to have a chance of winning an award from a website or that account. In these cases, the follower is not actually interested in the account it follows currently. We propose a metric called ghost follower rate to understand the number of ghost followers of an account. Initially, we gather the list of accounts that like the posts of the influencer and the list of the followers. The accounts which are in the follower list but not in the liked post list are called ghost followers.

Ghost follower rate = (total users who like posts - total followers count) /
*(total followers count * total post count)* (6)

4. Results

For the fake account analysis method used in this study, it was first thought to determine the number of likes, comments and story views in Instagram photos. However, the likes, comments, and stories may be generated automatically and be purchased. Hence, observing these attributes affects the analysis negatively. Therefore, the information of the influencer's followers is analyzed and the number of fake followers is determined with our proposed method.

The configuration of the computer where tests are performed in Section 4.1 is:

- Operating System: macOS BigSur
- CPU: Apple M1 chip 8-core CPU with 4 perform-ance cores and 4 efficiency cores
- GPU: Apple M1 chip 7-core GPU, 16-core Neural Engine

[4] https://web.archive.org/web/20211223130832/https://test.pypi.org/project/gender-guesser

- RAM: 8GB

The configuration of the computer where tests are performed in Section 4.2 and Section 4.3 is:

- Operating System: Windows 10 Pro 64 bit
- CPU: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs)
- GPU: NVIDIA GeForce GTX 1050 Ti
- RAM: 16GB

4.1. Tests about fake account prediction

An account with 298 followers was split into 234 real followers and 64 fake accounts. The training time of the model is 1.5 sec while the testing time is 0.51 for the prediction phase. After performing the test process on this account with our bot analysis algorithms, 230 accounts are detected (98.3% accuracy) as real and 61 accounts (89.1% accuracy) are detected as fake.

4.2. Tests about gender detection

The number of predicted genders in the dataset involving the followers of an account is not so high compared to the promise of DeepFace as seen in Table 1. DeepFace actually uses the most prominent algorithms and works hybrid for detection. The problem is not related to the algorithms indeed. The deep learning algorithms trained with the given datasets are mostly constituted with preprocessed images whereas the profile pictures of the accounts on Instagram are more complex for prediction. The profile pictures might not involve the face of the account owner or the face of the account owner might be captured from an angle to be predicted accurately. Hence, it is not simply enough to capture the gender profile of the followers of an influencer. Average processing time involves the time of fetching the personal information of the follower and the prediction time of the follower's gender together. The cause of the increase for operation time is related to accessing and fetching the data from Instagram. When it is a time-consuming scheme, we collect each user's information and store it not to process the same user repeatedly.

Table 1. *The results of DeepFace in our dataset.*

# of followers	Avg. processing time including Instagram access (sec)	Avg. detection time
100	141	67
250	316	169
500	736	378

Table 2 indicates the number of predicted genders of the followers with the help of their given names on their accounts. The composing solution approach from the outputs in Table 1 and Table 2 presents us better results to have an intuition about the portfolio of an influencer. The problem in this processing phase is the same in Table 1. It takes a long time to access and gather data from Instagram compared to processing of the names.

Table 2. *The results of the gender_guesser module in our dataset.*

# of followers	# of detected genders of followers	Avg. process time (sec)
100	67	0.4
250	184	0.43
500	357	0.47

The gender detection results via DeepFace and gender_guesser are seen in Figure 2 and Figure 3 in detail.

4.3. Tests about age range detection

Age detection is another important issue for matching the influencers and the potential customers. Because each person within an age range has its own hobbies or interesting products. The advertising companies have a list of profiles to match the products and the potential customers. Age range is an important attribute for these profiles as it is clearly seen. Figure 4 presents the detected age ranges of the users on our dataset collected from Instagram.

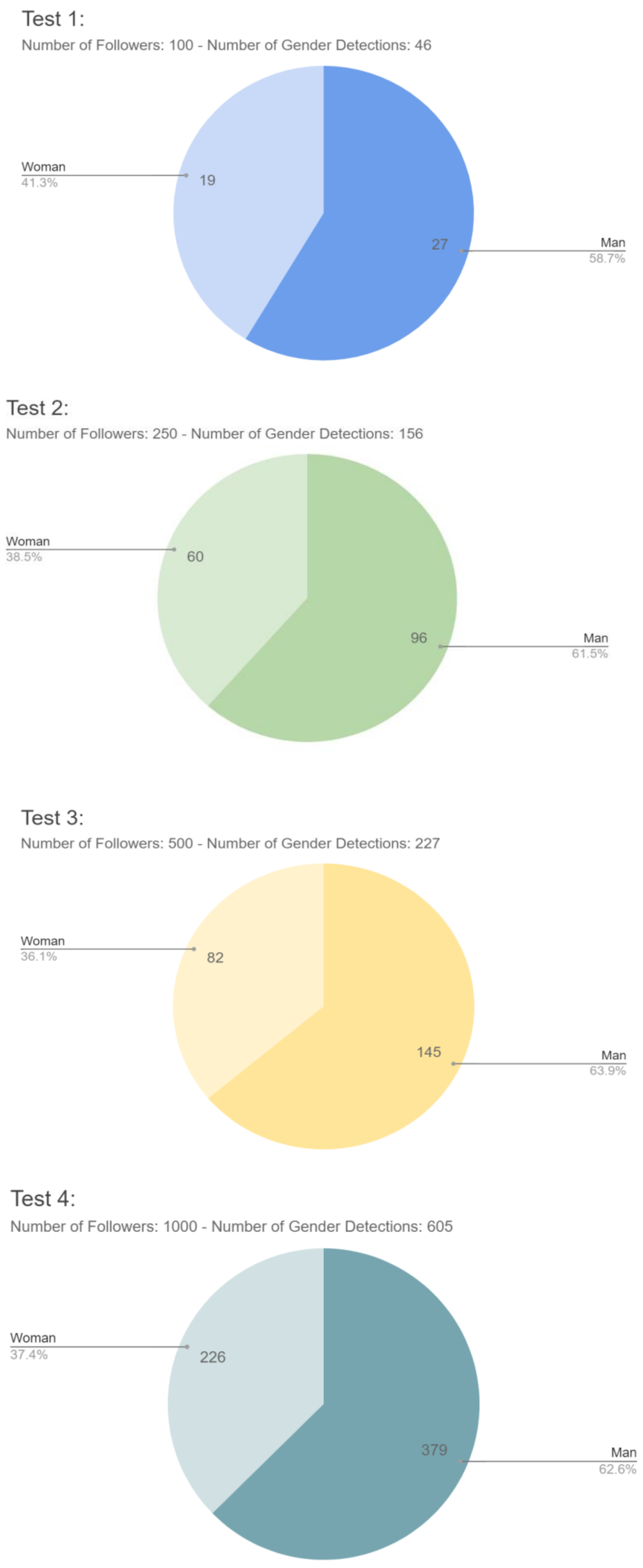


Figure 2. *Tests about gender detection via DeepFace*.

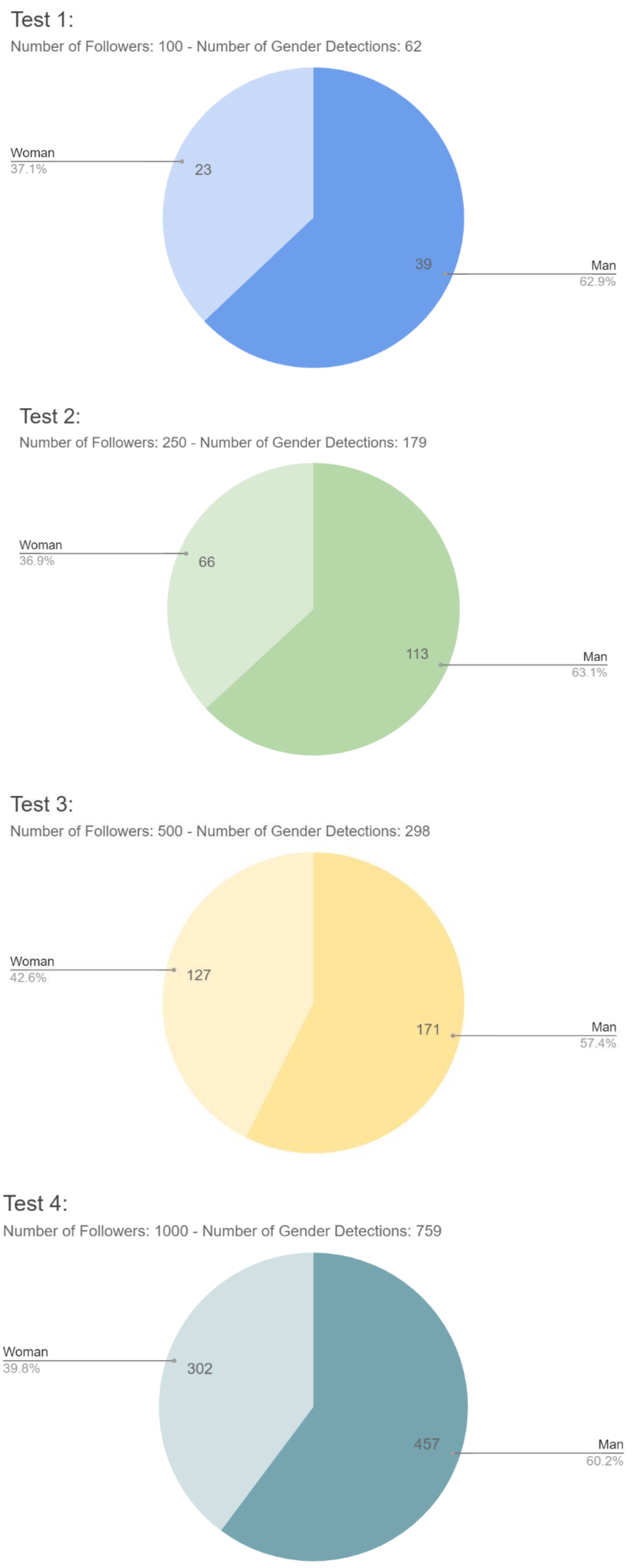


Figure 3. *Tests about gender detection via gender_guesser*.

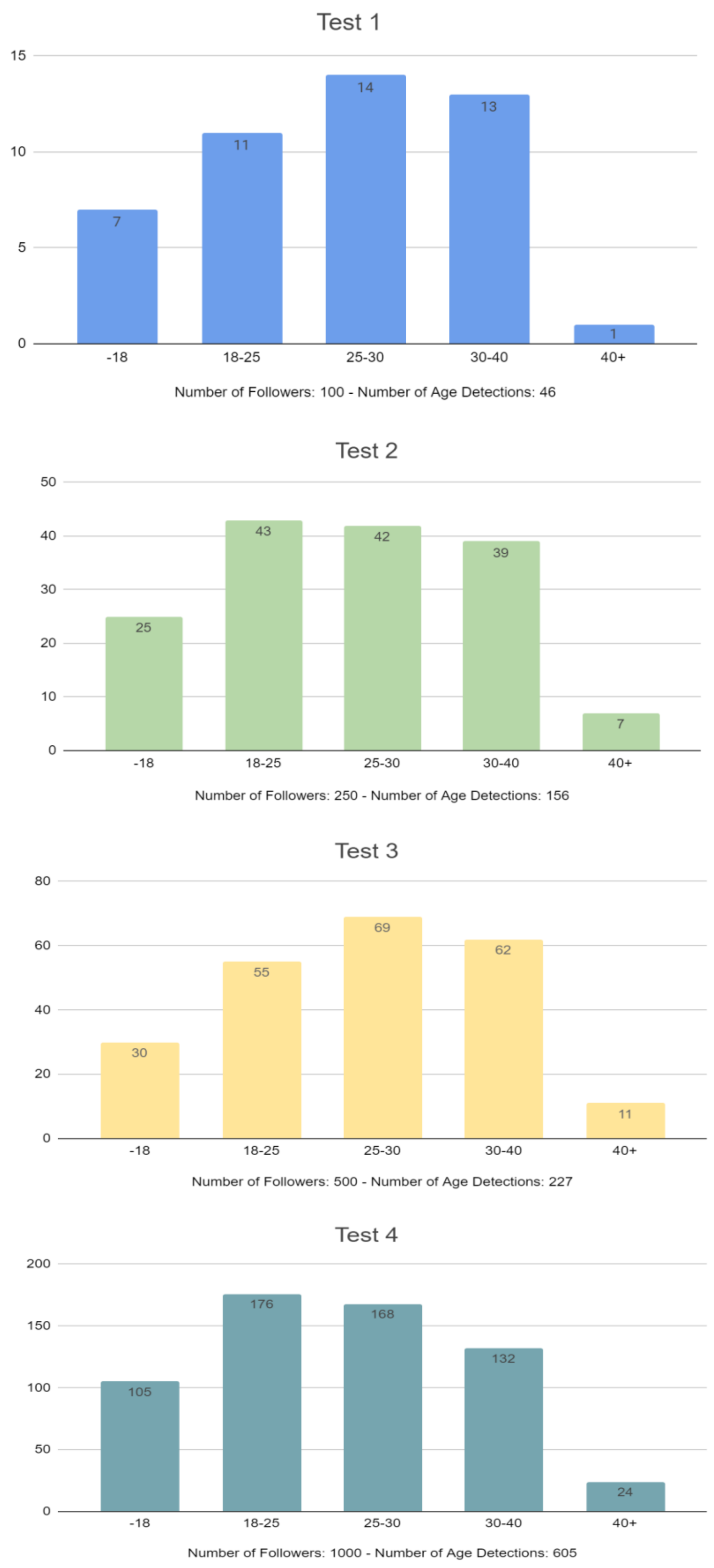


Figure 4. *Tests about age detection* .

5. Conclusion and Future Work

It is known that many influencers have fake followers on their profiles to provide an attraction for potential followers. At the same time, most companies advertise on social media through influencers. Our paper focuses on helping the companies to cooperate with the right influencer and receive the return of their advertisements. Hence, our studies have been carried out to determine whether the followers of an influencer on Instagram are fake. We also find the gender and age range of the influencers to obtain an intuition about the potential market portfolio of that influencer.

Existing solutions in the literature have not paid much attention to fake account detection. Our difference from others is to provide fake account detection and to recommend influencers with real followers to companies that will advertise. In addition to giving the gender and age determination values as percentages, our study also aims to give these values numerically that means we may supply more specific information if needed.

The main problem of our work is to run our application at a less acceptable speed for real-time usage. Our future goal is to run the deep learning solutions on multi-core architectures (GPUs, TPUs) to gain acceptable resulting times. We will also make evaluations with the proposed metrics in Section 4 with the extended dataset to gain more information about the follower of the influencer to offer a better portfolio to the company that requires advertisement for their products.

Declaration of Interest

As authors, we declare that we have no conflict of interest with anyone related to our work.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Sosyal Medyada Fenomenler ve Müşterileri Eşleyen Bir Çözüm Önerisi”).

References

[1] M. Pütter, “The impact of social media on consumer buying intention”, Journal of International Business Research and Marketing, 2017, 3(1), pp. 7-13.

[2] C. Schwemmer and S. Ziewiecki, “Social media sellout: The increasing role of product promotion on YouTube”, Social Media + Society, 2018, 4(3), pp. 1-20.

[3] M. Delbaere, B. Michael and B. J. Phillips, “Social media influencers: A route to brand engagement for their followers”, Psychology and Marketing, 2021, 38(3), pp. 101-112.

[4] M. Bruhn, V. Schoenmueller and D. B. Schäfer, “Are social media replacing traditional media in terms of brand equity creation?”, 2012, Management Research Review, 35(9), pp. 770-790.

[5] E. Constantinides, “Foundations of social media marketing”, Procedia-Social and Behavioral Sciences, 2014, 148, pp. 40-57.

[6] X. J. Lim, A. M. Radzol, J. Cheah and M. W. Wong, “The impact of social media influencers on purchase intention and the mediation effect of customer attitude”, Asian Journal of Business Research, 2017, 7(2), pp. 19-36.

[7] S. Woods, “#Sponsored: The emergence of influencer marketing”, 2016, https://trace.tennessee.edu/utk_chanhonoproj/1976.

[8] T. Gan, S. Wang, M. Liu, X. Song, Y. Yao, and Liqiang Nie, “Seeking Micro-influencers for Brand Promotion”, 2019, In Proceedings of the 27th ACM International Conference on Multimedia (MM '19). Association for Computing Machinery, New York, NY, USA, 1933–1941. DOI:https://doi.org/10.1145/3343031.3351080

[9] S. Wang, T. Gan, Y. Liu, L. Zhang, J. Wu and L. Nie, "Discover Micro-influencers for Brands via Better Understanding," in IEEE Transactions on Multimedia, 2021, doi: 10.1109/TMM.2021.3087038.

[10] F. C. Akyon and E. Kalfaoglu, “Instagram fake and automated account detection” In Proc. IEEE Innovations in Intelligent Systems and Applications Conference, 2019, pp. 1-7.

[11] Y. Jeon, S.G. Jean and K. Han, “Better targeting of consumers: Modeling multifactorial gender and biological sex from Instagram posts”, Journal of User Modeling and User-Adapted Interaction, 2020, vol. 30, pp. 833-866.

[12] K. Han, S. Lee, J. Y. Jang, Y. Jung, and D. Lee, “Teens are from mars, adults are from venus: analyzing and predicting age groups with behavioral characteristics in instagram.”, In Proceedings of the 8th ACM Conference on Web Science (WebSci '16), Association for Computing Machinery, 2016, pp. 35-44.

[13] A. Farseev, K. Lepikhin, H. Schwartz and E. K. Ang., “SoMin.ai: social multimedia influencer discovery marketplace” In Proc. of the 26th ACM International Conference on Multimedia, pp. 1234-1236, 2018.

[14] T. Niciporuc, "Comparative analysis of the engagement rate on Facebook and Google Plus social networks," Proceedings of International Academic Conferences 0902287, International Institute of Social and Economic Sciences, 2014.

[15] E. Naumanen and M. Pelkonen, “Celebrities of Instagram - What Type of Content Influences Followers’ Purchase Intentions and Engagement Rate?”, Master’s Thesis, Aalto University. School of Business, 2017.

[16] R. L. H. Yew, S. B. Suhaidi, P. Seewoochurn and V. K. Sevamalai, "Social Network Influencers' Engagement Rate Algorithm Using Instagram Data," 2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), pp. 1-8, doi: 10.1109/ICACCAF.2018.8776755, 2018.

[17] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition", BMVC 2015, 2015, pp. 41.1–41.12, 2015.

[18] S. I. Serengil and A. Ozpinar, "LightFace: A Hybrid Deep Face Recognition Framework, " 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1-5, doi: 10.1109/ASYU50717.2020.9259802, 2020.

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 160-164, 2021
Research Article

Artificial Neural Network Based Microstrip Reflectarray Unit Element Design

Mehmet BEREKET[1*], Aysu BELEN[2], Mehmet Ali BELEN[3]

[1*] İskenderun Technical University, Faculty of Engineering and Natural Sciences, Electrical Electronic Engineering, Hatay, Turkey, mehmet.bereket@iste.edu.tr
[2]İskenderun Technical University, Iskenderun Vocational School of Higher Education, Turkey, aysu.belen@iste.edu.tr
[3] İskenderun Technical University, Faculty of Engineering and Natural Sciences, Electrical Electronic Engineering, Hatay, Turkey, mali.belen@iste.edu.tr

Abstract
Microstrip reflectarray antennas (RAs) are designs that can achieve equivalent performance of parabolic reflector, but with simple and light electromagnetic and mechanical structures. The challenging problem in design of RA is the fast and accurate modelling of the unit element for the array optimization. 3D EM simulators are computationally very ineffective, thus in this study artificial neural network based unit element modelling for characterization of the reflection phase of the unit element in terms of its geometry, and operation frequency is studied. For this mean, a Malta Cross shaped design for X-band applications is taken into the consideration using Multilayer Perceptron (MLP) neural network trained the 3D CST microwave Studio simulator data. Validation of the MLP model is also worked out successfully with the 3D CST data. By this mean, a continuous function is obtained for the reflection phase of the unit element with respect to the variation of geometrical design parameters and operation frequency had been achieved which can be used for a design optimization process fast as analytical approach design while being accurate as 3D EM simulator tools.

Keywords: Artificial neural network; multi-layer perceptron; reflectarray antenna; X-band.

1. Introduction

Reflective array antennas (RAs) have the advantages of both conventional parabolic reflectors and phased array antennas without using complex and lossy transmission line feed networks [1-6]. Microstrip Reflective Array antennas (MRAs) have many advantages that can be categorized both electromagnetically and mechanically. Electromagnetically, its advantages are high gain, low side lobes, and beam directing ability. When we look at the mechanical aspect, it has the advantages of being a light and planar design and easy production. Since the phase distribution of reflective array antennas is performed with a large number of unit cells, a great deal of flexibility is provided. By optimizing the unit cell geometry and their arrangement, beam forming can be achieved at a low cost. Generating a pencil beam in a specific (θ^o, φ^o) direction can be achieved by designing a phase compensation proportional to the distance from the feed horn antenna phase center so that each RAs element reflects the incoming wave independently, as is well known from classical array theory. Therefore, phase tuning is a very important process in reflector array design. In the phase tuning method, patches of variable size are preferred because simplicity is desired. To meet requirements such as the ability to reflect a shaped patch or multiple patches, or also to improve frequency behavior and bandwidth, it is necessary to use advanced element configurations with several degrees of freedom. The need to manage different parameters and provide phase compensation, increase the bandwidth, and meet the requirements such as high gain opposite to each other makes the necessary multi-objective design optimization a challenging problem.

In order to have a computationally efficient optimization process, a fast and accurate unit element model is needed to act as a continuous function of the unit element's input variables. Then, using this fast and accurate model, it is possible to perform a computationally efficient optimization process. One of the commonly used solutions for achieving computationally efficient optimization process is the usage of data driven surrogate models [7-12]. These numerical models are efficient solution to create a mapping between input and outputs of the selected design problems. Some of the recently published applications of data driven surrogate modelling technique for design optimization of microwave designs can be named as application of machine learning for optimal selection of antenna for wireless communication [7], low cost modelling of antenna designs [8-9], and novel deep learning and ensemble based surrogate models for modelling of different types of microwave stages [10-12].

Here, it is aimed to create a fast and accurate model for the characterization of the reflection angle of the microstrip unit element in terms of geometric parameters and frequency by using artificial intelligence. For this purpose, it is aimed to model the Malta Cross-shaped patch in the X-band by using a Multilayer Perceptron (MLP)

*Corresponding author e-mail address: mehmet.bereket@iste.edu.tr

Received: 01/12/2021; Accepted: 14/12/2021

neural network trained with 3D electromagnetic modeling program data. The validation of the MLP model has also been successfully done with the 3D CST data design of the Artificial Neural Network. First, a Malta Cross shaped unit cell with variable parameters given in Figure 1-2 is investigated. In order to create the ANN model, a series of training and test sets were created using the parametric values given in Table 1. The proposed ANN-based cell model was used together with the MLP black box model using simulated reflection phase values obtained from the 3D simulator, the flow chart of the proposed design methodology had been presented in Fig. 1.

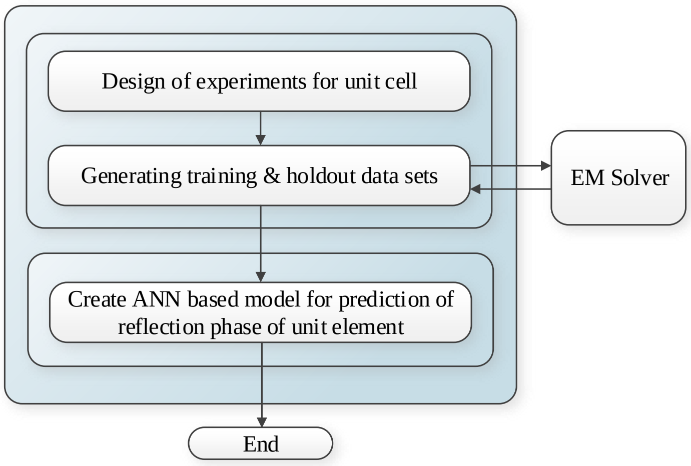

Design of experiments for unit cell
Generating training & holdout data sets
EM Solver
Create ANN based model for prediction of reflection phase of unit element
End

Figure 1. *Flow chart of the proposed design methodology.*

2. Unit Cell Design

In this study, a 3D simulation-based model of the Malta Cross unit cell [13] in Figure 2 is presented with its variables to design the unit cell RAs. In Table 1, the limits of the design variables to be used in the creation of the training and test data sets are given, and the variation of the shape within the given limits is given in Figure 3. For the sake of simplicity, the height values of the microstrip dielectric material are taken as 1.52 mm and 6.15 mm, respectively (compatible with Rogers 3006 material).

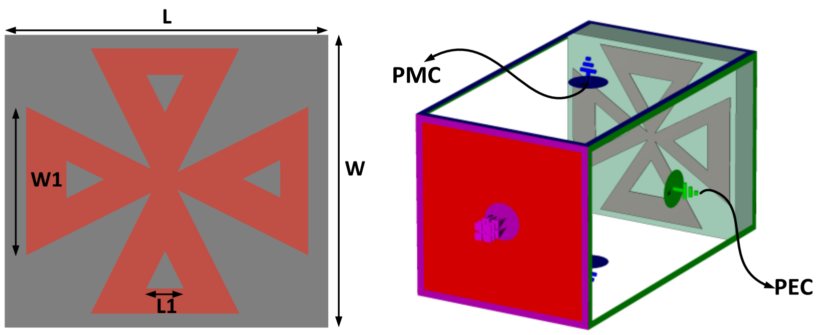


Figure 2. *3D view of the Malta Cross unit cell.*

Table 1. *Range of Unit Cell Parameters (Here W=L=15 mm).*

Parameters	Range	Step Range
W1	2-7	0.1
L1	0.2-2	0.1
f (GHz)	8-12	0.5
Data Set	8721	

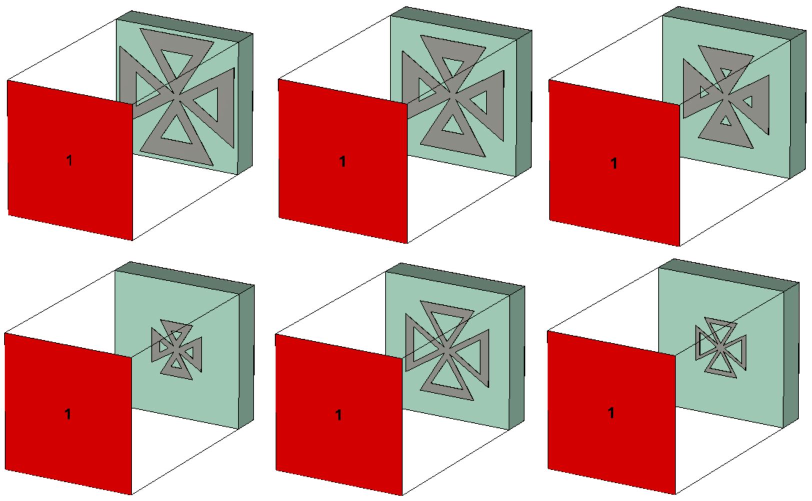


Figure 3. *The view of the unit cell in different parametric values.*

3. Unit Cell Modeling with ANN

Data from the parametric scan of the unit cell will be split into training and test data for the construction of the ANN-based model of the Malta Cross RAs element. For this process, the data generated using the k-fold validation method were divided into 2 equal parts (K=2). At this stage, a Multilayer Perceptron Network (MLP) was used to generate the ANN-based unit element model of the Malta Cross-shaped RAs using the training and test data. The network design parameters of the MLP given in Figure 4 are discussed as in Table 2.

Table 2. *User Defined Parameters of MLP Network.*

Number of Neurons in the Hidden Layer (N)	5, 10, 15
Activation Function	Tangent Sigmoid
Training Algorithm	Levenberg–Marquardt
Maximum epoch	2x(number of samples)

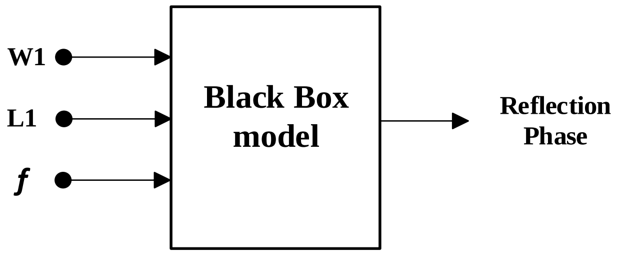


Figure 4. *Black Box model of the RAs element.*

The performance of the MLP network will be evaluated by the commonly used Mean Absolute Error (MAE) for 10 independent runs.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|T_i - P_i| \qquad (1)$$

For better performance evaluation of the ANN model, the training and test datasets were made using cross-fold validation with k=2. The results obtained are presented in Table 3-5.

Table 3. *Performance Result of a Single Layer ANN Model with 5 Neurons.*

Performance		MAE
K-Fold 1	Best	26.3
	Worst	115.6
	Average	75.8
K-Fold 2	Best	36.3
	Worst	94.2
	Average	55.1
Average Performance MAE [degree]	65.5	

Table 4. *Performance Results of 5-10 Neuron ANN Model.*

Performance		MAE
K-Fold 1	Best	14.7
	Worst	60.3
	Average	33.7
K-Fold 2	Best	18.4
	Worst	55.2
	Average	29.7
Average Performance MAE [degree]	31.7	

Table 5. *Performance Results of ANN Model with 5-10-15 Neurons.*

Performance		MAE
K-Fold 1	Best	4.1
	Worst	9.8
	Average	5.4
K-Fold 2	Best	4.8
	Worst	10.7
	Average	6.1
Average Performance MAE [degree]	5.8	

As seen in Table 3-5, the number of neurons in the hidden layer and the number of layers are the most important design parameters in the ANN model. The optimal number of neurons was determined as N=5-10-15 and the number of hidden layers was determined as 3. In Fig. 5, the simulated results of the ANN model for selected cases are presented. As it can be seen over the required operation range the ANN model achieves a very good agreement with the targeted data. Furthermore, a performance comparison of the proposed unit element with counterpart designs in literature is presented in Table 6. As it can be seen form the table, the proposed design has a very good performance with in means of size, operation band range and range of reflection manipulation compared to the state of the art designs in literature.

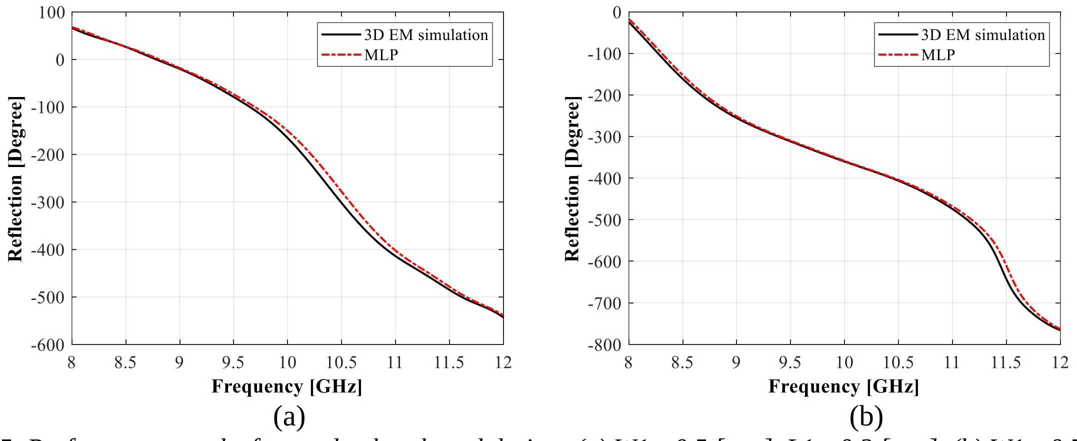


Figure 5. *Performance results for randomly selected designs (a) W1= 0.5 [mm], L1= 0.2 [mm], (b) W1= 0.7 [mm], L1= 2.0 [mm].*

Table 6. *Table of comparison of unit element with counterpart designs in literature.*

	Frequency [GHz]	Phase Range [Degree]	Size [mm]	Material
This study	8-12	360	15x15	FR4
[14]	8-12	203	---	
[15]	9.1-12	420	17x17	Arlon AD300
[16]	8-12	400	15x15	PLA

4. Conclusion

In future studies, it is aimed to use these ANN-based models and optimization algorithms in X-band large-scale reflective array antenna design. As can be seen from the simulation results, the proposed ANN-based model of Malta Cross-shaped microstrip patch RA unit cell has high accuracy with 3D simulation results. Thus, the proposed methodology can be used for design and optimization of a large scale RA design for X band applications.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Yapay Sinir Ağı Tabanlı Mikroşerit Yansıtıcı Dizi Anten Birim Hücre Tasarımı”).

References

[1] D.G. Berry, R.G. Malech, and W.A. Kennedy, The reflectarray antenna, IEEE Trans Antennas Propagat 11 (1963), 645–651.

[2] P. Mahouti, “Application of artificial intelligence algorithms on modeling of reflection phase characteristics of a nonuniform reflectarray element.” International Journal of Numerical Modelling: Electronic Networks, Devices and Fields 33, no. 2 (2020): e2689.

[3] D.M. Pozar, S.D. Targonski, and H.D. Syrigos, Design of millimeter wave microstrip reflectarrays, IEEE Trans Antennas Propagat 45 (1997), 287–297.

[4] J. Huang and J.A. Encinar, Reflectarray antennas, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2007. ISBN:978-0-470- 08491-4.

[5] A. Belen, F. Güneş, M. A. Belen, and P. Mahouti. “3D printed wideband flat gain multilayer nonuniform reflectarray antenna for X-band applications.” International Journal of Numerical Modelling: Electronic Networks, Devices and Fields 33, no. 6 (2020): e2753.

[6] M. Mahouti, N. Kuskonmaz, P. Mahouti, M. A. Belen, and M. Palandoken, “Artificial neural network application for novel 3D printed nonuniform ceramic reflectarray antenna,” International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 33(6), e2746, 2020.

[7] J. Jingon. "Machine learning-based antenna selection in wireless communications." IEEE Communications Letters 20, no. 11 (2016): 2241-2244.

[8] S. Koziel "Low-cost data-driven surrogate modeling of antenna structures by constrained sampling." IEEE Antennas and Wireless Propagation Letters 16 (2016): 461-464.

[9] S. Koziel, and A. Pietrenko-Dabrowska. "Performance-based nested surrogate modeling of antenna input characteristics." IEEE Transactions on Antennas and Propagation 67, no. 5 (2019): 2904-2912.

[10] H. Kalayci, U. E. Ayten, and P. Mahouti. "Ensemble-based surrogate modeling of microwave antennas using XGBoost algorithm." International Journal of Numerical Modelling: Electronic Networks, Devices and Fields (2021): e2950.

[11] S. Koziel, N. Çalık, P. Mahouti, and M A. Belen. "Accurate Modeling of Antenna Structures by Means of Domain Confinement and Pyramidal Deep Neural Networks." IEEE Transactions on Antennas and Propagation (2021).

[12] S. Koziel, P. Mahouti, N. Calik, M. A. Belen, and S. Szczepanski. "Improved Modeling of Microwave Structures Using Performance-Driven Fully-Connected Regression Surrogate." IEEE Access 9 (2021): 71470-71481.

[13] M. Bereket, A. Belen, and M. A. Belen, “Yapay Sinir Ağı Tabanlı Mikroşerit Yansıtıcı Dizi Anten Birim Hücre Tasarımı” 1st International Congress on Artificial Intelligence and Data Science, 2021

[14] S. Finich, N. A. Touhami, and A. Farkhsi, “Design and analysis of different shapes for unit-cell reflectarrray antenna”, Procedia Engineering 181, pp. 526-537, 2017.

[15] H. Bodur, and S.Çimen, “X-bant uygulamalar için tek katmanlı değişken birim eleman boyutlu yansıtıcı dizi anten tasarımı,” Journal of the Faculty of Engineering and Architecture of Gazi University 34:4 (2019) 1861-1869

[16] A. Belen, F. Günes¸ M. A. Belen, P. Mahouti, “3D printed wideband flat gain multilayer nonuniform reflectarray antenna for X-band applications,” Int J Numer Model El. 2020;e2753. https://doi.org/10.1002/jnm.2753

Journal of Artificial Intelligence and Data Science (JAIDA)
e-ISSN: 2791-8335
https://jaida.ikcu.edu.tr

Vol. 1, No. 2, pp. 165-175, 2021
Research Article

Siamese Inception Time Network for Remaining Useful Life Estimation

Ugur CEYLAN [*], Yakup GENC

[1] Gebze Technical University, Computer Engineering Department, Turkey

Abstract
Predictive maintenance tries to reduce cost in engine maintenance in power plants, aircraft, and factories by predicting when maintenance is needed (or by estimating the remaining useful life). Recently, with significant advances in deep learning and the availability of high volumes of data extracted from manufacturing processes, data-driven methods for predicting remaining useful life (RUL) have received a lot of attention. A major problem with data-based prognosis is that it is costly and requires large amounts of training data that could be difficult to obtain in large numbers of failure cases (often impossible and expensive to obtain in the real world). This paper proposes a learning-based method for fault diagnosis requiring fewer failure data. To this end, we use a Siamese network architecture based on a specific deep Convolutional Neural Network (CNN) called InceptionTime. The Siamese part of the network allows repeated use of the existing data to establish a similarity metric for two separate time windows. The turbofan engines C-MAPSS dataset supplied by NASA is used to verify the proposed model. Experiments are conducted using various sizes of the turbofan engines data to compare the performance on small datasets between the proposed model and a base-line model. The results demonstrate that our model can be used in fault diagnosis and provide satisfying prediction results with fewer data with comparable performances against state-of-art methods for RUL prediction.
Keywords: C-MAPSS, deep learning, prognostics and health management, remaining useful life, siamese networks.

1. Introduction

Prognostics and health management (PHM) of mechanical material has attracted attention and predicting remaining useful life (RUL) which is the total amount of cycles/time a given machinery/component can still work before failure is in the core of PHM [1, 2]. By estimating the RUL of the machinery, it is likely to provide preventive mechanical actions and offer a maintenance plan in a targeted way [3]. There are two main methods to estimate mechanical equipment RUL: data-driven and model-based approaches. Basically, in the former, machinery is run until a fault occurs and the runs are repeated multiple times to collect data. The latter requires knowledge of the physics of the breakdown progression [4]. In recent years, the remaining useful life (RUL) studies for data-driven approaches have drawn considerable attention when physical modeling is difficult to obtain for complex systems. In this paper, RUL estimation is predicted using a data-driven method. Data-driven approaches offer a solution for fault detection after certain faults occur (diagnostics) and forecasts of the future operating situations and the time to failure (prognosis). The main challenge in data-driven approaches is that it is usually difficult to obtain a large number of samples for failure progression which could expensive and labor intensive [4]. This can appear in a few different scenarios: (1) industrial systems are not permitted to work until failure due to the outcomes, particularly for crucial systems and failures; (2) most of the mechanical breakdowns happen gradually and result in a degradation path such that failure degradation of a system might take months or even years. Various techniques have been used to address this challenge. The first is to quicken aging by operating the system in a lab with excessive loads, increased speed, or doing simulations of real components that are made by weak materials so that a failure proceeds faster than normal [5]. Another technique is organizing failure progress unnaturally using exponential degeneration to model natural failure progression [6]. Both techniques have their benefits and limitations with the ability to interpret the failure degradation. But, in real-world applications where the circumstance is very distinct from the lab conditions, these techniques are challenging to perform and to obtain sufficient predicting achievement.

Recently, it has been shown that machine learning approaches, especially deep learning methods, perform powerful outcomes in areas such as computer vision, image processing, and natural language processing [7]. Deep learning methods have been employed for fault diagnosis and applied to the RUL estimation problem, in particular using ensemble networks (different types of networks combination). Using a combination of a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) neural network proposes an encouraging method to estimate RUL with a convenient and precise method that can be extended and adapted to obtain more accurately. However, one of the biggest limitations is that they require a large number of labeled failure cases data. In many applications, it is sometimes not possible to collect a lot of data. We also need to retrain the model whenever we want to classify class or estimate the RUL of a new failure case data. Therefore, the proposed method

*Corresponding author e-mail address: uceylan@gtu.edu.tr

Received: 02/12/2021; Accepted: 27/12/2021

aims to solve those problems by using a Siamese network. The Siamese networks tries to model the discriminant nature of the data from two opposite samples. This can be leveraged to generalize the network's predictive ability not only to new data but to entirely new types from unknown distributions [8]. By allowing repeated use of the existing data to establish a similarity metric for two separate time windows, the network is able to generalize predictive ability on small datasets and address the problem of lacking training samples.

This paper proposes a Siamese network that is based on a deep Convolutional Neural Network (CNN), called Inception-Time to increase the precision of RUL prediction performance. The problem is posed as modeling a metric between two time-windows in the observed data, similar being on the same side of the failure start point, and different being on the opposite sides.

The rest of the paper is organized as Section 2 provides a brief literature review for RUL estimation methods, with focus on machine learning (ML) methods and deep learning (DL) approaches. Section 3 describes the preprocessing of input data to the Siamese InceptionTime network. Also, in this section details about the proposed Siamese InceptionTime Network are given. Experiments and results are explained in Section 4 with Section 5 concluding the paper.

2. Background

The state-of-art for predictive maintenance shows that the newest studies on the topic are data-driven approaches and model-driven studies are out of date. Data-driven studies can be classified into two main groups. The first group tries to predict the remaining useful life. Historical sensor data from the machinery/engine is employed as a time series and the problem is converted into a regression problem [9, 10, and 11]. In the second group study, the status of the machinery/engine is characterized as healthy, slightly damaged, or very damaged, and transformed into a classification problem [12, 13]. Many distinct methodologies have been stated in the literature to estimate RUL in this manner.

Early studies on predictive maintenance for RUL estimation are based on statistical methods and ML-based approaches. For instance, Ordóñez *et al.* [9] proposed a data-based approach on turbofan engine dataset using auto-regressive integrated moving average (ARIMA) forecasting prediction as input of the support vector machine (SVM) model. The best kernel function selection is performed using genetic algorithms strategies. The predictive capability of their proposed hybrid method is shown to be good compared to the ones obtained solving the problem once using a multivariate Vector Autoregressive Moving-Average (VARMA) models. Susto *et al.* [10] also proposed SVM for predictive maintenance. The model is based on the decision boundary produced by the SVM model. The data used is synthetic and has been generated using Monte Carlo simulation. The study does not propose a comparison between SVMs and other ML techniques. Khelif *et al.* [11] used SVM to model the direct relationship between sensor values or health index (HI) and predicted RUL directly from sensor values without estimating deterioration state or failure threshold.

Durbhaka *et al.* [14] use k-means to analyze the behavior of wind turbines using vibration signal analysis. In this study, k-NN and SVM algorithms are compared with the k-means algorithm to classify failure types in wind turbines. Recently as an ML-based approach, S. Behera *et al.* [12] proposed a predictive maintenance model that applies a data-driven method for identification of the condition engine and solution formulated as classification manner with one of the three possible values critical, warning, and normal depending upon the RUL value. Utilizing advanced feature engineering, they obtained more insight and used ensemble tree learning. Comprehensive experiments are conducted on a widely used C-MAPSS dataset. It has been seen that Gradient Boosted Trees (GBT) accuracy performs better than Random Forest (RF) accuracy. However, RF competitively performed with a much faster compute time in comparison to the GBT.

Deep neural network (DNN) based methods have gained dominance in the last decade. For instance, Xu *et al.* [15] the healthy status of the engine with the data obtained from 21 sensors belonging to 100 different engines was tried to be monitored. The degradation behaviors of the 21 sensory signals observed and seven of them were selected in this study. The remaining data were estimated by a mixed system consisting of Dempster-Shafer Regression (DSR), SVM, and Recurrent neural networks (RNN). Babu *et al.* [25] built the first attempt of a deep CNN for RUL estimation across two public data sets. The CNN structure is employed to extract the local data features through the deep learning network for better prognostics. Their approach outperformed SVM and MLP based methods. Zhang *et al.* [26] proposed a neural network-based approach that a multi-objective deep belief networks ensemble (MODBNE) method. Deep belief networks (DBNs) are used to handle two issues. The first issue is utilizing handcrafted features which can hardly adjust to different predictive applications. It is employed DBNs as a solution since DBNs offer a promising solution to extracting the most useful features relevant to the estimation task via learning powerful hierarchical attribute representations from data. The other issue is utilizing one single model which can hardly keep good generalization performance across diverse predictive methods. They handle this problem with a multi-objective evolutionary ensemble learning framework incorporated with the traditional DBN.

Li *et al.* [16] proposed a deep learning CNN architecture to estimate RUL. For better feature extraction with CNN, the sliding the time window approach is used for sample preparation. The turbofan engine raw sensor data measurements with scaled between [-1, 1] and then used as input into the proposed network deep CNN architecture. This helps the industrial applications that do not need any prior expertise in signal processing. The high-level abstract features can be extracted with deep CNN architecture and related RUL can be estimated from the basis of the representation learned. The proposed method using time window, data normalization, and deep CNN structure is expected to achieve higher prognostic accuracy compared to traditional machine learning methods. Hanga *et al.* [17] proposed an applied Long Short-Term Memory (LSTM) network, an architecture that specializes in exploring fundamental patterns embedded in time series to monitor system degeneration and eventually predict RUL. The purposes of this paper firstly explain raw sensor data into an interpretable health index to better characterize the health status, and secondly monitor system corruption of the past to predict the future health situation correctly. For those purposes they used the C-MAPSS engine dataset and results show that LSTM produces more robust and accurate in explaining degeneration patterns, enabled by discovering the variation pattern underlying time-series is examined to track the system degradation.

Hinchi *et al.* [18] proposed a deep neural network framework that is based on convolutional and LSTM recurrent units for RUL estimation. First, the neural network subtracts the local features directly from sensor data using the convolutional layer, followed by an LSTM layer to catch the degradation process is added, RUL estimated finally using LSTM output and estimated time value. Experiments used ball-bearing data with good performances. Once again, LSTM models requires representative large training data sets. Zhang *et al.* [19] proposed for RUL estimation, a transfer learning algorithm based on Bi-directional Long Short-Term Memory (BLSTM) recurrent neural networks that models can be trained on different but related datasets first and then fine-tuned by the target dataset. Experimental results show that transfer learning can improve prediction models in the dataset with a small number of samples with one exception when transferring learning leads to a worse result when changing from multi-type to single operating conditions. Transfer learning methods however rely on a large training data. Li *et al.* [20] proposed a directed acyclic graph (DAG) network combining LSTM and a CNN for RUL estimation. The proposed method rather than just using CNN for extracting features it combines CNN and LSTM. The resulting model shows improved performance. We will use this method as our baseline model.

Finally, Siamese networks have been used in the literature quite a lot. Most of these are in the image processing domain. Koch *et al.* [8] use a Siamese network and employed it for identifying characters that consist of comprehensive alphabet sets. By using the Siamese convolutional architecture, they can achieve powerful results on one-shot classification tasks that exceed those of other deep learning models with state-of-the-art performance. Generally, with the Siamese neural networks, they learn image representations through a supervised metrics-based strategy and then reuse the properties of that network without any retraining. Hsiao *et al.* [21] applied Siamese neural networks to the malware image classification task. They implemented a method for identifying distinct types of malware for malware propagation using malware image classification. They also address the issue of lack of training samples at the early stage of new malware appearance by implementing Siamese networks.

3. Methodology

This paper presents a Siamese network based on InceptionTime architecture for RUL estimation using time series data. It is used in the turbofan engine degradation dataset. We first provide the details of the turbofan engine degradation dataset. Then we present the proposed Siamese InceptionTime network, as well as its key components, i.e., InceptionTime and Siamese networks.

3.1. Data overview

The turbofan engine degradation dataset is a well-known data set based on a simulation by Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) tool developed by NASA [22]. The dataset consists of four sub-datasets based on different operating conditions and fault modes. Each sub-dataset includes training data, test data, and actual RUL that corresponds to the last cycle of the test data. In this study, sub-dataset FD001 was used. A short description of the FD001 sub-datasets is given in Table 1. The training data comprises a certain number of engines data from a specific healthy state to failure, while the test data includes a certain number of engine data that end a bit before failure. Furthermore, the training and test data with different initial healthy states, respectively. Because of different initial healthy states, the operating cycles of distinct engines are different in the same database. For example, in sub-dataset FD001 of the database, the training dataset includes 100 engines, with a minimum operating cycle of 128 and a maximum operating cycle of 362 and the test dataset includes 100 engines, with a minimum operating cycle of 31 and a maximum operating cycle of 303, and collected under one operating condition with one fault mode.

Table 1. *Description FD001 sub-dataset of the C-MAPSS.*

Sub-dataset	FD001
Training set engines #	100
Test set engines #	100
Training max/min cycles	362/128
Test max/min cycles	303/31
Operating condition	1
Fault modes	1

3.2. Siamese InceptionTime network

InceptionTime is inspired by the Inception-v4 architecture and an ensemble of deep Convolutional Neural Network (CNN) models [23] for time series classification (TSC), each created by cascading multiple Inception modules, where each module has the same architecture [24]. The main building block of InceptionTime is the inception module. The core idea of an inception module is that utilize multiple filters simultaneously to an input time series. The module involves filters of different lengths enabling the network to automatically extract related features from both long and short time series.

The network consists of a sequence of Inception modules followed by a Global Average Pooling layer and a Dense layer with a softmax activation function. We make some changes after the Global Average Pooling layer in the framework of the proposed method to adapt our case. Furthermore, InceptionTime introduces a further element within its network's layers: residual connections at every third inception module. Thus decreasing the vanishing gradient problem by allowing a direct flow of the gradient.

The main idea of the Siamese network is that two input signals go through a shared network generating fixed-length feature vectors. Assuming that the appropriately trained neural network model, we can make the following hypotheses: If two input instances belong to the same instance classes, their feature vectors must be similar, if two input instances belong to different instance classes, their feature vectors will also be different. So absolute differences of the two feature vectors must be very different in both the above cases. The proposed network in shown in Figure 1.

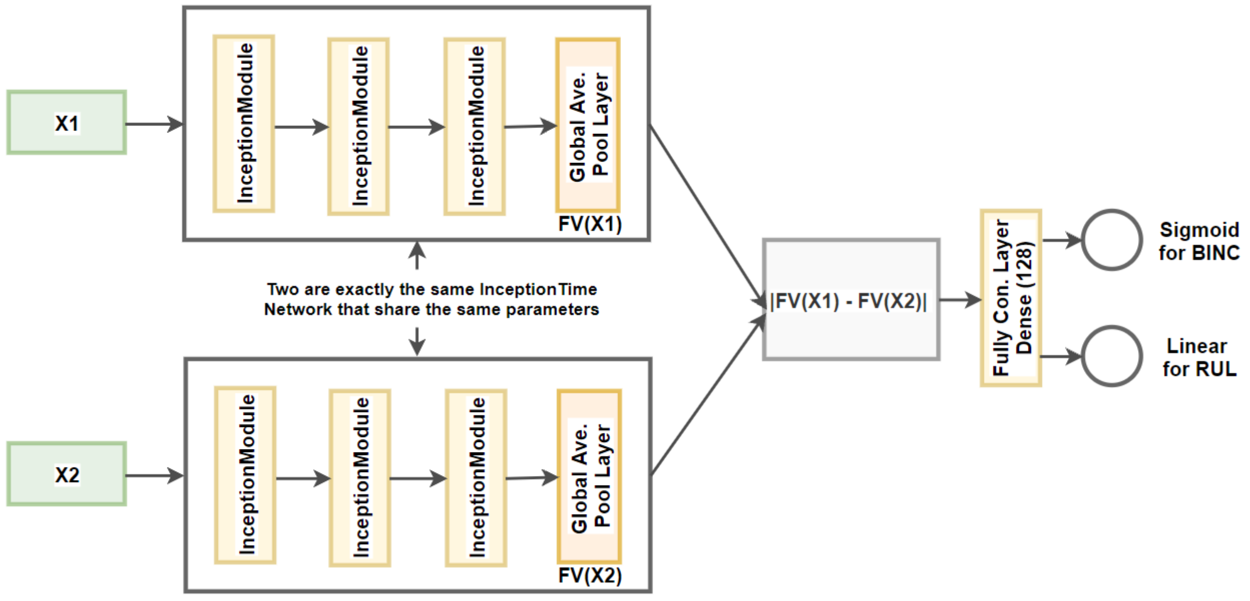


Figure 1. *The architecture of proposed Siamese InceptionTime Network.*

1) **Data preparation:** We first prepare pairs of healthy and failure samples as input for the Siamese network. The sub-dataset has N attributes, and the lifetime of the sensors are L_t cycles, i.e., machinery lifetime (see

Figure 2). The data is obtained by sliding the time window with window size w_l cycles, and the sliding step size is one cycle. For a given sequence we will have [L_t-w_l, L_t- w_l-1,. . .,0]. w_l is selected as 30 for the FD001 sub-dataset, because it gives the highest precision between results from other time window sizes as proposed [13] and with large time window size includes more data points within the time frame, so the degradation pattern may be simpler. To evaluate all engine's performance in sub-test data and make the comparison between the compared and the results of the proposed method, we cannot increase the time window size because of min cycle in the test engine is 31. In our experiment, we select 14 sensor outputs as in [20]. We further normalize these values within [0, 1].

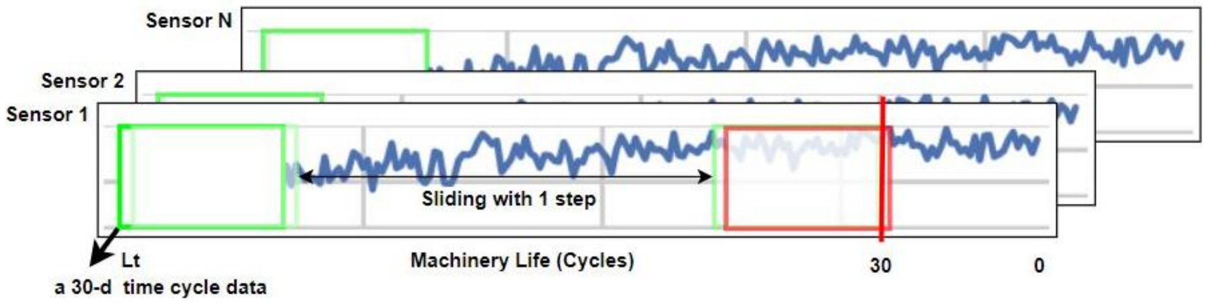


Figure 2. *Extracting health/failure samples*

Then, a breakpoint RUL value which is 30 specified (which is illustrated with a solid red vertical line in Figure 2) to discriminate between healthy and failure states for the FD001 sub-dataset, and according to this breakpoint value, healthy and failure samples were prepared. If the RUL value of input data is less than the breakpoint value, it was determined as a failure sample and other cases as an example of healthy samples. So, the input of the Siamese InceptionTime network is pairs of the combination of healthy and failure samples. For binary classification (BINC output, see Figure 1), data labeled as if a pair input data in the same status (in this study only pairing healthy sample vs healthy sample applied) then the label is 1 other case which is failure vs healthy samples then the label is 0. On the other hand, for output RUL (see Figure 1), for each pair of input data labeled as minimum RUL value of input pair samples.

2) **InceptionTime Siamese network:** We propose a Siamese network that is using InceptionTime Network for RUL estimation. The network includes two identical paths that share weights, and each is composed of three Inception modules followed by a Global Average Pooling layer. So, two input samples (x_1 and x_2) are passed through the Siamese InceptionTime Network to generate a fixed-length feature vector for each ($f_v(x_1)$ and $f_v(x_2)$), then, we compute the absolute distance between these feature vectors and passed to a fully connected layer (with a rectified linear unit activation) followed by another fully connected layer (with a sigmoid activation) determining the similarity of the two input. Depth of inception network set to 3 with 64 filters at each and three convolutions with [3, 5, 7] filters lengths. For training our network, we used Adam optimizer with learning rate 10^{-5} and batch size 64.

To evaluate the performance of the proposed Siamese Inceptiontime Network, root mean square error (RMSE) and Score metrics were used. RMSE is a frequently utilized metric for RUL prediction that gives the same penalties while the score has different penalties for early and late predictions [20]. If there is a prediction error *e* and the error between the predicted RUL and actual RUL is *e* = $RUL_{predict}$ - RUL_{actual} then RMSE and Score formulas are given in Eqs. (1) and (2) separately:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e_t^2} \tag{1}$$

$$Score = \sum_{t=1}^{N} s_t,\ s_t = \begin{cases} exp^{-e_t/13} - 1 & e_t < 0 \\ exp^{e_t/10} - 1 & e_t >= 0 \end{cases} \tag{2}$$

where e_t is the prediction error, *exp* denotes Euler's number and N is the test sample size.

4. Experiments and Results

We conducted experiments in the first dataset (FD001) of C-MAPSS to show the performance of the proposed method. We evaluate the performance of the proposed method and compare its results to a state-of-art best model [20]. First, we only used the data from 15 engines that were randomly selected for training. Selected engines ids are [52, 42, 37, 35, 100, 68, 41, 66, 40, 81, 53, 49, 61, 76, 18]. We obtained 30000 pairs of data from the selected engines with the balanced version of the dataset by weighting randomly selecting samples from each engine according to the total dataset size to train the proposed method.

The Siamese InceptionTime network was employed to estimate the RUL of the C-MAPSS engine dataset. For testing, each sample is to be predicted compared with the first healthy sample of a specific engine, and pairs are given to the network for prediction. The test root mean square error (RMSE) is 16.35 and the score is 460.77 (compare these with the best results in the literature [20] with 11.96 and 229.00 respectively). The predicted RUL of each test engine is shown in Figure 3. With the proposed method generally, the estimated value of the RUL can be observed close to the true value. We also compare the proposed model with some other model results in the literature.

The comparison result is quantified using RMSE of RUL prediction for the FD001 test dataset as listed in Table 2. It is noted that the developed proposed model outperforms SVR, MLP, DCNN, and RF. Also, the results of the proposed method are close to GB and MODBNE. DAG LSTM-CNN gives better than the proposed result. However, it should not be forgotten that the proposed method uses only 15 engines data for training.

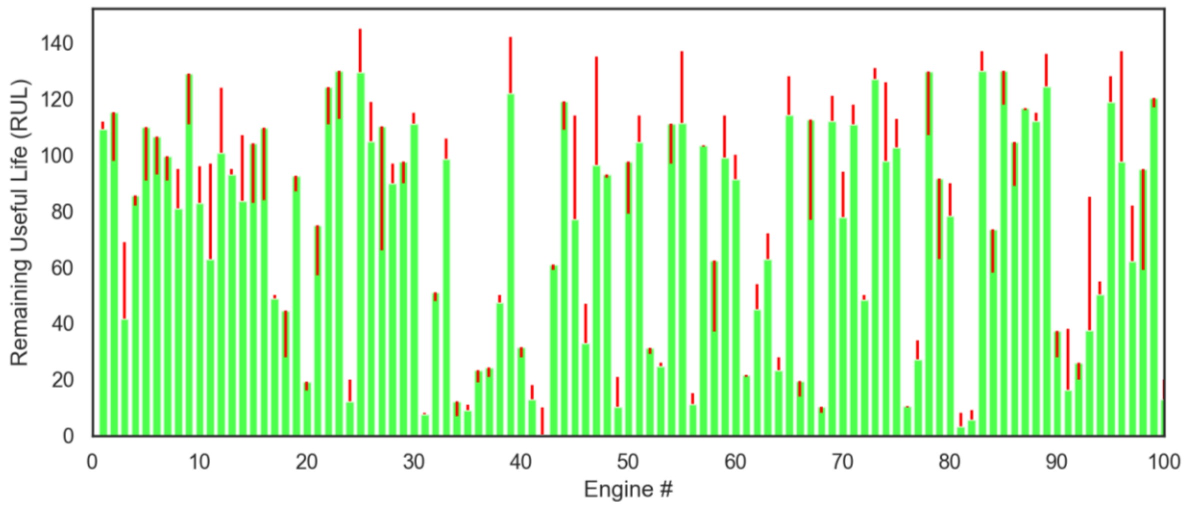


Figure 3. *Predicted RUL for each engine in the FD001 test sub-dataset by the Siamese InceptonTime network. The wider green bar shows the predicted RUL and the narrower red line exhibits the differences of RUL values between actual and predicted.*

Table 2. *Engine RUL prediction comparisons, in RMSE on FD001 test dataset.*

Methods	RMSE
SVR [25]	20.96
MLP [25]	37.56
DCNN [25]	18.44
RF [26]	17.91
GB [26]	15.67
MODBNE [26]	15.04
DAG LSTM-CNN [20]	11.96
Proposed method	**16.35**

Figure 4 demonstrates the predicted RUL and binary similarities generated by our algorithm on the two of the engines for the FD001 test sub-datasets. The green line shows the binary similarity (the first healthy sample is compared sequentially with other samples of the engine) and gives some signals before the failure point indicated in the dashed red line via decreasing score between data pairs. Also, the RUL prediction results are shown in blue and successfully estimate predictions as close to actual values and give the alarm signal before the failure point by lowering the predicted RUL value.

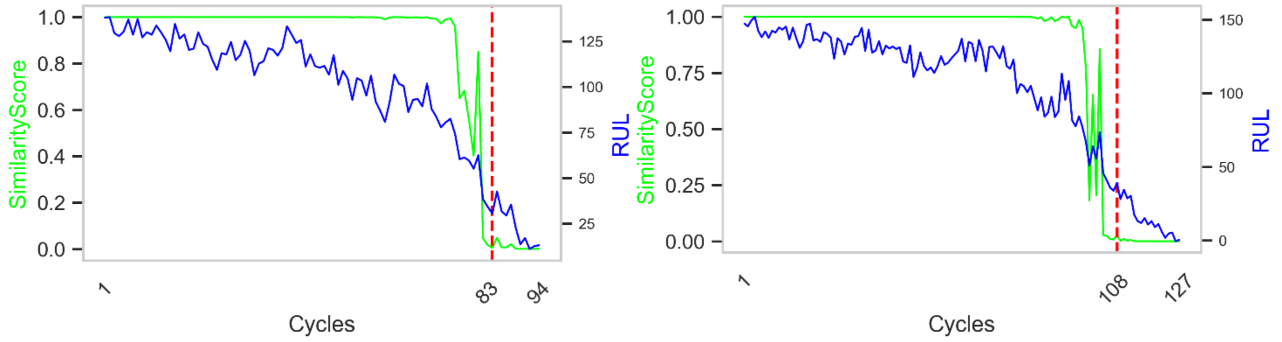


Figure 4. *Input data of first healthy sample vs others to the Siamese InceptionTime Network output results on engine #41 and #42 in FD001 test sub-dataset. Green shows binary similarity while blue shows the estimated RUL (best seen in color). The vertical dashed red line illustrates the degradation point of the engine.*

Classification results of the FD001 test dataset are illustrated in Figure 5 in a confusion matrix. The accuracy of our classifier is 95% while the F1 score is 0.96. This shows the discriminant nature of the windows before and during failure. The Siamese network captures this very accurately.

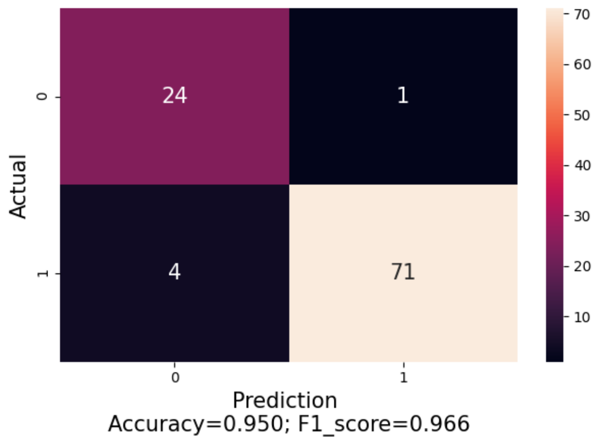


Figure 5. *Classification results input of pairs healthy/failure samples of test data.*

In order to evaluate compare the proposed method to a base-line system when a smaller amount of data is used, we have implemented the network described in [20]. For both proposed and compared models we obtained the best hyperparameters and run the model over one thousand epochs. We train these models with random hyperparameters and compared their results in RMSE and Score metrics.

We simulate the amount of data (smaller to larger) by using a fixed number of engines in the training (e,g., 5 engines randomly chosen from the training set). We change the number of engines from 5 to 24 indicating an increasing amount of data. We expect that 5 engines for training will be challenging and as the number increased, the models will perform better. For a fixed number of engines, models are trained 30 times. We call each of these as an *experiment*, and for each iteration, we select engines randomly and used the same engine's data for both models to compare performances on test data and recorded theirs results from with both hyperparameters and random parameters. For each experiment, we prepare 30000 data pairs from selected engines and train the proposed model for one thousand epochs with a simple early stopping strategy. The selected engines data for the baseline model is trained as described in [20]. The training procedure was repeated 30 times for both models.

RMSE and Score results that were obtained from hyperparameters on test data for both models are illustrated in Figure 6 while random parameters results are shown in Figure 7 with boxplots. Box plots result cut of values bigger than 50 for RMSE and 5000 for Score metrics for hyperparameter results while those values are 50 for RMSE and 15000 for Score metrics for random parameters. The proposed model gives a reasonable performance compared to the base-line model when optimal hyperparameters are used. For random parameters, the proposed model gives better results.

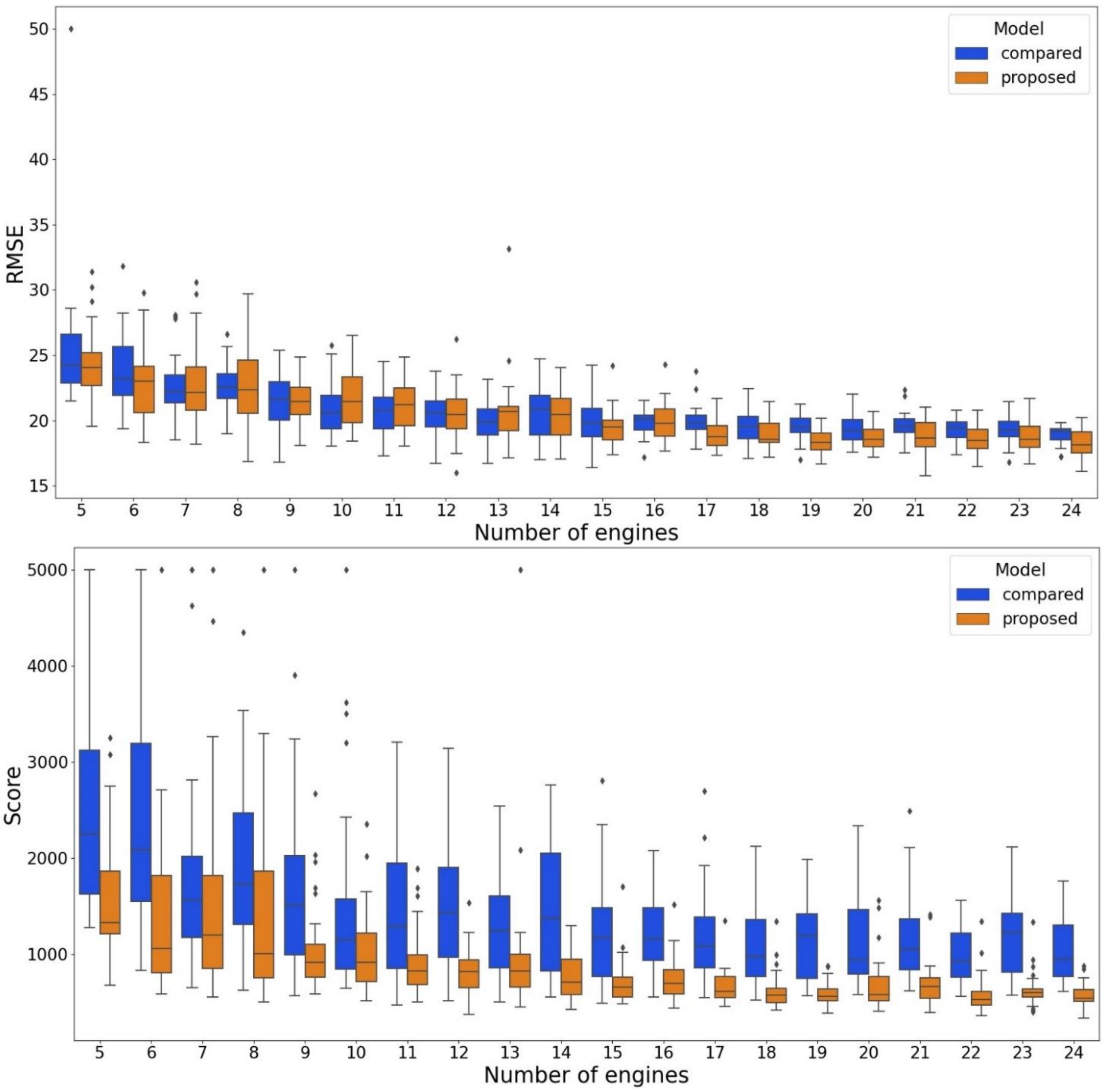


Figure 6. *Hyperparameters RMSE (top) and Score (bottom) results. Boxplot results were obtained from 30 experiments for changing number of engines.*

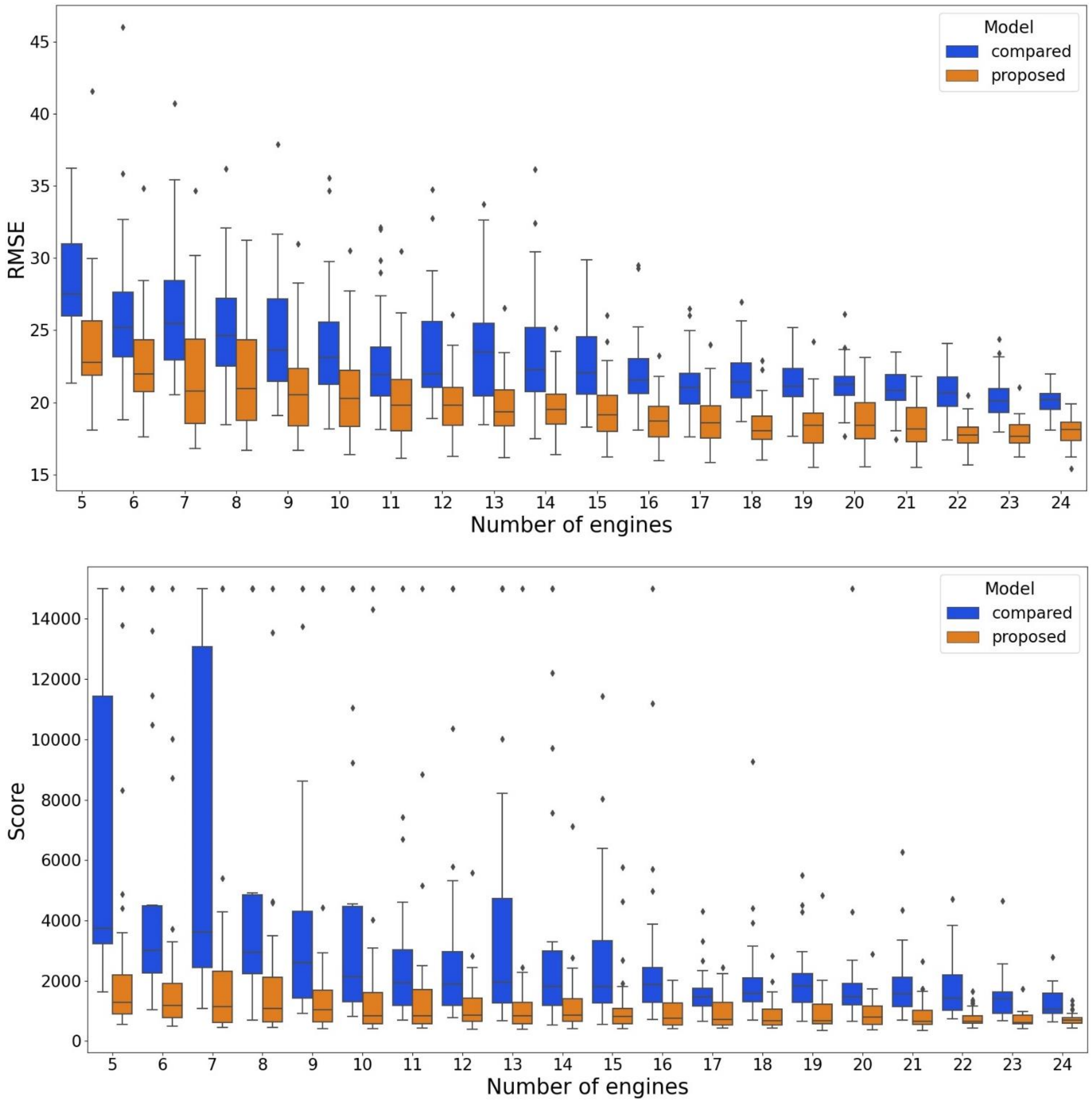


Figure 7. *Random parameters RMSE (top) and Score (bottom) results. Boxplot results were obtained from 30 experiments for changing number of engines.*

In order to see the effect of shortening the available sequence hence reducing the training data, we have trained the two models first with full data and then with first 15% of the data from the beginning is removed. The RMSE and Score metrics show slight improvements as seen in Figure 8. The proposed method performs slightly better when the sequence is shortened. This confirms our expectation that the model performs well with less data.

The improvement is slightly more with the base-line algorithm with shorter sequences. This might seem a bit counter intuitive. Yet, this result can still be explained with the fact that we are using a linear model for machine degradation. Ignoring the begging of the sequence removes any potential noise before the linear degradation starts.

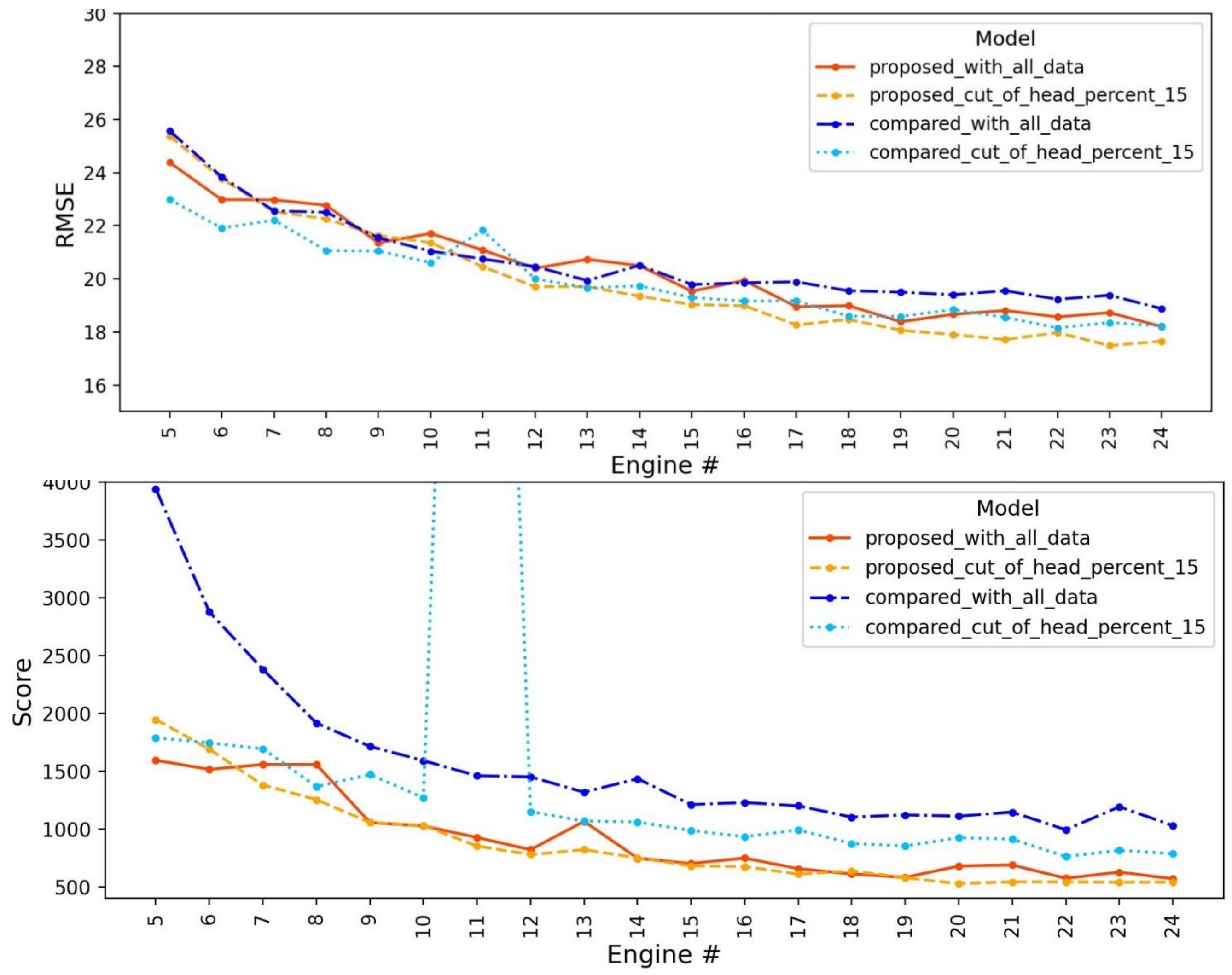


Figure 8. *Performance (top – RMSE, bottom – Score) change when the first 15% of each data is ignored during training. Mean of the metrics are given for changing number of engines used in training. Ignoring data in the beginning of a sequence improves results slightly.*

5. Conclusions

Accurate estimation of RUL has important advantages in many industrial applications where safety efficiency and reliability are among primary concerns. In the absence of a lot of failure data, we propose a Siamese network with temporal data processing capabilities for fault diagnosis classification and regression. Since typical deep learning models do not train well with small sets of training samples, the proposed method allows multiple uses of the data for learning a window-based similarity metric. InceptionTime network allows temporal processing. The experiments comparing the proposed method with a state-of-art baseline method show improved performance when less data is available. While these results are promising, further architecture and hyperparameters optimizations are still needed as the reported performance is slightly lower than the best state-of-art methods. Further experimental validations will be done on the other C-MAPSS sub-datasets.

Declaration of Interest

The authors declare that there is no conflict of interest.

Acknowledgements

An earlier version of this paper was presented at the ICADA 2021 Conference and was published in its Proceedings (Title of the conference paper: “Siamese Networks for Predicting Remaining Useful Life of Machinery”)

References

[1] D. Wang, K. Tsui, and Q. Miao, “Prognostics and Health Management: A Review of Vibration Based Bearing and Gear Health Indicators”. IEEE Access, 2018, pp. 665–676.

[2] R. Zhao, “Deep Learning and its Applications to Machine Health Monitoring”. Mechanical Systems and Signal Processing 115, 2019, pp. 213–237.

[3] X. Si, “Remaining Useful Life Estimation - A Review on the Statistical Data Driven Approaches”. European Journal Operation Research, 213, 2011, pp. 1–14.

[4] Ö. Eker, F. Camci, and I. Jennions, “Major Challenges in Prognostics: Study on Benchmarking Prognostic Datasets”, Proc. European Conference of the Prognostics and Health Management (PHM) Society, 2012.

[5] N. Gebraeel, A. Elwany, and Jing Pan. “Residual Life Predictions in the Absence of Prior Degradation Knowledge”. IEEE Transactions on Reliability 58. 2009, pp. 106–117.

[6] O. Eker et al. “A Simple State-Based Prognostic Model for Railway Turnout Systems”. IEEE Transactions on Industrial Electronics 58, 2011, pp. 1718–1726.

[7] Y. Bengio, Y. LeCun, and G. E. Hinton, “Deep learning for AI”, Communications of the ACM. 64, 2021, pp. 58–65.

[8] G. Koch, R. Zemel, and R. Salakhutdinov. “Siamese Neural Networks for One-shot Image Recognition”. Proc. International Conference on Machine Learning, 2015.

[9] C. Gal´an et al. “A hybrid ARIMA-SVM Model for the Study of the Remaining Useful Life of Aircraft Engines”. Journal of Computational and Applied Mathematics, 346, 2019, pp. 184–191.

[10] G. A. Susto and A. Beghi. “Dealing with Timeseries Data in Predictive Maintenance Problems”. Proc. IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016, pp. 1–4.

[11] R. Khelif et al. “Direct Remaining Useful Life Estimation Based on Support Vector Regression”. IEEE Transactions on Industrial Electronics 64, 2017, pp. 2276–2285.

[12] S. Behera et al. “Ensemble Trees Learning based Improved Predictive Maintenance using IoT for Turbofan Engines”. Proc. ACM/SIGAPP Symposium on Applied Computing, 2019.

[13] C. Zhang, J. H. Sun, and K. Tan. “Deep Belief Networks Ensemble with Multi-objective Optimization for Failure Diagnosis”. Proc. IEEE International Conference on Systems, Man, and Cybernetics, 2015, pp. 32–37.

[14] G. K. Durbhaka and B. Selvaraj. “Predictive Maintenance for Wind Turbine Diagnostics using Vibration Signal Analysis based on Collaborative Recommendation Approach”. Proc. International Conference on Advances in Computing, Communications, and Informatics (ICACCI), 2016, pp. 1839–1842.

[15] J. Xu, Y. Wang, and L. Xu. “PHM Oriented Integrated Fusion Prognostics for Aircraft Engines Based on Sensor Data”. IEEE Sensors Journal 14, 2014, pp. 1124–1132.

[16] X. Li, Q. Ding, and J. Sun. “Remaining Useful Life Estimation in Prognostics using Deep Convolution Neural Networks”. Reliability Engineering Systems, 172 (2018), pp. 1–11.

[17] J. Zhang et al. “Long Short-term Memory for Machine Remaining Life Prediction”. Journal of Manufacturing Systems 48 (2018), pp. 78–86.

[18] A. Z. Hinchi and M. Tkiouat. “Rolling Element Bearing Remaining Useful Life Estimation Based on a Convolutional Long-short-term Memory Network”. Procedia Computer Science 127 (2018), pp. 123–132.

[19] A. Zhang et al. “Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation”. Applied Sciences 8 (2018).

[20] J. Li, X. Li, and D. He, “A Directed Acyclic Graph Network Combined with CNN and LSTM for Remaining Useful Life Prediction”, IEEE Access, June 2019, pp. 75464–75475.

[21] S. C. Hsiao, “Malware Image Classification Using One-Shot Learning with Siamese Networks”. Proc. Knowledge-Based and Intelligent Information & Engineering Systems (KES2019). 2019.

[22] A. Saxena, “Damage Propagation Modeling for Aircraft Engine Run-to-failure Simulation”. Proc. International Conference on Prognostics and Health Management, 2008, pp. 1–9.

[23] I. Fawaz, H. Lucas, B. Forestier, “InceptionTime: Finding AlexNet for Time Series Classification”. Data Mining and Knowledge Discovery, 34, pp. 1936–1962, 2020.

[24] C. Szegedy et al. “Going Deeper with Convolutions”. Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[25] G. S. Babu, P. Zhao, X-L. Li, “Deep Convolutional Neural Network-based Regression Approach for Estimation of Remaining Useful Life”. Proc. International Conference on Database Systems for Advanced Applications, 2016. pp. 214–28.

[26] C. Zhang, P. Lim, AK. Qin, KC. Tan, “Multi-objective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics”. IEEE Transactions on Neural Network and Learning System 2016, pp. 1–13.