VOLUME 2 | ISSUE 2 | YEAR 2024

ISSN: 2980-3152

CURRENT TRENDS IN COMPUTING

https://dergipark.org.tr/en/pub/ctc

Current Trends in Computing (CTC)

Advisory Board

Scope

Current Trends in Computing (CTC) is an international double-blind peer-reviewed journal. It publishes original and high-quality unpublished research papers in all computer sciences/engineering areas. CTC allows researchers and academic professors to share their knowledge with other researchers and professors worldwide.

Contents

Follow this issue and upcoming issues at: https://dergipark.org.tr/en/pub/ctc

Vol: 2, Issue: 2, 2024
Pages:86-95
Received: 1 August 2024
Accepted: 26 August 2024

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE
DOI: 10.71074/CTC.1526313

APPLICATION OF AUTOMATED MACHINE LEARNING AND BAGGING TECHNIQUES TO CLASSIFY RICE VARIETIES

CIHAN BAYRAKTAR[1*]

[1] *Computer Technologies Department, Karabük University, 78050, Karabük, Türkiye*

ABSTRACT. **Nowadays, the research for digitalization in the agricultural sector recent years has significantly increased. In particular, machine learning and artificial intelligence have applications in agricultural product classification, quality control, and species identification. The fast processing times, high accuracy levels, and cost-effectiveness offered by digital solutions for quality control and classification accelerate these studies. Classifying rice species using traditional methods is a process that requires expertise, is time-consuming, and costly. Errors and differences of opinion due to human factors constitute essential limitations in this process. In order to eliminate these limitations, this study proposes a collaborative learning model utilizing Automated Machine Learning and Bagging techniques for rice species detection and classification. The model uses a dataset from the UCI Irvine Machine Learning Repository, which contains characteristics specific to the Osmancık and Cammeo rice varieties grown in Turkey. The dataset consists of 3810 data points, 2180 of which belong to Osmancık rice and 1630 to Cammeo rice. During the analysis, MLBox, an Automated Machine Learning library, was used to determine the optimal algorithm (Light Gradient Boosting Machine - LGBM) and its hyperparameters. Later, by applying the Bagging technique within the developed learning model, an accuracy rate of 93.54% was achieved in rice-type classification.**

1. INTRODUCTION

Agriculture is a sector of vital importance, especially for countries that are still developing. The reasons for this importance include ensuring sufficient and safe food supply, contributing to national income by providing job opportunities to large masses, encouraging the development of industry with the demand for agricultural inputs, developing exports, and contributing to the general development of the country. For these reasons, developing and supporting the agricultural sector should be among the priorities of every country [1]. In summary, a country's high agricultural productivity increases the economic welfare level of the country as a whole [2].

According to 2021 statistics, rice is one of the most important basic food products for the world population, producing more than 1 billion tons worldwide [3]. The criteria applied to detect quality rice vary according to regions and countries. However, among consumers, physical appearance, taste,

E-mail address: cihanbayraktar@karabuk.edu.tr (*).
Key words and phrases. Rice Types, Machine Learning, AutoML, Bagging, Classification.

aroma, smell, and cooking ability generally stand out as factors that are taken into consideration when determining the quality of rice [4].

In evaluating the quality of rice, the use of machine learning and artificial intelligence techniques instead of manual methods is increasing thanks to the low time and cost opportunities it provides. Manual methods at this stage result in long-term errors and high error rates due to human factors. In addition, the fact that evaluation can only be made by manual methods by experts of the relevant product brings another limitation. Differences in people's own opinions also cause differences in evaluation results. For this reason, the use of automatic systems instead of manual methods can enable more effective results in the quality evaluation of products [5].

Machine learning algorithms enable complex, high-dimensional data to be analyzed quickly and accurately. Fault detection, fraud detection, and product quality analysis can be given as examples. One of the main reasons for the widespread use of machine learning algorithms is that they enable the use of Graphics Processing Unit (GPU) on a large scale. Because GPUs can show much higher performance in data analysis operations than the Central Processing Unit (CPU) of computers. The development of these technologies in data analysis gives hope for solving evaluation problems in agricultural products [6]. Among the techniques used in machine learning applications, data analysis with Automated Machine Learning (AutoML) libraries can significantly impact different segments of agriculture and industry with the ease of hyperparameter optimization.

In the study conducted by Çınar et al., images of two rice species, Osmancık and Cammeo rice grains grown and registered in Turkey, were taken and processed, and a dataset was created by making feature extractions. The results were compared by applying machine learning algorithms and classification techniques to the produced dataset. As a result of the study, they stated that they got the best accuracy rate, 93.02%, from the model produced with the Logistic Regression algorithm [7].

In their previously published study on the dataset used in this study, İlhan et al. created a model with Deep Neural Networks for the classification of Osmancık and Cammeo rice varieties. As a result of the study, they reached an accuracy rate of 93.04% with the model prepared with Deep Neural Networks. It was stated that the model created in the study made successful classification [8].

In a study conducted by Jin et al., data analysis was carried out using deep learning algorithms such as LeNet, GoogleNet, and ResNet on the seeds imaged with hyperspectral imaging technology to classify rice seed varieties. As a result of the combination of hyperspectral imaging and deep learning algorithms, it has been determined that effective models can be produced in distinguishing rice seeds and the ResNet algorithm shows the best performance with 86.08% [9].

In their study, Jaithavil et al. created transfer learning models with VGG16, InceptionV3 and MobileNetV2 systems for the classification of paddy seeds and performed data analysis tests. It was announced that the proposed transfer learning model achieved high accuracy in classifying steel seeds, and the InceptionV3 model achieved the best result with an accuracy rate of 83.33%. Within the scope of the study, it was also stated that the MobileNetV2 model reached the same level of accuracy, but the classification performance of this model was not considered sufficient as the test loss occurred at 61.95% [10].

In the study conducted by Jumi et al. to classify rice types, the shape, color, and texture characteristics of rice were extracted using the Invariant Moment, Hue Saturation Value, and Local Dual Axis methods,

and a dataset was created. Afterwards, the relevant dataset was analyzed with the k-nearest Neighbor classification algorithm, and an accuracy rate of 86.22% was obtained. It was stated that the data obtained within the scope of the study reached a promising result on the subject [11].

Hoang et al. investigated the difference between manual methods and the CNN algorithm that can be used to classify rice varieties. The VNRICE dataset was used in the study. After testing various CNN models, they found the best result of 99.04% with the learning model created by the DenseNet algorithm with 121 layers. In the study, it was stated that the 121-layer DenseNet model showed the highest performance, as the accuracy result, as well as the memory and resource consumption of the models, were taken into consideration [12].

In the study conducted by Mrutyunjaya and Harish Kumar, ensemble machine-learning algorithms were used to classify five different rice varieties with high accuracy. It has been stated that the learning model, which was created based on machine learning techniques and image processing methods, was successful in correctly classifying different rice varieties. In the study, the highest average classification accuracy among all tested algorithms was achieved by the Extreme Gradient Boosting (XGBoost) algorithm with 99.60% [13].

Köklü et al. carried out a classification study with deep learning algorithms for Arborio, Basmati, İpsala, Jasmine, and Karacadağ rice types. In this study, Artificial Neural Networks (ANN), Deep Neural Networks (DNN), and Conventional Neural Networks (CNN) algorithms were preferred for classification processes, and the results obtained as a result of the classification were compared. As a result of the analysis, ANN reached 99.87%, DNN 99.95%, and CNN 100% performance rates. The findings obtained in the study stated that the learning models used can be successfully applied in the classification of rice varieties and can help determine seed quality [5].

In this study, the difficulties encountered in the quality classification processes of agricultural products are discussed based on the classification of Osmancık and Cammeo rice varieties grown in Turkey. The slowness and high error rates of traditional methods require the use of more effective and accurate techniques in the quality control of agricultural products. In this context, the proposed solution is a learning model developed using AutoML and Bagging methods. This model aims to increase efficiency in agricultural production processes by classifying rice varieties more quickly and accurately. In the following sections of the study, detailed information is given about the dataset used, explanations are made regarding the production of the proposed models, and the findings obtained from the analysis of the data are conveyed.

2. Materials and Methods

In this study, in order to make a prediction and classify between Osmancık and Cammeo rice, the dataset was produced by Çınar and Köklü [7] and donated to the UC Irvine Machine Learning Pool, where analyses were carried out [14]. In the study conducted by [7], it was stated that the dataset used was created by transferring the images of 50g Osmancık and Cammeo rice to the computer environment and determining their properties, with a camera placed on a box that does not receive any external light

but has an internal lighting mechanism. In this context, the dataset consists of 3810 lines of data in total. The attribute definitions determined for the dataset are given in Table 1.

TABLE 1. **Dataset attribute definitions**

Attribute Name	Attribute Definition
Area	Number of pixels within the boundaries of a grain of rice
Perimeter	The sum of the distances between pixels around the boundaries of the grain of rice
Major Axis Length	The longest line that can be drawn on a grain of rice
Minor Maxis Length	The shortest line that can be drawn on a grain of rice
Eccentricity	The degree of roundness of the ellipse with the same moments as a grain of rice
Convex Area	Number of pixels of the smallest convex hull of the area formed by the rice grain
Extend	The ratio of the area formed by the rice grain to the bounding box
Class	Result tag (Osmancık / Cammeo)

The class label distribution in the dataset was Osmancık 2180 and Cammeo 1630 (Figure 1).

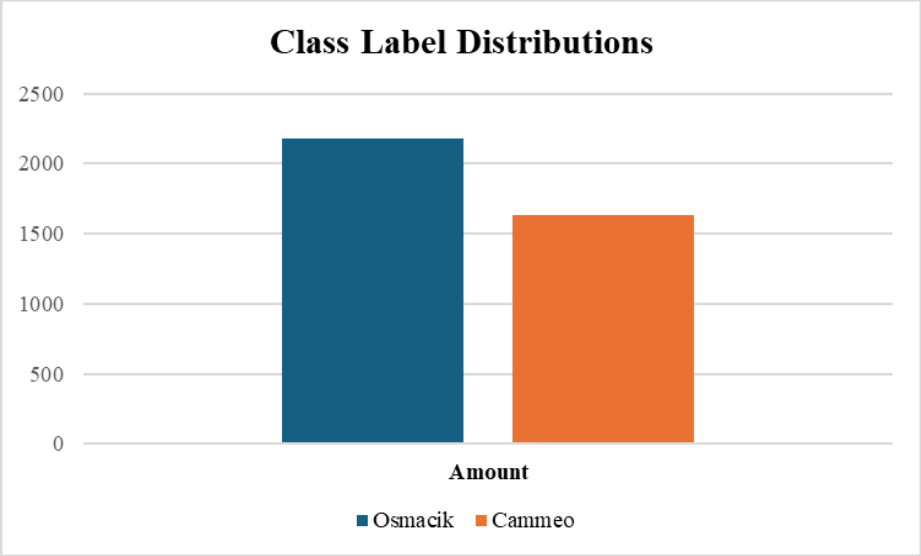


FIGURE 1. **Dataset class label distributions.**

In the analyses performed on the dataset, a computer with Intel Core i7 9750H CPU, 16 GB RAM Memory, Nvidia GeForce GTX 1050 3 GB Graphics Card, and Windows 11 operating system was used. Analyzes were carried out using the Python 3.7 programming language and the Jupyter Notebook editor.

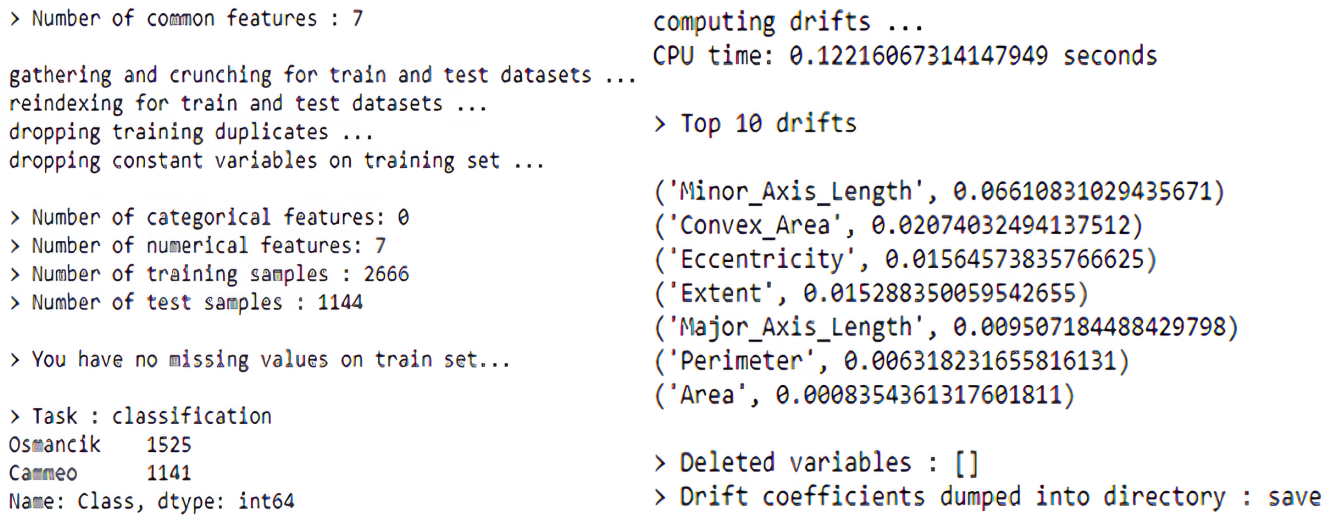


FIGURE 2. **MLBox Data preprocessing results.**

2.1. **Data Preprocessing:**

Data preprocessing, also called data preparation, is the process of processing raw data and cleaning, modifying, and rearranging it before analysis. This step often requires formatting, adjustment, and integration to improve the information contained in the datasets. Data preprocessing is an important step in preparing data for processing and reducing the possibility of bias, but it can be a laborious task [15].

Preprocessing processes were run on the dataset analyzed in this study before analysis with MLBox, an AutoML library. In line with the results shown in Figure 2, no missing/erroneous data was found in the data, and no categorical data type was detected. In addition, in terms of the dataset order, it was seen that the attributes did not have a quality that would disrupt the order in reaching the result. Therefore, no attribute extraction was performed. With the current state of the dataset, the analysis process has begun for the creation and testing of learning models.

2.2. **Data Analyses:**

AutoML technology was used to determine the most suitable algorithms and hyperparameters before the bagging process to be used in the classification of Osmancık and Cammeo rice. AutoML is the preferred technique for performing complex analyses on large datasets. This technique significantly facilitates the analysis of large-scale data compared to traditional analysis methods [16]. AutoML aims to enable machine learning applications to produce better results by making it easier for data analysis experts to easily apply machine learning techniques, as well as to make appropriate hyperparameter adjustments for data analysis experts [17]. AutoML uses an automatic technique that allows machine learning algorithms to be configured at an optimal level. For this reason, the prevalence of its use among researchers continues to increase [18].

TABLE 2. **LGBM algorithm performance values**

Model	Precision	Sensitivity	Specifity	F1-Scrore	Accuracy
LGBM	0.912	0.925	0.928	0.918	0.927

AutoML technology covers the following processes in terms of data analysis [19]:

- Data Preprocessing: It ensures that the quality of the data is maintained by helping to perform various cleaning and preparation operations on the datasets before creating the learning models.
- Model Selection: AutoML allows the most appropriate model to be automatically selected according to the characteristics of the dataset and the classification method to be applied.
- Hyperparameter Optimization: AutoML technology automatically adjusts hyperparameter options that will optimize performance and accuracy without the need for manual intervention by the data analysis researcher.
- Binary, Multi-Class, and Multi-Label Classification: With AutoML tools, they can act in multiple ways in classification scenarios by creating effective solutions to such classification problems.

Many libraries implement AutoML techniques. Within the scope of this study, data analysis was carried out using the AutoML library named MLBox. MLBox is a library developed to perform distributed data processing, cleaning, and formatting processes. In order to provide these features, it supports state-of-the-art machine learning algorithms. In addition to individual algorithms, it can also work with ensemble learning algorithms such as LightGbm and XgBoost [20]. In addition, it can perform feature selection processes in an extremely robust manner and apply accurate hyperparameter optimizations in high-dimensional data structures [21]. MLBox performs data analysis with three basic sub-packages that work in a determined order. The first of these packages, preprocessing, ensures that the data is read and preprocessed if necessary. The second package, optimization, enables the application of appropriate hyperparameter optimizations and the testing processes of the created learning models. The third step, prediction, carries out the process of predicting the result using the obtained learning models and input data. The working order of MLBox occurs automatically since it is an AutoML library [22].

3. RESULTS

In our study, in order to achieve the best level of accuracy in classifying rice grains, the appropriate algorithm and hyperparameters were determined with the MLBox library, and then the detected algorithm was subjected to the Bagging process with hyperparameters, aiming to increase the performance rate incorrect predictions.

In the first stage, as a result of the analysis performed with the MLBox library, it was concluded that the Light Gradient Boosting Machine (LGBM) algorithm was suitable for the dataset with the hyperparameters given in Figure 3. The classification success of the LGBM algorithm on the dataset was determined as 92.70%, and it was observed that it achieved successful classification. Table 2 gives the performance values of the LGBM algorithm on the dataset.

```
LGBMClassifier(max_depth=3, boosting_type='gbdt', class_weight=None, colsample_bytree=0.8,
               importance_type='split', learning_rate=0.05, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=500, n_jobs=-1, num_leaves=31, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=0.9,
               subsample_for_bin=200000, subsample_freq=0, nthread=-1, seed=0)
```

FIGURE 3. **LGBM classification algorithm hyperparameters.**

```
BaggingClassifier(LGBMClassifier(max_depth=3, boosting_type='gbdt', class_weight=None, colsample_bytree=0.8,
                  importance_type='split', learning_rate=0.05, min_child_samples=20, min_child_weight=0.001,
                  min_split_gain=0.0, n_estimators=500, n_jobs=-1, num_leaves=31, objective=None,
                  random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=0.9,
                  subsample_for_bin=200000, subsample_freq=0, nthread=-1, seed=0),
                 max_samples=0.1, max_features=0.5, n_estimators=50)
```

FIGURE 4. **Bagging process hyperparameters.**

TABLE 3. **Performance of the learning model produced with bagging technique**

Tests	Precision	Sensitivity	Specifity	F1-Scrore	Accuracy
Test 1	0.905	0.951	0.925	0.927	0.936
Test 2	0.909	0.949	0.928	0.929	0.937
Test 3	0.916	0.938	0.932	0.927	0.935
Test 4	0.902	0.946	0.923	0.924	0.933
Test 5	0.907	0.938	0.926	0.922	0.931
Test 6	0.909	0.944	0.928	0.927	0.935
Test 7	0.909	0.951	0.928	0.930	0.938
Test 8	0.909	0.951	0.928	0.930	0.938
Test 9	0.902	0.944	0.923	0.923	0.932
Test 10	0.916	0.947	0.933	0.931	0.939
Mean	**0.909**	**0.946**	**0.927**	**0.927**	**0.9354**

In the second stage, the Bagging process was applied to the learning model using the LGBM algorithm and the determined hyperparameters, and the aim was to increase the classification performance rate. The bagging process was carried out using the hyperparameters shown in Figure 4, and the performance rate increased as targeted. As a result of the tests carried out using the learning model created with the bagging method, the correct classification success of the model reached an average level of 93.54% (Table 3). This result shows that the Bagging technique, one of the ensemble learning methods, has a positive effect on the model performance.

TABLE 4. **Performance values of models obtained from different studies**

Model	Precision	Sensitivity	Specifity	F1-Scrore	Accuracy
LR	0.915	0.923	0.937	0.918	0.9302
DNN	0.911	0.925	0.935	0.918	0.9304
Bagging	0.909	0.946	0.927	0.927	0.9354

Analyzes were carried out in different studies on the dataset we used in our study and the results were written. According to these studies, various classification algorithms were tested with the dataset by Çınar and Köklü [**?**] and it was stated that the best accuracy rate was determined as 93.02% with the Logistic Regression (LR) algorithm. In addition, in a study conducted by İlhan et al. [8], a learning model was created with Deep Neural Networks (DNN) and it was written that they reached an average accuracy rate of 93.04%. In our study, an average accuracy rate of 93.54% was achieved in the learning model created using the Bagging technique, which was carried out after the algorithm and Hyperparameter determination process with the MLBox library. Table 4 shows the comparison of performance values of the models obtained in the studies.

The structure of the dataset used in the study and the maintenance data specified in Table 4 stopped the initiation of the correct feature extraction process for the classification of the analyzed certificates. The high accuracy rates obtained in the analyses performed indicate these features. It is foreseen that analyses performed with different machine learning and deep learning techniques will also realize these developments in the near future. However, the proposed learning model structure can produce better results than preferring section structures instead of relying on a single model.

4. CONCLUSION

Within the scope of the study, the dataset containing the characteristics of Osmancık and Cammeo rice types, published as open source in the UCI Irvine Machine Learning Repository, was used. In the study, to perform analyses on the dataset, the MLBox library, one of the AutoML libraries, was used to determine the classification algorithm suitable for analysis and the hyperparameters that gave the most accurate results. At this stage, MLBox suggested the LGBM algorithm as a result, and an accuracy rate of 92.70% was achieved with the created learning model. In the next stage, the Bagging technique was applied to the LGBM algorithm to improve the learning model in order to achieve better results. An average accuracy level of 93.54% was obtained in the analyses made using the new learning model developed with the bagging technique.

When the performance values obtained in the study were examined, it was concluded that the learning model created was successful. In addition, a comparison was made with the performance values obtained in different studies on the same dataset (Table 4). As a result of this comparison, it was observed that all models had performance values close to each other and made successful classification. It can be said that the learning model obtained in our study gives slightly better results than the models in other studies.

In the future, in line with the results obtained from these studies, it is recommended to create automation structures for rapid classification of agricultural products and identification of their types. In this way, it will be possible to control the products produced in the agricultural sector faster and with minimum errors.

DECLARATIONS

- **Contribution Rate Statement:** Cihan BAYRAKTAR has conducted this study as a single author.
- **Conflict of Interest:** The author declares that they have no conflict of interest.
- **Data Availability:** Data are available at https://doi.org/10.24432/C5MW4Z
- **Statement of Support and Acknowledgment:** We would like to thank the researchers who carried out the necessary studies to prepare the dataset used in this study and made it available under the CC BY 4.0 license [7]. The dataset used in the study can be accessed at https://doi.org/10.24432/C5MW4Z Additionally, we thank the anonymous referees for their thoughtful comments and suggestions on the manuscript.

REFERENCES

[1] R. Alamyar, I. Boz, Marketing problems encountered by rice producers and their solutions: A case study of takhar-afghanistan, ISPEC Journal of Agricultural Sciences 5 (2) (2021) 381–392. doi:10.46291/ISPECJASVOL5ISS2PP381-392.

[2] E. Veziroglu, I. Pacal, A. Coskuncay, Derin evrişimli sinir ağları kullanılarak pirinç hastalıklarının sınıflandırılması, Journal of the Institute of Science and Technology 13 (2) (2023) 792–814. doi:10.21597/JIST.1265769.
URL https://dergipark.org.tr/en/pub/jist/issue/77307/126576

[3] Food and Agriculture Organization of the United Nations, FAOSTAT dataset, accessed: 2021 (2021).
URL https://www.fao.org/faostat/en/#data

[4] Z. C. Mutafcilar, Türkiye'de tescilli çeltik çeşitlerinin moleküler karakterizasyonu, Ph.d. thesis, Trakya University (2018).

[5] M. Koklu, I. Cinar, Y. S. Taspinar, Classification of rice varieties with deep learning methods, Computers and Electronics in Agriculture 187 (2021) 106285. doi:10.1016/j.compag.2021.106285.

[6] D. I. Patrício, R. Rieder, Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review, Computers and Electronics in Agriculture 153 (2018) 69–81. doi:10.1016/j.compag.2018.08.001.

[7] I. Cinar, M. Koklu, Classification of rice varieties using artificial intelligence methods, International Journal of Intelligent Systems and Applications in Engineering 7 (3) (2019) 188–194. doi:10.18201/IJISAE.2019355381.

[8] U. Ilhan, A. Ilhan, K. Uyar, E. I. Iseri, Classification of osmancik and cammeo rice varieties using deep neural networks, in: ISMSIT 2021 - 5th International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings, Institute of Electrical and Electronics Engineers Inc., Ankara, Turkey, 2021, pp. 587–590. doi:10.1109/ISMSIT52890.2021.9604606.

[9] B. Jin, C. Zhang, L. Jia, Q. Tang, L. Gao, G. Zhao, H. Qi, Identification of rice seed varieties based on near-infrared hyperspectral imaging technology combined with deep learning, ACS Omega 7 (6) (2022) 4735–4749. doi:10.1021/acsomega.1c04102.

[10] D. Jaithavil, S. Triamlumlerd, M. Pracha, Paddy seed variety classification using transfer learning based on deep learning, in: Proceedings of the 2022 International Electrical Engineering Congress, iEECON 2022, Institute of Electrical and Electronics Engineers Inc., Khon Kaen, Thailand, 2022, pp. 1–4. doi:10.1109/IEECON53204.2022.9741677.

[11] J. Jumi, A. Zaenuddin, T. Mulyono, Identification of rice types based on shape, color and texture using k-nearest neighbors method as classifier, International Journal of Engineering Research Technology 9 (12) (2020). doi:10.17577/IJERTV9IS120013.

[12] V. T. Hoang, D. P. Van Hoai, T. Surinwarangkoon, H. T. Duong, K. Meethongjan, A comparative study of rice variety classification based on deep learning and hand-crafted features, ECTI Transactions on Computer and Information Technology (ECTI-CIT) 14 (1) (2020) 1–10. doi:10.37936/ECTI-CIT.2020141.204170.

[13] M. S. Mrutyunjaya, K. S. Harish Kumar, Non-destructive machine vision system based rice classification using ensemble machine learning algorithms, Recent Advances in Electrical Electronic Engineering (Formerly Recent Patents on Electrical Electronic Engineering) 16 (jul 2023). doi:10.2174/2352096516666230710144614.

[14] I. Cinar, M. Koklu, Rice (cammeo and osmancik) (2019). doi:10.24432/C5MW4Z.

[15] C. El Morr, M. Jammal, H. Ali-Hassan, W. El-Hallak, Data preprocessing, Springer International Publishing, Cham, 2022, pp. 117–163. doi:10.1007/978-3-031-16990-8_4.

[16] V. Kovalevsky, E. Stankova, N. Zhukova, O. Ogiy, A. Tristanov, Automl framework for labor potential modeling, in: Advances in Intelligent Systems and Computing, Vol. 13957, Springer, Cham, 2023, pp. 87–98. doi:10.1007/978-3-031-36808-0_6.

[17] Y. Sun, Q. Song, X. Gui, F. Ma, T. Wang, Automl in the wild: Obstacles, workarounds, and expectations, in: Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, Hamburg, Germany, 2023, pp. 1–15. doi:10.1145/3544548.3581082.

[18] C. Wang, Z. Chen, M. Zhou, Automl from software engineering perspective: Landscapes and challenges, in: Proceedings - 2023 IEEE/ACM 20th International Conference on Mining Software Repositories, MSR 2023, IEEE Inc., Melbourne, Australia, 2023, pp. 39–51. doi:10.1109/MSR59073.2023.00019.

[19] M. Vinícius, C. Aragão, A. Guimarães Afonso, R. C. Ferraz, R. Gonçalves Ferreira, S. Gomes Leite, R. G. Ferreira, A practical evaluation of automl tools for binary, multiclass, and multilabel classification, TechRxiv (oct 2023). doi:10.36227/TECHRXIV.21792959.V1.

[20] S. Das, U. M. Cakmak, Hands-On Automated Machine Learning: A Beginner's Guide to Building Automated Machine Learning Systems Using AutoML and Python, Packt Publishing, Birmingham, UK, 2018.

[21] A. Aronio De Romblay, N. Cherel, M. Maskani, H. Gerard, Mlbox (2017).

[22] S. Ozdemir, S. Orslu, Makine ögrenmesinde yeni bir bakış açısı: Otomatik makine Ögrenmesi (automl), Journal of Information Systems and Management Research 1 (1) (2019) 23–30.

Vol: 2, Issue: 2, 2024
Pages:96-107
Received: 15 October 2024
Accepted: 13 December 2024

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE
DOI: 10.71074/CTC.1562363

ON THE POLYNOMIAL MULTIPLICATION ALGORITHMS FOR POST-QUANTUM CRYPTOGRAPHY

EBRU YALÇIN[1*], FIDAN NURIYEVA[2,3] AND ERDEM ALKIM[2]

[1] *The Graduate School of Natural and Applied Sciences, Department of Computer Science, Dokuz Eylul University, 35390, Izmir, Türkiye*
[2] *Department of Computer Science, Dokuz Eylul University, 35390, Izmir, Türkiye*
[3] *Institute of Control Systems, The Ministry of Science and Education of the Republic of Azerbaijan, Baku, Azerbaijan*

ABSTRACT. **This study explores the multiplication operations carried out on polynomial rings within lattice-based systems used in post-quantum cryptography. Polynomial rings of high degree are utilized to enhance security in post-quantum cryptography. Since multiplication is the most time-consuming arithmetic operation on polynomial rings, several algorithms have been suggested to optimize newly developed systems by enhancing their efficiency. Typically, these algorithms use the properties of the chosen polynomial ring to minimize the number of multiplications, however, some arithmetical tricks can be used to use them for other rings. Therefore, the systems are optimized in terms of efficiency and cost. In this study, we investigated several multiplication algorithms based on their complexity and reported the results from the literature for their implementation efficiency. We have compared those algorithms when they were implemented to perform multiplications on the same polynomial ring and reported that the ring of the coefficients should be also considered when comparing the efficiency.**

1. INTRODUCTION

In the modern day, as technology advances and becomes more widely utilized, the need to guarantee the security of systems and networks has become a significant concern due to the possibility of vulnerabilities such as data breaches and cyber threats. Cryptology safeguards the authenticity and secrecy of delicate and classified data, shielding it from illegal intrusion. Ongoing research is being conducted to address emerging challenges in the field of computational difficulties and vulnerabilities in systems, which have arisen as a result of advancements and contributions to the existing body of knowledge. Due to advancements in technology and recent research, the introduction of quantum computing has revolutionized the field of science and prompted a reassessment of current cryptography methods.

E-mail address: ebru.yalcin305@gmail.com, fidan.nuriyeva@deu.edu.tr(*), erdem.alkim@deu.edu.tr.
Key words and phrases. Lattice-based cryptography, Polynomial multiplication algorithms, Number theoretic transform, Bruun algorithm.

Shor's algorithm is a quantum algorithm that provides a polynomial solution to the discrete logarithm problem, which is used in the current cryptographic protocol [1]. Shor's algorithm is a significant technique that utilizes quantum computing principles to perform operations on big integers, namely for factoring and solving fractional logarithms. These mathematically challenging problems seem to serve as the foundation for numerous encryption methods. Shor's algorithm is a prominent method that leverages the concepts of quantum computing to carry out computations on large integers, namely factoring and solving fractional logarithms. These mathematically complex difficulties appear to be the basis for many encryption systems. Shor's method presents a substantial risk to the security of widely utilized public key cryptosystems such as RSA and ECC. In 2018, the National Institute of Standards and Technology (NIST) in the United States launched a standardization project to tackle these emerging challenges. Developed specifically to provide a long-term defense against quantum computers, these innovative techniques are based on universally accepted mathematical problems that are difficult for both classical and quantum technology to solve [2].

Lattice-based systems are the most promising and prominent approach among recently developed systems. Lattice-based systems are notable due to the elevated complexity of lattice problems, which come from the challenging nature of mathematical issues. Their characteristics enable lattice-based systems to offer a resilient encryption mechanism and an effective defense against attacks. Lattice-based systems perform computations on polynomial rings. The primary benefit of utilizing these systems operating on polynomial rings lies in their inherent algebraic structure, which enables rapid expression of polynomial coefficients and proper execution of operations. The efficient storage of polynomial coefficients and the facilitation of effective operations are made feasible by this structure [3]. While lattice-based systems offer numerous advantages, polynomial multiplication is a computationally expensive operation. The computational load of processing the polynomials increases significantly due to the quick increase in multiplication complexity, which depends on the degree of the polynomials. Novel polynomial multiplication algorithms have been suggested to address this issue. These novel multiplication algorithms employ several techniques to decrease the computational complexity of the point-wise multiplication process and enhance and optimize overall efficiency.

The primary instances of these multiplication algorithms include the School-Book, Karatsuba, Toom-Cook, The Number Theoretic Transform (NTT), and Toeplitz Matrix-Vector Multiplication (TMVP) algorithms. The School-Book algorithm is the most fundamental and commonly used method for polynomial multiplication in literature. This algorithm is implemented by performing a straight multiplication of two polynomials. Karatsuba is a multiplication algorithm that reduces the total number of multiplications by employing the divide-and-conquer approach. It accomplishes this by dividing the polynomials into smaller segments while executing the multiplication. The NTT algorithm is a mathematical transformation method derived from the Fast Fourier Transform (FFT). It is an enhanced version of the FFT that has been further developed using number field theory. The NTT algorithm is mostly used for polynomial multiplication. This study also investigates the TMVP method, which is a specific algorithm that exploits the Toeplitz matrix structure commonly encountered in lattice-based systems. The Method section examines a polynomial multiplication algorithm known as the Bruun algorithm [4].

These multiplication algorithms are seen to be used on many different schemes today. For example, Kyber [5], Falcon [6], and Dilithium [7], among the projects that made it to NIST's standardization competition final in post-quantum cryptography are lattice-based systems. These schemes use polynomial multiplication extensively in their different stages and aim to ensure efficiency and security. They aim to speed up polynomial multiplications and reduce the complexity of the operation by using polynomial multiplication algorithms such as NTT and FFT. In this way, large-degree polynomials can be operated on, allowing complex calculations to be made. Additionally, these algorithms appear to produce accurate and reliable results. Due to these features, it appears to reduce the load on the processor and optimize energy consumption. For lower-power devices and embedded systems, these features are important. Thus, the schemes used in post-quantum cryptography are expected to work successfully in real-life applications.

Lattice-based cryptography has become a leading candidate for post-quantum security due to its robustness and reliance on complex mathematical problems. A critical aspect of these systems is polynomial multiplication, a resource-intensive operation that significantly impacts performance. Efficient algorithms such as NTT and TMVP play a vital role in optimizing cryptographic schemes like NTRU. This study focuses on improving polynomial multiplication to enhance the efficiency and practicality of post-quantum cryptographic systems for real-world applications.

In this study, information is given on polynomial multiplication algorithms used in lattice-based systems. In section two, firstly, the definition and mathematical representation of the polynomial ring and the definition of polynomial multiplication are given. In the same section, polynomial multiplication algorithms frequently used in lattice-based systems; NTT algorithm, and TMVP algorithm were examined. In the Third section, Bruun's algorithm is introduced. In chapter four, the results are given. Finally, in chapter five, we conclude our paper.

2. Polynomial Multiplication

Polynomial multiplication refers to the process of multiplying two polynomials inside the same polynomial ring. The current scenario can be expressed in the following manner.

Definition 2.1. Let $\mathscr{R}$ be an accumulative ring, $N \in \mathbb{N}$, and $0 \leq i < N$. Let $a_i, d_i, c \in \mathscr{R}$ be coefficients. The polynomials $a(x)$ and $d(x)$ are subjected to the polynomial multiplication operation within the same polynomial ring, resulting in:

$$c = a(x) \cdot d(x) \tag{1}$$

$$c_k = \sum_{i=0}^{k} a_i d_{k-i} + \sum_{i=k+1}^{N-1} a_i d_{N+k-i} = \sum_{\substack{j+i \equiv k \\ (\text{mod } N)}} a_i d_j \tag{2}$$

The polynomial multiplication operation takes place in the ring $\mathscr{R} = \mathbb{Z}_q/(x^N - 1)$, and the factors and product elements become elements of the ring $\mathscr{R} = \mathbb{Z}_q/(x^N - 1)$. Specifically, if $2x+1 = c_1$, then $2x+1$ serves as a factor and a product element within the ring $\mathscr{R} = \mathbb{Z}_q/(x^N - 1)$ [8].

Polynomial multiplication is typically carried out on polynomials with high degrees. Multiplication, which is one of the arithmetic operations carried out on polynomials, requires a greater amount of time and computational power compared to other operations. Consequently, researchers have conducted investigations to enhance this circumstance by developing polynomial multiplication algorithms. These algorithms are implemented on various systems based on specific requirements. Polynomial multiplication algorithms can be categorized as follows: School-Book, Toom-Cook, Number Theoretic Transform, Toeplitz Matrix-Vector Product, and Bruun.

The subsequent part delves into a thorough examination of polynomial multiplication algorithms, analyzing each one individually. Explicit formulations and efficient algorithms are provided.

2.1. **Number Theoretic Transform:**

The Number Theoretic Transform (NTT) is a mathematical technique mostly employed for solving the factorization issue. Its output is derived from the Fast Fourier Transform. According to [9] it is also claimed that this is a version of DFT that operates on finite fields rather than complex numbers. The method originated as an extension of the FFT, although its precise output remains uncertain. Several mathematicians and computer scientists made significant contributions to the initial investigations of NTT. S.S. Winograd introduced a polynomial evaluation procedure utilizing the Chinese remainder theorem [10] after examining the literature. The investigation led to the invention of NTT, which focuses on the remaining parts based on prime numbers. Using the findings from conducted studies, an algorithm was designed that utilizes CRT and modular arithmetic operations to accomplish polynomial multiplication, incorporating novel features. He enhanced the development of NTT's applications by incorporating this algorithm. NTT has achieved its current level of usage through the contributions of fundamental research areas like FFT and modular arithmetic. These areas have been crucial in developing efficient algorithms for polynomial multiplication and polynomial factorization.

NTT is a mathematical operation called the fractional Fourier transformation, which is defined on the ring $\mathscr{R}_q = \mathbb{Z}_q/\Phi_m(x)$. It performs fast calculations on polynomials, hence improving the efficiency of polynomial multiplication. A polynomial $a(x)$ over the ring $\mathscr{R}_q$, with a degree of $n-1$, can be represented as:

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \tag{3}$$

The NTT of a polynomial $\bar{a}(x)$ of degree $n-1$ on the ring $\mathscr{R}_q$ is represented in polynomial form as:

$$\bar{a} = \sum_{i=0}^{n-1} \bar{a}_i x^i \tag{4}$$

where the coefficients $\bar{a}$ can be defined using the following (2.38):

$$\bar{a} = \sum_{j=0}^{n-1} \bar{a}_j w^{i \cdot j} \pmod{q} \quad \text{for } i = 0, 1, 2, \ldots, n-1. \tag{5}$$

The equation involves a twiddle factor, denoted as ω, which must satisfy the criteria $\omega^n \equiv 1 \pmod{q}$, and $\omega^i \neq 1 \pmod{q}$ for all $i < n$. The NTT operation is executed by computing this equation for every value of i ranging from 0 to $n-1$.

The NTT operation employs a constant known as the twiddle factor, $\omega \in \mathbb{Z}_q$, which represents the n-th root of unity. The method utilizes a basic n-th root of unity, $\omega \in \mathbb{Z}_q$, which fulfills the requirements $\omega^n \equiv 1 \pmod{q}$, $\omega^i \not\equiv 1 \pmod{q}$ for all $i < n$, and $q \equiv 1 \pmod{n}$. The inverse Number Theoretic Transform (INTT) operation follows a similar method, except in the last step, the element $\omega^{-1} \in \mathbb{Z}_q$ is employed instead of ω. Furthermore, in the mathematical field $\mathbb{Z}_q$, while performing the final step of the Inverse Number Theoretic Transform (INTT) calculation, the resulting coefficients are multiplied by the inverse of n^{-1} [11].

2.2. **Toeplitz Matrix-Vector Product:**

The TMVP algorithm is a highly efficient method for multiplying matrices and vectors, specifically designed to exploit the distinctive characteristics of Toeplitz matrices. The actual origin and date of the proposal for TMVP are uncertain, although it is believed that the concept of utilizing Topelitz matrices for efficient computations is derived from the research conducted by Otto Toeplitz. German mathematicians specialized in the areas of algebraic and numerical analysis. Toeplitz matrices are square matrices whose each diagonal has constant values. These matrices possess mathematical properties that make them useful for efficient calculations [12]. Subsequent works further investigated and elaborated on the concept of utilizing Toeplitz matrices for polynomial multiplication. Utilizing Toeplitz matrices aids in diminishing the overall intricacy through the pre-calculation and reuse of certain pieces. Furthermore, when multiple multiplication operations are required on a single matrix, TMVP executes these operations efficiently by avoiding redundant calculations, hence greatly enhancing efficiency. Given this circumstance, utilizing it in polynomial multiplication operations, which are crucial in lattice-based systems prevalent in post-quantum cryptography, offers several benefits. Toeplitz matrices are commonly employed in various cryptographic applications, as evidenced by their frequent appearance in the literature [13], [14], [15].

A TMVP (n-dimensional) can be computed by utilizing three TMVPs ($n/2$-dimensional) in a 2-way TMVP formula. Denote the half-dimensional divisions of T as T_0, T_1, and T_2, and the half-dimensional partitions of the vector V as V_0 and V_1. The calculation of the N-dimensional matrix-vector multiplication is performed in the following manner:

$$T \cdot V = \begin{pmatrix} T_1 & T_0 \\ T_2 & T_1 \end{pmatrix} \begin{pmatrix} V_0 \\ V_2 \end{pmatrix} = \begin{pmatrix} P_0 + P_1 \\ P_0 - P_2 \end{pmatrix} \tag{6}$$

and

$$
\begin{aligned}
P_0 &= T_1(V_0 + V_1), \\
P_1 &= (T_0 - T_1)V_1, \\
P_2 &= (T_1 - T_2)V_0
\end{aligned} \tag{7}
$$

The complexity of TMVP_2 can be expressed as

$$M_{\text{TMVP}_2}(n) = 3M(n/2) + 3n - 1$$

based on the operations mentioned above. Furthermore, an n-dimensional Total Mean Value Projection (TMVP) can be computed by utilizing three $n/3$-dimensional TMVPs in a 3-way TMVP equation. By following the same procedure as for TMVP_2, we can determine the existence of TMVP_3. The complexity of TMVP_3 can be expressed as

$$M_{\text{TMVP}_3}(n) = 6M(n/3) + 5n - 1.$$

It is evident that when performing polynomial multiplication, various formulas are generated for TMVPs based on their size and effectiveness. The article [16] provides a more detailed analysis of the process of developing additional TMVP formulas using the given formulas mentioned above.

2.2.1. *Polynomial Multiplication Modulo $x^n \pm 1$ via TMVP.*

In the polynomial multiplication operation, when performed on $\mathbb{Z}[x]/(x^n \pm 1)$, the resulting polynomial $\mathbb{Z}[x]$ should be reduced by using $x^n \pm 1$. TMVP leverages the structure of Toeplitz matrices and the properties of polynomial multiplication to optimize the operations to be executed. TMVP utilizes polynomial multiplication operations on $x^n \pm 1$ to answer a wide range of issues across several disciplines. The modulo operation is of great importance in the coding and decoding procedures of Error-correcting codes as it enables the identification and correction of errors. Cryptography employs it in several systems, including digital signatures and encryption algorithms. It offers a highly effective computational capability for the systems in which it is employed. Since reduction modulo $x^n \pm 1$ is only a addition or subtraction T_2 becomes $\pm T_0$, thus equ. 6 and equ. 7 becomes:

$$T \cdot V = \begin{pmatrix} T_1 & T_0 \\ \pm T_0 & T_1 \end{pmatrix} \begin{pmatrix} V_0 \\ V_2 \end{pmatrix} = \begin{pmatrix} P_0 + P_1 \\ P_0 - P_2 \end{pmatrix} \tag{8}$$

and

$$
\begin{aligned}
P_0 &= T_1(V_0 + V_1) \\
P_1 &= (T_0 - T_1)V_1 \\
P_2 &= (T_1 \pm T_0)V_0
\end{aligned} \tag{9}
$$

3. Factorization of the Cyclotomic polynomial $x^{2^k}+1$

The Bruun technique, as described by George Bruun in his 1978 publication, is a Discrete Fourier Transform algorithm specifically designed for real numbers with a logarithmic basis [17]. These processes, which are associated with the traditional complex FFT, allow for the utilization of new FFT variations that exclusively operate with real coefficients. Additionally, the implementation of new FFT algorithms involves utilizing only half the number of real multiplications compared to existing FFT methods.

The Bruun method is the method employed to ascertain the factors of a polynomial that encompasses all unit roots. This method utilizes the structural characteristics of the unit roots of polynomials to expedite the computation of polynomial factors. This approach, employed in areas such as number theory and modular arithmetic, is said to enhance the efficiency of polynomial calculations in post-quantum cryptography. Bruun's approach is designed to factorize the roots of a polynomial of degree n. It achieves this by recursively finding the explicit roots of the polynomial.

The new structure demonstrates a logarithmic reduction in calculation time and achieves processing efficiency by dividing the DFT operations into segments and executing certain parallel operations. Therefore, it can be stated that intricate discrete Fourier transform (DFT) processes have experienced an increase in efficiency. In the classical approach of FFT, $N/2\log N$ complex multiplication operations are performed, where two complex numbers are multiplied in each operation. The new method, in contrast to the old one, was demonstrated to involve the multiplication of a real number and a complex number.

Using Bruun's algorithm, the same outcome as the classical technique can be achieved by employing the multiplication of real and complex numbers instead of complex multiplications. It is evident that the new algorithm has decreased the utilization of intricate multiplication in the classical way by half. Therefore, it is widely acknowledged that the Bruun algorithm operates with greater speed and efficiency.

This section demonstrates the complete separation of the polynomial $x^{2^k}+1$ on $\mathbb{F}_p$ into separable polynomials. p is a prime number that fulfills the criterion $p \equiv 3 \pmod 4$. Therefore, it is demonstrated that it is possible to create an irreducible polynomial over $\mathbb{F}_p$ with a degree that is a power of 2. Therefore, it is evident that this approach can be effectively utilized in FFT applications within limited regions.

The following theorem pertains to the situation when p is a prime integer and is entirely irreducible on $\mathbb{F}_p$, subject to the constraints of $p \equiv 3 \pmod 4$, meaning $p \equiv -1 \pmod{2^{k+1}}$ [18].

Let p be a prime number that satisfies $p \equiv 3 \pmod 4$. The process of complete factorization of the polynomial $x^{2^k}+1$ on the $\mathbb{F}_p$ field is investigated. To simplify the problem, it can be seen that the roots of the polynomial $x^{2^k}+1$ are actually the primitive $x^{2^k}+1$-th roots of unity in an expansion field of $\mathbb{F}_p$. Therefore, the goal is to construct the smallest polynomials on $\mathbb{F}_p$ for primitive $x^{2^k}+1$-th roots of unity where k is an integer greater than or equal to 1. If we consider that the degree of every i

Let the highest exponent of 2 in $p+1$ be denoted as 2^a. The expression p^2-1, where p is a variable, represents the maximum power of 2 and is denoted as 2^{a+1}. Assuming $\alpha \in \mathbb{F}_{p^2}$, let's consider that it has a degree of 2^{a+1}. It should be noted that the polynomial $x^{2^e}+\alpha$ is irreducible over the field $\mathbb{F}_{p^2}$ for $e \geq 0$. According to this information, the 2^{e+1}-th order cannot be broken down on $\mathbb{F}_p$, and the primitive roots

of the 2^{a+e+1}-st order are given as $(x^{2^e}+\alpha)(x^{2^e}+\alpha^p)$. The article states that $\mathbb{F}_{p^2}=\mathbb{F}_p(i)$, where i is the square root of -1. Additionally, since f is defined as $\mathbb{F}_{p^2}\to\mathbb{F}_{p^2}$, it is expressed as $(1+x)^{\frac{p-1}{2}}$ [19], [20].

The following formula provides the steps for calculating the square root in $\mathbb{F}_{p^2}$. The formula applicable to any second-order α residue in $\mathbb{F}_p(i)$ is given as follows.

$$\sqrt{\alpha}=\begin{cases} i\alpha^{\frac{p+1}{4}}, & \text{if } \alpha^{\frac{p-1}{2}}=-1,\\ \left(1+\alpha^{\frac{p-1}{2}}\right)^{\frac{p+1}{2}}\alpha^{\frac{p+1}{4}}, & \text{otherwise.}\end{cases} \tag{10}$$

If k is greater than 0, and the order of the element α is 2^k, both $\sqrt{\alpha}$ and $\sqrt{-\alpha}$ have an order of $x^{2^k}+1$. If we define the starting point as $i=\sqrt{-1}$, it becomes evident that the numbers with an exponent of 2^k can be determined using a recursive process. Prior to factoring the polynomial $x^{2^k}+1$, it is essential to calculate the minimum polynomials of all the elements produced in $\mathbb{F}_p$.

Theorem 3.1. *Let* $H_1=\{0\}$.

$$H_k=\pm\left\{\left(\frac{u+1}{2}\right)^{\frac{p+1}{4}}\right\} \quad \textit{for } u\in H_{k-1} \tag{11}$$

For every value of k from 1 to $a-1$*, the cardinality of* H_k *is equal to* 2^{k-1}*,*

$$x^{2^k}+1=\prod_{u\in H_k}\left(x^2-2ux+1\right) \tag{12}$$

For any integer $e\geq 0$*,*

$$x^{2^k}+1=\prod_{u\in H_k}\left(x^{2^{e+1}}-2ux^{2^e}-1\right) \tag{13}$$

The aforementioned theorem can be used for additional cyclotomic polynomials; however, it should be noted that these polynomials are not directly connected to the Bruun paper. Nevertheless, there exists a connection between them. Bruun's Algorithm utilizes polynomial factorization to conduct DFT computations. This is analogous to the factorization of cyclotomic polynomials, as both approaches involve dividing polynomials into smaller components. The relationship between Bruun's algorithm and cyclotomic polynomials is established by the factorization of polynomials and the utilization of unit roots. This connection offers enhanced efficiency and rapidity in DFT calculations and polynomial factorizations.

4. Results

The concluding part is organized into three distinct topics. Initially, an analysis was conducted on the number of multiplications and their complexity, which are determined by specific parameters (n), for the multiplication operation. This operation is known to be the most time-consuming and costly among the various operations performed on polynomial rings. The second title provides a general explanation of the Bruun approach and includes some inferred information. In the last heading provides details on

polynomial multiplication methods and discusses their efficiency.

4.1. **Multiplication Algorithms:**

Below is a table quoted from the article [21], examining the number of multiplications and time complexities in multiplication algorithms depending on certain n parameters. In this table, one of the frequently used polynomial multiplication algorithms; Scool-Book, Karatsuba, Toom-Cook-way, $TMVP_2$, $TMVP_3$, $TMVP_4$ and NTT algorithms were examined.

TABLE 1. **Complexities of multiplication algorithms**

No	Multiplication Algorithms	Complexity
1	School-Book	$T(n) = 2n^2 - 2$
2	Karatsuba	$T(n) = 3T(n/2)$
3	Toom-Cook-k	$T(n) = (2k-1)T(n/3)$
4	TMVP-2	$T(n) = 3T(n/2) + 3n - 1$
5	TMVP-3	$T(n) = 6T(n/3)$
6	TMVP-4	$T(n) = 7T(n/4) + 5n - 1$
7	NTT	$T(n) = \frac{3}{2}n\log n + n$
8	Bruun	$T(n) = \frac{3}{2}n\log n + n$

Using the information in Table 1, some inferences for multiplication algorithms for certain parameters are given in Table 2 below.

We have chosen a specific parameter set of NTRU, known as `ntruhrss701`, for the purpose of comparing algorithms. In this parameter set, the value of q is 2^{13}, n is 701, and $f(x)$ is defined as $x^{701} - 1$. Consequently, the polynomial ring can be represented as $\mathbb{Z}_{2^{13}}[x]/(x^{701} - 1)$.

To carry out multiplication in the ring $\mathbb{Z}_{2^{13}}[x]/(x^{701} - 1)$, TMVP must divide the input polynomials. Given that n is a prime integer, it is necessary to select a size that is larger than n to perform the multiplication calculation. Since the majority of implementations focus on sizes in the form of $a' = 2^k 3^l t$, where t is less than 16 for optimal performance, we computed the number of recursion steps for the two shortest possibilities within the chosen polynomial ring.

$$704 \xrightarrow{\text{TMVP}_4} 176 \xrightarrow{\text{TMVP}_2} 88 \xrightarrow{\text{TMVP}_2} 44 \xrightarrow{\text{TMVP}_2} 22 \xrightarrow{\text{TMVP}_2} 11 \tag{14}$$

$$720 \xrightarrow{\text{TMVP}_4} 180 \xrightarrow{\text{TMVP}_3} 60 \xrightarrow{\text{TMVP}_3} 20 \xrightarrow{\text{TMVP}_2} 10 \tag{15}$$

The $T(n')$ values in Table 2 were computed based on the complexity provided in Table 1. The number of cycles reported by [15], [22], and [23] were all measured on the ARM Cortex-M4 Discovery board, which served as the common target platform. It can be seen that Table 2 below was created using the parameters 704, 720, 1440, and 1536.

Table 2 comprises $T(n)$ values computed based on the time complexity algorithms provided in Table 1. Upon examination and comparison of the TMVP, NTT, and Toom-Cook multiplication algorithms, it is evident that NTT-based polynomial multiplication necessitates fewer operations, even for dimensions of $2x$. However, it is important to note that NTT does require modular arithmetic operations. In contrast, the TMVP and Toom-Cook algorithms are capable of operating with two basis powers, eliminating the need for additional modular reduction following elementary arithmetic operations. While the TMVP algorithm incorporates polynomial reduction in its multiplication operations, it necessitates fewer operations.

TABLE 2. **Multiplication Cycles for TMVP, NTT, and Toom-Cook Algorithms**

	TMVP		NTT		Toom-Cook	
n'	$T(n')$	#Cycles	$T(n')$	#Cycles	$T(n')$	#Cycles
704	115331	142252 [24]	-	-	68607	172882 [22]
720	125519	-	-	-	52500	-
1440	-	-	45360	141000 [23]	-	-
1536	-	-	42240	148000 [23]	-	-

This study examines various multiplication algorithms utilized to enhance the efficiency of Lattice-based cryptographic protocols. It provides comparisons of these algorithms based on their theoretical complexity and current usage in the field. An assessment was conducted based on certain criteria on the NTRU scheme, which is one of the lattice-based systems that reached the final stage in the standardization competition hosted by NIST [21].

Lattice-based systems have excelled among the various schemes in the competition set by NIST in the field of post-quantum cryptography. These systems operate on polynomial rings. To optimize the efficiency of the systems, several novel polynomial multiplication algorithms have been developed to minimize the number of multiplication operations required. The algorithms used in this study are Karatsuba, Toom-Cook, NTT, TMVP, and the Bruun polynomial multiplication algorithm mentioned in the technique section.

When the research on polynomial multiplication algorithms in the literature is generally examined, it is seen that various comparisons and evaluations have been made for these algorithms in terms of efficiency, security, and suitability for different applications. Lattice-based algorithms such as Karatsuba, Toom-Cook, NTT, and TMVP will be used in post-quantum cryptography. It can be said that it is among the polynomial multiplication algorithms that are thought to be very important for systems. It is obvious that the proposed algorithms will bring new features, optimization techniques, and efficiency improvements to this field.

5. CONCLUSION

In this study, multiplication operations performed in lattice-based systems on polynomial rings in post-quantum cryptography are examined. It is obvious that the operation that causes the most time

and cost among the operations on polynomial rings is the multiplication operation. If the value of the parameter n is chosen as 701 for the NTRU scheme, it has been shown that the NTT-based polynomial multiplication algorithm exhibits the highest performance, despite its higher memory usage. TMVP-based polynomial multiplication algorithms have demonstrated superior memory efficiency compared to other algorithms, resulting in only a 1% decrease in multiplication operations. Bruun's algorithm, another suggested approach, achieves the same outcome as other methods by performing multiplication operations on complex and real values. As a future work, we plan to compare the implementation of the Bruun algorithm with NTT for the same polynomial ring.

DECLARATIONS

- **Contribution Rate Statement:** Ebru Yalcin: writing – original draft, resources, methodology, conceptualization, Fidan Nuriyeva: writing – review & editing, methodology, Erdem Alkım: writing – review & editing, methodology.
- **Conflict of Interest:** The authors have not disclosed any competing interests.
- **Data Availability:** Data sharing is not applicable to this article as no datasets were generated or analyzed.
- **Statement of Support and Acknowledgment:** None.

REFERENCES

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing 26 (5) (1997) 1484–1509.

[2] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, Y.-K. Liu, Status report on the first round of the NIST post-quantum cryptography standardization process, National Institute for Standards and Technology Internal Report 8240, https://doi.org/10.6028/NIST.IR.8240 (2019).

[3] D. Micciancio, O. Regev, Lattice-based cryptography, in: Post-quantum cryptography, Springer Berlin Heidelberg, 2009, pp. 147–191.

[4] V. Hwang, A survey of polynomial multiplications for lattice-based cryptosystems, Cryptology ePrint Archive, Paper 2023/1962 (2023).
URL https://eprint.iacr.org/2023/1962

[5] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, D. Stehlé, Crystals-kyber: a cca-secure module-lattice-based kem, in: 2018 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2018, pp. 353–367.

[6] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, D. Stehlé, Crystals–dilithium: Digital signatures from module lattices, Cryptology ePrint Archive (2018).

[7] P. A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, Z. Zhang, Falcon: Fast-fourier lattice-based compact signatures over NTRU, Submission to the NIST's post-quantum cryptography standardization process 36 (5) (2018) 1–75.

[8] R. T. Moenck, Practical fast polynomial multiplication, in: Proceedings of the third ACM symposium on Symbolic and algebraic computation, ACM, 1976, pp. 136–148.

[9] D. Harvey, Faster arithmetic for number-theoretic transforms, Journal of Symbolic Computation 60 (2014) 113–119.

[10] S. Winograd, On computing the discrete fourier transform, Mathematics of computation 32 (141) (1978) 175–199.

[11] K. Derya, A. C. Mert, E. Öztürk, E. Savaş, CoHA-NTT: A configurable hardware accelerator for NTT-based polynomial multiplication, Microprocessors and Microsystems 89 (2022).

[12] O. Toeplitz, Das algebraische analogon zu einem satze von fejér, Mathematische Zeitschrift 2 (1) (1918) 187–197.
[13] S. Ali, M. Cenk, Faster residue multiplication modulo 521-bit mersenne prime and an application to ECC, IEEE Transactions on Circuits and Systems I: Regular Papers 65 (8) (2018) 2477–2490.
[14] M. A. Hasan, N. Meloni, A. H. Namin, C. Negre, Block recombination approach for subquadratic space complexity binary field multiplication based on toeplitz matrix-vector product, IEEE Transactions on Computers 61 (2) (2010) 151–163.
[15] I. K. Paksoy, M. Cenk, TMVP-based multiplication for polynomial quotient rings and application to saber on ARM cortex-M4, cryptology ePrint Archive (2020).
[16] S. Winograd, On multiplication of polynomials modulo a polynomial, SIAM Journal on Computing 9 (2) (1980) 225–229.
[17] G. Bruun, z-transform DFT filters and FFT's, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 56–63.
[18] I. F. Blake, S. Gao, R. C. Mullin, Explicit factorization of $x^{2^k}+1$ over F_p with prime $p \equiv 3 mod 4$, Applicable Algebra in Engineering, Communication and Computing 4 (2) (1993) 89–94.
[19] H. W. Lenstra, Finding isomorphisms between finite fields, Mathematics of Computation 56 (193) (1991) 329–347.
[20] V. Shoup, New algorithms for finding irreducible polynomials over finite fields, Mathematics of computation 54 (189) (1990) 435–447.
[21] E. Yalçın, F. Nuriyeva, E. Alkım, A comparative study on polynomial multiplication algorithms in context on post-quantum cryptography, in: DEU International Symposium Series on Graduate Researches-2022 DataScience, DEU, 2022, pp. 1–10.
[22] M. Kannwischer, P. Bissmeyer, S. Schmidt, Optimizing lattice-based cryptography schemes with structured noise, in: Post-Quantum Cryptography: 5th International Conference, PQCrypto 2019, Fukuoka, Japan, July, Springer, 2019, pp. 81–97.
[23] E. Alkim, V. Hwang, B.-Y. Yang, Multi-parameter support with ntts for ntru and ntru prime on cortex-m4, IACR Transactions on Cryptographic Hardware and Embedded Systems 2022 (4) (2022) 349–371.
[24] I. K. Paksoy, M. Cenk, Faster ntru on arm cortex-m4 with tmvp-based multiplication, IEEE Transactions on Circuits and Systems I: Regular Papers 69 (10) (2022) 4083–4092.

Vol: 2, Issue: 2, 2024
Pages:108-139
Received: 01 December 2024
Accepted: 25 December 2024

CURRENT TRENDS IN COMPUTING
https://dergipark.org.tr/en/pub/ctc
RESEARCH ARTICLE
DOI: 10.71074/CTC.1594291

ENHANCING GREEN COMPUTING THROUGH ENERGY-AWARE TRAINING: AN EARLY STOPPING PERSPECTIVE

ABDULKADIR TAŞDELEN *

Department of Software Engineering, Faculty of Engineering and Natural Sciences, Ankara Yıldırım Beyazıt University, Ankara 06010, Türkiye

ABSTRACT. **This study delves into energy-efficient training strategies, emphasizing their alignment with green computing principles. In particular, it highlights the utility of early stopping mechanisms in optimizing the training process of deep learning models. Early stopping works by monitoring performance metrics, such as validation accuracy or loss, and halting the training process once these metrics stabilize or show no improvement over a predefined number of epochs. This approach eliminates redundant computations, leading to significant reductions in energy consumption and computational costs while preserving model accuracy. The research is centered on transfer learning models, specifically MobileNetV2, InceptionV3, ResNet50V2, and Xception, which are well-regarded for their versatility and performance in image classification tasks. By systematically varying patience criteria, the study explores their impact on training duration, model accuracy, and computational efficiency. Each patience criteria determine how many epochs the training continues without improvement before stopping, allowing for a nuanced examination of its effects across different architectures. Additionally, the Rock Paper Scissors dataset, used for this study, is thoroughly described, including its structure, size, and pre-processing steps applied. The findings reveal that early stopping not only streamlines the training process but also aligns well with the broader goals of sustainable artificial intelligence development. Supported by statistical analyses, such as Kruskal-Wallis H and Conover-Iman tests, the results demonstrate that early stopping significantly reduces training time without compromising accuracy. By effectively balancing computational efficiency with performance optimization, this strategy exemplifies how environmentally responsible practices can be integrated into AI workflows. This study contributes valuable insights into how adopting such techniques can mitigate the environmental impact of AI model training, highlighting their importance in the context of advancing green computing initiatives.**

1. INTRODUCTION

The development and application of technology increasingly demand environmentally conscious approaches, particularly in response to the global challenges of sustainability and energy consumption [1],

E-mail address: abdulkadirtasdelen@aybu.edu.tr (*).
Key words and phrases. Artificial Intelligence, Green Computing, Green AI, Early Stop Strategy .

[2]. In this context, green computing emerges as a critical paradigm, aiming to minimize the environmental footprint of computational processes by enhancing energy efficiency and reducing operational costs. Its applications span diverse areas, including power management, server virtualization, data center optimization, and energy-efficient resource utilization, significantly influencing fields such as business, environmental management, and artificial intelligence (AI) [3], [4].

AI, while transformative across numerous industries, presents a considerable environmental challenge due to its high energy demands. Since the success of AlexNet in the 2012 ImageNet competition [5], the computational requirements for model training have grown exponentially, leading to substantial energy consumption [4], [6]. Addressing these challenges necessitates strategies that align AI development with sustainability goals. Recent advancements in AI training have highlighted the critical role of energy-efficient strategies. For instance, studies on large-scale language models such as GPT-3 have shown significant energy demands, emphasizing the necessity of integrating green computing principles into AI workflows [7]. Among these, early stopping—a technique that halts training once performance metrics stabilize—has shown promise. Early stopping is a regularization technique used to terminate training when performance on a validation set ceases to improve. This prevents over-fitting by halting before the model memorizes training data. By curbing unnecessary computations, early stopping not only enhances computational efficiency but also embodies the principles of green computing, offering a pathway toward sustainable AI practices [8].

Another critical aspect of green computing is the efficient management of energy distribution and usage, as exemplified by smart grid technologies. These systems manage renewable energy sources more effectively, boosting energy efficiency while reducing greenhouse gas emissions. However, ensuring reductions in energy costs and CO_2 emissions remains a priority. Research in commercial and institutional buildings demonstrates that human intervention, supported by energy-saving techniques and information systems, can significantly minimize energy losses. Similarly, sorting tasks based on time and power requirements exemplifies strategies for reducing power consumption during decision-making processes [6], [9], [10].

In AI, early stopping stands out as a pivotal approach to mitigating the environmental impact of training processes. This method halts the training process when performance metrics, such as validation accuracy, plateau or decline. By reducing computational demands, early stopping directly addresses the sustainability challenges of AI, balancing computational efficiency with model performance [11], [12]. Such optimization techniques illustrate how green computing principles can be effectively integrated into AI workflows, fostering a future where technological progress aligns with environmental responsibility.

The increasing demand for energy-efficient solutions in AI underscores the need for interdisciplinary approaches that prioritize sustainability without sacrificing performance. As global concerns about climate change and resource scarcity grow, embedding green computing principles into emerging technologies becomes imperative. Innovations like energy-efficient consensus mechanisms, smart grids, and energy-aware training strategies enable technological advancements to align with the objectives of sustainable development, ensuring that progress benefits both society and the environment [4], [6], [12], [13].

This study specifically investigates the application of early stopping mechanisms in the training of deep learning models, focusing on their energy efficiency and impact on model performance. Transfer learning architectures—MobileNetV2, InceptionV3, ResNet50V2, and Xception—were selected for their versatility and strong performance in image classification tasks. These architectures are systematically evaluated by varying patience criteria (PC), which define thresholds for halting training in the absence of performance improvements, facilitating an analysis of training duration, model accuracy, and computational efficiency [11].

The Rock Paper Scissors dataset [14] serves as the basis for evaluation, with detailed descriptions of its structure, size, and pre-processing steps ensuring clarity and reproducibility. Statistical analyses, including Kruskal-Wallis H [15], Mann-Whitney U [16], and Conover-Iman [17] tests, validate the results rigorously.

The novelty of this work lies in its comparative analysis of multiple deep learning architectures, underpinned by statistical rigor. By employing Kruskal-Wallis H and Conover-Iman tests, the study ensures the reliability and generalizability of its findings. This approach establishes a benchmark for future research on energy-aware AI methodologies.

Furthermore, this research underscores the interdisciplinary nature of integrating green computing principles into AI model training. By fostering collaboration across fields such as computer science, environmental science, and data engineering, it paves the way for innovations that prioritize sustainability without compromising technological advancements.

2. Background

The rising demand for machine learning (ML)-enabled systems has significantly increased energy consumption across various computational tasks. As ML applications proliferate, their environmental impact becomes a growing concern. To address this, researchers and practitioners have emphasized green computing practices, which focus on minimizing energy usage while maintaining model performance [13].

Xu *et al.* [12], empirically evaluated the impact of experimental design on the energy efficiency of the training process by analyzing three different convolutional neural network (CNN) architectures across two large image classification datasets. The training sessions were assessed using three efficiency metrics: CO_2 emissions produced, total energy consumed, and the number of floating-point operations (FLOPs) required. Statistical evidence revealed that carbon emissions and energy consumption are closely linked to the experimental design of neural network architectures. Furthermore, external factors, such as the geographical location of cloud-hosted services, also influence the computational impact, highlighting challenges beyond the researcher's control. These findings emphasize the importance of incorporating energy-efficient strategies into deep learning research to ensure that advancements in model performance align with sustainable computing practices.

A catalog of green architectural tactics for ML-enabled systems highlights a structured approach to energy efficiency. These tactics span multiple dimensions, including data-centric methods, algorithm design, model optimization, model training, deployment, and management. Among these, energy-aware

training strategies, such as quantization-aware training, leveraging checkpoints, and designing for memory constraints, are pivotal in reducing computational overhead during the training phase [8], [13].

Recent studies emphasize the importance of addressing over-parameterization in pre-trained CNN models to enhance energy efficiency and computational performance. For example, a systematic analysis of 27 pre-trained models identified EfficientNetB0 as the most energy-efficient candidate for Eimeria parasite detection, reducing parameter counts by up to 8% through pruning without sacrificing classification accuracy. This approach not only saves energy but also demonstrates the potential for holistic model design, combining multiple species into a single model for improved efficiency [18].

In this context, early stopping emerges as a key energy-aware training method that halts model training once a predefined performance threshold is met. By preventing unnecessary computations, this approach not only conserves energy but also accelerates the development cycle of ML models [13]. While early stopping has been widely studied in the context of energy-aware machine learning, its comparative effects across diverse transfer learning architectures remain underexplored. This study addresses this gap by systematically evaluating multiple architectures using a standardized dataset and statistical rigor.

Early exit strategies in deep learning have emerged as an effective approach to balancing model performance and computational efficiency. These strategies allow intermediate predictions within neural networks, enabling the termination of computations for certain inputs when a confident prediction is achieved. This approach has been extensively studied across various domains and tasks, including image classification, machine translation, text ranking, and quality enhancement [11], [19].

Recent studies on early existing, as summarized in [20], highlight the diversity of applications and metrics employed to evaluate these strategies. For instance, Teerapittayanon *et al.* [21], [22] applied early exit strategies to image classification tasks using datasets such as MNIST and CIFAR-10, with base models like LeNet-5, AlexNet, and ResNet, focusing on metrics like accuracy and latency. Similarly, Wang *et al.* [23] and Li *et al.* [24] investigated the efficiency of early exits in large-scale datasets like ImageNet, employing models such as ResNet, DenseNet, and MSDNet. Notably, energy consumption was explicitly considered in several works, such as those by Laskaridis *et al.* [25] and Wang *et al.* [23], underscoring the role of early exits in energy-efficient computing.

By reducing latency, computational complexity, and energy consumption, early exit strategies align with green computing principles, offering a pathway to sustainable deep learning practices. Integrating these strategies into model design and training processes could significantly reduce the environmental footprint of machine learning applications. Future research in this area is expected to expand the scope of early exits to additional tasks and domains, further advancing the synergy between performance optimization and energy efficiency in deep learning [11].

In line with the objectives outlined in the introduction, this study underscores the necessity of interdisciplinary efforts to integrate green computing principles into the life-cycle of emerging technologies. By prioritizing energy efficiency and sustainability, we pave the way for innovations that are not only technically advanced but also environmentally responsible. A key focus of this research is the application of early stopping strategies within the context of deep learning, specifically exploring how these strategies influence model performance across various deep learning architectures. By investigating different PC,

this study provides a comparative analysis that highlights the impact of early stopping on both training time and model accuracy.

The primary goal of this study is to explore how early stopping strategies can affect the training process of deep learning models, offering a comprehensive analysis of different model architectures and their performance under varying parameters. While this study does not focus on directly measuring energy consumption or computational resources, it does provide valuable insights into how early stopping can influence training time and model performance. By halting training when the performance metrics, such as validation accuracy or loss, no longer show significant improvement, early stopping reduces unnecessary epochs, which in turn decreases training time.

Validation loss measures the model's error on the validation dataset after each training epoch. It serves as a key indicator of over-fitting, where a rising validation loss suggests that the model is no longer generalizing well to unseen data. Similarly, validation accuracy measures the proportion of correctly predicted instances in the validation dataset after each training epoch, providing insight into the model's generalization performance. A stagnating or declining validation accuracy, despite improvements in training accuracy, may suggest over-fitting.

This research examines the relationship between the number of epochs before stopping, the final test accuracy, and the time taken for training. By analyzing different PC (such as 3, 5, 7, 10, and 15 epochs), we determine the optimal stopping point for achieving a good balance between model performance and training duration. The comparative analysis focuses on how these parameters impact the test accuracy and the training time, ultimately showing that early stopping can be a practical technique to improve the efficiency of deep learning processes.

Ultimately, this study aims to demonstrate how early stopping can optimize deep learning workflows by reducing unnecessary training time, allowing researchers and practitioners to achieve efficient models without over-fitting. By offering a framework for understanding how different PC influence model accuracy and training duration, this research supports the integration of efficiency-oriented strategies into machine learning practices, contributing to more sustainable development cycles.

3. Materials and Methods

3.1. **Transfer Learning Methods:**

To investigate the impact of early stopping strategies on different model architectures, we utilize transfer learning with four well-established transfer learning models [26]: MobileNetV2, InceptionV3, ResNet50V2, and Xception. These pre-trained models are fine-tuned on the Rock Paper Scissors Dataset [14]. Transfer learning is particularly beneficial for tasks with limited labeled data, as the models have already learned relevant features from large-scale datasets such as ImageNet. The fine-tuning process adapts the pre-trained models to the specific classification task, thus accelerating the training process and potentially enhancing model performance. Each of these models has been selected due to its proven effectiveness in image classification tasks, providing a diverse range of model architectures to assess the generalizability and performance of early stopping strategies.

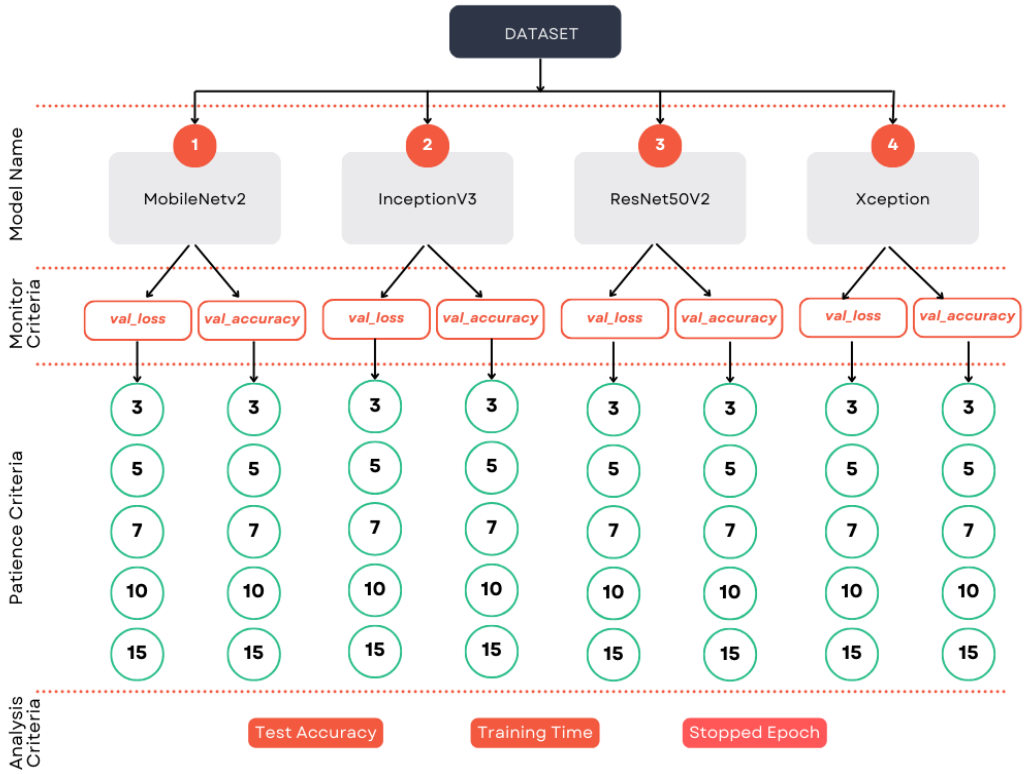


FIGURE 1. **Early stopping parameter configuration and monitoring across transfer learning models.**

Figure 1 illustrates the experimental setup for evaluating the impact of different PC on four transfer learning models: MobileNetV2, InceptionV3, ResNet50V2, and Xception. The monitored criteria, including validation loss and validation accuracy, are analyzed to determine the effects of early stopping strategies on key performance metrics: test accuracy, training time, and stopped epoch. The visual framework emphasizes the systematic evaluation process to identify optimal PC settings for efficient and effective model training.

3.2. **Dataset:**

The Rock Paper Scissors Dataset [14] is used for this study and consists of labeled images categorized into different types. Each category contains 975 images, resulting in a total of 2,925 images. The dataset is divided into two main subsets: 80% (2,340 images) for the training dataset (70% for training and 10% for validation), and 20% (585 images) for the test dataset (Figure 2). This division ensures that the model is trained on a substantial portion of the data, while also leaving a portion for validation and testing to evaluate its performance on unseen data. Moreover, to account for potential variability and to ensure that

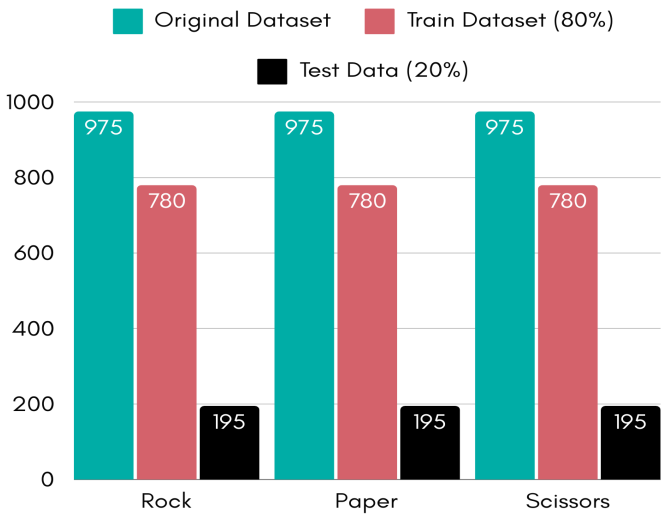


FIGURE 2. **The distribution of each class.**

the models are assessed fairly, 5-fold cross-validation (CV) is employed. CV is a statistical method used to evaluate a model's generalization performance by splitting the dataset into complementary subsets for training and testing. This ensures that the evaluation is robust and not overly dependent on a single split. This method ensures that each fold of the CV process maintains a proportional distribution of the classes, providing a more reliable estimate of the model's generalization ability. In each fold, a different partition is used for training, validation, and testing, and the process is repeated for each fold, allowing the model to be trained and tested on all available data [27], [28].

The dataset serves as the basis for evaluation, with detailed descriptions of its structure, size, and pre-processing steps ensuring clarity and reproducibility. In selecting the dataset, this study aims to analyze model performance on a medium-scale dataset that allows for efficient experimentation with various early stopping strategies. The dataset is well-suited for evaluating energy-efficient training techniques, as it provides a manageable size for computational experiments while maintaining enough complexity to demonstrate key differences in performance. This choice also facilitates reproducibility and comparability with other studies in the field. Future research will consider larger and more complex datasets, enabling the examination of early stopping strategies on a broader range of tasks and more demanding computational settings.

Figure 3 displays samples from different classes of the dataset, which represent the classic game of Rock, Paper, or Scissors. Each class corresponds to one of the three possible moves in the game: Rock, Paper, or Scissors. These samples are crucial for training machine learning models, as they help the

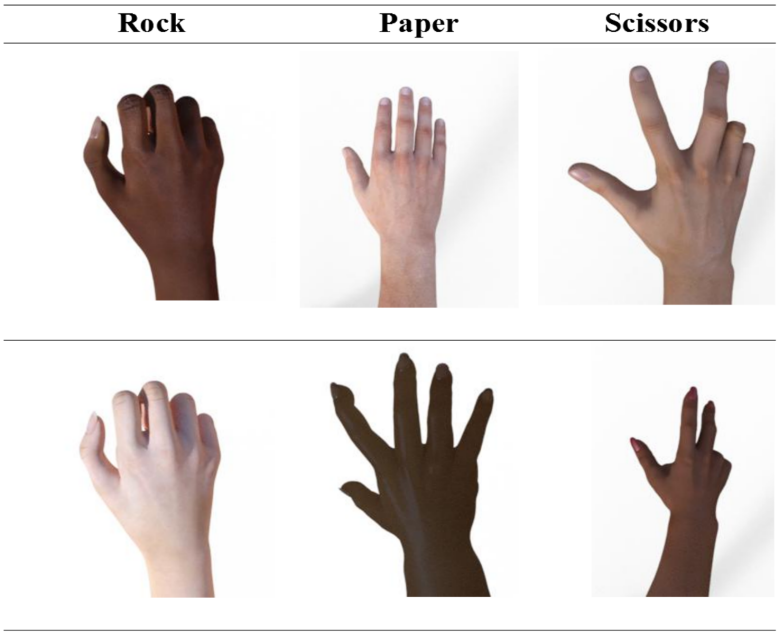


FIGURE 3. **Samples from different classes of the dataset.**

algorithm distinguish between the various input categories. The dataset likely includes images or representations of each move, enabling the model to learn patterns and make predictions based on the visual features associated with Rock, Paper, or Scissors.

3.3. **Statistical Analysis:**

The performance of early stopping strategies is evaluated using several metrics, including classification accuracy, training time, and validation loss. To determine whether early stopping strategies lead to statistically significant differences in performance, a Kruskal-Wallis H test [15] is employed. This non-parametric test is appropriate for comparing multiple independent groups—such as different PC—across performance metrics, as it does not assume the data to be normally distributed. If the Kruskal-Wallis H test yields a significant result, indicating differences in performance across groups, a post-hoc Conover-Iman Test [17] is conducted. This test is particularly suitable when sample sizes within groups are small, providing a robust method for pairwise comparisons while controlling for the family-wise error rate. Additionally, for pairwise comparisons between groups of PC, the Mann-Whitney U test [16] is utilized. This test is a non-parametric alternative to the t-test and is applied when comparing two independent groups, such as specific pairs. It assesses whether the distributions of the two groups differ significantly. The Mann-Whitney U test provides a complementary approach to the Conover-Iman Test by offering

more granular insights into pairwise differences [17]. All statistical tests are conducted with a significance level of and $a = 0.05$. If the p-value from any test is less than 0.05, the null hypothesis is rejected, indicating statistically significant differences between the groups being compared. This rigorous statistical approach ensures that the conclusions drawn from the analysis are reliable and valid. By evaluating the impact of early stopping strategies on model performance through Kruskal-Wallis H, Conover-Iman, and Mann-Whitney U tests, this study aims to identify optimal training configurations that balance computational efficiency with model accuracy.

3.4. **Common Specifications:**

The common specifications for the model are outlined in Table 1. CV is employed with a fold size of 5 to ensure balanced class representation. Data shuffling is enabled with a random state set to 1 for reproducibility. The base model is configured with an input shape of 96x96, utilizes pre-trained weights from the ImageNet dataset. and is set as non-trainable. The model architecture includes a dense layer with 64 units, using the Rectified Linear Unit (ReLU) activation function [29] followed by a dropout layer with a rate of 0.05 to mitigate over-fitting and improve generalization. The Adam optimizer is used during compilation, with a learning rate of 0.0001. Training is conducted using a batch size of 32 and spans a maximum of 50 epochs.

TABLE 1. **Common specifications for each transfer learning models**

Category	**Name**	**Value**
Monitor Criteria	Validation Loss Metric	val_loss
	Validation Accuracy Metric	val_accuracy
Patience Criteria	Stopped Epochs	3, 5, 7, 10, 15
CV	Number of Folds	5
	Shuffle	True
	Random Seed	1
Base Model	Input Shape	96*96
	Weights	ImageNet
	Trainable	False
Dense Layer	Units	64
	Activation Function	ReLU
Dropout Layer	Dropout Rate	0.50
Compile	Optimizer	Adam
	Learning Rate	0.0001
Model Training	Batch Size	32
	Number of Epochs	50

To improve efficiency and avoid over-fitting, early stopping is applied based on the monitoring criteria (MC). Specifically, the training process monitors validation loss and validation accuracy, stopping automatically if no improvement is observed across consecutive epochs. The evaluation is performed at specific checkpoints (epochs 3. 5. 7. 10. and 15), offering insights into model performance at different stages of training. These configurations aim to balance computational efficiency, model generalization, and robustness. ensuring reliable results across all folds of the CV process.

4. RESULTS

The performance of four deep learning models—MobileNetV2, InceptionV3, ResNet50V2, and Xception—was evaluated based on several key metrics, including training accuracy, training loss, validation accuracy, validation loss, test accuracy, test loss, and training time. These metrics were averaged over various early stop criteria, such as *val_loss* and *val_accuracy*, with results reported for different PC. Table 2 summarizes the average performance across these models for each validation criterion.

In general, MobileNetV2 consistently demonstrated high training and test accuracies with relatively low training and test losses. Similarly, InceptionV3 and ResNet50V2 achieved competitive results, although with some variations in performance depending on the validation criteria. Xception, on the other hand, showed a robust performance, especially in terms of validation accuracy and test accuracy, though with higher training and test losses in comparison to the other models. The average training times for all models were also considered, providing an indication of computational efficiency.

The analysis of model performance (Table 3), grouped by MC and PC, revealed distinct patterns in the test accuracy, training time, and early stopping behavior for each architecture.

For MobileNetv2, the test accuracy did not show a statistically significant difference ($p=0.382$, $p>0.05$), indicating consistent performance across groups. However, both training time ($p=0.000$, $p<0.05$) and the stopped epoch count ($p=0.000$, $p<0.05$) exhibited significant variations, suggesting the model's sensitivity to these parameters.

Similarly, InceptionV3 displayed no significant difference in test accuracy ($p=0.403$, $p>0.05$), while training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) were significantly impacted, reflecting comparable trends to MobileNetv2.

For ResNet50V2, test accuracy remained consistent ($p=0.269$, $p<0.05$) without notable variation. However, as observed with the other models, both training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) varied significantly, reinforcing the importance of these parameters in model training dynamics.

In contrast, Xception demonstrated a significant difference in test accuracy ($p=0.030$, $p<0.05$). indicating variability in performance across groups. Additionally, training time ($p=0.000$, $p<0.05$) and stopped epoch count ($p=0.000$, $p<0.05$) also showed significant variation, further emphasizing the model's sensitivity to MC and PC.

The Kruskal-Wallis H test results, summarized in Table 4, show the comparison of different monitor parameters across various deep learning architectures. For most models, including MobileNetV2, InceptionV3, and ResNet50V2, the test results for test accuracy did not show significant differences ($p > 0.05$),

TABLE 2. **Average performance metrics for MobileNetV2, InceptionV3, ResNet50V2, and Xception models**

MN	MC	PC	SE	TE	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	TT (s)
MobileNetV2	val_loss	3	9.4	50	0.995	0.028	0.841	0.414	0.984	0.056	25.055
	val_loss	5	13.4	50	0.998	0.017	0.848	0.414	0.983	0.051	33.112
	val_loss	7	18	50	0.999	0.011	0.870	0.390	0.986	0.046	42.814
	val_loss	10	19	50	0.999	0.009	0.863	0.429	0.987	0.049	48.824
	val_loss	15	25	50	0.999	0.007	0.827	0.518	0.981	0.061	61.902
	val_acc	3	8.8	50	0.994	0.034	0.831	0.419	0.979	0.068	22.719
	val_acc	5	11.6	50	0.997	0.021	0.857	0.390	0.986	0.054	32.113
	val_acc	7	15	50	0.997	0.017	0.863	0.421	0.985	0.064	39.614
	val_acc	10	18.4	50	0.998	0.011	0.829	0.461	0.980	0.062	48.845
	val_acc	15	25.2	50	0.999	0.006	0.859	0.440	0.986	0.063	75.320
	Original	**No**	**No**	**50**	**1**	**0.002**	**0.840**	**0.580**	**0.978**	**0.070**	**121.674**
InceptionV3	val_loss	3	8.4	50	0.986	0.057	0.758	0.743	0.965	0.125	33.622
	val_loss	5	17	50	0.996	0.023	0.798	0.720	0.976	0.093	70.758
	val_loss	7	17.2	50	0.994	0.027	0.814	0.683	0.972	0.117	82.924
	val_loss	10	30.6	50	0.998	0.010	0.825	0.678	0.982	0.077	182.771
	val_loss	15	30.6	50	0.998	0.009	0.823	0.726	0.981	0.077	181.658
	val_acc	3	10.8	50	0.988	0.049	0.814	0.659	0.969	0.128	68.307
	val_acc	5	19.4	50	0.996	0.017	0.799	0.767	0.979	0.086	125.599
	val_acc	7	27.8	50	0.999	0.008	0.816	0.735	0.979	0.077	191.087
	val_acc	10	31.8	50	0.999	0.008	0.857	0.595	0.983	0.075	223.856
	val_acc	15	36.2	50	0.999	0.005	0.847	0.695	0.982	0.075	256.737
	Original	**No**	**No**	**50**	**0.999**	**0.004**	**0.835**	**0.772**	**0.979**	**0.0991**	**416.361**
ResNet50V2	val_loss	3	17.8	50	0.995	0.025	0.958	0.121	0.993	0.026	139.693
	val_loss	5	16	50	0.993	0.032	0.918	0.194	0.991	0.044	127.430
	val_loss	7	26.6	50	0.997	0.018	0.923	0.166	0.993	0.026	197.787
	val_loss	10	26.2	50	0.997	0.016	0.891	0.232	0.987	0.040	201.758
	val_loss	15	41.6	50	0.998	0.008	0.934	0.133	0.996	0.019	364.172
	val_acc	3	11.6	50	0.985	0.053	0.939	0.1401	0.983	0.056	151.258
	val_acc	5	16.2	50	0.993	0.031	0.939	0.132	0.990	0.037	176.910
	val_acc	7	19.2	50	0.994	0.028	0.941	0.137	0.991	0.040	238.396
	val_acc	10	23.4	50	0.997	0.018	0.955	0.106	0.991	0.036	303.144
	val_acc	15	36.4	50	0.997	0.009	0.919	0.166	0.992	0.025	520.313
	Original	**No**	**No**	**50**	**0.999**	**0.004**	**0.911**	**0.209**	**0.985**	**0.033**	**723.026**
Xception	val_loss	3	9.4	50	0.975	0.116	0.691	0.653	0.942	0.184	71.368
	val_loss	5	15.4	50	0.990	0.065	0.719	0.660	0.958	0.138	118.122
	val_loss	7	16.4	50	0.990	0.068	0.704	0.663	0.951	0.150	125.397
	val_loss	10	22.8	50	0.995	0.042	0.719	0.699	0.958	0.137	175.587
	val_loss	15	28.8	50	0.998	0.025	0.741	0.702	0.965	0.113	233.605
	val_acc	3	10	50	0.974	0.125	0.701	0.648	0.940	0.218	95.685
	val_acc	5	16	50	0.988	0.066	0.716	0.664	0.956	0.140	129.949
	val_acc	7	21.6	50	0.992	0.052	0.698	0.757	0.956	0.159	171.013
	val_acc	10	26.8	50	0.997	0.029	0.722	0.752	0.963	0.116	219.265
	val_acc	15	37.4	50	0.998	0.019	0.728	0.823	0.966	0.105	313.890
	Original	**No**	**No**	**50**	**0.999**	**0.010**	**0.732**	**0.874**	**0.965**	**0.120**	**429.153**

*MN: Transfer Learning Model Name. MC: Monitor Criteria. PC: Patience Criteria. SE: Stopped epoch. TE: Total Number of Epochs. TT: Training Times in Seconds. Note: All results except TE are mean of each fold. See **Appendix A** for details.

TABLE 3. **Summary of each model by groups**

#	Group Name	Test Statistic	*p*-value	Result
	Test Accuracy	10.693	0.382	+
MobileNetV2	Training Time	42.263	0.000	**x**
	Stopped Epoch	40.643	0.000	**x**
	Test Accuracy	10.434	0.403	+
InceptionV3	Training Time	42.982	0.000	**x**
	Stopped Epoch	43.074	0.000	**x**
	Test Accuracy	12.251	0.269	+
ResNet50V2	Training Time	43.426	0.000	**x**
	Stopped Epoch	41.092	0.000	**x**
	Test Accuracy	19.895	0.030	**x**
Xception	Training Time	41.517	0.000	**x**
	Stopped Epoch	41.719	0.000	**x**

Note: "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

meaning the null hypothesis (H_0) was accepted. However, for training time and the number of stopped epochs, significant differences were observed ($p < 0.05$), leading to the rejection of the null hypothesis (H_0) for those parameters, indicating that early stopping significantly affected these factors. Only Xception model's test accuracy did show a significant result ($p < 0.05$), while other models exhibited no significant difference in test accuracy. Overall, the results suggest that early stopping strategies had a noticeable impact on training time and number of epochs, but did not always affect the test accuracy in a significant way.

The Mann-Whitney U Test was conducted to evaluate the pairwise differences in performance metrics across four transfer learning architectures: MobileNetV2, ResNet50V2, InceptionV3, and Xception. The metrics analyzed included test accuracy, training time, and stopped epoch under various early stopping PC (Table 5).

The test accuracy results across all comparisons yielded $p > 0.05$, indicating that the null hypothesis (H_0) could not be rejected. This signifies that there were no statistically significant differences in test accuracy between the models across the considered PC. The early stopping strategies employed do not significantly affect the accuracy of the models. This consistency suggests that the transfer learning architectures are robust to changes in PC with respect to test accuracy.

For most pairwise comparisons, the $p > 0.05$, indicating no significant differences in training time across architectures and PC. At a PC of 7, the comparison involving MobileNetV2 resulted in a p-value = 0.008, which is statistically significant ($p < 0.05$). This suggests that the training time for MobileNetV2 is significantly different compared to another model for this specific PC. While early stopping does not generally lead to significant differences in training time, the significant result for MobileNetV2 indicates that certain architectures may exhibit more sensitivity to specific PC in terms of computational efficiency.

TABLE 4. **Summary of Kruskal-Wallis Test results for two different MC**

		MC= *val_loss*			MC= *val_accuracy*		
#	**Group Name**	**Test Statistic**	***p*-value**	**Result**	**Test Statistic**	***p*-value**	**Result**
	Test Accuracy	5.515	0.356	+	7.881	0.163	+
MobileNetV2	Training Time	23.885	0.000	**x**	24.474	0.000	**x**
	Stopped Epoch	25.137	0.000	**x**	23.540	0.000	**x**
	Test Accuracy	6.147	0.292	+	4.715	0.452	+
InceptionV3	Training Time	22.925	0.000	**x**	24.025	0.000	**x**
	Stopped Epoch	23.655	0.000	**x**	24.159	0.000	**x**
	Test Accuracy	8.418	0.135	+	4.632	0.462	+
ResNet50V2	Training Time	24.561	0.000	**x**	23.622	0.000	**x**
	Stopped Epoch	23.698	0.000	**x**	22.883	0.000	**x**
	Test Accuracy	12.703	0.026	x	8.430	0.134	+
Xception	Training Time	25.552	0.000	**x**	22.533	0.000	**x**
	Stopped Epoch	25.493	0.000	**x**	22.943	0.000	**x**

Note: "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

For most pairwise comparisons of stopped epochs, the $p > 0.05$, meaning no significant differences were observed between the models. At a PC of 3, one comparison yielded a $p = 0.020$, which is statistically significant ($p < 0.05$). This indicates a significant difference in the number of epochs at which training is halted for smaller PC between the models. Early stopping with smaller PC may result in varying training dynamics across architectures, influencing the point at which training is terminated. The boxplot (Figure 4) illustrates the test accuracy of the Xception model across various MC and PC. Each group represents a specific monitoring criterion (e.g. validation loss or validation accuracy) evaluated at different epochs, as well as a no-monitoring baseline. The mean and median test accuracies are annotated for each group, providing insights into consistency and performance variations.

The results highlight that the *val_accuracy_15* and *no_monitoring_No* groups achieved the highest median and mean test accuracies (0.97), while *val_loss_3* and *val_accuracy_3* showed comparatively lower performance, with median values around 0.94. This indicates that MC and the choice of PC significantly influence the model's ability to generalize effectively.

Although the test accuracy results for the Xception model suggest that there is a statistically significant difference between at least one pair of monitoring groups ($p < 0.05$), further pairwise analysis using the Conover-Iman test reveals a nuanced interpretation (Table 6). Specifically, while some groups, such as *no_strategy* vs. *val_loss_3* ($p = 0.046$) and *val_accuracy_3* vs. *val_accuracy_15* ($p = 0.046$) show *p*-values below the significance threshold of 0.05, these values are very close to the threshold. Additionally, other comparisons, such as *val_loss_3* vs. *val_loss_5* ($p = 0.297$) and *val_accuracy_5* vs. *val_accuracy_7* (p=0.687), yield much higher *p*-values, indicating no statistically significant difference.

TABLE 5. **Summary of Mann-Whitney U Test results for pair comparison by monitoring strategy**

PC	Group Name	Test Statistic	*p*-value	Result	Test Statistic	*p*-value	Result
	Test Accuracy	8.5	0.462	+	6	0.206	+
3	Training Time	10	0.690	+	15	0.690	+
	Stopped Epoch	10.5	0.747	+	1	**0.020**	**x**
	Test Accuracy	16	0.523	+	11	0.829	+
5	Training Time	12	1.000	+	22	0.056	+
	Stopped Epoch	9	0.522	+	10.5	0.750	+
	Test Accuracy	13	1.000	+	8	0.395	+
7	Training Time	10	0.690	+	16	0.548	+
	Stopped Epoch	8.5	0.461	+	5	0.151	+
	Test Accuracy	4	0.090	+	10	0.671	+
10	Training Time	12	1.000	+	20	0.151	+
	Stopped Epoch	12.5	1.000	+	7.5	0.340	+
	Test Accuracy	15.5	0.600	+	5	0.134	+
15	Training Time	15	0.690	+	19	0.222	+
	Stopped Epoch	10	0.674	+	8.5	0.462	+
	Test Accuracy	16	0.530	+	10.5	0.753	+
3	Training Time	20	0.151	+	14	0.841	+
	Stopped Epoch	17	0.421	+	10	0.675	+
	Test Accuracy	13	1.000	+	16	0.530	+
5	Training Time	22	0.056	+	17	0.421	+
	Stopped Epoch	17	0.402	+	15.5	0.599	+
	Test Accuracy	14	0.832	+	17.5	0.344	+
7	Training Time	25	**0.008**	**x**	18	0.310	+
	Stopped Epoch	22	0.059	+	16.5	0.458	+
	Test Accuracy	12.5	1.000	+	16	0.530	+
10	Training Time	17	0.421	+	18	0.310	+
	Stopped Epoch	14	0.841	+	16.5	0.463	+
	Test Accuracy	15.5	0.599	+	14	0.834	+
15	Training Time	21	0.095	+	21	0.095	+
	Stopped Epoch	16	0.530	+	17.5	0.344	+

Note: • *MobileNetV2,* • *ResNet50V2,* • *InceptionV3,* • *Xception*, PC: Patience Criteria, "+": H_0 is accepted ($p > 0.05$), "x": H_0 is rejected ($p < 0.05$).

Given these findings, while some differences appear significant at first glance, the proximity of the *p*-values to the threshold in certain cases, along with the consistency across most comparisons, suggests that

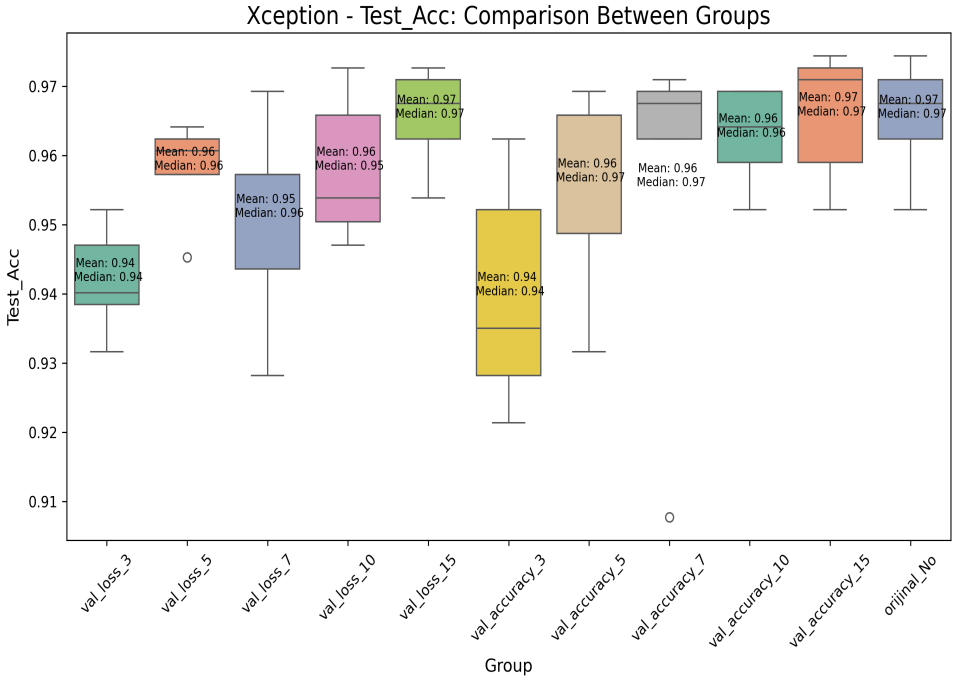


FIGURE 4. **Boxplot of test accuracy for different PC with Xception model.**

these results may not represent meaningful differences in practical terms. As such, the overall outcomes can be interpreted as lacking substantial evidence for significant performance differences between the monitoring strategies. This indicates that the choice of MC might not critically impact the Xception model's test accuracy under the conditions evaluated.

5. DISCUSSION

The results from the application of early stopping strategies to various transfer learning models reveal significant insights into balancing computational efficiency and model performance. In this section, we interpret the findings, their implications for green computing.

5.1. **Impact on Training Time and Computational Efficiency:**

Hussein *et al.* [11] highlighted the critical interplay between early stopping PC and the number of epochs in deep learning models, demonstrating that higher PC values often require more epochs to achieve optimal validation accuracy. Conversely, lower PC values can lead to premature stopping and suboptimal model performance. Importantly, they also noted that prolonged training times do not necessarily enhance validation accuracy, reinforcing the utility of early stopping in mitigating over-fitting and conserving computational resources. These findings underscore the necessity of carefully calibrating PC

TABLE 6. **Pairwise comparisons of monitoring strategies for the Xception model Using Conover-Iman Test**

Group Name	**no strategy**	**val_acc_3**	**val_accuracy_5**	**val_accuracy_7**	**val_accuracy_10**	**val_accuracy_15**	**val_loss_3**	**val_loss_5**	**val_loss_7**	**val_loss_10**	**val_loss_15**
no strategy (*original_no*)	1.000	**0.049**	0.425	0.687	0.690	0.935	**0.046**	0.383	0.236	0.425	0.991
val_accuracy_3	**0.049**	1.000	0.297	0.140	0.136	**0.046**	0.934	0.383	0.565	0.297	**0.049**
val_accuracy_5	0.425	0.297	1.000	0.687	0.687	0.383	0.236	0.934	0.687	0.991	0.425
val_accuracy_7	0.687	0.140	0.687	1.000	0.991	0.685	0.110	0.685	0.406	0.687	0.687
val_accuracy_10	0.690	0.136	0.687	0.991	1.000	0.687	0.110	0.653	0.383	0.687	0.687
val_accuracy_15	0.935	**0.046**	0.383	0.685	0.687	1.000	0.046	0.342	0.202	0.383	0.936
val_loss_3	**0.046**	0.934	0.236	0.110	0.110	**0.046**	1.000	0.297	0.434	0.236	**0.046**
val_loss_5	0.383	0.383	0.934	0.685	0.653	0.342	0.297	1.000	0.712	0.934	0.383
val_loss_7	0.236	0.565	0.687	0.406	0.383	0.202	0.434	0.712	1.000	0.687	0.236
val_loss_10	0.425	0.297	0.991	0.687	0.687	0.383	0.236	0.934	0.687	1.000	0.425
val_loss_15	0.991	**0.049**	0.425	0.687	0.687	0.936	0.046	0.383	0.236	0.425	1.000

to achieve a balance between computational efficiency and model performance. Building upon this, our results further confirm that early stopping consistently reduced training time across all four models (MobileNetV2, InceptionV3, ResNet50V2, and Xception), with statistically significant reductions observed in most cases. This demonstrates that early stopping effectively prevents unnecessary computations, contributing to energy conservation and aligning with the principles of green computing. However, the sensitivity of training time reduction to PC underscores the importance of carefully tuned parameters. For instance, shorter PCs drastically reduce epochs but may compromise the achievement of optimal weights for certain architectures, as evidenced by the significant test accuracy variations observed in Xception. These findings align with Chen's [8] assertion that "Algorithms aimed at early termination of training or dynamic adjustment of model complexity based on performance can prevent over-computation and conserve resources."

5.2. **Consistency of Model Accuracy:**

The findings underline that early stopping does not significantly degrade test accuracy for most models. Models like MobileNetV2 and ResNet50V2 exhibited stable performance across varying PCs, reaffirming the robustness of these architectures. However, Xception's test accuracy displayed significant

variability, indicating that the model's performance is more sensitive to the choice of PC and monitoring criteria. These results resonate with Patterson *et al.*'s perspective that prioritizing metrics beyond accuracy, such as training efficiency and carbon footprint, could foster innovations in ML algorithms, systems, and hardware, ultimately advancing both performance consistency and sustainability [7].

5.3. **Implications for Green AI:.**

The study supports the hypothesis that early stopping strategies contribute to reducing the environmental impact of AI training. By halting the training process earlier, models consume less energy, thereby reducing carbon footprints. This aligns with sustainable development goals and emphasizes the role of energy-efficient practices in AI research.

5.4. **Limitations and Challenges:**

While the findings are promising, several challenges remain:

Dataset Dependency: The experiments were conducted on the Rock Paper Scissors dataset. The generalizability of these findings to more complex datasets with higher feature dimensionality requires further exploration.

Model Sensitivity: The variability in performance for Xception suggests that some architectures might require additional adaptive mechanisms to ensure consistent accuracy while leveraging early stopping.

Energy Measurement: Although training time reductions imply energy savings, the study does not provide direct measurements of energy consumption, limiting the precision of conclusions about environmental impact.

6. Conclusion and Future Studies

The results of this study underscore the effectiveness of early stopping as a practical approach to enhancing the energy efficiency of deep learning model training. By preventing unnecessary epochs, early stopping significantly reduces training time and computational resources. The comparative analysis across multiple transfer learning architectures demonstrated consistent performance in accuracy, with substantial gains in energy savings. These outcomes highlight the role of energy-aware strategies in addressing the environmental challenges posed by the growing computational demands of artificial intelligence. Early stopping represents a straightforward yet impactful optimization that aligns well with global sustainability objectives, such as the United Nations' Sustainable Development Goals. This research provides a foundation for integrating such strategies into standard deep learning practices, paving the way for environmentally responsible AI development. To further advance the insights from this study, future research could focus on the following areas:

Broader Dataset Evaluation: Expand experiments to include larger and more diverse datasets, ensuring the generalizability of early stopping strategies across various domains and data types.

Direct Energy Measurement: Develop and employ methodologies to directly measure energy consumption during training, enabling a more precise assessment of energy savings.

Advanced Early Stopping Criteria: Investigate more sophisticated stopping mechanisms, such as adaptive PC thresholds and hybrid criteria combining multiple performance metrics.

Renewable Energy Integration: Evaluate the performance and energy efficiency of early stopping in data centers powered by renewable energy sources, providing insights into its sustainability impact under different energy scenarios.

Interdisciplinary Approaches: Collaborate across fields to incorporate principles of green computing into AI research, exploring synergies with smart grid technologies and sustainable data center designs.

Optimization for Real-Time Systems: Explore early stopping strategies in real-time AI applications, such as edge computing and Internet of Things (IoT) systems, where energy constraints are critical.

By addressing these areas, future studies could expand the application of energy-aware training strategies and enhance their effectiveness in promoting sustainable computing practices. This will not only benefit the AI community but also contribute to broader efforts toward reducing the environmental impact of advanced technologies.

DECLARATIONS

- **Contribution Rate Statement:** Abdulkadir TAŞDELEN has conducted this study as a single author.
- **Conflict of Interest:** The author has no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.
- **Data Availability:** The dataset has been properly cited, and a comprehensive analysis is included in the Appendices.
- **Statement of Support and Acknowledgment:** None.

REFERENCES

[1] K. I. Ibekwe, A. A. Umoh, Z. Q. S. Nwokediegwu, E. A. Etukudoh, V. I. Ilojianya, A. Adefemi, Energy efficiency in industrial sectors: A review of technologies and policy measures, Engineering Science & Technology Journal 5 (1) (2024) 169–184. doi:10.51594/estj.v5i1.742.

[2] A. Tasdelen, M. H. Habaebi, M. R. Islam, Exploring blockchain technologies: Insights into consensus mechanisms, mining pool dynamics, and energy consumption patterns, in: 2024 9th International Conference on Mechatronics Engineering (ICOM), IEEE, 2024, p. 95–100. doi:10.1109/icom61675.2024.10652588.

[3] V. Bolón-Canedo, L. Morán-Fernández, B. Cancela, A. Alonso-Betanzos, A review of green artificial intelligence: Towards a more sustainable future, Neurocomputing 599 (2024) 128096. doi:10.1016/j.neucom.2024.128096.

[4] Z. Vale, L. Gomes, D. Ramos, P. Faria, Green computing: a realistic evaluation of energy consumption for building load forecasting computation, Journal of Smart Environments and Green Computing 2 (2) (2022) 34–45. doi:10.20517/jsegc.2022.06.

[5] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. V. Esesn, A. A. S. Awwal, V. K. Asari, The history began from alexnet: A comprehensive survey on deep learning approaches (2018). arXiv:1803.01164. URL https://arxiv.org/abs/1803.01164

[6] Y. Zhou, X. Lin, X. Zhang, M. Wang, G. Jiang, H. Lu, Y. Wu, K. Zhang, Z. Yang, K. Wang, Y. Sui, F. Jia, Z. Tang, Y. Zhao, H. Zhang, T. Yang, W. Chen, Y. Mao, Y. Li, D. Bao, Y. Li, H. Liao, T. Liu, J. Liu, J. Guo, X. Zhao, Y. WEI, H. Qian, Q. Liu, X. Wang, W. Kin, Chan, C. Li, Y. Li, S. Yang, J. Yan, C. Mou, S. Han, W. Jin, G. Zhang, X. Zeng, On

the opportunities of green computing: A survey (2023). arXiv:2311.00447.
URL https://arxiv.org/abs/2311.00447

[7] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, J. Dean, Carbon emissions and large neural network training (2021). arXiv:2104.10350.
URL https://arxiv.org/abs/2104.10350

[8] X. Chen, Optimization strategies for reducing energy consumption in ai model training, ACS 6 (1) (Mar. 2023).

[9] L. Lannelongue, J. Grealey, M. Inouye, Green algorithms: Quantifying the carbon footprint of computation, Advanced Science 8 (12) (May 2021). doi:10.1002/advs.202100707.

[10] S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, Y. Zou, Green ai: do deep learning frameworks have different costs?, in: Proceedings of the 44th International Conference on Software Engineering, ICSE '22, ACM, 2022, p. 1082–1094. doi:10.1145/3510003.3510221.

[11] B. M. Hussein, S. M. Shareef, An empirical study on the correlation between early stopping patience and epochs in deep learning, ITM Web of Conferences 64 (2024) 01003. doi:10.1051/itmconf/20246401003.

[12] Y. Xu, S. Martínez-Fernández, M. Martinez, X. Franch, Energy efficiency of training neural network architectures: An empirical study (Feb. 2023). arXiv:2302.00967.

[13] H. Järvenpää, P. Lago, J. Bogner, G. Lewis, H. Muccini, I. Ozkaya, A synthesis of green architectural tactics for ml-enabled systems, in: Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS'24, ACM, 2024, p. 130–141. doi:10.1145/3639475.3640111.

[14] L. Maroney, Rock paper scissors classification dataset, https://laurencemoroney.com/datasets.html, accessed: 2024-10-17.

[15] W. H. Kruskal, W. A. Wallis, Use of ranks in one-criterion variance analysis, Journal of the American Statistical Association 47 (260) (1952) 583–621. doi:10.1080/01621459.1952.10483441.

[16] H. B. Mann, D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1) (1947) 50–60.

[17] W. Conover, R. Iman, Multiple-comparisons procedures. Informal report, 1979. doi:10.2172/6057803.

[18] S. S. Acmali, Y. Ortakci, H. Seker, Green ai-driven concept for the development of cost-effective and energy-efficient deep learning method: Application in the detection ofeimeriaparasites as a case study, Advanced Intelligent Systems 6 (7) (Jun. 2024). doi:10.1002/aisy.202300644.

[19] D. Reguero, S. Martínez-Fernández, R. Verdecchia, Energy-efficient neural network training through runtime layer freezing, model quantization, and early stopping, Computer Standards & Interfaces 92 (2025) 103906. doi:10.1016/j.csi.2024.103906.

[20] Y. Matsubara, M. Levorato, F. Restuccia, Split computing and early exiting for deep learning applications: Survey and research challenges, ACM Computing Surveys 55 (5) (2022) 1–30. doi:10.1145/3527155.

[21] S. Teerapittayanon, B. McDanel, H. Kung, Branchynet: Fast inference via early exiting from deep neural networks, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, p. 2464–2469. doi:10.1109/icpr.2016.7900006.

[22] S. Teerapittayanon, B. McDanel, H. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, p. 328–339. doi:10.1109/icdcs.2017.226.

[23] Y. Wang, J. Shen, T.-K. Hu, P. Xu, T. Nguyen, R. Baraniuk, Z. Wang, Y. Lin, Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference, IEEE Journal of Selected Topics in Signal Processing 14 (4) (2020) 623–633. doi:10.1109/jstsp.2020.2979669.

[24] H. Li, H. Zhang, X. Qi, R. Yang, G. Huang, Improved techniques for training adaptive deep networks (2019).
URL https://arxiv.org/abs/1908.06294

[25] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, N. D. Lane, Spinn: synergistic progressive inference of neural networks over device and cloud, in: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, MobiCom '20, ACM, 2020, p. 1–15. doi:10.1145/3372224.3419194.
[26] Keras applications, https://keras.io/api/applications/, accessed: 2024-10-15.
[27] T. Fontanari, T. C. Fróes, M. Recamonde-Mendoza, Cross-validation Strategies for Balanced and Imbalanced Datasets, Springer International Publishing, 2022, p. 626–640. doi:10.1007/978-3-031-21686-2_43.
[28] D. Berrar, Cross-Validation, Elsevier, 2019, p. 542–545. doi:10.1016/b978-0-12-809633-8.20349-x.
[29] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: G. Gordon, D. Dunson, M. Dudík (Eds.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Vol. 15 of Proceedings of Machine Learning Research, PMLR, Fort Lauderdale, FL, USA, 2011, pp. 315–323.
URL https://proceedings.mlr.press/v15/glorot11a.html

APPENDIX (A)

TABLE A.1. Detailed results for MobileNetV2 with various MC and PC (Appendix A.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	10	0.998046	0.02271	0.716724	0.61063	0.97265	0.07408	24.89913
3	loss	2	8	0.989741	0.039663	0.897611	0.407444	0.991453	0.046021	21.42021
3	loss	3	10	0.998046	0.02114	0.829352	0.383397	0.977778	0.058188	31.18301
3	loss	4	12	0.996092	0.0236	0.863481	0.39339	0.984615	0.052646	31.40667
3	loss	5	7	0.994138	0.034424	0.897611	0.27528	0.991453	0.046647	16.36674
5	loss	1	12	0.994626	0.022884	0.836177	0.407061	0.977778	0.061897	40.35201
5	loss	2	13	0.998046	0.017082	0.8157	0.505994	0.991453	0.037977	31.8802
5	loss	3	13	0.998534	0.017046	0.805461	0.482879	0.97265	0.070374	27.50178
5	loss	4	12	0.999023	0.018486	0.887372	0.335181	0.981197	0.054642	30.86905
5	loss	5	17	1	0.011072	0.894198	0.338853	0.989744	0.031267	34.95643
7	loss	1	20	0.999023	0.008709	0.829352	0.448269	0.982906	0.050633	48.03739
7	loss	2	13	0.998534	0.014379	0.853242	0.470612	0.984615	0.048726	29.33452
7	loss	3	15	0.998534	0.013665	0.921502	0.284624	0.984615	0.0479	35.74343
7	loss	4	26	1	0.006078	0.856655	0.471619	0.982906	0.059258	54.84729
7	loss	5	16	0.999023	0.010669	0.887372	0.27569	0.996581	0.025798	46.10693
10	loss	1	18	0.999023	0.009387	0.866894	0.40527	0.982906	0.061318	44.18391
10	loss	2	16	0.999511	0.009909	0.866894	0.479844	0.996581	0.032309	40.4024
10	loss	3	16	1	0.009493	0.846416	0.489362	0.982906	0.068769	48.54023
10	loss	4	22	0.999511	0.007761	0.860068	0.441097	0.981197	0.054182	56.33341
10	loss	5	23	0.999511	0.008148	0.87372	0.329433	0.991453	0.029512	54.65796
15	loss	1	34	1	0.003683	0.836177	0.465539	0.979487	0.054873	90.97743
15	loss	2	23	0.999023	0.006659	0.788396	0.644614	0.991453	0.044725	51.89995
15	loss	3	24	0.999511	0.007204	0.713311	0.718376	0.957265	0.098596	51.59951
15	loss	4	25	0.998046	0.007837	0.918089	0.349919	0.986325	0.04279	69.64649
15	loss	5	19	0.999023	0.00891	0.877133	0.413882	0.989744	0.062436	45.38881
3	accuracy	1	8	0.995115	0.035462	0.812287	0.452551	0.977778	0.075911	19.19615
3	accuracy	2	7	0.991695	0.03949	0.750853	0.558056	0.969231	0.086702	22.83207
3	accuracy	3	8	0.990718	0.044302	0.890785	0.298727	0.986325	0.06379	22.52161
3	accuracy	4	10	0.995603	0.028642	0.883959	0.357354	0.981197	0.063921	23.85584
3	accuracy	5	11	0.996092	0.022401	0.8157	0.428575	0.982906	0.049576	25.19012
5	accuracy	1	9	0.997557	0.02813	0.924915	0.235847	0.989744	0.052855	26.95682
5	accuracy	2	14	0.997069	0.01464	0.897611	0.355202	0.991453	0.031014	35.12124
5	accuracy	3	14	0.999023	0.014261	0.778157	0.556217	0.981197	0.0605	34.65687
5	accuracy	4	9	0.995115	0.028258	0.887372	0.304796	0.984615	0.071368	23.06869
5	accuracy	5	12	0.995603	0.019862	0.798635	0.499943	0.981197	0.054467	40.76033
7	accuracy	1	10	0.995603	0.027834	0.843003	0.472188	0.989744	0.073563	30.75511
7	accuracy	2	10	0.99658	0.025613	0.849829	0.40949	0.988034	0.07837	26.90913
7	accuracy	3	21	0.998534	0.007147	0.880546	0.373008	0.982906	0.051573	47.40013
7	accuracy	4	21	0.999511	0.00838	0.87372	0.394475	0.977778	0.060484	48.73546
7	accuracy	5	13	0.997069	0.016314	0.866894	0.455652	0.986325	0.056186	44.27132
10	accuracy	1	13	0.99658	0.018139	0.808874	0.418347	0.974359	0.100734	39.47001
10	accuracy	2	17	0.997069	0.011614	0.78157	0.638499	0.988034	0.051562	40.93397
10	accuracy	3	25	0.999511	0.00633	0.866894	0.357318	0.981197	0.046204	57.84523
10	accuracy	4	18	0.999511	0.010149	0.83959	0.434623	0.976068	0.061002	41.20016
10	accuracy	5	19	0.999511	0.006501	0.849829	0.45403	0.981197	0.050926	64.77641
15	accuracy	1	40	0.999511	0.002785	0.856655	0.440164	0.981197	0.055093	123.3092
15	accuracy	2	23	1	0.005775	0.822526	0.555798	0.993162	0.035381	66.58583
15	accuracy	3	24	1	0.005983	0.836177	0.522272	0.976068	0.063149	65.34113

(Continued on next page)

(Table A.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	17	0.998534	0.009715	0.911263	0.307966	0.988034	0.120658	42.9821
15	accuracy	5	22	0.999511	0.006387	0.870307	0.375544	0.991453	0.04107	78.37715
-	no monitoring	1	N/A	1	0.001575	0.880546	0.457774	0.977778	0.075568	148.2634
-	no monitoring	2	N/A	1	0.002043	0.771331	0.749586	0.981197	0.048413	117.9772
-	no monitoring	3	N/A	1	0.001433	0.822526	0.563311	0.97265	0.091081	113.3214
-	no monitoring	4	N/A	1	0.001629	0.866894	0.492891	0.977778	0.079798	114.794
-	no monitoring	5	N/A	1	0.001867	0.856655	0.635862	0.982906	0.056227	114.0149

TABLE A.2. Descriptive statistics for MobileNetV2 model performance (Appendix A.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.995	0.028	0.841	0.414	0.984	0.056	25.055
val_loss	3	**std**	0.003	0.008	0.075	0.122	0.008	0.011	6.454
val_loss	3	**min**	0.990	0.021	0.717	0.275	0.973	0.046	16.367
val_loss	3	**25%**	0.994	0.023	0.829	0.383	0.978	0.047	21.420
val_loss	3	**50%**	0.996	0.024	0.863	0.393	0.985	0.053	24.899
val_loss	3	**75%**	0.998	0.034	0.898	0.407	0.991	0.058	31.183
val_loss	3	**max**	0.998	0.040	0.898	0.611	0.991	0.074	31.407
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.998	0.017	0.848	0.414	0.983	0.051	33.112
val_loss	5	**std**	0.002	0.004	0.041	0.079	0.008	0.016	4.844
val_loss	5	**min**	0.995	0.011	0.805	0.335	0.973	0.031	27.502
val_loss	5	**25%**	0.998	0.017	0.816	0.339	0.978	0.038	30.869
val_loss	5	**50%**	0.999	0.017	0.836	0.407	0.981	0.055	31.880
val_loss	5	**75%**	0.999	0.018	0.887	0.483	0.990	0.062	34.956
val_loss	5	**max**	1.000	0.023	0.894	0.506	0.991	0.070	40.352
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.999	0.011	0.870	0.390	0.986	0.046	42.814
val_loss	7	**std**	0.001	0.003	0.036	0.101	0.006	0.012	10.181
val_loss	7	**min**	0.999	0.006	0.829	0.276	0.983	0.026	29.335
val_loss	7	**25%**	0.999	0.009	0.853	0.285	0.983	0.048	35.743
val_loss	7	**50%**	0.999	0.011	0.857	0.448	0.985	0.049	46.107
val_loss	7	**75%**	0.999	0.014	0.887	0.471	0.985	0.051	48.037
val_loss	7	**max**	1.000	0.014	0.922	0.472	0.997	0.059	54.847
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	1.000	0.009	0.863	0.429	0.987	0.049	48.824
val_loss	10	**std**	0.000	0.001	0.010	0.065	0.007	0.018	6.763
val_loss	10	**min**	0.999	0.008	0.846	0.329	0.981	0.030	40.402
val_loss	10	**25%**	1.000	0.008	0.860	0.405	0.983	0.032	44.184
val_loss	10	**50%**	1.000	0.009	0.867	0.441	0.983	0.054	48.540
val_loss	10	**75%**	1.000	0.009	0.867	0.480	0.991	0.061	54.658
val_loss	10	**max**	1.000	0.010	0.874	0.489	0.997	0.069	56.333
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.999	0.007	0.827	0.518	0.981	0.061	61.902
val_loss	15	**std**	0.001	0.002	0.080	0.157	0.014	0.023	18.603
val_loss	15	**min**	0.998	0.004	0.713	0.350	0.957	0.043	45.389
val_loss	15	**25%**	0.999	0.007	0.788	0.414	0.979	0.045	51.600
val_loss	15	**50%**	0.999	0.007	0.836	0.466	0.986	0.055	51.900
val_loss	15	**75%**	1.000	0.008	0.877	0.645	0.990	0.062	69.646
val_loss	15	**max**	1.000	0.009	0.918	0.718	0.991	0.099	90.977

Table A.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.994	0.034	0.831	0.419	0.979	0.068	22.719
val_accuracy	3	**std**	0.002	0.009	0.058	0.099	0.007	0.014	2.228
val_accuracy	3	**min**	0.991	0.022	0.751	0.299	0.969	0.050	19.196
val_accuracy	3	**25%**	0.992	0.029	0.812	0.357	0.978	0.064	22.522
val_accuracy	3	**50%**	0.995	0.035	0.816	0.429	0.981	0.064	22.832
val_accuracy	3	**75%**	0.996	0.039	0.884	0.453	0.983	0.076	23.856
val_accuracy	3	**max**	0.996	0.044	0.891	0.558	0.986	0.087	25.190
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.997	0.021	0.857	0.390	0.986	0.054	32.113
val_accuracy	5	**std**	0.002	0.007	0.065	0.134	0.005	0.015	7.048
val_accuracy	5	**min**	0.995	0.014	0.778	0.236	0.981	0.031	23.069
val_accuracy	5	**25%**	0.996	0.015	0.799	0.305	0.981	0.053	26.957
val_accuracy	5	**50%**	0.997	0.020	0.887	0.355	0.985	0.054	34.657
val_accuracy	5	**75%**	0.998	0.028	0.898	0.500	0.990	0.060	35.121
val_accuracy	5	**max**	0.999	0.028	0.925	0.556	0.991	0.071	40.760
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.997	0.017	0.863	0.421	0.985	0.064	39.614
val_accuracy	7	**std**	0.002	0.010	0.016	0.042	0.005	0.011	10.067
val_accuracy	7	**min**	0.996	0.007	0.843	0.373	0.978	0.052	26.909
val_accuracy	7	**25%**	0.997	0.008	0.850	0.394	0.983	0.056	30.755
val_accuracy	7	**50%**	0.997	0.016	0.867	0.409	0.986	0.060	44.271
val_accuracy	7	**75%**	0.999	0.026	0.874	0.456	0.988	0.074	47.400
val_accuracy	7	**max**	1.000	0.028	0.881	0.472	0.990	0.078	48.735
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.998	0.011	0.829	0.461	0.980	0.062	48.845
val_accuracy	10	**std**	0.001	0.005	0.034	0.106	0.005	0.022	11.659
val_accuracy	10	**min**	0.997	0.006	0.782	0.357	0.974	0.046	39.470
val_accuracy	10	**25%**	0.997	0.007	0.809	0.418	0.976	0.051	40.934
val_accuracy	10	**50%**	1.000	0.010	0.840	0.435	0.981	0.052	41.200
val_accuracy	10	**75%**	1.000	0.012	0.850	0.454	0.981	0.061	57.845
val_accuracy	10	**max**	1.000	0.018	0.867	0.638	0.988	0.101	64.776
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	1.000	0.006	0.859	0.440	0.986	0.063	75.319
val_accuracy	15	**std**	0.001	0.002	0.034	0.102	0.007	0.034	29.723
val_accuracy	15	**min**	0.999	0.003	0.823	0.308	0.976	0.035	42.982
val_accuracy	15	**25%**	1.000	0.006	0.836	0.376	0.981	0.041	65.341
val_accuracy	15	**50%**	1.000	0.006	0.857	0.440	0.988	0.055	66.586
val_accuracy	15	**75%**	1.000	0.006	0.870	0.522	0.991	0.063	78.377
val_accuracy	15	**max**	1.000	0.010	0.911	0.556	0.993	0.121	123.309
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	1.000	0.002	0.840	0.580	0.978	0.070	121.674
no_monitoring	No	**std**	0.000	0.000	0.044	0.117	0.004	0.018	14.970
no_monitoring	No	**min**	1.000	0.001	0.771	0.458	0.973	0.048	113.321
no_monitoring	No	**25%**	1.000	0.002	0.823	0.493	0.978	0.056	114.015
no_monitoring	No	**50%**	1.000	0.002	0.857	0.563	0.978	0.076	114.794
no_monitoring	No	**75%**	1.000	0.002	0.867	0.636	0.981	0.080	117.977
no_monitoring	No	**max**	1.000	0.002	0.881	0.750	0.983	0.091	148.263

TABLE B.1. Detailed results for InceptionV3 with various MC and PC (Appendix B.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	9	0.983878851	0.063424252	0.788395882	0.691795886	0.962393165	0.137821734	34.0100146
3	loss	2	8	0.989252567	0.040872496	0.733788371	0.860006809	0.979487181	0.088228665	29.8984313
3	loss	3	13	0.994137764	0.030255197	0.825938582	0.536911786	0.974358976	0.092576943	47.7669581
3	loss	4	7	0.986809969	0.060364712	0.679180861	0.84068495	0.960683763	0.123133942	28.6042797
3	loss	5	5	0.974596977	0.090318508	0.761092126	0.78642565	0.948717952	0.182484463	27.8326678
5	loss	1	32	0.999022961	0.00739672	0.853242338	0.577991843	0.984615386	0.070216581	117.0095957
5	loss	2	9	0.988764048	0.043512646	0.709897637	1.017861128	0.960683763	0.128654212	37.7095276
5	loss	3	14	0.997557402	0.02145461	0.726962447	0.860041678	0.970940173	0.120089293	53.3798204
5	loss	4	15	0.996580362	0.021638783	0.815699637	0.625344396	0.982905984	0.062162023	79.2635385
5	loss	5	15	0.997068882	0.019070214	0.883959055	0.518541694	0.982905984	0.08363767	66.42613
7	loss	1	19	0.996091843	0.019235797	0.771331072	0.681577265	0.977777779	0.080463678	78.0538619
7	loss	2	8	0.982901812	0.066090435	0.791808903	0.86845696	0.938461542	0.24873282	36.247579
7	loss	3	15	0.995114803	0.025213875	0.832764506	0.57030046	0.981196582	0.105363898	64.5173208
7	loss	4	15	0.997068882	0.019120131	0.849829376	0.585848749	0.979487181	0.069590084	88.7693096
7	loss	5	29	1	0.005879726	0.825938582	0.708134949	0.982905984	0.080251314	147.0300085
10	loss	1	20	0.997557402	0.015320398	0.788395882	0.742725492	0.979487181	0.077499501	78.6837716
10	loss	2	17	0.995603323	0.016541081	0.713310599	1.11474967	0.969230771	0.117452495	68.8851666
10	loss	3	35	0.99951148	0.005891902	0.829351544	0.576556921	0.981196582	0.094332993	153.3307557
10	loss	4	47	0.99951148	0.004547769	0.883959055	0.550591946	0.988034189	0.03828479	375.8302146
10	loss	5	34	0.999022961	0.005897738	0.91126281	0.403910398	0.98974359	0.057163749	237.1262536
15	loss	1	31	0.999022961	0.008081561	0.866894186	0.523374856	0.981196582	0.068272196	184.3225278
15	loss	2	26	0.999022961	0.010207516	0.761092126	1.104023457	0.976068377	0.084269352	121.4915938
15	loss	3	33	0.99951148	0.006300624	0.825938582	0.654499173	0.981196582	0.093615212	140.6734394
15	loss	4	26	0.994137764	0.017620478	0.781569958	0.797062635	0.974358976	0.081692502	121.4853378
15	loss	5	37	1	0.004083774	0.877133131	0.553094745	0.991452992	0.059600405	340.3184283
3	accuracy	1	17	0.995114803	0.0206411	0.808873713	0.725548565	0.970940173	0.102678411	140.3074823
3	accuracy	2	11	0.992183685	0.034354091	0.76791811	0.876152992	0.976068377	0.092387527	65.2484093
3	accuracy	3	4	0.965315104	0.124897584	0.815699637	0.536990285	0.933333337	0.271629155	25.8223064
3	accuracy	4	12	0.991695166	0.028814746	0.829351544	0.577801764	0.986324787	0.071021616	58.943908
3	accuracy	5	10	0.993649244	0.034925677	0.846416354	0.57682842	0.976068377	0.10173586	51.2127445
5	accuracy	1	20	0.995603323	0.015065268	0.829351544	0.646792293	0.981196582	0.082639068	154.8791887
5	accuracy	2	21	0.996091843	0.014288138	0.713310599	1.207827926	0.979487181	0.074940607	145.3536055
5	accuracy	3	22	0.997068882	0.01275574	0.798634827	0.676660776	0.972649574	0.118803278	137.5336144
5	accuracy	4	24	0.998534441	0.009583861	0.883959055	0.497871906	0.98974359	0.032772083	133.823382
5	accuracy	5	10	0.991206646	0.035472624	0.771331072	0.80417043	0.972649574	0.119832106	56.4036108
7	accuracy	1	28	0.998045921	0.007867916	0.798634827	0.770256519	0.981196582	0.089092769	154.7163437
7	accuracy	2	31	0.998534441	0.006880392	0.771331072	0.958002925	0.974358976	0.072101355	162.4083133
7	accuracy	3	22	0.997557402	0.010819421	0.860068262	0.450244308	0.974358976	0.093822978	190.629765
7	accuracy	4	36	1	0.003621859	0.819112599	0.797214568	0.981196582	0.05228468	278.316256
7	accuracy	5	22	0.99951148	0.009172016	0.829351544	0.701360941	0.986324787	0.077255376	169.3633921
10	accuracy	1	40	1	0.003867939	0.846416354	0.548633695	0.979487181	0.082988761	284.900779
10	accuracy	2	25	0.999022961	0.010678066	0.808873713	0.794085383	0.977777779	0.069033876	173.9952877
10	accuracy	3	39	0.99951148	0.0046171	0.849829376	0.599616289	0.977777779	0.11157576	247.5588835
10	accuracy	4	24	0.998045921	0.010440554	0.873720109	0.534663856	0.98974359	0.043696053	141.4675116
10	accuracy	5	31	0.998045921	0.009225765	0.907849848	0.497515827	0.98974359	0.068952538	271.3575743
15	accuracy	1	28	1	0.005100978	0.832764506	0.671540976	0.977777779	0.089632653	226.8121977
15	accuracy	2	50	1	0.002206131	0.778156996	1.169290781	0.976068377	0.07546217	379.2302903
15	accuracy	3	30	0.998534441	0.007315609	0.863481224	0.531567454	0.982905984	0.089072227	207.6188399

(Continued on next page)

(Table B.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	44	1	0.00308408	0.877133131	0.580116212	0.986324787	0.046814781	282.2350076
15	accuracy	5	29	0.998534441	0.009574798	0.883959055	0.522773325	0.988034189	0.071746752	187.7881733
-	no monitoring	1	N/A	0.998534441	0.00314847	0.907849848	0.450603038	0.982905984	0.085012116	317.1966538
-	no monitoring	2	N/A	0.999022961	0.003437786	0.812286675	1.013028383	0.977777779	0.075830355	344.6338879
-	no monitoring	3	N/A	0.99951148	0.003941682	0.78498292	0.774207115	0.976068377	0.136283711	544.4648693
-	no monitoring	4	N/A	0.999022961	0.004351126	0.788395882	1.109158397	0.970940173	0.12248259	494.416349
-	no monitoring	5	N/A	0.99951148	0.002637709	0.883959055	0.515361607	0.98974359	0.076655284	381.0932598

TABLE B.2. Descriptive statistics for InceptionV3 model performance (Appendix B.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.986	0.057	0.758	0.743	0.965	0.125	33.622
val_loss	3	**std**	0.007	0.023	0.056	0.132	0.012	0.038	8.258
val_loss	3	**min**	0.975	0.030	0.679	0.537	0.949	0.088	27.833
val_loss	3	**25%**	0.984	0.041	0.734	0.692	0.961	0.093	28.604
val_loss	3	**50%**	0.987	0.060	0.761	0.786	0.962	0.123	29.898
val_loss	3	**75%**	0.989	0.063	0.788	0.841	0.974	0.138	34.010
val_loss	3	**max**	0.994	0.090	0.826	0.860	0.979	0.182	47.767
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.996	0.023	0.798	0.720	0.976	0.093	70.758
val_loss	5	**std**	0.004	0.013	0.077	0.211	0.010	0.030	30.102
val_loss	5	**min**	0.989	0.007	0.710	0.519	0.961	0.062	37.710
val_loss	5	**25%**	0.997	0.019	0.727	0.578	0.971	0.070	53.380
val_loss	5	**50%**	0.997	0.021	0.816	0.625	0.983	0.084	66.426
val_loss	5	**75%**	0.998	0.022	0.853	0.860	0.983	0.120	79.264
val_loss	5	**max**	0.999	0.044	0.884	1.018	0.985	0.129	117.010
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.994	0.027	0.814	0.683	0.972	0.117	82.924
val_loss	7	**std**	0.007	0.023	0.032	0.120	0.019	0.075	40.881
val_loss	7	**min**	0.983	0.006	0.771	0.570	0.938	0.070	36.248
val_loss	7	**25%**	0.995	0.019	0.792	0.586	0.978	0.080	64.517
val_loss	7	**50%**	0.996	0.019	0.826	0.682	0.979	0.080	78.054
val_loss	7	**75%**	0.997	0.025	0.833	0.708	0.981	0.105	88.769
val_loss	7	**max**	1.000	0.066	0.850	0.868	0.983	0.249	147.030
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.998	0.010	0.825	0.678	0.982	0.077	182.771
val_loss	10	**std**	0.002	0.006	0.079	0.272	0.008	0.031	127.375
val_loss	10	**min**	0.996	0.005	0.713	0.404	0.969	0.038	68.885
val_loss	10	**25%**	0.998	0.006	0.788	0.551	0.979	0.057	78.684
val_loss	10	**50%**	0.999	0.006	0.829	0.577	0.981	0.077	153.331
val_loss	10	**75%**	1.000	0.015	0.884	0.743	0.988	0.094	237.126
val_loss	10	**max**	1.000	0.017	0.911	1.115	0.990	0.117	375.830
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.998	0.009	0.823	0.726	0.981	0.077	181.658
val_loss	15	**std**	0.002	0.005	0.051	0.237	0.007	0.013	92.332
val_loss	15	**min**	0.994	0.004	0.761	0.523	0.974	0.060	121.485
val_loss	15	**25%**	0.999	0.006	0.782	0.553	0.976	0.068	121.492
val_loss	15	**50%**	0.999	0.008	0.826	0.654	0.981	0.082	140.673
val_loss	15	**75%**	1.000	0.010	0.867	0.797	0.981	0.084	184.323
val_loss	15	**max**	1.000	0.018	0.877	1.104	0.991	0.094	340.318

Table B.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.988	0.049	0.814	0.659	0.969	0.128	68.307
val_accuracy	3	**std**	0.013	0.043	0.029	0.141	0.020	0.081	42.948
val_accuracy	3	**min**	0.965	0.021	0.768	0.537	0.933	0.071	25.822
val_accuracy	3	**25%**	0.992	0.029	0.809	0.577	0.971	0.092	51.213
val_accuracy	3	**50%**	0.992	0.034	0.816	0.578	0.976	0.102	58.944
val_accuracy	3	**75%**	0.994	0.035	0.829	0.726	0.976	0.103	65.248
val_accuracy	3	**max**	0.995	0.125	0.846	0.876	0.986	0.272	140.307
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.996	0.017	0.799	0.767	0.979	0.086	125.599
val_accuracy	5	**std**	0.003	0.010	0.064	0.270	0.007	0.036	39.515
val_accuracy	5	**min**	0.991	0.010	0.713	0.498	0.973	0.033	56.404
val_accuracy	5	**25%**	0.996	0.013	0.771	0.647	0.973	0.075	133.823
val_accuracy	5	**50%**	0.996	0.014	0.799	0.677	0.979	0.083	137.534
val_accuracy	5	**75%**	0.997	0.015	0.829	0.804	0.981	0.119	145.354
val_accuracy	5	**max**	0.999	0.035	0.884	1.208	0.990	0.120	154.879
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.999	0.008	0.816	0.735	0.979	0.077	191.087
val_accuracy	7	**std**	0.001	0.003	0.033	0.185	0.005	0.016	50.563
val_accuracy	7	**min**	0.998	0.004	0.771	0.450	0.974	0.052	154.716
val_accuracy	7	**25%**	0.998	0.007	0.799	0.701	0.974	0.072	162.408
val_accuracy	7	**50%**	0.999	0.008	0.819	0.770	0.981	0.077	169.363
val_accuracy	7	**75%**	1.000	0.009	0.829	0.797	0.981	0.089	190.630
val_accuracy	7	**max**	1.000	0.011	0.860	0.958	0.986	0.094	278.316
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.999	0.008	0.857	0.595	0.983	0.075	223.856
val_accuracy	10	**std**	0.001	0.003	0.037	0.117	0.006	0.025	62.886
val_accuracy	10	**min**	0.998	0.004	0.809	0.498	0.978	0.044	141.468
val_accuracy	10	**25%**	0.998	0.005	0.846	0.535	0.978	0.069	173.995
val_accuracy	10	**50%**	0.999	0.009	0.850	0.549	0.979	0.069	247.559
val_accuracy	10	**75%**	1.000	0.010	0.874	0.600	0.990	0.083	271.358
val_accuracy	10	**max**	1.000	0.011	0.908	0.794	0.990	0.112	284.901
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.999	0.005	0.847	0.695	0.982	0.075	256.737
val_accuracy	15	**std**	0.001	0.003	0.043	0.272	0.005	0.017	77.001
val_accuracy	15	**min**	0.999	0.002	0.778	0.523	0.976	0.047	187.788
val_accuracy	15	**25%**	0.999	0.003	0.833	0.532	0.978	0.072	207.619
val_accuracy	15	**50%**	1.000	0.005	0.863	0.580	0.983	0.075	226.812
val_accuracy	15	**75%**	1.000	0.007	0.877	0.672	0.986	0.089	282.235
val_accuracy	15	**max**	1.000	0.010	0.884	1.169	0.988	0.090	379.230
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	0.999	0.004	0.835	0.772	0.979	0.099	416.361
no_monitoring	No	**std**	0.000	0.001	0.057	0.292	0.007	0.028	98.394
no_monitoring	No	**min**	0.999	0.003	0.785	0.451	0.971	0.076	317.197
no_monitoring	No	**25%**	0.999	0.003	0.788	0.515	0.976	0.077	344.634
no_monitoring	No	**50%**	0.999	0.003	0.812	0.774	0.978	0.085	381.093
no_monitoring	No	**75%**	1.000	0.004	0.884	1.013	0.983	0.122	494.416
no_monitoring	No	**max**	1.000	0.004	0.908	1.109	0.990	0.136	544.465

TABLE C.1. Detailed results for ResNet50V2 with various MC and PC (Appendix C.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	18	0.998046	0.02271	0.716724	0.61063	0.97265	0.07408	24.89913
3	loss	2	18	0.989741	0.039663	0.897611	0.407444	0.991453	0.046021	21.42021
3	loss	3	18	0.998046	0.02114	0.829352	0.383397	0.977778	0.058188	31.18301
3	loss	4	20	0.996092	0.0236	0.863481	0.39339	0.984615	0.052646	31.40667
3	loss	5	15	0.994138	0.034424	0.897611	0.27528	0.991453	0.046647	16.36674
5	loss	1	16	0.994626	0.022884	0.836177	0.407061	0.977778	0.061897	40.35201
5	loss	2	16	0.998046	0.017082	0.8157	0.505994	0.991453	0.037977	31.8802
5	loss	3	21	0.998534	0.017046	0.805461	0.482879	0.97265	0.070374	27.50178
5	loss	4	12	0.999023	0.018486	0.887372	0.335181	0.981197	0.054642	30.86905
5	loss	5	15	1	0.011072	0.894198	0.338853	0.989744	0.031267	34.95643
7	loss	1	18	0.999023	0.008709	0.829352	0.448269	0.982906	0.050633	48.03739
7	loss	2	37	0.998534	0.014379	0.853242	0.470612	0.984615	0.048726	29.33452
7	loss	3	19	0.998534	0.013665	0.921502	0.284624	0.984615	0.0479	35.74343
7	loss	4	23	1	0.006078	0.856655	0.471619	0.982906	0.059258	54.84729
7	loss	5	36	0.999023	0.010669	0.887372	0.27569	0.996581	0.025798	46.10693
10	loss	1	27	0.999023	0.009387	0.866894	0.40527	0.982906	0.061318	44.18391
10	loss	2	27	0.999511	0.009909	0.866894	0.479844	0.996581	0.032309	40.4024
10	loss	3	27	1	0.009493	0.846416	0.489362	0.982906	0.068769	48.54023
10	loss	4	18	0.999511	0.007761	0.860068	0.441097	0.981197	0.054182	56.33341
10	loss	5	32	0.999511	0.008148	0.87372	0.329433	0.991453	0.029512	54.65796
15	loss	1	50	1	0.003683	0.836177	0.465539	0.979487	0.054873	90.97743
15	loss	2	48	0.999023	0.006659	0.788396	0.644614	0.991453	0.044725	51.89995
15	loss	3	33	0.999511	0.007204	0.713311	0.718376	0.957265	0.098596	51.59951
15	loss	4	49	0.998046	0.007837	0.918089	0.349919	0.986325	0.04279	69.64649
15	loss	5	28	0.999023	0.00891	0.877133	0.413882	0.989744	0.062436	45.38881
3	accuracy	1	17	0.995115	0.035462	0.812287	0.452551	0.977778	0.075911	19.19615
3	accuracy	2	8	0.991695	0.03949	0.750853	0.558056	0.969231	0.086702	22.83207
3	accuracy	3	9	0.990718	0.044302	0.890785	0.298727	0.986325	0.06379	22.52161
3	accuracy	4	10	0.995603	0.028642	0.883959	0.357354	0.981197	0.063921	23.85584
3	accuracy	5	14	0.996092	0.022401	0.8157	0.428575	0.982906	0.049576	25.19012
5	accuracy	1	16	0.997557	0.02813	0.924915	0.235847	0.989744	0.052855	26.95682
5	accuracy	2	14	0.997069	0.01464	0.897611	0.355202	0.991453	0.031014	35.12124
5	accuracy	3	14	0.999023	0.014261	0.778157	0.556217	0.981197	0.0605	34.65687
5	accuracy	4	25	0.995115	0.028258	0.887372	0.304796	0.984615	0.071368	23.06869
5	accuracy	5	12	0.995603	0.019862	0.798635	0.499943	0.981197	0.054467	40.76033
7	accuracy	1	17	0.995603	0.027834	0.843003	0.472188	0.989744	0.073563	30.75511
7	accuracy	2	20	0.99658	0.025613	0.849829	0.40949	0.988034	0.07837	26.90913
7	accuracy	3	16	0.998534	0.007147	0.880546	0.373008	0.982906	0.051573	47.40013
7	accuracy	4	30	0.999511	0.00838	0.87372	0.394475	0.977778	0.060484	48.73546
7	accuracy	5	13	0.997069	0.016314	0.866894	0.455652	0.986325	0.056186	44.27132
10	accuracy	1	26	0.99658	0.018139	0.808874	0.418347	0.974359	0.100734	39.47001
10	accuracy	2	15	0.997069	0.011614	0.78157	0.638499	0.988034	0.051562	40.93397
10	accuracy	3	18	0.999511	0.00633	0.866894	0.357318	0.981197	0.046204	57.84523
10	accuracy	4	25	0.999511	0.010149	0.83959	0.434623	0.976068	0.061002	41.20016
10	accuracy	5	33	0.999511	0.006501	0.849829	0.45403	0.981197	0.050926	64.77641
15	accuracy	1	50	0.999511	0.002785	0.856655	0.440164	0.981197	0.055093	123.3092
15	accuracy	2	26	1	0.005775	0.822526	0.555798	0.993162	0.035381	66.58583
15	accuracy	3	26	1	0.005983	0.836177	0.522272	0.976068	0.063149	65.34113

(Continued on next page)

(Table C.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	45	0.998534	0.009715	0.911263	0.307966	0.988034	0.120658	42.9821
15	accuracy	5	35	0.999511	0.006387	0.870307	0.375544	0.991453	0.04107	78.37715
-	no monitoring	1	N/A	1	0.001575	0.880546	0.457774	0.977778	0.075568	148.2634
-	no monitoring	2	N/A	1	0.002043	0.771331	0.749586	0.981197	0.048413	117.9772
-	no monitoring	3	N/A	1	0.001433	0.822526	0.563311	0.97265	0.091081	113.3214
-	no monitoring	4	N/A	1	0.001629	0.866894	0.492891	0.977778	0.079798	114.794
-	no monitoring	5	N/A	1	0.001867	0.856655	0.635862	0.982906	0.056227	114.0149

TABLE C.2. Descriptive statistics for ResNet50V2 model performance (Appendix C.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.995	0.026	0.958	0.121	0.994	0.026	139.693
val_loss	3	**std**	0.002	0.005	0.007	0.010	0.004	0.006	14.402
val_loss	3	**min**	0.993	0.017	0.949	0.108	0.988	0.020	126.999
val_loss	3	**25%**	0.994	0.026	0.952	0.117	0.990	0.020	132.744
val_loss	3	**50%**	0.996	0.026	0.959	0.123	0.995	0.023	133.583
val_loss	3	**75%**	0.996	0.029	0.962	0.123	0.997	0.031	141.364
val_loss	3	**max**	0.998	0.030	0.966	0.136	0.998	0.033	163.777
val_loss	5	**count**	5	5	5	5	5	5	5
val_loss	5	**mean**	0.993	0.032	0.918	0.194	0.991	0.044	127.430
val_loss	5	**std**	0.004	0.012	0.034	0.069	0.006	0.018	26.791
val_loss	5	**min**	0.988	0.018	0.884	0.107	0.983	0.029	89.900
val_loss	5	**25%**	0.991	0.024	0.887	0.153	0.986	0.029	112.390
val_loss	5	**50%**	0.995	0.031	0.925	0.183	0.995	0.037	134.345
val_loss	5	**75%**	0.997	0.041	0.928	0.256	0.995	0.060	141.700
val_loss	5	**max**	0.997	0.046	0.966	0.271	0.995	0.066	158.813
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.997	0.015	0.923	0.166	0.993	0.026	197.787
val_loss	7	**std**	0.002	0.007	0.031	0.046	0.002	0.008	60.948
val_loss	7	**min**	0.996	0.007	0.870	0.131	0.990	0.017	140.188
val_loss	7	**25%**	0.996	0.007	0.922	0.142	0.991	0.021	150.817
val_loss	7	**50%**	0.996	0.017	0.935	0.151	0.993	0.022	173.366
val_loss	7	**75%**	0.999	0.020	0.939	0.162	0.995	0.033	247.801
val_loss	7	**max**	1.000	0.022	0.949	0.246	0.995	0.037	276.762
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.997	0.016	0.891	0.232	0.987	0.040	201.758
val_loss	10	**std**	0.001	0.006	0.091	0.204	0.017	0.038	38.513
val_loss	10	**min**	0.996	0.010	0.730	0.115	0.957	0.020	142.963
val_loss	10	**25%**	0.996	0.011	0.908	0.127	0.993	0.023	201.046
val_loss	10	**50%**	0.997	0.016	0.932	0.139	0.993	0.025	203.378
val_loss	10	**75%**	0.999	0.018	0.939	0.184	0.995	0.025	210.735
val_loss	10	**max**	0.999	0.024	0.949	0.593	0.997	0.109	250.669
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.999	0.008	0.934	0.133	0.996	0.019	364.172
val_loss	15	**std**	0.001	0.003	0.025	0.033	0.003	0.008	78.947
val_loss	15	**min**	0.996	0.003	0.894	0.104	0.991	0.013	273.448
val_loss	15	**25%**	0.999	0.006	0.928	0.112	0.995	0.015	287.525
val_loss	15	**50%**	0.999	0.007	0.939	0.121	0.997	0.015	392.789
val_loss	15	**75%**	1.000	0.011	0.949	0.141	0.997	0.016	419.770
val_loss	15	**max**	1.000	0.011	0.959	0.187	1.000	0.034	447.328

Table C.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.985	0.053	0.939	0.141	0.983	0.056	151.258
val_accuracy	3	**std**	0.009	0.025	0.042	0.062	0.011	0.023	33.598
val_accuracy	3	**min**	0.972	0.027	0.891	0.074	0.974	0.029	120.028
val_accuracy	3	**25%**	0.982	0.034	0.904	0.093	0.976	0.040	133.953
val_accuracy	3	**50%**	0.986	0.053	0.939	0.126	0.976	0.062	139.552
val_accuracy	3	**75%**	0.991	0.065	0.980	0.197	0.990	0.063	155.989
val_accuracy	3	**max**	0.995	0.088	0.983	0.214	0.998	0.088	206.765
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.993	0.031	0.939	0.132	0.990	0.037	176.910
val_accuracy	5	**std**	0.003	0.013	0.021	0.043	0.003	0.009	53.507
val_accuracy	5	**min**	0.989	0.012	0.915	0.095	0.986	0.027	142.929
val_accuracy	5	**25%**	0.992	0.031	0.922	0.097	0.988	0.035	146.108
val_accuracy	5	**50%**	0.993	0.031	0.945	0.113	0.990	0.037	147.014
val_accuracy	5	**75%**	0.994	0.035	0.949	0.165	0.991	0.037	179.683
val_accuracy	5	**max**	0.998	0.049	0.966	0.189	0.995	0.050	268.818
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.994	0.028	0.941	0.137	0.991	0.040	238.396
val_accuracy	7	**std**	0.004	0.013	0.046	0.084	0.005	0.019	100.550
val_accuracy	7	**min**	0.988	0.012	0.860	0.075	0.986	0.021	162.213
val_accuracy	7	**25%**	0.991	0.019	0.949	0.084	0.986	0.030	190.180
val_accuracy	7	**50%**	0.995	0.028	0.962	0.110	0.991	0.032	193.103
val_accuracy	7	**75%**	0.996	0.034	0.962	0.134	0.991	0.045	234.178
val_accuracy	7	**max**	0.999	0.046	0.969	0.280	0.998	0.071	412.303
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.997	0.018	0.955	0.106	0.991	0.036	303.144
val_accuracy	10	**std**	0.002	0.009	0.010	0.021	0.006	0.020	103.501
val_accuracy	10	**min**	0.994	0.010	0.945	0.073	0.981	0.020	185.947
val_accuracy	10	**25%**	0.997	0.013	0.945	0.102	0.991	0.023	217.471
val_accuracy	10	**50%**	0.999	0.015	0.956	0.113	0.991	0.028	315.892
val_accuracy	10	**75%**	0.999	0.020	0.959	0.121	0.993	0.038	356.140
val_accuracy	10	**max**	0.999	0.034	0.969	0.124	0.998	0.069	440.271
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.999	0.009	0.919	0.166	0.992	0.025	520.313
val_accuracy	15	**std**	0.002	0.007	0.040	0.071	0.003	0.006	152.293
val_accuracy	15	**min**	0.997	0.002	0.874	0.089	0.988	0.019	356.875
val_accuracy	15	**25%**	0.998	0.005	0.891	0.092	0.991	0.020	384.393
val_accuracy	15	**50%**	0.999	0.007	0.911	0.189	0.991	0.025	514.190
val_accuracy	15	**75%**	1.000	0.015	0.952	0.227	0.993	0.030	651.946
val_accuracy	15	**max**	1.000	0.018	0.969	0.234	0.997	0.033	694.159
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	1.000	0.004	0.911	0.209	0.985	0.033	723.026
no_monitoring	No	**std**	0.000	0.001	0.057	0.147	0.010	0.026	41.621
no_monitoring	No	**min**	0.999	0.004	0.829	0.102	0.969	0.014	694.063
no_monitoring	No	**25%**	1.000	0.004	0.874	0.103	0.985	0.016	705.354
no_monitoring	No	**50%**	1.000	0.004	0.942	0.128	0.988	0.021	707.042
no_monitoring	No	**75%**	1.000	0.005	0.949	0.271	0.991	0.035	712.132
no_monitoring	No	**max**	1.000	0.005	0.962	0.440	0.993	0.076	796.537

TABLE D.1. Detailed results for Xception with various MC and PC (Appendix D.1)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
3	loss	1	9	0.972642899	0.129837066	0.648464143	0.726279438	0.94700855	0.195763454	64.5046287
3	loss	2	9	0.965803623	0.145221606	0.713310599	0.582341194	0.940170944	0.190947622	66.8170586
3	loss	3	8	0.969711781	0.121941559	0.662116051	0.679573655	0.931623936	0.209745377	62.8957683
3	loss	4	11	0.98534441	0.084378332	0.737201393	0.631100476	0.938461542	0.164128602	86.5575404
3	loss	5	10	0.979482174	0.099705689	0.69624573	0.643902957	0.952136755	0.161598474	76.0659985
5	loss	1	14	0.98583293	0.070808701	0.692832768	0.699655771	0.95726496	0.147542983	108.2439275
5	loss	2	12	0.989252567	0.076574892	0.709897637	0.663573325	0.964102566	0.136994839	91.7473245
5	loss	3	16	0.990229607	0.064519256	0.69624573	0.740498126	0.960683763	0.13576889	122.3358252
5	loss	4	21	0.993649244	0.04727627	0.757679164	0.623574436	0.945299149	0.132990971	162.5682723
5	loss	5	14	0.989741087	0.066146471	0.740614355	0.570958138	0.962393165	0.135764197	105.715716
7	loss	1	19	0.994137764	0.043756392	0.720136523	0.658100367	0.95726496	0.122135416	145.7735281
7	loss	2	12	0.979482174	0.105163231	0.668941975	0.66026026	0.928205132	0.212660953	96.5790529
7	loss	3	17	0.992672205	0.058356903	0.648464143	0.803342044	0.95726496	0.13514547	127.8505467
7	loss	4	15	0.989741087	0.064322673	0.757679164	0.545565844	0.943589747	0.165480867	119.8481848
7	loss	5	19	0.995603323	0.042332146	0.726962447	0.64823091	0.969230771	0.11589212	136.9359018
10	loss	1	23	0.996580362	0.032822847	0.716723561	0.705110371	0.965811968	0.111317508	172.6875957
10	loss	2	25	0.996091843	0.037626501	0.692832768	0.806456029	0.972649574	0.096915461	192.6734316
10	loss	3	16	0.991206646	0.064496234	0.713310599	0.656079113	0.950427353	0.195622265	122.5274311
10	loss	4	33	0.998045921	0.023534197	0.757679164	0.660158396	0.953846157	0.11040625	262.3697482
10	loss	5	17	0.992183685	0.052593071	0.713310599	0.668919325	0.94700855	0.171621963	127.6774122
15	loss	1	29	0.998534441	0.023043793	0.757679164	0.580849946	0.967521369	0.108793311	214.6257377
15	loss	2	23	0.996580362	0.036374774	0.713310599	0.73689121	0.962393165	0.131636575	174.2537773
15	loss	3	32	0.997557402	0.022166649	0.747440279	0.742685497	0.970940173	0.097075753	252.1042346
15	loss	4	30	0.999022961	0.022557253	0.754266202	0.713742137	0.953846157	0.121158503	250.3157561
15	loss	5	30	0.998534441	0.023030076	0.730375409	0.734522462	0.972649574	0.107649416	276.7263841
3	accuracy	1	15	0.993649244	0.061796885	0.69624573	0.670105934	0.962393165	0.130404502	142.4387054
3	accuracy	2	5	0.957498789	0.191257089	0.621160388	0.741892576	0.928205132	0.319094092	50.9358952
3	accuracy	3	7	0.966780663	0.159885839	0.737201393	0.562528431	0.935042739	0.244990811	65.6215291
3	accuracy	4	17	0.994137764	0.045600958	0.774744034	0.577316344	0.952136755	0.127913699	154.4101072
3	accuracy	5	6	0.959941387	0.16798079	0.675767899	0.688473523	0.921367526	0.265415251	65.0164075
5	accuracy	1	16	0.990718126	0.060059696	0.692832768	0.812040746	0.965811968	0.127139732	149.1743443
5	accuracy	2	10	0.975085497	0.113430224	0.651877105	0.691287875	0.931623936	0.21590881	76.2116969
5	accuracy	3	15	0.989741087	0.065249637	0.744027317	0.619207919	0.965811968	0.122518174	123.6166012
5	accuracy	4	22	0.994137764	0.040192954	0.761092126	0.607570052	0.948717952	0.125213459	170.1709438
5	accuracy	5	17	0.991695166	0.052661575	0.730375409	0.589609325	0.969230771	0.109279864	130.5702227
7	accuracy	1	25	0.996091843	0.032614194	0.69624573	0.786663473	0.967521369	0.111560121	190.436003
7	accuracy	2	17	0.991206646	0.05449627	0.682593882	0.746500134	0.969230771	0.112729013	130.1836983
7	accuracy	3	19	0.992672205	0.047206469	0.672354937	0.820098698	0.962393165	0.125569016	162.6656513
7	accuracy	4	9	0.980459213	0.111395694	0.69624573	0.665015817	0.907692313	0.355802476	74.0074007
7	accuracy	5	38	0.999022961	0.015742909	0.740614355	0.766431868	0.970940173	0.089645736	297.7733321
10	accuracy	1	34	0.997557402	0.020063197	0.675767899	0.878209054	0.969230771	0.105710268	277.3612384
10	accuracy	2	31	0.999022961	0.020484067	0.686006844	0.948435187	0.969230771	0.086290933	252.9607732
10	accuracy	3	22	0.995114803	0.037949301	0.737201393	0.683958352	0.958974361	0.119537391	187.3716481
10	accuracy	4	30	0.99951148	0.022190876	0.78498292	0.569951594	0.952136755	0.123341084	241.4252356
10	accuracy	5	17	0.995114803	0.043765735	0.726962447	0.678048849	0.964102566	0.145041451	137.2071482
15	accuracy	1	46	0.998534441	0.010064815	0.720136523	0.874718249	0.974358976	0.100223176	365.9279238
15	accuracy	2	50	0.999022961	0.010573568	0.709897637	0.986595809	0.972649574	0.073904678	388.8055961
15	accuracy	3	29	0.998045921	0.023716006	0.703071654	0.846732259	0.958974361	0.121186987	273.9335451

(Continued on next page)

(Table D.1 Continued from previous page)

PC	MC	Fold	Stop Epoch	Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time (s)
15	accuracy	4	26	0.997068882	0.028543573	0.76791811	0.628445148	0.952136755	0.132668525	234.2733131
15	accuracy	5	36	0.998534441	0.022074558	0.740614355	0.778797984	0.970940173	0.098766908	306.510873
-	no monitoring	1	N/A	0.999022961	0.010553496	0.706484616	0.941983461	0.974358976	0.118602358	439.8904622
-	no monitoring	2	N/A	1	0.01098085	0.686006844	1.034381628	0.967521369	0.094622567	437.3137553
-	no monitoring	3	N/A	0.99951148	0.010696846	0.75085324	0.825193703	0.962393165	0.125199303	396.7657198
-	no monitoring	4	N/A	0.999022961	0.011571754	0.764505148	0.73913306	0.952136755	0.154828563	463.3622841
-	no monitoring	5	N/A	0.99951148	0.008148649	0.75085324	0.829228759	0.970940173	0.104553312	408.4304722

TABLE D.2. Descriptive statistics for Xception model Performance (Appendix D.2)

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_loss	3	**count**	5	5	5	5	5	5	5
val_loss	3	**mean**	0.975	0.116	0.691	0.653	0.942	0.184	71.368
val_loss	3	**std**	0.008	0.024	0.036	0.054	0.008	0.021	9.904
val_loss	3	**min**	0.966	0.084	0.648	0.582	0.932	0.162	62.896
val_loss	3	**25%**	0.970	0.100	0.662	0.631	0.938	0.164	64.505
val_loss	3	**50%**	0.973	0.122	0.696	0.644	0.940	0.191	66.817
val_loss	3	**75%**	0.979	0.130	0.713	0.680	0.947	0.196	76.066
val_loss	3	**max**	0.985	0.145	0.737	0.726	0.952	0.210	86.558
val_loss	5	**count**	v	.5	.5	.5	.5	.5	.5
val_loss	5	**mean**	0.990	0.065	0.719	0.660	0.958	0.138	118.122
val_loss	5	**std**	0.003	0.011	0.028	0.066	0.008	0.006	27.112
val_loss	5	**min**	0.986	0.047	0.693	0.571	0.945	0.133	91.747
val_loss	5	**25%**	0.989	0.065	0.696	0.624	0.957	0.136	105.716
val_loss	5	**50%**	0.990	0.066	0.710	0.664	0.961	0.136	108.244
val_loss	5	**75%**	0.990	0.071	0.741	0.700	0.962	0.137	122.336
val_loss	5	**max**	0.994	0.077	0.758	0.740	0.964	0.148	162.568
val_loss	7	**count**	5	5	5	5	5	5	5
val_loss	7	**mean**	0.990	0.063	0.704	0.663	0.951	0.150	125.397
val_loss	7	**std**	0.006	0.025	0.045	0.092	0.016	0.040	18.812
val_loss	7	**min**	0.979	0.042	0.648	0.546	0.928	0.116	96.579
val_loss	7	**25%**	0.990	0.044	0.669	0.648	0.944	0.122	119.848
val_loss	7	**50%**	0.993	0.058	0.720	0.658	0.957	0.135	127.851
val_loss	7	**75%**	0.994	0.064	0.727	0.660	0.957	0.165	136.936
val_loss	7	**max**	0.996	0.105	0.758	0.803	0.969	0.213	145.774
val_loss	10	**count**	5	5	5	5	5	5	5
val_loss	10	**mean**	0.995	0.042	0.719	0.699	0.958	0.137	175.587
val_loss	10	**std**	0.003	0.016	0.024	0.063	0.011	0.044	56.882
val_loss	10	**min**	0.991	0.024	0.693	0.656	0.947	0.097	122.527
val_loss	10	**25%**	0.992	0.033	0.713	0.660	0.950	0.110	127.677
val_loss	10	**50%**	0.996	0.038	0.713	0.669	0.954	0.111	172.688
val_loss	10	**75%**	0.997	0.053	0.717	0.705	0.966	0.172	192.673
val_loss	10	**max**	0.998	0.064	0.758	0.806	0.973	0.196	262.370
val_loss	15	**count**	5	5	5	5	5	5	5
val_loss	15	**mean**	0.998	0.025	0.741	0.702	0.965	0.113	233.605
val_loss	15	**std**	0.001	0.006	0.019	0.068	0.008	0.013	39.886
val_loss	15	**min**	0.997	0.022	0.713	0.581	0.954	0.097	174.254
val_loss	15	**25%**	0.998	0.023	0.730	0.714	0.962	0.108	214.626
val_loss	15	**50%**	0.999	0.023	0.747	0.735	0.968	0.109	250.316
val_loss	15	**75%**	0.999	0.023	0.754	0.737	0.971	0.121	252.104
val_loss	15	**max**	0.999	0.036	0.758	0.743	0.973	0.132	276.726

Table D.2 Continued from previous page

MC	PC		Train_Acc	Train_Loss	Val_Acc	Val_Loss	Test_Acc	Test_Loss	Train Time
val_accuracy	3	**count**	5	5	5	5	5	5	5
val_accuracy	3	**mean**	0.974	0.125	0.701	0.648	0.940	0.218	95.685
val_accuracy	3	**std**	0.018	0.067	0.059	0.076	0.017	0.085	48.686
val_accuracy	3	**min**	0.957	0.046	0.621	0.563	0.921	0.128	50.936
val_accuracy	3	**25%**	0.960	0.062	0.676	0.577	0.928	0.130	65.016
val_accuracy	3	**50%**	0.967	0.160	0.696	0.670	0.935	0.245	65.622
val_accuracy	3	**75%**	0.994	0.168	0.737	0.688	0.952	0.265	142.439
val_accuracy	3	**max**	0.994	0.191	0.775	0.742	0.962	0.319	154.410
val_accuracy	5	**count**	5	5	5	5	5	5	5
val_accuracy	5	**mean**	0.988	0.066	0.716	0.664	0.956	0.140	129.949
val_accuracy	5	**std**	0.008	0.028	0.044	0.091	0.016	0.043	35.056
val_accuracy	5	**min**	0.975	0.040	0.652	0.590	0.932	0.109	76.212
val_accuracy	5	**25%**	0.990	0.053	0.693	0.608	0.949	0.123	123.617
val_accuracy	5	**50%**	0.991	0.060	0.730	0.619	0.966	0.125	130.570
val_accuracy	5	**75%**	0.992	0.065	0.744	0.691	0.966	0.127	149.174
val_accuracy	5	**max**	0.994	0.113	0.761	0.812	0.969	0.216	170.171
val_accuracy	7	**count**	5	5	5	5	5	5	5
val_accuracy	7	**mean**	0.992	0.052	0.698	0.757	0.956	0.159	171.013
val_accuracy	7	**std**	0.007	0.036	0.026	0.058	0.027	0.111	83.055
val_accuracy	7	**min**	0.980	0.016	0.672	0.665	0.908	0.090	74.007
val_accuracy	7	**25%**	0.991	0.033	0.683	0.747	0.962	0.112	130.184
val_accuracy	7	**50%**	0.993	0.047	0.696	0.766	0.968	0.113	162.666
val_accuracy	7	**75%**	0.996	0.054	0.696	0.787	0.969	0.126	190.436
val_accuracy	7	**max**	0.999	0.111	0.741	0.820	0.971	0.356	297.773
val_accuracy	10	**count**	5	5	5	5	5	5	5
val_accuracy	10	**mean**	0.997	0.029	0.722	0.752	0.963	0.116	219.265
val_accuracy	10	**std**	0.002	0.011	0.044	0.156	0.007	0.022	56.463
val_accuracy	10	**min**	0.995	0.020	0.676	0.570	0.952	0.086	137.207
val_accuracy	10	**25%**	0.995	0.020	0.686	0.678	0.959	0.106	187.372
val_accuracy	10	**50%**	0.998	0.022	0.727	0.684	0.964	0.120	241.425
val_accuracy	10	**75%**	0.999	0.038	0.737	0.878	0.969	0.123	252.961
val_accuracy	10	**max**	1.000	0.044	0.785	0.948	0.969	0.145	277.361
val_accuracy	15	**count**	5	5	5	5	5	5	5
val_accuracy	15	**mean**	0.998	0.019	0.728	0.823	0.966	0.105	313.890
val_accuracy	15	**std**	0.001	0.008	0.026	0.132	0.010	0.023	63.855
val_accuracy	15	**min**	0.997	0.010	0.703	0.628	0.952	0.074	234.273
val_accuracy	15	**25%**	0.998	0.011	0.710	0.779	0.959	0.099	273.934
val_accuracy	15	**50%**	0.999	0.022	0.720	0.847	0.971	0.100	306.511
val_accuracy	15	**75%**	0.999	0.024	0.741	0.875	0.973	0.121	365.928
val_accuracy	15	**max**	0.999	0.029	0.768	0.987	0.974	0.133	388.806
no_monitoring	No	**count**	5	5	5	5	5	5	5
no_monitoring	No	**mean**	0.999	0.010	0.732	0.874	0.965	0.120	429.153
no_monitoring	No	**std**	0.000	0.001	0.034	0.115	0.009	0.023	26.601
no_monitoring	No	**min**	0.999	0.008	0.686	0.739	0.952	0.095	396.766
no_monitoring	No	**25%**	0.999	0.011	0.706	0.825	0.962	0.105	408.430
no_monitoring	No	**50%**	1.000	0.011	0.751	0.829	0.968	0.119	437.314
no_monitoring	No	**75%**	1.000	0.011	0.751	0.942	0.971	0.125	439.890
no_monitoring	No	**max**	1.000	0.012	0.765	1.034	0.974	0.155	463.362

Vol: 2, Issue: 2, 2024
Pages:140-149
Received: 05 December 2024
Accepted: 13 January 2025

CURRENT TRENDS IN COMPUTING
Research Article
DOI: https://doi.org/10.71074/CTC.1596824

CLASSIFICATION OF EEG SPECTROGRAM IMAGES WITH DEEP LEARNING MODELS FOR ALCOHOLISM DETECTION

ÖZNUR YILDIRIM[1], YAHYA CIHAT SÖKER[2], MEHMET ZAHID YILDIRIM[2*] AND EMRAH ÖZKAYNAK[2]

[1] *TOBB Technical Sciences Vocational School, Karabük University, 78050, Karabük, Türkiye*
[2] *Computer Engineering Department, Karabük University, 78050, Karabük, Türkiye*

ABSTRACT. **Electroencephalogram (EEG) signals are time series that play an essential role in understanding the electrical behavior of the brain. The complex structure of the brain makes the interpretation of EEG signals difficult. In this study, the classification of EEG signals based on image processing with deep learning is performed differently from traditional methods. Images of EEG signals obtained for the detection of alcoholism were used to classify healthy and alcohol-addicted individuals using a Convolutional Neural Network (CNN). Three models have been implemented in the experiments conducted on the EEG images: Resnet50, Xception, and custom CNN. The findings demonstrate that Xception achieves the best accuracy with 100% classification success.**

1. INTRODUCTION

Electroencephalography (EEG) is a method that records electrical brain activity [1]. This method provides feedback related to brain activity by detecting time-dependent frequency information [2,3]. According to the data provided, pre-diagnostic systems have been developed for several nervous system diseases such as epilepsy [4], Alzheimer's [5], emotion analysis [6] and alcohol addiction [7]. Due to the complexity and multidimensional structure of EEG signals, plenty of methods have been proposed to analyze the signals. Fourier and Wavelet Transformations are classical methods that aim to reveal the spectral features of EEG signals with their frequency-based approaches. These methods have been frequently applied to track the frequency components of EEG signals over time and to detect specific events in the signals [8, 9]. In addition, feature extraction techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Independent Component Analysis (ICA) have been used to reduce noise in the signal and analyze the significant components [10] . However, these methods are inadequate, especially in the detection of nonlinear features. In recent years, the preliminary diagnosis and classification of nervous system diseases has become much faster and easier, especially with

E-mail address: oznuryildirim@karabuk.edu.tr, yahyacihat@protonmail.ch, m.zahidyildirim@karabuk.edu.tr(*), eozkaynak@karabuk.edu.tr.
Key words and phrases. Machine learning, Electroencephalography (EEG), Spectrogram, Convolutional neural network (CNN).

machine learning-based algorithms applied to large and complex data. In particular, deep learning-based methods are more successful than traditional methods for complex and difficult-to-analyze data, such as EEG signals [11]. Deep learning methods applied to EEG signals can be divided into two different concepts. The first is based on recurrent neural networks (RNN). RNN-based methods such as Long Short-Term Memory (LSTM) can be used to model temporal dependencies in EEG signals [12,13]. Second, CNN-based methods can be used to process EEG signals in image format [14]. CNN models are deep learning models that can work specifically on images. These models are algorithms that can extract features from images and classify them at the same time.

In this study, alcoholism detection is performed using images obtained from EEG signals. The time-dependent frequency features of EEG signals are converted into spectrogram images, and CNN-based classification is performed. Popular CNN-based algorithms, Resnet50 and Xception models, and a custom CNN model were applied to the spectrogram images. Among these methods, the Xception model achieved 100% classification success in detecting alcoholism on spectrogram images in the experimental studies and gave more successful results compared to other methods. This success in classifying spectrogram images of EEG signals with the CNN model seems promising for classifying complex and multidimensional signals with more straightforward methods.

2. METHODS

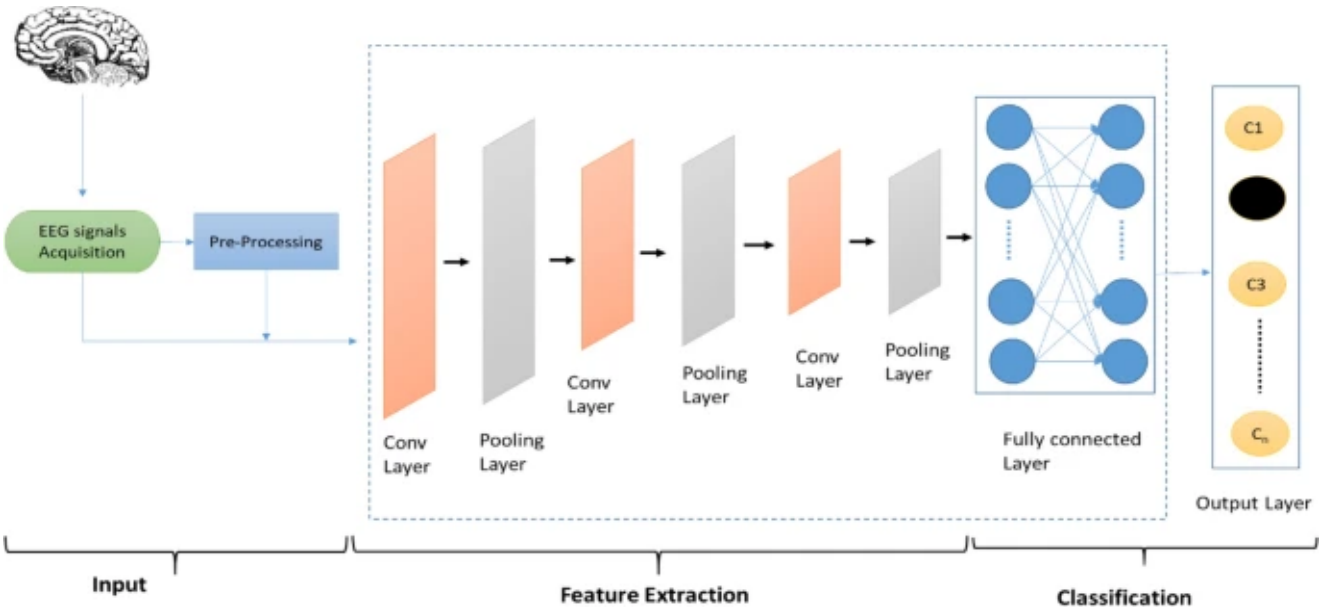


FIGURE 1. **A typical CNN architecture used for EEG analysis [15] .**

CNN is a deep-learning algorithm commonly used in image processing. It is able to detect and classify features in images through the operations performed in its different layers. The input data passes

through convolutional, pooling, and fully connected layers, respectively. Figure 1 shows a general CNN architecture used in EEG signal processing.

The convolutional layer is where various filters are used to extract meaningful features from images. This layer passes filters over the entire image to extract feature maps. In the pooling layer, dimension reduction is performed to reduce the computational cost. The fully connected layer passes the extracted features to a classifier.

The CNN algorithm, with its layered structure, extracts important features from the data without the need for manual feature selection. It also provides more effective learning by preserving spatial and temporal features in the data [16] . Many special models of the CNN algorithm have been developed in accordance with the structure of the data used. ResNet50 and Xception are some of the most popular ones [17].

ResNet50 and Xception are two deep-learning models usually used for image recognition. The ResNet50 model consists of 50 layers. Each layer provides a transferred copy of its output to the following layer via unique connections known as "residual connections." The structure of the ResNet50 model enables the establishment of deeper networks, which means the learning of the network is faster and more efficient. Xception is a modified version of the Inception model. This model requires less computation and parameter selection with an unusual technique called "depthwise separable convolution," which enables this method to operate faster alongside high performance. Both methods are very efficient and widely desired, especially in image classification problems [17]. Additionally, a custom CNN model is implemented in this paper. This model consists of three convolutional layers: a pooling layer, a flattening layer, and a fully connected output layer. The first three layers of the model are 3x3 convolutional layers with 64, 128, and 256 filters. The ReLU activation function is used at the output of each convolutional layer in Equation 1.

$$f(x) = \max(0, x) \tag{1}$$

After the third convolution layer, a max pooling layer is added. This layer reduces the size of the feature maps by selecting the highest values of the image at a given filter size. The max pooling process is expressed as follows, as given in Equation 2.

$$P(i, j) = \max(S(x, y)) \tag{2}$$

After the pooling layer, the flattened layer transforms the multidimensional feature maps into a one-dimensional vector. It then prepares this vector to be transferred to the fully connected layer. The final layer of the model is a fully connected layer consisting of two neurons. It converts the output of the model into a value between 0 and 1 using the sigmoid activation function given in Equation 3.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

The binary cross-entropy function is used to optimize the model and calculate the losses. The Adam optimization algorithm is also used to update the parameters of the model.

2.1. **Evaluation Metrics.**

The complexity matrix is used to evaluate the model performance in detail. The complexity matrix provides four main components by comparing the true and predicted classes. Precision indicates how many of the positively classified examples are actually positive, as seen in Equation 4. Recall indicates how many of the positively classified examples are correctly classified as positive, as seen in Equation 5. F1-Score is a metric that expresses the balance between Precision and Recall, as seen in Equation 6, and is calculated with the harmonic mean. Accuracy is the rate of correct classification of all examples, as seen in Equation 7 [18].

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$F1-Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

2.2. **K-Fold Cross Validation.**

In order to evaluate the performance of the models used in the study more consistently and to minimize the bias in the dataset, a 4-step (4-fold) K-fold cross validation method was applied. This method consists of the following steps as shown in Figure 2.

- The dataset is divided into 4 equal subgroups (folds).
- In each iteration, one of these subgroups is used as the test dataset and the rest as the training dataset.
- The model is trained in each iteration and evaluated on the test dataset.
- The evaluation metrics of all iterations are calculated and the average is taken and the overall performance of the model is evaluated.

This method is an effective strategy to obtain an overall view of model performance and reduce the risk of overfitting.

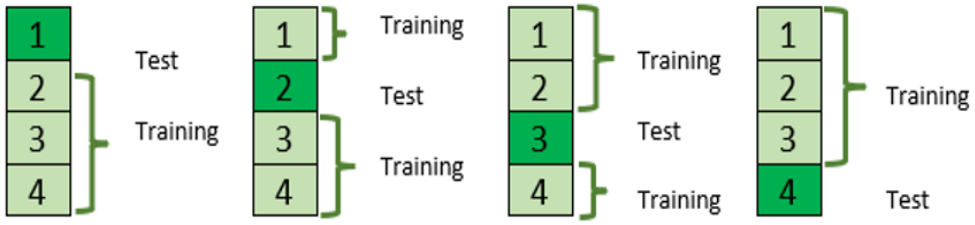


FIGURE 2. **K-Fold Cross Validation for k=4.**

3. Experimental Study

In this study, a dataset of EEG spectrogram images created for alcoholism detection was used [19] . The dataset consists of 7200 one-second images from 12 different brain channels. 5400 images were used for training, 900 of the remaining 1800 images were used for testing, and 900 were used for validation. There are equal numbers of alcohol-dependent and normal subjects in the training and test data. Figure 3 shows examples of normal subjects and dependent on alcohol.

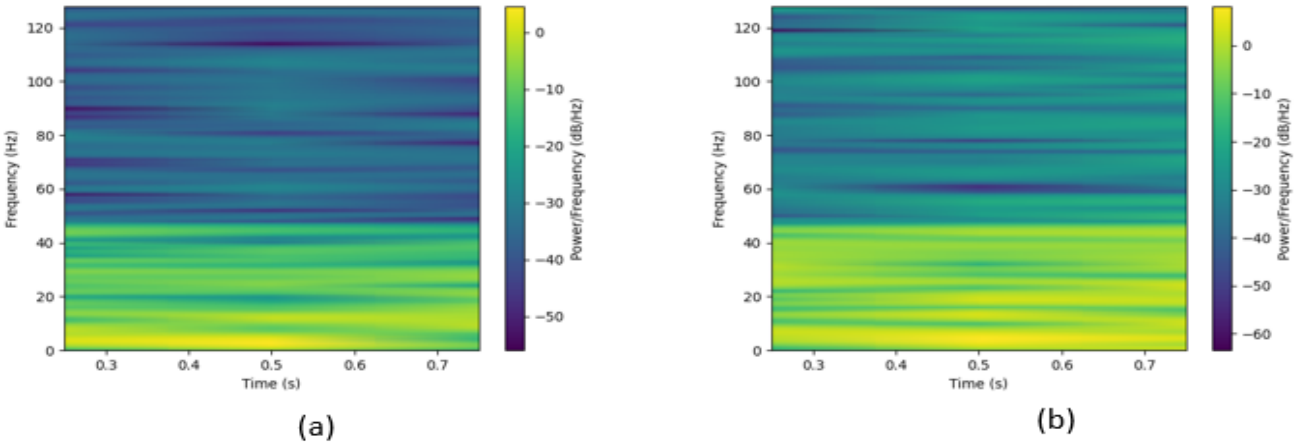


Figure 3. **Spectrogram image of EEG signal. (a) alcoholism (b) normal.**

The success of deep learning in image classification has influenced the preference for deep learning methods in artificial intelligence technologies in recent years. Unlike traditional machine learning methods, the ability of deep learning methods to extract features directly from images through deep neural networks has been effective in the preference of deep learning methods in our study. ResNet50, Xception, and custom CNN models, which are among the most preferred deep learning models in image classification, were used.

The hyper parameters given in Table 1 were selected in the training process of the deep learning models used in the study.

Table 1. **Hyperparameters used in training the models.**

Parameter	Parameter Selection
Initial Learning Rate	1
Rho(ρ)	0.95
Epochs	120
Cluster Size	5400

Learning rate shows how fast the network parameters are updated. In case of using adaptive optimization algorithms, this value is learned automatically during learning and is constantly changing. The constant ρ value is a measure used in updating the parameters in the backpropagation phase of the network. Epoch number is the number of iterations. Cluster size shows the number of training data processed in a single step in an epoch during the training phase.

TABLE 2. **Classification results of EEG spectrogram images**

Algorithm	Precision	Recall	F1-score	Accuracy
Resnet50	100%	86%	92%	97%
Xception	100%	100%	100%	100%
Custom CNN	99%	99%	99%	99%

In the results of the experimental study in Table 2, the classification of spectrogram images of EEG signals with deep learning models had a high success rate. Among the applied models, it is seen that the Xception model gives the most successful results in the detection of alcoholic individuals. The successful performance of the Xception model is also seen in the Complexity matrix given in Figure 4, where it has high sensitivity and accuracy in addiction detection, and there are no false positive or false negative classification rates. The deep discrimination capacity of the Xception model and the harmonious optimization of its parameters have been an important factor in the high performance of this model in classification accuracy.

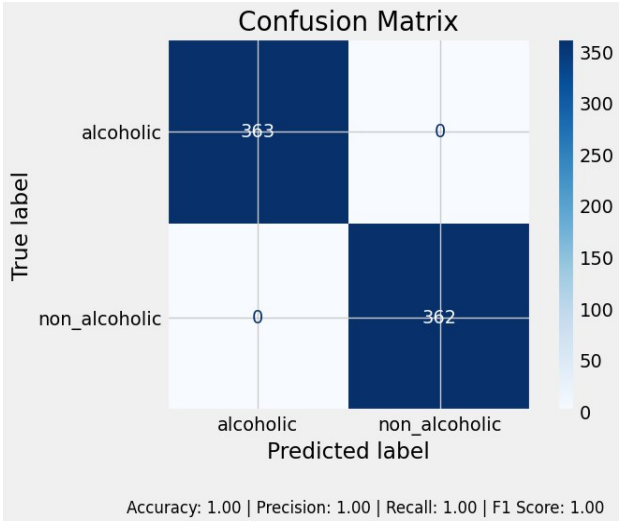


FIGURE 4. **Complexity matrix of the Xception model according to classification results.**

The box plot in Figure 5 shows the consistency of the results obtained with the 4-step K-fold method of the Xception model. The graph shows that the model offers a high and constant success rate in each fold, thus the generalization ability of the model is quite strong. No anomalies or large variances are observed in the graph, indicating the stable performance of the model.

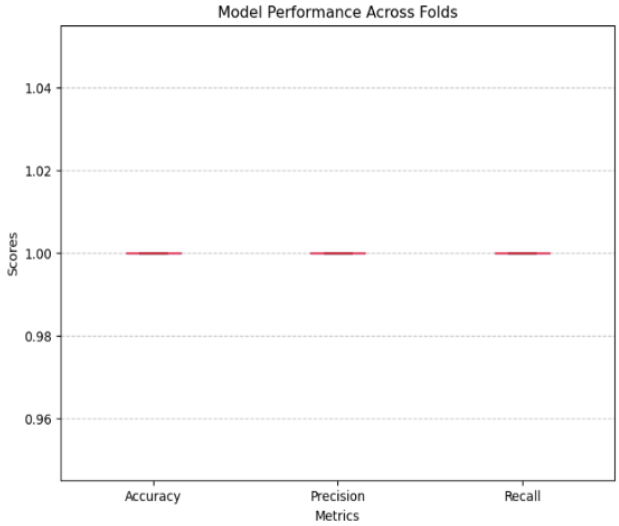


FIGURE 5. **Box plot of the results of the Xception model.**

Table 2 shows that the second most successful model for alcoholism classification is custom CNN. The Custom CNN model, which is close to the Xception model with 99% success in all metrics, correctly classified alcoholic individuals at a very high rate. As can be seen in the complexity matrix given in Figure 6, a performance loss of 1% is due to the fact that it could not classify very few test data correctly due to not learning some spectral variations sufficiently. This shows that the custom CNN model works quite effectively for alcoholism detection, but the Xception model outperforms it by a small margin due to the depth of optimization and parameter adjustments.

The ResNet50 model, another deep learning model used in the study, is seen to be lagging behind among the compared models, although it yields successful results as seen in Table 2. Although the ResNet50 model reached 97% in the accuracy rate, it only achieved 86% in the sensitivity metric. This result shows that the ResNet50 model has difficulty in correctly identifying some dependent individuals in the classification process and causes missing detections. The success rate of 92% for the F1-Score metric can be attributed to the relatively low sensitivity. The complexity matrix in Figure 7 shows that this decrease in the success of the ResNet50 method is due to the classification of non-alcoholic individuals as alcoholics.

The experimental results show that the Xception model shows the most successful performance among the deep learning models ResNet50, Xception and custom CNN models applied for classification. The experimental results also show that the Xception model has the best accuracy and reliability in classifying

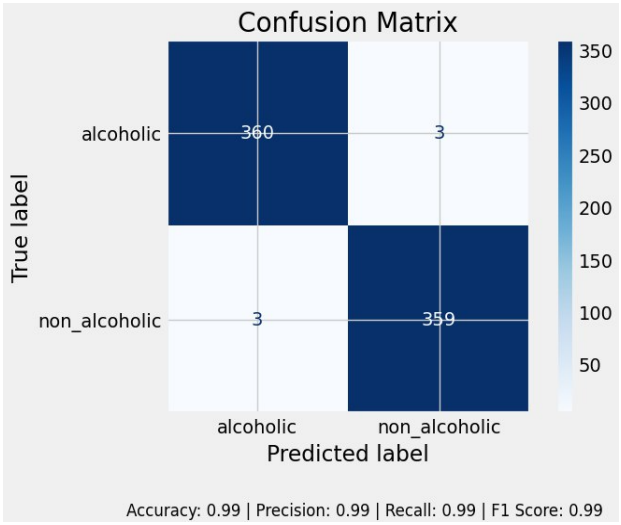


FIGURE 6. **Complexity matrix of the Custom CNN model according to classification results.**

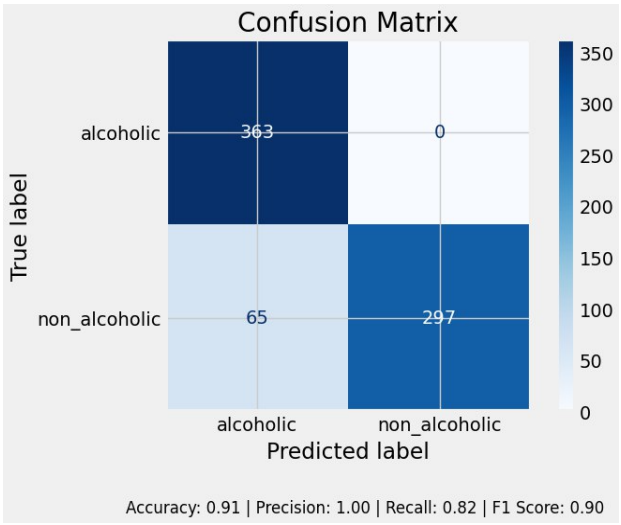


FIGURE 7. **Complexity matrix of ResNet50 model according to classification results.**

spectrogram images obtained from EEG signals for alcoholism analysis. The parameter optimization of the Xception model, its in-depth layered structure and its ability to effectively learn the intrinsic properties of the data have provided superiority over other models in classification success.

4. Conclusion

This study aims to classify EEG signals for alcoholism detection by converting their time-dependent frequency features into spectrogram images. It is shown that alcoholism can be successfully classified with deep learning models. The applied ResNet50, Xception, and custom CNN models achieve 91%, 100%, and 99% classification success, respectively. The Xception model shows superior performance in classification by achieving 100% sensitivity and 100% F1-score as well as classification accuracy. The Custom CNN model is close to the Xception model, achieving 99% sensitivity and 99% F1-score. The ResNet50 model, on the other hand, achieved 97% accuracy, but underperformed with 86% in sensitivity and 92% in F1-score. These results indicate that the parametric fit and the deep structure of the Xception model improve the classification performance. The results prove that analyzing spectrogram images of EEG signals with deep learning algorithms is a powerful tool for the detection of nervous system diseases. In future studies, since the conversion of EEG signals into spectrograms can reveal more details in the image format, this approach may be potentially useful in the diagnosis of other neurological disorders and other areas requiring signal analysis.

Declarations

- **Contribution Rate Statement:** All authors have contributed equally.
- **Conflict of Interest:** The authors report no declarations of interest.
- **Data Availability:** Dataset is available online.
- **Statement of Support and Acknowledgment:** None.

References

[1] H. Berger, Über das elektroenkephalogramm des menschen, Archiv für psychiatrie und nervenkrankheiten 87 (1) (1929) 527–570.

[2] S. Q. O. Omar, C. Tepe, Eeg sinyallerini işlemek için makine öğreniminin kullanıldığı konular üzerine bir inceleme, Bayburt Üniversitesi Fen Bilimleri Dergisi 5 (1) (2022) 124–137.

[3] G. Zhang, V. Davoodnia, A. Sepas-Moghaddam, Y. Zhang, A. Etemad, Classification of hand movements from eeg using a deep attention-based lstm network, IEEE Sensors Journal 20 (6) (2019) 3113–3122.

[4] M. Savadkoohi, T. Oladunni, L. Thompson, A machine learning approach to epileptic seizure prediction using electroencephalogram (eeg) signal, Biocybernetics and Biomedical Engineering 40 (3) (2020) 1328–1341.

[5] V. Doma, M. Pirouz, A comparative analysis of machine learning methods for emotion recognition using eeg and peripheral physiological signals, Journal of Big Data 7 (1) (2020) 18.

[6] D. Pirrone, E. Weitschek, P. Di Paolo, S. De Salvo, M. C. De Cola, Eeg signal processing and supervised machine learning to early diagnose alzheimer's disease, Applied sciences 12 (11) (2022) 5413.

[7] L. Farsi, S. Siuly, E. Kabir, H. Wang, Classification of alcoholic eeg signals using a deep learning method, IEEE Sensors Journal 21 (3) (2020) 3552–3560.

[8] M. M. Shaker, Eeg waves classifier using wavelet transform and fourier transform, brain 2 (3) (2006) 169–174.

[9] S. Lekshmi, V. Selvam, M. P. Rajasekaran, Eeg signal classification using principal component analysis and wavelet transform with neural network, in: 2014 International Conference on Communication and Signal Processing, IEEE, 2014, pp. 687–690.
[10] A. Subasi, M. I. Gursoy, Eeg signal classification using pca, ica, lda and support vector machines, Expert systems with applications 37 (12) (2010) 8659–8666.
[11] B. Arı, Alkolik ve normal eeg sinyallerinin zaman-alan tanımlayıcı analizi tabanlı otomatik sınıflandırılması, Fırat Üniversitesi Mühendislik Bilimleri Dergisi 35 (1) 291–300.
[12] B. Adhikari, A. Shrestha, S. Mishra, S. Singh, A. K. Timalsina, Eeg based directional signal classification using rnn variants, in: 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), IEEE, 2018, pp. 218–223.
[13] N. Olgun, İ. Türkoğlu, Defining materials using laser signals from long distance via deep learning, Ain Shams Engineering Journal 13 (3) (2022) 101603.
[14] V. Bajaj, Y. Guo, A. Sengur, S. Siuly, O. F. Alcin, A hybrid method based on time–frequency images for classification of alcohol and control eeg signals, Neural Computing and Applications 28 (2017) 3717–3723.
[15] S. Rajwal, S. Aggarwal, Convolutional neural network-based eeg signal analysis: A systematic review, Archives of Computational Methods in Engineering 30 (6) (2023) 3585–3615.
[16] S. Rajwal, S. Aggarwal, Evrişimsel sinir ağı tabanlı eeg sinyal analizi: Sistematik bir İnceleme, Archives of Computational Methods in Engineering 30 (2023) 3585–3615.
[17] N. Thapliyal, M. Manwal, V. Kukreja, R. Sharma, Artificial intelligence-based resnet50, xception, and vgg16 models for an efficient detection of lung cancer, in: 2024 5th International Conference for Emerging Technology (INCET), IEEE, 2024, pp. 1–5.
[18] P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation, Encyclopedia of database systems (2009) 532–538.
[19] S. Mohammed, Eeg spectrogram images, accessed: 2024-12-01 (2024).
URL https://www.kaggle.com/datasets/sayeemmohammed/eeg-spectrogram-images

Vol: 2, Issue: 2, 2024
Pages:150-171
Received: 04 November 2024
Accepted: 13 January 2025

CURRENT TRENDS IN COMPUTING
Research Article
https://doi.org/10.71074/CTC.1578917

FINE-GRAINED CLASSIFICATION OF MILITARY AIRCRAFT USING PRE-TRAINED DEEP LEARNING MODELS AND YOLO11

HASAN KARACA[1], NESRIN AYDIN ATASOY[2*]

[1] *The Institute of Graduate Programs, Computer Engineering Department, Karabük University, 78050, Karabük, Türkiye*
[2] *Computer Engineering Department, Karabük University, 78050, Karabük, Türkiye*

ABSTRACT. **This research examines the potential of pre-trained deep learning models for the fine-grained classification of military aircraft, to achieve accurate identification and extraction of unique tail numbers. The study uses a publicly available dataset comprising 43 classes of military aircraft, with a total of 24,164 images for training and 6,042 images for testing. The performance of five distinct pre-trained convolutional neural network (CNN) architectures, including DenseNet121, MobileNetV2, ResNet50, ResNet101, and VGG19, is evaluated and compared. Furthermore, the paper examines the effectiveness of the YOLO11 model family for aircraft classification,with the YOLO11x-cls model achieving the highest accuracy of 95.9, demonstrating its superior performance. particularly emphasizing the YOLO11x-cls model's superior performance. The study analyses the training results and confusion matrix of the YOLO11x-cls model, demonstrating its accuracy and ability to generalize well to unseen data. This work contributes to the advancement of AI-powered image recognition for military aviation applications, potentially improving data collection, monitoring, and analysis processes.**

1. INTRODUCTION

Aircraft are an important part of military forces when it comes to performing duties related to national defense and security. In this advanced technological area, Artificial Intelligence (AI) and Optical Character Recognition (OCR) have combined to provide an unprecedented boost in processing textual information, while persistently gaining interest in the art of classifying the aircraft themselves and reading off their wing numbers. OCR technology, being the process of extraction of written content from the visual appearance of data to textual form, has undergone a profound change by integrating AI capabilities [1]. Besides the critical role that military aircraft play in national defense, their efficient classification, and precise identification, including the reading of the wing numbers precisely, is also an essential aspect of

E-mail address: hasankaraca9163@yandex.com , nesrinaydin@karabuk.edu.tr(*).
Key words and phrases. Fine-Grained Classification, Deep Learning, Convolutional Neural Networks, YOLO11, ResNet50.

ensuring maximum operational effectiveness. AI and OCR came together to cause a revolution in processing textual information and turned out to be important tools for dealing with complications regarding military aircraft data [2].

This integration has augmented not only the accuracy and speed of OCR but also its application in various fields such as finance, healthcare, and legal services [3]. Among the innovative techniques is Layout Agnostic Alignment (LAA) [4], which solves the problem of harmonizing document layouts across different systems using OCRs. Very recently, the integration of OCR and Text-based Visual Question Answering has reached a milestone, underlining the exacting integration of both technologies seamlessly. This includes the design of specific deep neural network models for dedicated tasks like automatic license plate recognition with BLPnet. Newer efforts like Clip-OCR and Master Object (COME) [5] have moved the goalposts further in the case of representations for text and images by contrastive learning and representation learning through multi-modal feature extraction. Moreover, post-correction of the errors generated by OCR and optimization of document recognition and data extraction [6, 7] have acted to stress the emerging primacy of AI-based OCR. In particular, the recent innovative research in the unsupervised ranking of name entities from garbled OCR text [8] and OCR-based product classification in the retail sector [9] has widened the horizon for the spread of OCR applications. Finally, the statistical learning models built recently for correction of OCR errors are extremely promising in further raising accuracy and reliability [10].

Another related area in which the recent works have significantly brought about a revolution in image processing and classification related to aircraft is the damage detection in aircraft engine bore scope images using deep learning where a new benchmark was established regarding the accuracy of inspection. The Scattering Characteristics Analysis Network (SCAN) has significantly altered the way the type of aircraft classification was done in few-shot image settings where high-quality Synthetic Aperture Radar images are available. Besides, several deep learning approaches have been revealing their encouraging performance for small aircraft detection. In particular, a modified ResNet-50 architecture applied to the large-satellite image processing and the Scattering Topology Network-ST-Net significantly reduces the processing time, thereby improving object recognition in the Synthetic Aperture Radar (SAR) images. Also, deep learning frameworks have been designed in the case of automatically detecting aircraft in remotely sensed satellite images to find small objects in a complex scene. On the other hand, machine learning models using radar data from small unmanned aerial systems have opened ways for scalable traffic management and safety improvements. One of the recent end-to-end aircraft detection algorithms outperformed other methods by a margin: [11]. Aircraft classification research has focused on Principle Component Analysis (PCA) and feature fusion techniques. This has vastly improved the performance of feature classification. Specific research on identifying aircraft types by Mask R-CNN enables the accurate identification and classification of aircraft types from high-resolution satellite images: [12]. Comparative studies have finally established the efficacy of deep learning methods for object detection, further enhancing the accuracy of detection [13].

Deep learning represents a class of machine learning methodologies using neural networks that can perform complex tasks on vast volumes of data [14]. It employed several robust pre-trained models

including ResNet50 [15], ResNet101 [16], ShuffleNet [17], Xception [18], GoogLeNet [19], Inception-V3 [20], MobileNet-V2 [21], Inception-ResNet-V2 [22] and NASNet-Mobile [23]. These pre-trained models achieved a lot of success in deep learning and, hence, are considered to show excellent performance in several parts of computer vision tasks, object recognition, natural language processing [24], and other artificial intelligence applications.

Additionly Gao and Wen-jun presented the IDBO-KELM model, which remarkably enhanced the accuracy of identification of aerodynamic parameters due to the reduction of errors in transonic regions [25], hence proving its potential in precise aircraft performance analysis. Proposed the MPSA-DenseNet model, a multi-task learning model with attention mechanisms that had achieved high accuracy for complex datasets classification tasks [26]. This methodology can also be extended to aerial data analysis. Also applied the Harris Hawks Optimization algorithm in feature selection to optimize model efficiency by minimizing feature sets while retaining high predictive accuracy [27]. This is important for computationally intensive domains in aircraft classification. Further evidence is derived from health diagnostics applications [28], where Rough Neutrosophic Attribute Reduction was combined with DL-based techniques to show how deep learning frameworks are really strong in handling big and complex datasets and improving decision-making processes therein. Collectively, these works present an overview of the developments around deep learning and machine learning models, emphasizing aspects related to accuracy, efficiency, and adaptability that make them highly relevant for advanced classification tasks, including aircraft identification.

2. Methodology

2.1. **Data:**

The dataset shown in Figure 1 used in this study was obtained from Kaggle and is named ”Military Aircraft Detection Dataset” [29]. The dataset consists of 43 classes, containing 24,164 images in the training set and 6,042 in the test set. Each image belonging to a specific aircraft is stored in a folder with the name of the corresponding class. Notably, there is no dedicated test dataset; therefore, a separation has been implemented that allocates twenty percent of the images from each class as test data, while the remaining eighty percent constitutes the training data.

Table 1 provides a tabular format that has different object categories, presumably aircraft, with numerical values provided in two separate columns labeled as ”Train” and ”Validation.” The tabular structure represents one format of a dataset to train and then validate types of aircraft through machine learning.

Each row in the table represents a unique category, potentially corresponding to a specific aircraft model or type. The ”Train” column indicates the number of instances available in the training set for a given category. The training set serves as the basic data used to instruct the model to recognize and distinguish between the different categories. Conversely, the ”Validation” column denotes the number of instances present within the validation set, which serves as a means to assess the model’s accuracy when confronted with previously unseen data, thereby ensuring its ability to effectively generalize to novel instances.

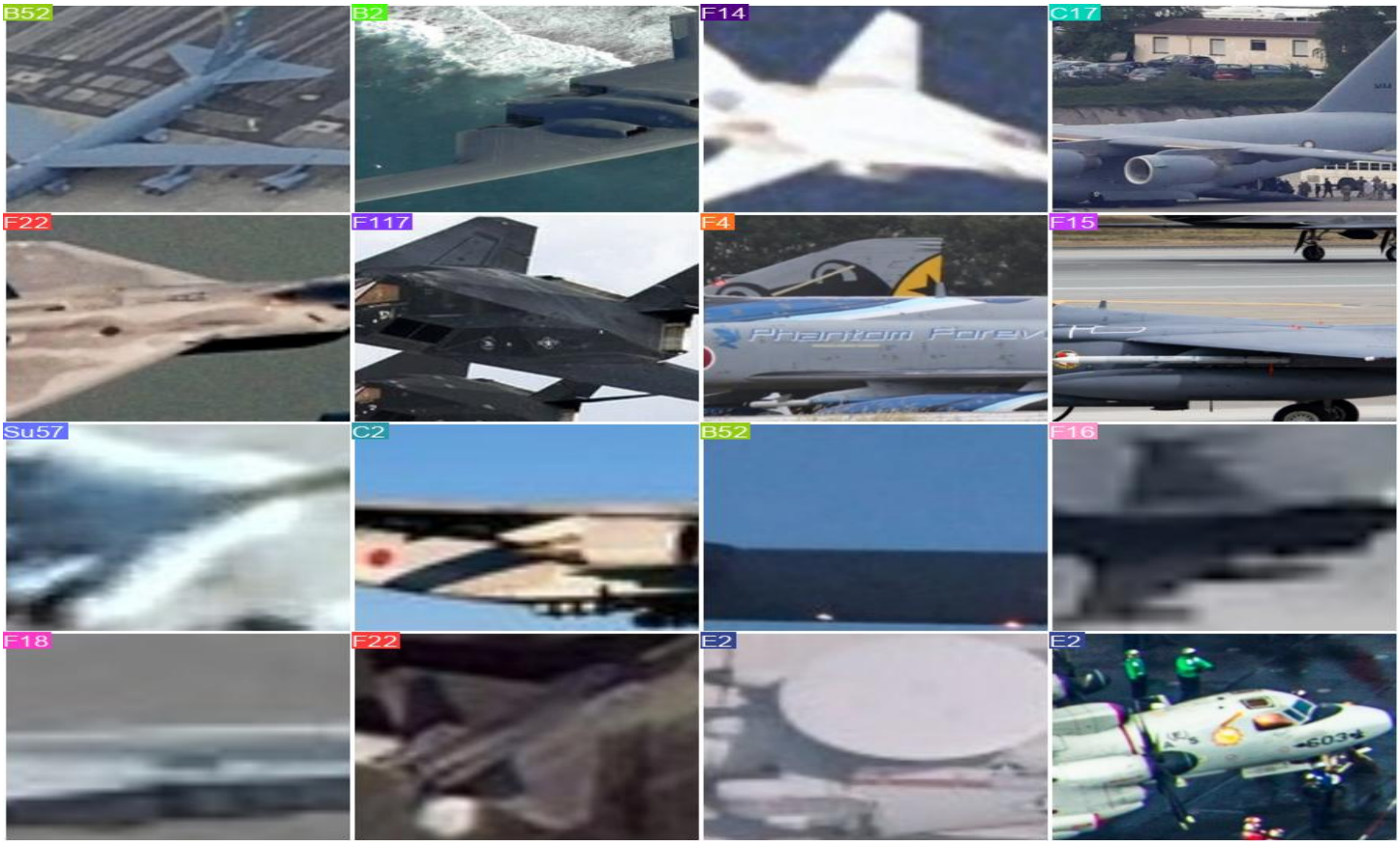


FIGURE 1. **Military aircraft class examples from different angles.**

This dataset is to be used in a k-fold cross-validation framework. K-fold cross-validation is a procedure where the data is divided into a number of K equally sized folds or subsets. Then, the model is going to be trained on k-1 of these folds, while the remaining fold will be used for its validation. It repeats K times where each fold acts once as a validation set. This technique enables the testing of the model's performance against independent datasets, thus becoming helpful when there is a lack of data resources.

The classes listed above represent some of the known model variants for aircraft models, such as F16, Rafale, and B52. Numerical values show how many images or data points within a particular category are available to train and validate the model for gaining knowledge and improving the model classification capability.

The information in Table 1 clearly elucidates how many training and validation data as shown Figure 2 each class of aircraft has. The composition of the dataset has been rich by applying data augmentation techniques as per the poor count of instances for some classes of military aircraft. Though there are classes with more than 890 instances in their training data, some classes have less than 400 instances.

TABLE 1. **Distribution of aircraft classes**

Class Name	Train	Validation	Class	Train	Validation	Class	Train	Validation
MQ9	561	140	B2	631	158	J20	562	141
JAS39	639	160	C130	611	153	F22	513	129
V22	546	137	YF23	424	106	F16	1339	335
Rafale	662	165	SR71	519	130	Vulcan	299	75
Mig31	554	139	U2	516	130	A400M	366	92
C17	666	167	C2	549	137	B1	500	126
AG600	480	119	F18	890	223	F117	284	71
F14	640	160	B52	645	161	F15	1147	287
F35	741	185	Su57	541	136	E7	146	37
Tornado	599	149	C5	583	145	XB70	137	35
EF2000	351	88	Tu95	500	124	AV8B	345	87
P3	459	115	E2	600	151	Be200	224	56
Tu160	513	129	US2	448	111	A10	550	138
F4	378	95	Su34	540	135	-	-	-
Mirage2000	585	146	RQ4	236	59	-	-	-

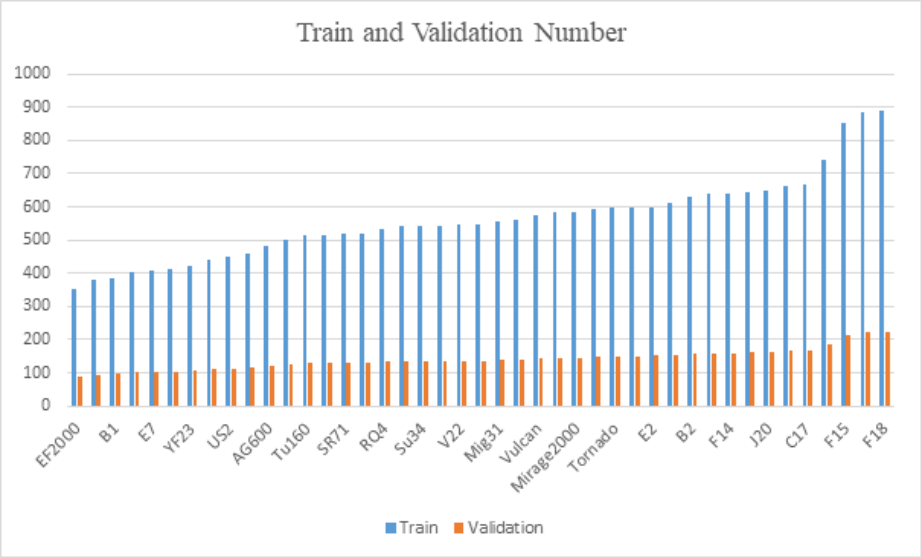


FIGURE 2. **Train and validation number of each class.**

2.2. **Methods:**

These models utilized for military aircraft classification in the current study are DenseNet121, MobileNetV2, ResNet50, ResNet101, and VGG19. These will be fine-tuned using ImageNet weights, the

weights that were developed during pre-training using the weights from the ImageNet dataset as training data. ImageNet is a dataset that has been trained for several object classification tasks and very frequently is used to fine-tune pre-trained models.

It has been shown that these models classify military aircraft with high accuracy. The goals of this study were to evaluate the performance of various deep learning architectures on tasks of military aircraft classification. Models with different architectures, such as DenseNet121, MobileNetV2, ResNet50, ResNet101, and VGG19, have been tried out to determine which is best for this particular task.

2.2.1. ***DenseNet121***. Among such famous CNN architectures is DenseNet121, which stands out because of its dense connectivity pattern. While in a traditional CNN, each layer feeds only its subsequent layer, DenseNet introduces direct connections from every layer to every other layer in a feed-forward fashion. The successive reuse of features through the dense connectivity makes it possible for gradients to propagate efficiently in the network, hence alleviating the vanishing gradient problem of deep networks. DenseNet121 explicitly contains 121 layers and is also built by stacking dense blocks composed of several convolutional layers with batch normalization and ReLU activation, followed by a transition layer for the purpose of reducing dimensionality. This architecture results in better parameter efficiency and feature extraction; hence, it is quite suitable for image classification tasks.

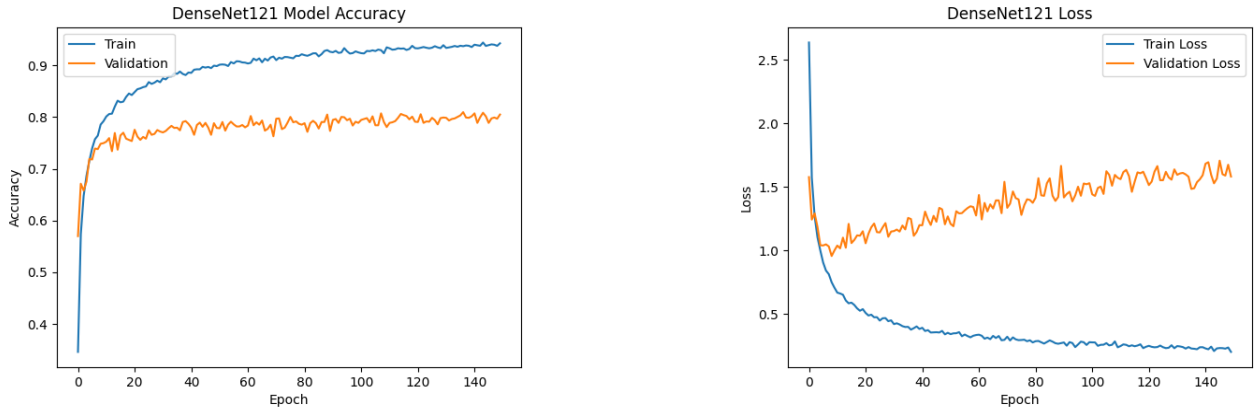
2.2.2. ***MobileNetV2***. MobileNetV2 is a light-weight CNN architecture that was proposed targeting mobile and embedded vision applications. It aimed at striking a good balance between model size and performance. Among the striking features include depthwise separable convolutions, whereby the standard convolution is split into two separate layers: depthwise convolution and pointwise convolution. This separation causes a dramatic reduction in the computational cost while keeping the capability of the network for representation intact. MobileNetV2 further uses the concept of an inverted residual with linear bottlenecks for swiftness. The bottlenecks expand the number of channels, apply depthwise convolution, and then project the features back to a lower-dimensional space to reduce the computational costs.

2.2.3. ***ResNet50***. ResNet50 is a part of the ResNet family, which initially introduced skip connections or shortcuts; this allowed the gradients to flow more directly through the network. That helped to mitigate the vanishing-gradient problem so that very deep networks could be trained. It has 50 layers and is constructed by a series of residual blocks. Considering the residual block, each of the blocks includes two convolutional layers with batch normalization and ReLU activation. Each block adds the input to the output, before passing on the result through the activation function. This enables the network to learn residual functions—that is, those where the input is close to the output. This architecture enables deeper networks to be trained, since optimization becomes considerably easier.

2.2.4. ***ResNet101***. ResNet101 is an extension of ResNet50 with 101 layers. It follows the same basic principles as ResNet50 but with a deeper architecture, which can capture more complex features and patterns in the data. The additional layers in ResNet101 allow for more refined feature extraction, potentially leading to improved performance on challenging tasks. However, deeper networks also require more computational resources and may be more prone to overfitting, so careful tuning and regularization are essential.

2.2.5. ***VGG19***. VGG19 is a variant of the VGG (Visual Geometry Group) network, known for its simplicity and effectiveness. It consists of 19 layers and is characterized by its uniform architecture, with a stack of convolutional layers followed by max-pooling layers, culminating in fully connected layers for classification. VGG networks are praised for their easy-to-understand architecture and strong performance, especially in capturing fine details in images. However, VGG19 is deeper and has more parameters than earlier VGG models, which can make it computationally expensive and prone to overfitting, especially on smaller datasets.

3. EXPERIMENTAL STUDIES

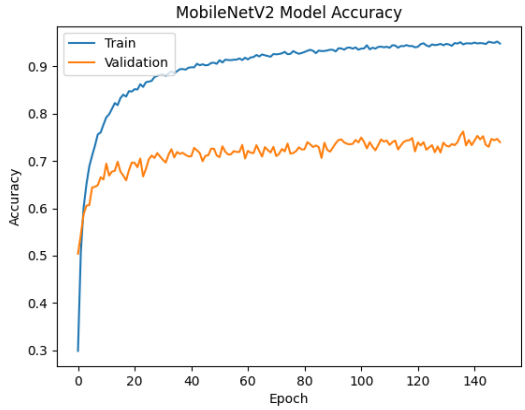


(A) Accuracy Plot (B) Loss Plot

FIGURE 3. **DenseNet121 Accuracy and Loss Plots.**

3.1. **DenseNet121.** The Figure 3 displays the training and validation accuracy and loss of a DenseNet121 model over 150 epochs. The training loss rapidly decreases in the first few epochs, from an initial value of approximately 2.5 to around 1.0 by epoch 10. It then continues to decrease gradually until it reaches a minimum of around 0.3 at epoch 150. Similarly, the validation loss also decreases in the first few epochs, from an initial value of around 2.0 to approximately 1.0 by epoch 10. However, after about 50 epochs, the value starts to increase again and reaches a maximum of approximately 1.8 at epoch 100. Subsequently, it decreases to around 1.5 at epoch 150. Optimization of the model was conducted utilizing the Adam optimizer with an initial learning rate of 0.001. The loss function applied was categorical cross-entropy, a method frequently employed in addressing multi-class classification problems.

3.2. **MobileNetV2.** Figure 4 shows the training and validation loss of a MobileNetV2 model over 150 epochs is displayed in the graph. Initially, the training loss decreases rapidly and then slows as it reaches a minimum value of approximately 0.3. The validation loss also drops quickly during the first few epochs but begins to increase after about 50 epochs, signaling potential overfitting. The provided details also

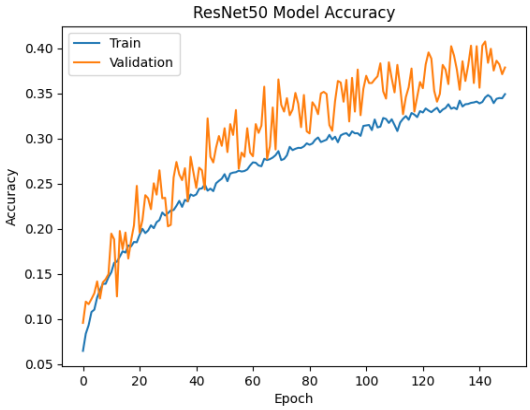


(A) Accuracy Plot

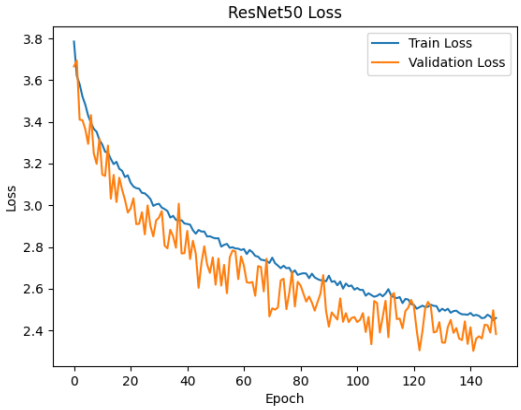
(B) Loss Plot

FIGURE 4. **MobileNetV2 Accuracy and Loss Plots.**

cover training and validation accuracy over the same period. Training accuracy starts at around 0.3 and quickly rises to about 0.95 by epoch 50. The validation accuracy begins similarly at 0.3 but climbs more slowly, reaching around 0.85 by epoch 150. The notable gap between training and validation accuracy suggests that the model is overfitting to the training data, excelling at recognizing training patterns but struggling to generalize to new data. The model was optimized using the Adam optimizer, initialized with a learning rate of 0.001. The categorical cross-entropy loss function, commonly used for multiclass classification tasks, was employed.



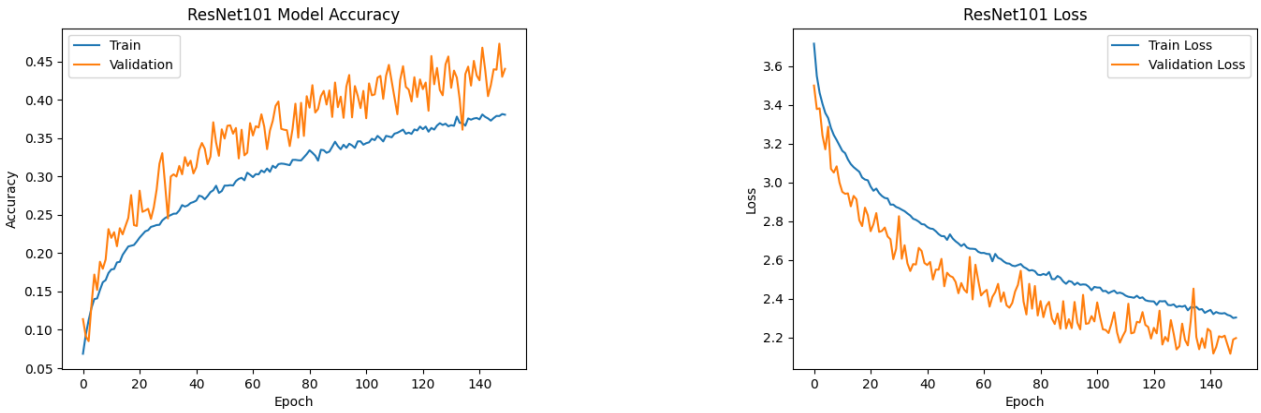
(A) Accuracy Plot



(B) Loss Plot

FIGURE 5. **ResNet50 Accuracy and Loss Plots.**

3.3. **ResNet50.** Figure 5 depicts the training and validation accuracy of ResNet50, after it has been trained for 150 epochs. The accuracy of the training starts at around 0.05 and goes all the way up to approximately 0.38. That of the validation also starts at approximately 0.05 but increases to around 0.35, though it sometimes fluctuated up and down in that process. Since the gap between training and validation accuracy is relatively small, overfitting does not happen in this model. This graph represents the training and verification loss for the ResNet50 model during 150 epochs. The training loss starts higher at approximately 3.8 and decreases linearly to about 2.4; also, the smooth validation loss starts from roughly 3.8 and goes down to about 2.4, with fluctuations during most of this training process. The relatively small difference between the training and verification loss suggests that this is not an overfitting model.



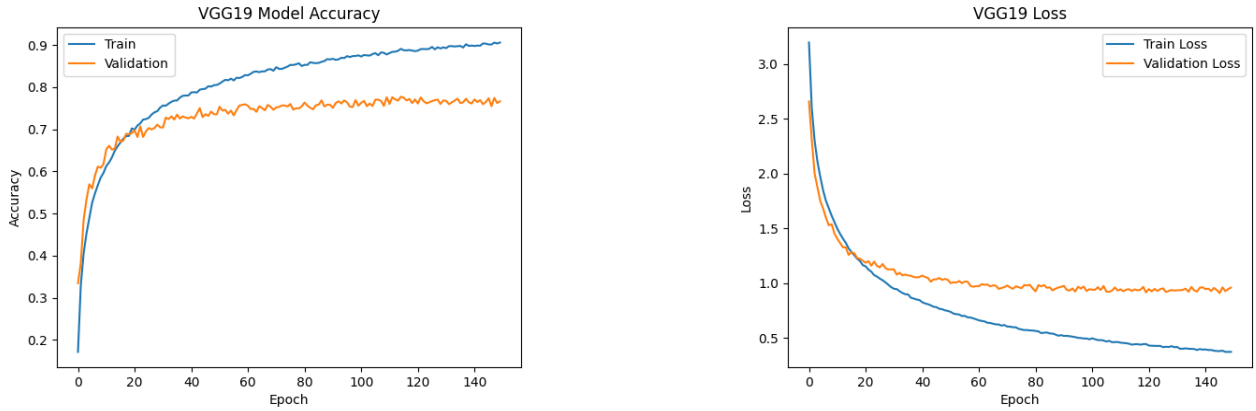
(A) Accuracy Plot (B) Loss Plot

FIGURE 6. **ResNet101 Accuracy and Loss Plots.**

3.4. **ResNet101.** These following graphs represent the accuracy and loss of the ResNet101 model concerning 150 epochs of training and validation in Figure 6. It is crystal clear that the training and validation accuracy both increase with time; however, the accuracy of training always outpaces that of validation. This would support the interpretation that the model overfits to the training dataset, learning the patterns in the training dataset perhaps too well and thereby limiting its eventual performance on new data. In similar fashion, while the training loss decreases steadily as time progresses, although validation loss is becoming less regular and skewed higher than training loss. That would mean the model has overfit to the training data. If improving the model's generalization ability, some techniques like regularization or data augmentation could be done.

3.5. **VGG19.** VGG19 was proposed by Simonyan and Zisserman in 2014 and also follows a deep CNN model architecture with stacks of convolution layers followed by fully connected layers. It is a very

simple yet effective network. VGG19 contains 19 layers, amounting to approximately 143.7 million parameters, and has achieved state-of-the-art performance for most image classification challenges.



(A) Accuracy Plot (B) Loss Plot

FIGURE 7. **VGG19 Accuracy and Loss Plots.**

Top graphic describes the accuracy curve, while the bottom graph depicts the loss curve, segregated for training and validation phases for VGG19 architecture in Figure 7.

It is observed from the accuracy figure that the model first increases in the initial epochs and then saturates at approximately 90 percent for the training set. Also, the validation accuracy increases, but it does so at a more gradual pace and stabilizes a little over 80 percent. That's good because it reflects appropriate generalization to unseen data.

This is reflected in the loss graph, wherein the training loss drops rapidly within the first few epochs before it stabilizes. The validation loss does decrease but at a much slower rate compared to the training loss, and it plateaus at a higher value; this is expected, as the validation set is not used for training.

In Figure 8 confusion matrix indicates that the model correctly classifies most images in the test set, although there are some misclassifications. There are instances where the model misclassifies images. For instance, some images of F-16s are misclassified as F-22s. However, this is a common occurrence with machine learning models. Overall, the results indicate that the VGG19 model can accurately classify images and generalize well to unseen data, making it a valuable deep learning model.

The confusion matrix reveals significant misclassifications between classes C130 and C17, with 14 occurrences of class C130 being erroneously labeled as C17 and 10 occurrences of class C17 being incorrectly identified as C130. This finding suggests that the model encounters difficulties in differentiating between these two classes, likely due to their visual similarities. A similar pattern is observed with classes F15 and F18, where 35 instances of F15 are misclassified as F18. This recurring misclassification suggests that these classes also possess visually similar attributes, posing challenges to the model's ability to accurately differentiate them.

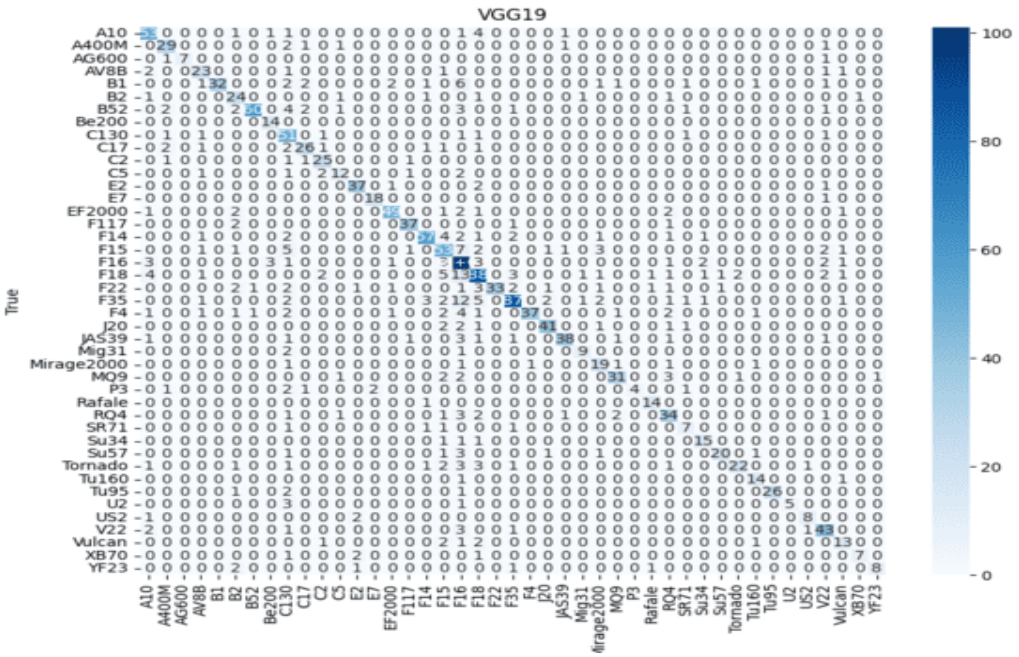


FIGURE 8. **Confusion matrix of VGG19.**

There is a notable level of misclassification between certain classes, such as F22 and F35, with observed values of 33 and 25, respectively. Though the model has some capacity to distinguish between these classes, a considerable overlap in their features leads to classification errors. Furthermore, the model incorrectly identifies Mig31 as Mig29 in 31 instances, illustrating the challenge of differentiating between similar classes. On the other hand, the model shows low misclassification rates for classes like A10 and A400M, and Tornado and Tu160, with values close to zero, indicating high accuracy for these presumably visually distinct classes. To enhance classification accuracy for often misclassified categories, several strategies can be employed. Increasing the size and diversity of training data for specific classes, such as C130, C17, F15, and F18, can improve the model's ability to distinguish among them. Moreover, employing data augmentation techniques, such as rotating, scaling, and flipping images, can help in creating a more robust training set.

3.6. **YOLO11.** The performance of the YOLO11x-cls model is evaluated in comparison with that of other YOLO11 classifier models, including YOLO11n-cls, YOLO11s-cls, YOLO11m-cls, and YOLO11l-cls. Table 2 of the paper presents a comprehensive comparison of these models based on key metrics, including accuracy, precision, recall, and F1-score.

The best performance is exhibited by the YOLO11x-cls model, which reaches class accuracy of 95.9, precision of 94.1, Recall of 95.6, and an F1-score as high as 94.8. Definitely the best results among all

TABLE 2. **YOLO11 models metrics**

Model	Accuracy	Precision	Recall	F1	Params (M)
YOLO11n-cls	0.919	0.900	0.908	0.904	1.6
YOLO11s-cls	0.935	0.918	0.933	0.925	5.5
YOLO11m-cls	0.950	0.937	0.941	0.939	10.4
YOLO11l-cls	0.955	0.936	0.952	0.944	12.9
YOLO11x-cls	**0.959**	**0.941**	**0.956**	**0.948**	**28.4**

four variants of YOLO 11. However, they are achieved at the expanse of a dramatic increase compared to the number of this model's parameters: 28.4 million, more than that of YOLO11L-cls (12.9M) by more than two times, and more than that one of YOLO 11n-cls in 17 times (1.6M). The same model, YOLO11x-cls, can serve as an example illustrative of tradeoffs between model complexity and achievements.

By comparing these models, YOLO11n-cls outperforms YOLO11scls with a moderate increase in network parameters, from 1.6M to 5.5M, yielding substantial precision improvement, from 91.9 to 93.5, and further improving the F1-score from 90.4 up to 92.5. A similar trend presents when going from YOLO11s-cls to more powerful YOLO11m-cls (increasing parameters to 10.4M), this further increases the accuracy to the level of 95.0 and F1 up to 93.9. However, this increase diminishes as the scaling factor increases. Considering YOLO11l-cls for example, it has a very limited increase of +0.4 in accuracy and +0.4 in F1-score compared to YOLO11x-cls, while the number of parameters increased 2.2 times.

Considering only the YOLO11 family of classifiers, the much larger YOLO11x-cls achieves state-of-the-art, but the two variants described here, the YOLO11m-cls and YOLO11l-cls, are offering top performance with considerably fewer parameters and hence may be excellent candidates for a practical realization. This clearly points towards at least further investigating some methodologies for optimizing such models—e.g., model pruning, quantization, knowledge distillation—for limited computing power contexts.

TABLE 3. **Military aircraft classification models metrics**

AI Model Used	**Classes**	**Methods**	**Accuracy**
Linear SVM [30]	20	CNN, data augmentation	96.8%
Artificial Neural Networks [31]	4	Sound signal processing, NN	96.2%
Feedforward NN [32]	4	Image processing, NN	97.0%
Signal Processing AI [33]	5	Radar signal feature extraction	95.0%
YOLO11x-cls **(Proposed)**	43	CNN, data augmentation	95.9%

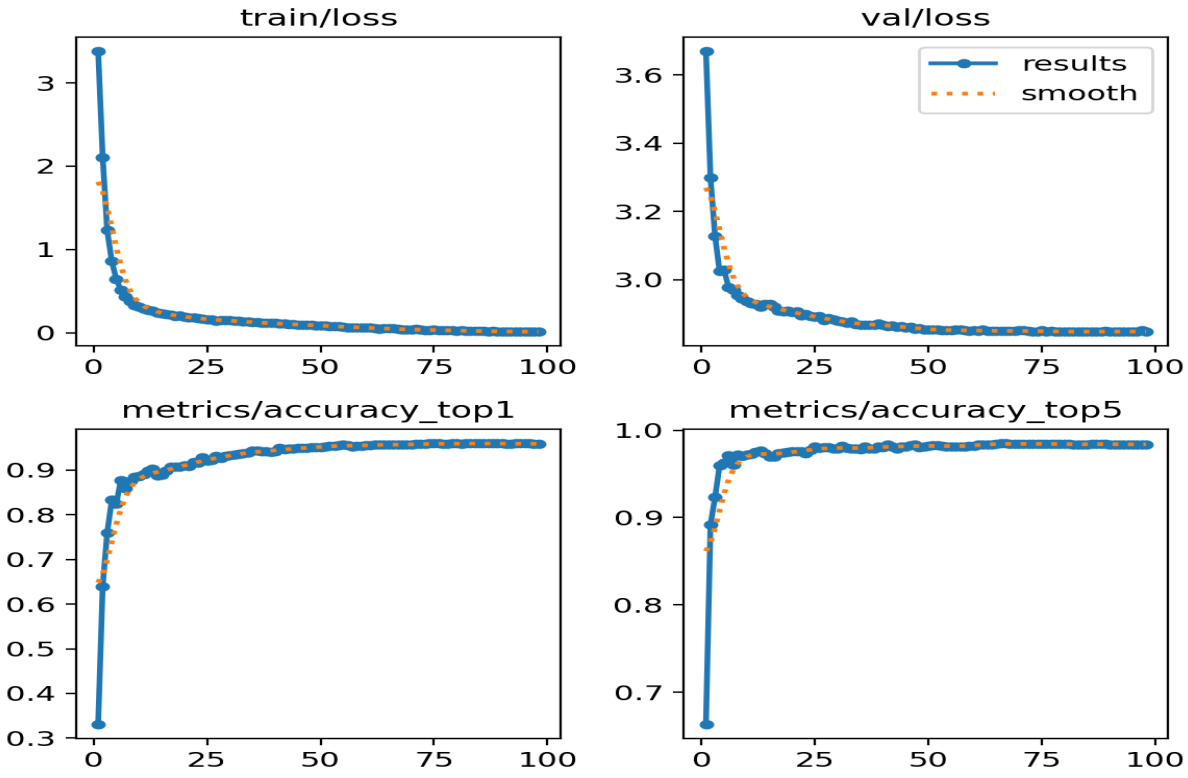


FIGURE 9. **YOLO11x-cls train metrics.**

As can be seen from the table, the YOLO11x-cls model outperforms all other models across all metrics. It exhibits the highest accuracy (0.959), precision (0.941), recall (0.956), and F1-score (0.948). These results indicate that the YOLO11x-cls model not only achieves high accuracy in classifying military aircraft but also demonstrates a remarkable ability to correctly identify positive instances (high recall) while minimizing false positives (high precision). The superior performance of the YOLO11x-cls model can be attributed to its larger size and complexity compared to other models. This enables the model to learn more complex features and patterns from the data, resulting in improved generalization and higher accuracy.

Figure 9 of the paper shows some training metrics for the YOLO11x-cls model. The focus of the above graph lies in the training loss and model accuracy over the epochs. It is observed that its training loss keeps decreasing step by step with growing epochs, a good signal for indicating this model learns well. More importantly, high training accuracy shows that the model learns from the training data and generalizes well. This would tend to suggest that the model is learning useful patterns in the training set and generalizing this by properly classifying military aircraft in the environment.

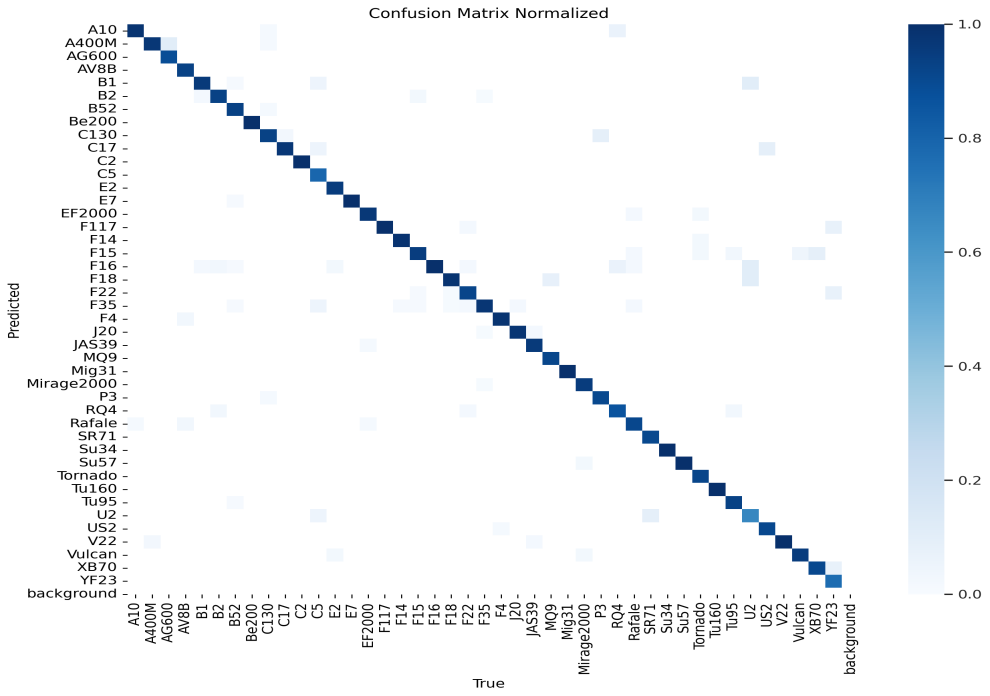


FIGURE 10. **Confusion metrics of YOLO11x-cls.**

Figure 10 is confusion matrix of the YOLO11x-cls model. From it, insight into the classification behavior of the model can be viewed. A confusion matrix depicts visually that the model has been able to classify most of the military aircraft in the test set correctly. It has, however, also revealed some misclassifications that are not out of the ordinary in any real-world machine learning application. A closer look into these misclassifications may hold a silver lining in the form of suggestions that could be obtained regarding the model's points of failure. For example, it could be very useful to study why some F-16 images were classified as F-22s with the aim of bringing improvement in the model's discrimination capability.

3.7. **Training Configuration for YOLO11.** YOLO11x classification was trained using the well-defined set of hyperparameters to have the best performance on the military airforce dataset. Training for 100 epochs, it has early stopping set with 10 epochs of patience to avoid overfitting by stopping when validation loss stops improving. Batch size 16 balances memory efficiency with gradient stability, and input image size 224x224 pixels was chosen for a good balance between computational efficiency and retaining enough spatial information. Optimization is guided by an initial learning rate (lr0) = 0.01 that is gradually decreased according to the learning rate factor (lrf) = 0.01. The momentum parameter is set to 0.937 for accelerated convergence. It stabilizes the gradient update, while weight decay equals 0.0005 and serves as a regularization factor against overfitting. During the first three epochs, the warmup policies

are used to smoothly increase the initial learning rate and momentum factor in favor of finding stability in training. Putting warmup momentum at 0.8 and warm up bias learning rate at 0.1 allows the model's bias to somewhat adapt more quickly.

The training regimen incorporates a suite of data augmentation techniques to enhance the robustness and generalization capacity of the classification model. Color jittering is applied through perturbations of hue, saturation, and value channels with respective magnitudes of 0.015, 0.7, and 0.4, introducing variations in image color characteristics. Geometric transformations, including translation up to 10% of the image dimensions and scaling by a factor up to 50%, are utilised, while rotation, shear, perspective, and vertical flips are intentionally deactivated. Horizontal flips are applied with a probability of 0.5. Mosaic augmentation, a technique that combines multiple images into a single training sample, is enabled.Furthermore, RandAugment is employed as an automated augmentation strategy to apply a diverse set of transformations. The training configuration is further augmented by the deliberate exclusion of random erasing (40% probability), cropping of the entire image during training, and the exclusion of mixup and copy-paste augmentations. These strategies are designed to diversify the training data, mitigate overfitting, and enhance the model's capacity for generalization to unseen data.

This work has designed an effective regularization that could balance the loss between the localization and classification tasks. The threshold for IoU is set to 0.7, and one can rest assured of getting high accuracy in overlap with the real results during the prediction stage. The box loss weighs 7.5; the classification loss is weighed at 0.5 while DFL takes 1.5-the loss weighs nicely manages the object detection and localization. Data augmentation involves a hue, saturation, and value adjustment in order to further generalize the model on hsv-h 0.015, hsv-s 0.7, and hsv-v is also 0.4. Additionally, the model does not use dropout, which was set to 0.0, and it is without label smoothing, while relying on weight decay and data augmentation for regularization. This combination of hyperparameters provides a robust and efficient training process, optimizing model generalization and convergence for the accurate classification of military aircraft.

3.8. **Classification Test on Real Data.** These testing results on your deep learning classifier illustrate that the model works excellent for aircraft type identification when the images are clear, frontal, or in ideal condition. The top-1 predictions have very high scores: 1.00 for F-16, C-130, and A-10, proving the strength of the model to pick out unique structural features like the configurations of wings, placement of engines, and shapes of fuselage. On challenging views of the aircraft, such as the F-22 at 0.59 confidence with alternative predictions of F-35 at 0.33, the model only shows slight ambiguity. While the classifier is robust in this matter, it struggles with classes that really do tend to look somewhat similar with the stealth factors of an aircraft taken into consideration. The meaningful secondary predictions in the top-5 results, such as F-18 with F-16 and Rafale, reflect the nuanced understanding of aircraft classes developed by the model but point to areas for improvement. To further improve the performance, more training on datasets that include diverse angles, lighting conditions, and occlusions would enhance the model's ability to tell apart visually similar aircraft.

This efficiency and applicability to real-world performance metrics further validate your classifier. The 11.8 ms/image preprocessing time just shows how well-optimized the pipeline in charge of preparing

(a) (b)

(c) (d)

FIGURE 11. **Performance evaluation of aircraft classification model: Top-5 predictions and confidence scores across diverse aircraft types [34] with processing metrics (Preprocess: 11.8 ms, Inference: 266.4 ms) on Dual Intel Xeon CPUs (2.20 GHz).**

the inputs before feeding into the model is. This gives a model inference of 266.4ms for the input shape [(1,3,224,224)] that becomes competitive in efficiency for any deep neural network and a rather complicated task of aircraft type recognition. Consequentially, the full processing time equals about 278.2 msec per image, as was noticed perfectly streamlined and fast end-to-end pipeline. This level of performance can be delivered by, but may not be limited to a hardware setup with 2 Intel Xeon CPUs-2.20 GHz since the system has multi-threading at the parallel processing level. Since this model has a very

FIGURE 12. **OCR output used without image preprocessing aircraft-1.**

high degree of confidence in addition with sub-second processing time will perform remarkably in real-life applications such as in military recognitions, auto surveillance and aircraft recognition systems. With further refinement, this classifier should be able to differentiate better between visually similar aircraft and optimize the inference time to perform well in complex and time-critical situations.

3.9. **Optical Character Recognition (OCR).** The presented image dataset demonstrates the application of OCR via the EasyOCR library on images of military aircraft tail numbers. An initial assessment shows that direct OCR application on the original images, which are characterised by varying lighting conditions and potentially low contrast, produces sub-optimal results with low confidence scores or inaccurate character recognition. This is due to the inherent challenges that OCR systems face when processing images that lack sharp transitions and distinct features. The presence of noise, blurred text and inconsistent lighting also contributes to the OCR's struggle to accurately decipher the alphanumeric sequences that make up the tail numbers. The result is an unacceptable level of recognition, indicating the need for pre-processing techniques.

The presented image dataset demonstrates the application of OCR via the EasyOCR on images of military aircraft tail numbers. An initial assessment shows that direct OCR application on the original images [35], which are characterised by varying lighting conditions and potentially low contrast, produces sub-optimal results with low confidence scores or inaccurate character recognition. This is due to the inherent challenges that OCR systems face when processing images that lack sharp transitions and distinct features. The presence of noise, blurred text and inconsistent lighting also contributes to the OCR's struggle to accurately decipher the alphanumeric sequences that make up the tail numbers. The result is an unacceptable level of recognition, indicating the need for pre-processing techniques.

FIGURE 13. **OCR result after histogram equalisation aircraft-1.**

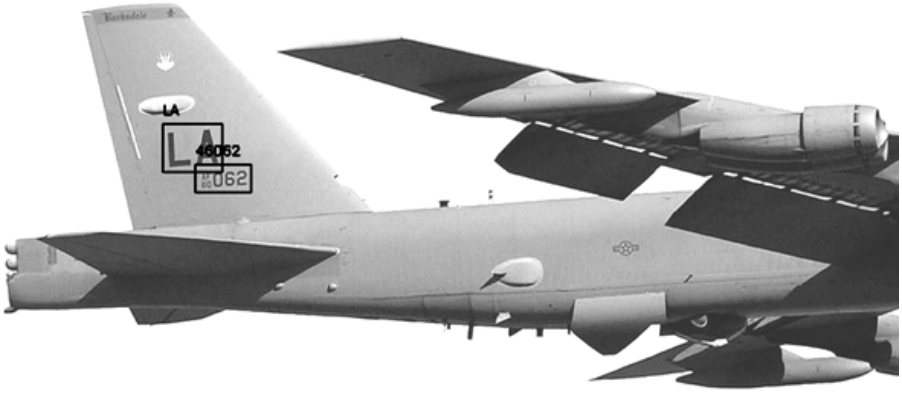


FIGURE 14. **OCR result of the image after histogram equalisation and contrast boosting process aircraft-1.**

To overcome the limitations of direct OCR, a pre-processing step using histogram equalisation is introduced. Histogram equalisation increases the contrast of the image by redistributing pixel intensities, effectively stretching the dynamic range of grey levels within the image. This adjustment significantly

FIGURE 15. **OCR output used without image preprocessing aircraft-2.**

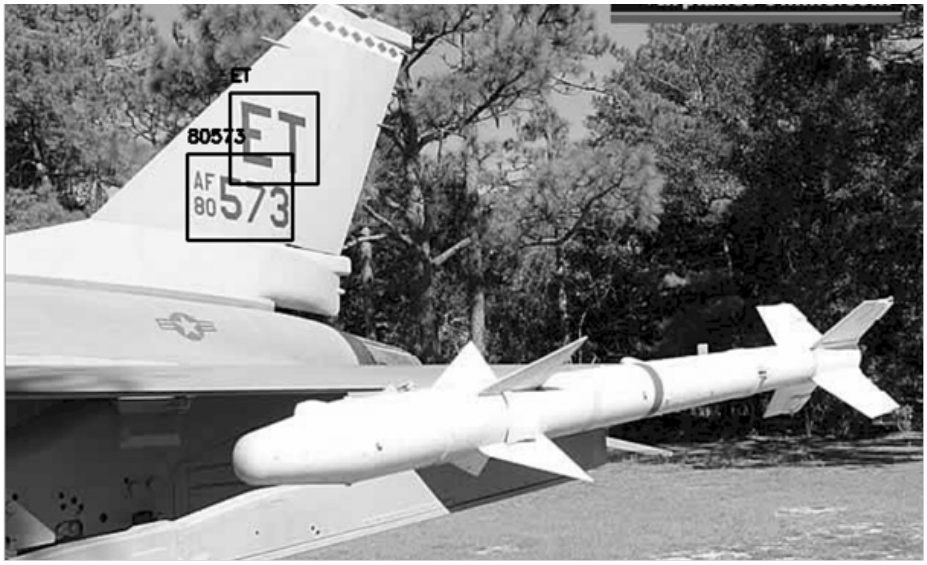


FIGURE 16. **OCR result of the image after histogram equalisation followed by contrast boosting aircraft-2.**

increases the contrast between the tail number characters and the background, improving the visual distinction of the text. After histogram equalisation, EasyOCR's performance shows a marked improvement, providing accurate character recognition with increased confidence scores. This confirms that

pre-processing techniques such as histogram equalisation play a key role in optimising the OCR process, particularly for images with difficult contrast or lighting conditions, and underlines the importance of pre-processing in text recognition applications.

4. Conclusion

This study focuses on the fine-grained classification of military aircraft using deep learning models that are pre-trained, focusing on uniquely identifying tail numbers correctly. Five CNN architectures were compared: DenseNet121, MobileNetV2, ResNet50, ResNet101, and VGG19. Whereas all the models built had a degree of success, the best performing family was the YOLO11 family, with a high accuracy achieved of 0.959 by YOLO11x-cls. The precision, recall, and F1-score were all superior with this model; hence, this classifier had great capability in recognizing the aircraft and generalizing to unseen data. This probably is due to the YOLO11x-cls being bigger in size and more complicated; hence, it can learn far more complicated features and patterns from data. The consistent decrease in training loss of the model and the high training accuracy are a witness to how effectively it gets to learn from the training data and does the right thing in classifying the aircraft correctly. More evidence can be seen with regard to the performance of the model from the confusion matrix; it performs high in classifying most of the aircraft in the test set.

In future investigations, the dataset will be greatly expanded to encompass a more diverse range of aircraft categories. The integration of Optical Character Recognition (OCR) techniques will enable the automatic extraction of tail numbers from images, thereby improving both the precision and efficiency of data processing. The research will also evaluate various pre-trained models to determine the most effective options for this task, potentially enhancing both performance and accuracy. These initiatives are essential for advancing the core technology and ensuring the solution's capability to manage a wider array of aircraft identification cases.

Moreover, the research will explore the adaptation of these models for real-time use cases, like video surveillance systems. This entails evaluating their performance in real-time scenarios and optimizing them for ongoing monitoring and swift data processing. The primary objective of this investigation is to create a highly reliable AI-driven image recognition solution specifically designed for military aviation purposes. Such advancements are anticipated to greatly enhance processes of data collection, monitoring, and analysis, ultimately bolstering national defense and security capabilities. The outcomes of these studies will form the foundation for developing a thorough and efficient system capable of meeting the stringent requirements of military operations.

Declarations

- **Contribution Rate Statement:** Hasan KARACA has conducted the study and wrote the first draft, Nesrin AYDIN ATASOY has supervised, reviewed and edited the manuscript.
- **Conflict of Interest:** The authors report no declarations of interest.
- **Data Availability:** Dataset is available online.
- **Statement of Support and Acknowledgment:** None.

REFERENCES

[1] Mori, S., H. Nishida, and H. Yamada, Optical character recognition. 1999: John Wiley & Sons, Inc.

[2] Mekonnen, I., Automated Aircraft Identification by Machine Vision. 2017.

[3] Tomovic, S., K. Pavlovic, and M. Bajceta, Aligning document layouts extracted with different OCR engines with clustering approach. Egyptian Informatics Journal, 2021. 22(3): p. 329-338.

[4] Kobayashi, Y., et al., Basic research on a handwritten note image recognition system that combines two OCRs. Procedia Computer Science, 2021. 192: p. 2596-2605.

[5] Zeng, G., et al., Beyond OCR + VQA: Towards end-to-end reading and reasoning for robust and accurate textvqa. Pattern Recognition, 2023. 138: p. 109337.

[6] Onim, M.S.H., et al., BLPnet: A new DNN model and Bengali OCR engine for Automatic Licence Plate Recognition. Array, 2022. 15: p. 100244.

[7] Lv, G., et al., COME: Clip-OCR and Master ObjEct for text image captioning. Image and Vision Computing, 2023. 136: p. 104751.

[8] Imam, N.H., V.G. Vassilakis, and D. Kolovos, OCR post-correction for detecting adversarial text images. Journal of Information Security and Applications, 2022. 66: p. 103170.

[9] Irimia, C., et al., Official Document Identification and Data Extraction using Templates and OCR. Procedia Computer Science, 2022. 207: p. 1571-1580.

[10] Dutta, H. and A. Gupta, PNRank: Unsupervised ranking of person name entities from noisy OCR text. Decision Support Systems, 2022. 152: p. 113662.

[11] Oucheikh, R., T. Pettersson, and T. Löfström, Product verification using OCR classification and Mondrian conformal prediction. Expert Systems with Applications, 2022. 188: p. 115942.

[12] Mei, J., et al., Statistical learning for OCR error correction. Information Processing & Management, 2018. 54(6): p. 874-887.

[13] Shen, Z., et al. Deep learning based framework for automatic damage detection in aircraft engine borescope inspection. in 2019 International Conference on Computing, Networking and Communications (ICNC). 2019. IEEE.

[14] Sun, X., et al., SCAN: Scattering characteristics analysis network for few-shot aircraft classification in high-resolution SAR images. IEEE Transactions on Geoscience and Remote Sensing, 2022. 60: p. 1-17.

[15] Kiyak, E. and G. Unal, Small aircraft detection using deep learning. Aircraft Engineering and Aerospace Technology, 2021. 93(4): p. 671-681.

[16] Khan, S.N., et al. Rapid Aircraft Classification in Satellite Imagery using Fully Convolutional Residual Network. in 2020 International Conference on Emerging Trends in Smart Technologies (ICETST). 2020. IEEE.

[17] Kang, Y., et al., ST-Net: Scattering Topology Network for Aircraft Classification in High-Resolution SAR Images. IEEE Transactions on Geoscience and Remote Sensing, 2023. 61: p. 1-17.

[18] Hassan, A., et al. A deep learning framework for automatic airplane detection in remote sensing satellite images. in 2019 IEEE Aerospace Conference. 2019. IEEE.

[19] Dolph, C., et al. Aircraft Classification Using RADAR from small Unmanned Aerial Systems for Scalable Traffic Management Emergency Response Operations. in AIAA AVIATION 2021 FORUM. 2021.

[20] Chen, Z., T. Zhang, and C. Ouyang, End-to-end airplane detection using transfer learning in remote sensing images. Remote Sensing, 2018. 10(1): p. 139.

[21] Azam, F., et al., Aircraft classification based on PCA and feature fusion techniques in convolutional neural network. IEEE Access, 2021. 9: p. 161683-161694.

[22] Alshaibani, W., et al., Airplane Type Identification Based on Mask RCNN and Drone Images. arXiv preprint arXiv:2108.12811, 2021.

[23] Alganci, U., M. Soydas, and E. Sertel, Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. Remote sensing, 2020. 12(3): p. 458.

[24] LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. nature, 2015. 521(7553): p. 436-444.
[25] Gao, Z., & Yi, W. (2025). Optimizing projectile aerodynamic parameter identification of kernel extreme learning machine based on improved Dung Beetle Optimizer algorithm. Measurement, 239, 115473.
[26] Song, T., Nguyen, L. T. H., & Ta, T. V. (2025). MPSA-DenseNet: A novel deep learning model for English accent classification. Computer Speech & Language, 89, 101676.
[27] Zhang, Y., Liu, R., Wang, X., Chen, H., & Li, C. (2021). Boosted binary Harris hawks optimizer and feature selection. Engineering with Computers, 37, 3741-3770.
[28] Prakash, N. N., Rajesh, V., Namakhwa, D. L., Pande, S. D., & Ahammad, S. H. (2023). A DenseNet CNN-based liver lesion prediction and classification for future medical diagnosis. Scientific African, 20, e01629.
[29] Data Statement Dataset is available at https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset
[30] Azam, F., Rizvi, A., Khan, W. Z., Aalsalem, M. Y., Yu, H., Zikria, Y. B. (2021). Aircraft classification based on PCA and feature fusion techniques in convolutional neural network. *IEEE Access*, *9*, 161683-161694.
[31] Barbarosou, M., Paraskevas, I., Ahmed, A. (2016). Military aircrafts' classification based on their sound signature. *Aircraft Engineering and Aerospace Technology: An International Journal*, *88*(1), 66-72.
[32] Karacor, A. G., Torun, E., Abay, R. (2011). Aircraft classification using image processing techniques and artificial neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *25*(08), 1321-1335.
[33] Luo, S., Yu, J., Xi, Y., Liao, X. (2022). Aircraft target detection in remote sensing images based on improved YOLOv5. *IEEE Access*, *10*, 5184-5192.
[34] Fine-Grained Visual Classification of Aircraft, S. Maji, J. Kannala, E. Rahtu, M. Blaschko, A. Vedaldi, arXiv.org, 2013
[35] https://www.airplanes-online.com/

Vol: 2, Issue: 2, 2024
Pages:172-178
Received: 24 December 2024
Accepted: 15 January 2025

CURRENT TRENDS IN COMPUTING
Research Article
DOI: 10.71074/CTC.1606104

DATASET OF EASY SCREEN P300 SPELLER BRAINCOMPUTER INTERFACE DESIGN

ABDULLAH BILAL AYGUN[1*], AHMET RESIT KAVSAOGLU[2]

[1]*Biomedical Engineering Department, Karabük University, 78050, Karabük, Türkiye*

ABSTRACT. **A dataset has been created to support advancements in brain-computer interface (BCI) research, particularly focusing on P300 speller systems and electroencephalography (EEG) signal analysis. This dataset provides detailed EEG recordings from 30 healthy participants during offline analysis, online character recognition, and word-writing tasks. A 16-channel Brain Products V-Amp device was utilized, and data were collected via a 7×7 visual stimulus matrix designed to evoke reliable P300 responses, with stimuli presented in randomized sequences. The dataset comprises raw EEG signals, binary labels, and stimulus timing information structured to facilitate the development of innovative BCI algorithms and real-time applications. This open-access resource enables novel approaches to EEG signal classification and supports the design of adaptive P300 speller interfaces, offering a foundation for advancing assistive technologies and neuroscience research.**

1. INTRODUCTION

The P300 speller paradigm represents a prominent approach in brain-computer interface (BCI) research, offering a reliable communication method for individuals with severe motor disabilities. This study emphasizes providing a detailed dataset derived from electroencephalography (EEG) recordings, enabling the development and evaluation of novel methodologies and algorithms for BCI applications. The Brain Products V-Amp 16 Channel EEG system has structured data into three main tasks: offline analysis, online character recognition, and word-writing experiments. This dataset builds upon earlier work presented by Aygun and Kavsaoglu (2022) in in [1] and is made openly accessible to encourage replication efforts and collaborative research. By structuring the dataset with detailed annotations and compatibility with widely-used analysis platforms such as MATLAB, this resource is intended to support innovation in BCI system design and signal processing.

2. EXPERIMENTAL DESIGN

EEG data have been collected to address three distinct objectives: offline analysis, online monitoring, and word-writing tasks. Each session involved a stimulus matrix with 7 rows and 7 columns, where the stimuli flashed in a randomized order. Each stimulus was presented 15 times, consisting of a 100ms ”on”

E-mail address: abilalaygun@karabuk.edu.tr (*), kavsaoglu@karabuk.edu.tr.
Key words and phrases. Brain-computer interface (BCI), P300 speller, EEG..

period followed by a 75ms ”off” period. Data were recorded for 700 milliseconds after the initiation of each flash, resulting in 210 individual stimulus events per session. Each session was treated as an independent trial.

During the offline analysis phase, EEG data were recorded from 30 participants across 25 sessions each, except for one participant who completed 24 sessions. Participants were instructed to focus on different regions of the visual stimulus matrix, as depicted in Figure 1, ensuring that any part of the matrix could serve as the source of the stimulus. This approach enabled robust and comprehensive data acquisition for developing a classification model.

FIGURE 1. **Focused offline characters in [1].**

For the online analysis, EEG data were recorded in a single session for each character recognition task. In the online analysis, 30 characters were focused on. The online analysis data includes 30 sessions and the corresponding label data.

In the word-writing application, participants could complete the entire word by matching one of the characters from E1 to E20 with the corresponding index number of the desired word after detecting the first character. These characters, as depicted in Figure 2, were displayed in a sequence ranging from 1 to 20 on the side of the interface. This approach enabled participants to write long words by detecting only two characters. The data were structured accordingly. The data were structured accordingly. Table 1 provides the list of characters that each participant focused on during the task. S1–S30 represents the numbering of participants. The session numbers are also specified in the table.

Participants engaged in the online sessions by focusing on their desired characters in specific areas of the stimulus matrix, such as the lower right, lower left, upper right, upper right, upper left, and center regions.

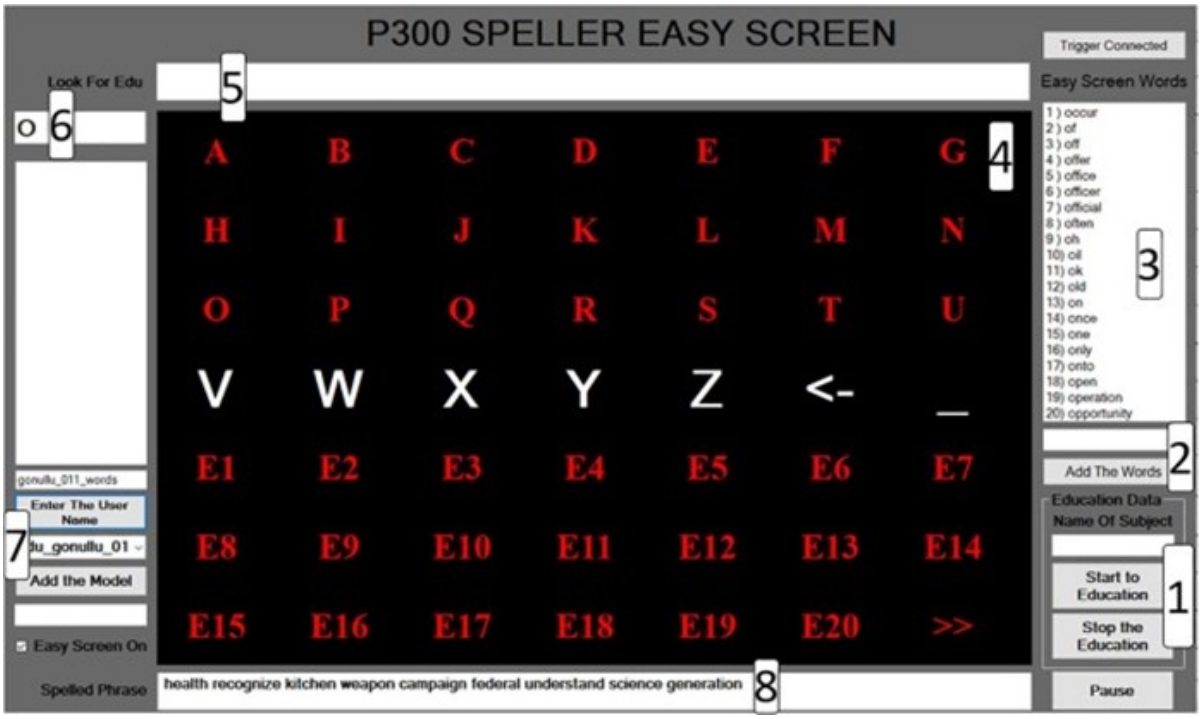


FIGURE 2. **These characters, as depicted in Figure 2, were displayed in a sequence ranging from 1 to 20 on the side of the interface [1].**

3. DATASET DESCRIPTION

The dataset is structured into three main categories:

- Offline Data: Includes raw EEG signals, binary labels indicating the presence or absence of P300 responses, and the timing of stimulus flashes.
- Online Characters: Contains single-session data specifically for individual character recognition experiments.
- Online Words: Comprises multi-session data for word-writing tasks, segmented into trials corresponding to each stimulus presentation.

All data files are provided in '.mat' format, ensuring compatibility with MATLAB and various other analysis platforms. Comprehensive annotations are included with each file to facilitate straightforward data interpretation and usage. Details of preprocessing steps and file naming conventions are also provided to enhance transparency.

This application was tested on 30 participants, 11 healthy females, and 19 healthy males, as shown in Table 2. Among the participants, 11 had mild farsightedness and wore glasses, while the remaining 19 participants had no vision impairments. Based on the information provided by the participants, no chronic, mental, or psychological disorders were identified.

TABLE 1. **The table indicates which character was focused on during each session**

Session Num	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
1	A	B	R	A	B	A	H	A	A	D
2	E16	E17	E14	E16	E4	E20	E11	E16	E7	E2
3	E	E	T	B	D	D	R	S	S	B
4	E9	E16	E10	E3	E2	E10	E18	E6	E14	E16
5	G	W	E	C	A	E	K	D	D	F
6	E13	E10	E18	E13	E20	E6	E6	E15	E3	E7
7	I	T	K	P	S	S	W	F	B	R
8	E20	E9	E6	E7	E8	E11	E10	E3	E17	E8
9	M	H	P	T	C	B	C	E	F	H
10	E10	E11	E18	E10	E1	E15	E3	E16	E8	E18
11	W	S	M	M	Z	C	F	R	R	L
12	E4	E6	E20	E5	E3	E6	E11	E7	E9	E14
13	L	K	A	L	W	M	U	N	H	O
14	E2	E8	E2	E9	E5	E4	E2	E8	E19	E5
15	S	A	C	D	E	R	S	L	L	M
16	E8	E7	E19	E20	E11	E6	E7	E12	E15	E9
17	D	F	D	E	R	T	G	B	O	A
18	E13	E3	E13	E1	E3	E9	E5	E2	E6	E6
19	P	P	S	O	K	K	O	H	M	S
20	E3	E20	E1	E4	E15	E8	E20	E10	E10	E14
Session Num	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**	**S17**	**S18**	**S19**	**S20**
1	H	U	K	H	E	G	A	T	S	T
2	E8	E8	E6	E19	E4	E5	E7	E7	E11	E12
3	B	F	S	V	M	T	R	H	A	B
4	E12	E16	E7	E6	E10	E17	E4	E12	E8	E6
5	D	T	B	A	R	A	T	P	P	M
6	E5	E10	E11	E16	E15	E11	E10	E13	E20	E8
7	P	N	O	S	S	R	N	C	E	K
8	E18	E4	E14	E11	E13	E18	E14	E2	E4	E1
9	T	O	R	T	G	D	C	F	R	O
10	E9	E20	E18	E7	E5	E2	E9	E11	E2	E3
11	A	H	U	U	A	K	P	K	T	F
12	E16	E5	E2	E2	E16	E8	E9	E10	E11	E6
13	N	L	W	P	U	F	G	E	B	D
14	E14	E12	E10	E9	E2	E6	E12	E6	E19	E12
15	W	R	E	F	H	L	O	O	V	N
16	E10	E18	E16	E13	E11	E9	E20	E19	E7	E8
17	K	S	T	W	Z	S	J	N	D	L
18	E8	E15	E9	E18	E9	E15	E3	E1	E4	E11
19	G	V	N	M	L	V	W	S	R	V
20	E13	E6	E8	E10	E12	E4	E16	E6	E6	E8
Session Num	**S21**	**S22**	**S23**	**S24**	**S25**	**S26**	**S27**	**S28**	**S29**	**S30**
1	T	B	S	P	I	T	B	B	A	A
2	E5	E6	E12	E15	E17	E17	E6	E8	E4	E14
3	V	M	N	C	R	S	K	D	B	D
4	E3	E4	E16	E8	E7	E12	E5	E14	E9	E1
5	D	S	O	L	S	P	M	G	E	Y
6	E5	E14	E7	E7	E6	E11	E12	E9	E3	E4
7	A	F	F	M	Q	V	N	P	H	K
8	E2	E5	E6	E13	E3	E7	E9	E6	E1	E7
9	U	V	H	Z	V	Z	O	Y	K	P
10	E9	E2	E11	E11	E2	E5	E12	E7	E10	E10
11	L	H	E	I	G	H	R	H	C	L
12	E17	E11	E14	E2	E13	E13	E8	E3	E5	E9
13	G	Z	P	N	O	A	S	K	T	N
14	E13	E10	E5	E14	E5	E15	E12	E8	E12	E8
15	H	H	A	T	M	B	V	U	G	O
16	E8	E16	E11	E9	E8	E11	E5	E4	E8	E19
17	P	E	R	O	A	J	Y	Z	S	J
18	E13	E18	E9	E3	E4	E2	E8	E1	E15	E3
19	K	A	M	D	L	N	T	N	R	U
20	E6	E17	E6	E16	E9	E10	E15	E12	E16	E10

TABLE 2. **The characteristics of the participants [1]**

	Age	Height	Weight	Glasses Users
Mean	34.47 ± 9.69	172.2 ± 8.75	75.43 ± 17.62	11
Range	(18 – 61)	(160 – 198)	(48 – 137)	

The experiments were conducted over 20 days in a quiet environment, involving two individuals at a time: one participant and one experiment assistant. Visual stimuli were presented to the participants on a computer screen placed 1 meter away under moderate brightness and daylight conditions. The EEG device was also connected to the same computer, which displayed the stimuli to record brain signals. Table 1 provides the characteristics of the participants.

Brain signals were recorded using a Brain Products V-Amp 16 Channel EEG device (V-Amp, Brain Products GmbH, Gilching, Germany). Electrodes were positioned according to the International 10-20 system, as shown in Figure 2, and signals were recorded from 16 channels. The signals were filtered using a 1-12 Hz Butterworth band-pass filter and a 50 Hz Notch filter. The EEG data was digitized at 2 kHz and subjected only to filtering, with no further processing applied. The dataset is presented without downsampling to retain its original resolution.

4. APPLICATIONS AND FUTURE DIRECTIONS

Applications and Future Directions This dataset offers a comprehensive foundation for advancing brain-computer interface (BCI) technology. Key potential applications include:

- Development of adaptive speller interfaces to enable efficient and user-friendly communication systems.
- Exploration of novel machine learning algorithms, facilitating innovative approaches to EEG signal classification.
- Validation of real-time BCI systems to support the development of assistive communication technologies.

This resource promotes reproducibility, fosters innovation, and facilitates collaboration within the neuroscience community by providing open access to a well-annotated and structured dataset. Researchers are encouraged to leverage the dataset to explore new directions in BCI system design and signal processing.

5. LIMITATIONS

While this dataset offers significant potential for advancing BCI research, several limitations should be considered. The participant pool primarily comprises healthy individuals, limiting its applicability to clinical populations. Furthermore, environmental conditions, such as lighting and potential distractions during experiments, were controlled but may still influence the EEG signals. Variability in participant focus and attention could also introduce noise into the data. Future studies could address these limitations

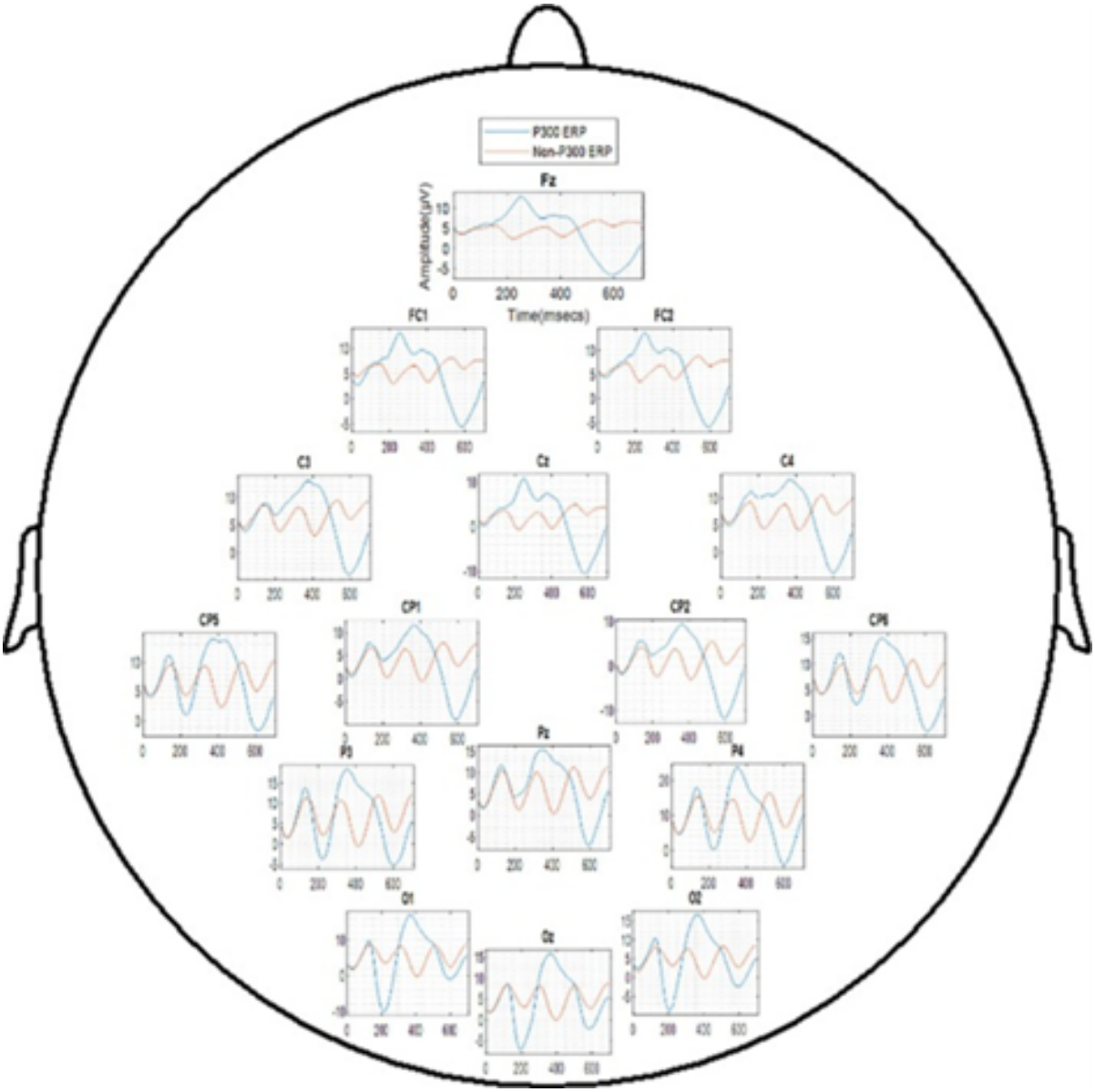


FIGURE 3. **The figure illustrates the channels from which the data were recorded [1].**

by incorporating clinical participants, expanding the dataset under varied environmental conditions, and improving signal acquisition protocols.

6. Conclusion

This dataset serves as a bridge between theoretical research and practical applications in BCI design. Making this data publicly available aims to promote open science initiatives and stimulate innovations in assistive technology. Researchers are encouraged to utilize this resource to enhance reproducibility and foster collaboration within the neuroscience community. Future studies are recommended to build upon this dataset by exploring novel adaptive algorithms, incorporating clinical populations, and validating real-time BCI applications in diverse settings.

Declarations

- **Contribution Rate Statement:** All authors have contributed equally.
- **Conflict of Interest:** The author declares no conflict of interest.
- **Data Availability:** The dataset is hosted on the Zenodo platform to ensure support scientific development and can be accessed using the link https://doi.org/10.5281/zenodo.13861638. It is distributed under the Creative Commons Attribution 4.0 license, permitting unrestricted use provided proper citation is given. Researchers are encouraged to utilize the raw EEG signals, stimulus timing files, and labels to develop machine learning models or investigate novel brain-computer interface (BCI) designs. Additional documentation is provided alongside the dataset to guide users in its application.
- **Statement of Support and Acknowledgment:** None

References

[1] A. B. Aygun, A. R. Kavsaoglu, An innovative P300 speller brain–computer interface design: Easy screen, Biomedical Signal Processing and Control, Volume 75, 2022, 103593, https://doi.org/10.1016/j.bspc.2022.103593.