ISSN 2723-6471

A Grid-search Method Approach for Hyperparameter Evaluation and Optimization on Teachable Machine Accuracy: A Case Study of Sample Size Variation

Edwin Ariesto Umbu Malahina[1,*] Gregorius Rinduh Iriane[2,] Yohanes Suban Belutowe[3,] Petrus Katemba[4,] Jimi Asmara[5,]

[1 2 3 4 5]*STIKOM Uyelindo Kupang, Jl. Perintis Kemerdekaan I, Kupang City, 85111, Indonesia*

(Received: May 27, 2024; Revised: June 27, 2024; Accepted: July 16, 2024; Available online: July 23, 2024)

Abstract

This study aims to evaluate the effectiveness of the grid-search method in hyperparameter optimization on Teachable Machine (TM) using a varying number of image samples. The hyperparameters studied include epoch (e), batch size (b), and learning rate (l). A structured grid-search method approach will be applied to test 216 hyperparameter combinations across 6 categories of sample size per class, namely 10, 25, 50, 100, 250, and 500. The results showed that the optimal combination findings were obtained based on variations in the number of samples as follows: 10 samples using e:100, b:256, l:0.001 get an accuracy range of ≥ 90%; for 25 samples using e:500, b:16, l:0.001 get an accuracy range ≥ 97%; for 50 samples using e:100, b:512, l:0.001 get an accuracy range ≥ 88%; for 100 samples using e:500, b:32, l:0.001 get an accuracy range ≥ 88%; for 250 samples using e:50, b:16, l:0.001 get an accuracy range ≥ 92%, and finally 500 samples using e:500, b:256, l:0.001 get an accuracy range ≥ 96% and on average are able to achieve 100% accuracy from the detection test results of the best value performed for each sample variation of the image object. This research provides significant contributions or benefits in finding the optimal hyperparameter configuration, minimizing overfitting, and shortening the search time for TM accuracy in image classification, particularly in human face recognition. The findings support the development of more efficient and accurate TMs and provide practical guidance for finding better hyperparameter optimization using the grid-search method approach. The results of this study have implications for improving the effectiveness and accuracy of TM models and their development in mobile web applications.

Keywords: Teachable Machine, Grid Search, Epoch, Batch Size, Learning Rate.

1. Introduction

TM is a platform designed by Google to facilitate the classification of objects, including images, audio, and poses, using open-source machine learning models that can be widely accessed and developed by users [1]. Previous research has shown that TM can achieve significant accuracy rates, up to 100% [2], [3], [4] depending on various variables that affect the performance of the object being classified. TM also adopts the Transfer Learning method, which utilizes the capabilities of Convolutional Neural Networks (CNNs) algorithms in deep learning and enables further model development by using the Tensorflow framework [5], [6]. So, the potential and service quality of TM in applying machine learning technology is very effective and efficient in the development of various object classification applications. The grid-search approach is a systematic search method used to find a combination of hyperparameters [7], [8]. With grid-search, each combination of hyperparameters is evaluated to determine the configuration that gives the best performance for the classification model on the training data in TM.

This research is different from previous studies in that it evaluates all hyperparameter values from minimum to maximum values that affect the variation of the number of samples on model performance, and uses a grid-search method approach which is a more effective and structured approach to optimizing hyperparameters compared to the trial-and-error approach often used in previous studies randomly based on previous experience. Grid-search ensures that no combinations are missed, which can lead to the identification of better hyperparameter settings and more optimized model performance based on the number of object sample variations.

***Corresponding author: Edwin Ariesto Umbu Malahina (edwinariesto@gmail.com)**
DOI: https://doi.org/10.47738/jads.v5i3.290

Hyperparameter optimization in machine learning, especially in TM, is very important as it can affect the accuracy rate, training time efficiency, and the ability of the model to generalize data. Hyperparameters such as epoch, batch size, and learning rate play a crucial role in determining the performance of object classification models. Epoch determines the number of times the entire dataset is trained, batch size affects the number of samples processed before the model is updated, and learning rate governs the speed at which the model learns. Without proper settings, the model can suffer from overfitting or underfitting, which negatively impacts the performance when applied to new data.

The main problems and challenges faced by developers in using TM services often occur in the process of very long training time, limited training data, and many combinations of hyperparameter values (epoch, batch size, and learning rate) in finding the most optimal value to train the dataset to achieve accurate and efficient detection [9], [10]. Without a systematic approach, finding the best combination of hyperparameters can be very time- and resource-consuming, where this research also aims not only to identify the ideal values of these parameters, but also to evaluate the factors that affect object detection, especially image objects with a research focus on human face images, in the use of TM. In TM, there are methods of adopting transfer learning and CNNs to help perform classification and data extraction quickly and accurately [11]. The results of this research will be able to make a significant contribution in providing optimal parameter value recommendations for future users with TM, as well as software development into the proposed mobile web service.

Given the important issues and applications of TM in image object classification, this study will use a grid-search approach and deep exploration techniques to test the model based on a varying number of face image recognition samples with 10, 25, 50, 100, 250, and 500 image samples per class, respectively. Hyperparameters tested include combinations of epoch values (10, 25, 50, 100, 250, and 500), batch sizes (16, 32, 64, 128, 256, and 512), and learning rates (0.1, 0.01, 0.001, 0.0001, 0.00001, and 0.000001). This approach aims to assess the optimal performance and ensure the TM model can minimize overfitting, ensure the efficiency of training time, and evaluate the model's ability to learn and generalize from a limited number of sample data variations. The use of varying the number of samples is expected to provide greater insight into the effect of the amount of data on model performance, as well as produce a more robust and efficient model in face image recognition.

2. Methods and Algorithms

In the research process carried out by the researcher, it involves various methodological techniques and process flows used in the research which will be explained at the following points. Among them are the exploration of human face samples, the implementation of transfer learning with CNNs, the use of a grid-search method approach for hyperparameter optimization, and the evaluation of model performance across various sample sizes.

2.1. Exploration

This research will be tested directly using four examples of human face samples in the testing process in TM, with various categories of the number of each class, namely: 10, 25, 50, 100, 250 and 500 image samples per class. Then, from each category, the number of classes will be tested with the parameter values in TM, namely epoch (10, 25, 50, 100, 250 and 500), batch size (16, 32, 64, 128, 256 and 512) and learning rate (0.1, 0.01, 0.001, 0.0001, 0.00001 and 0.000001).

2.2. Transfer Learning and CNNs

Transfer learning, as a methodology integrated in TM services, allows users to utilize pre-trained models, eliminating the need to start the learning process all over again. The technique significantly reduces the number of examples required to classify new items, contributing to time efficiency and reduced resource usage [12], [13]. In the training process, the main focus lies on customizing the last few layers of the model architecture, rather than the entire network, speeding up the overall training process. This approach is especially advantageous in the context of web browser applications, where computing resources can vary greatly depending on the user's device [14], [15]. In addition, this technique supports direct access to device sensors, easing data acquisition. The implementation of transfer learning in such an environment ensures that the application can operate efficiently, even with limited resources, while making the most of the potential of the available data for improved classification accuracy and effectiveness. The Transfer learning model architecture can be seen in figure 1.

 ISSN 2723-6471

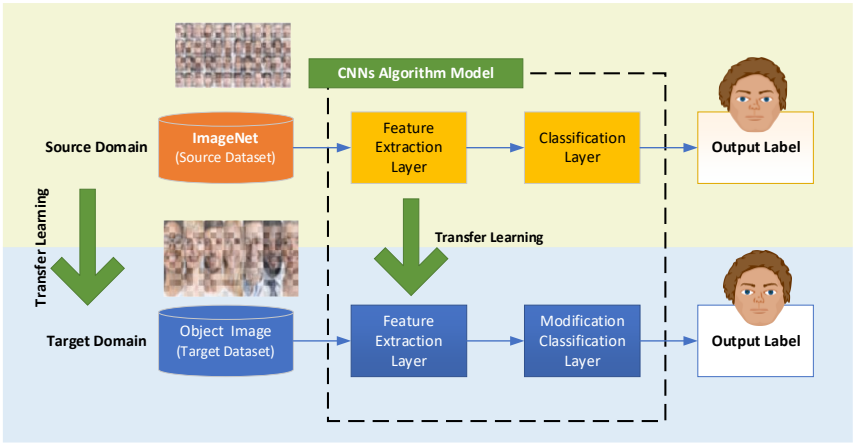


Figure 1. Transfer learning model architecture

In figure 1, which is the transfer learning architecture, TM works by using CNNs models that have been pre-trained on large datasets such as ImageNet. In this process, the initial layer of the pre-trained model, which already understands the basic features of the image, is frozen and not changed. Then, new custom layers are added and retrained (fine-tuning) using specific datasets from users, while the initial layers remain unchanged. This process allows the model to recognize the specific features of the new data quickly and efficiently. Once the retraining is complete, the model can be exported in a format that can be used in web or mobile applications, such as TensorFlow.js, so that users can develop accurate image classification models without requiring training from scratch. The benefit of transfer learning lies in the ability of the model to facilitate effective learning despite the limitations of a limited training dataset. This is in contrast to conventional machine learning approaches, where each iteration of learning requires the use of new and large datasets to achieve optimal performance [16], [17].

The transfer learning process in TM is also supported by CNNs algorithm, a simple yet powerful approach to object classification tasks [18], [19]. CNNs are structured from a series of specialized layers, including convolution layers, max pool layers, pooling layers, and fully connected layers, which play a role in the process of feature extraction and object classification with a high degree of accuracy. The architecture of convolution layers can be seen in figure 2.

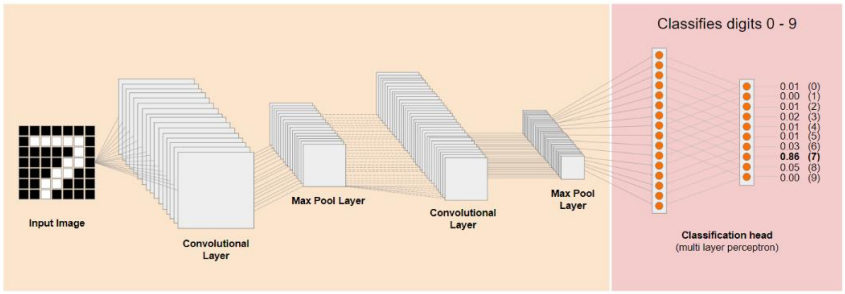


Figure 2. CNNs architecture [6]

In figure 2, CNNs have a structured architecture of a series of specialized layers, including convolution layers, max pool layers, pooling layers, and fully connected layers, which play a role in the process of feature extraction and object classification with a high degree of accuracy. Although TM has proven its usefulness in object classification by utilizing CNNs, recent developments have resulted in various object classification models that offer various advantages and efficiencies that need to be evaluated to differentiate the new algorithms. Some examples of models that have surfaced in this research include ResNet, VGGNet, AlexNet, and GoogleNet each offering a unique approach in the feature extraction and classification process.

So, it can be concluded that the respective workings of transfer learning methods and CNNs in TM are, where the transfer learning method is a technique in which a model pre-trained on a large dataset is reused for a different but related task. In TM, a pre-trained model from a large dataset such as ImageNet is used as the basis. These models have learnt various common features in images, such as edges and patterns. In TM, only the last layer of the pre-trained

 ISSN 2723-6471

model is adjusted to the new data, by adding a custom layer that will be trained with the user's specific dataset. This process is called fine-tuning, where the model learns to recognise specific features from the new dataset quickly and efficiently, saving the time and resources required compared to training the model from scratch. Whereas, CNN in TM consists of several main layers: a convolutional layer that detects basic features in the image, a pooling layer that reduces the dimension of the feature map while retaining important features, and a fully connected layer that combines all the extracted features to produce the final prediction. In TM, CNN is used to extract important features from input images and then perform classification based on user training data. By utilizing pre-trained CNN models, TM can quickly adapt to new classification tasks, allowing users to develop accurate models with limited data and short training time.

2.3. Teachable Machine Classification Process

Teachable Machine is an online platform that makes it easy for anyone to build machine learning models quickly and easily. It is designed for people from various backgrounds, such as educators, artists, students, and developers, without requiring specialized knowledge of machine learning. With TM, users can train models to recognize images, sounds, and movements with just a few clicks, without the need to write complex code. Once the model is trained, it can be directly used in various projects, such as websites, apps, and more [20], [21]. The TM workspace view can be seen in figure 3.

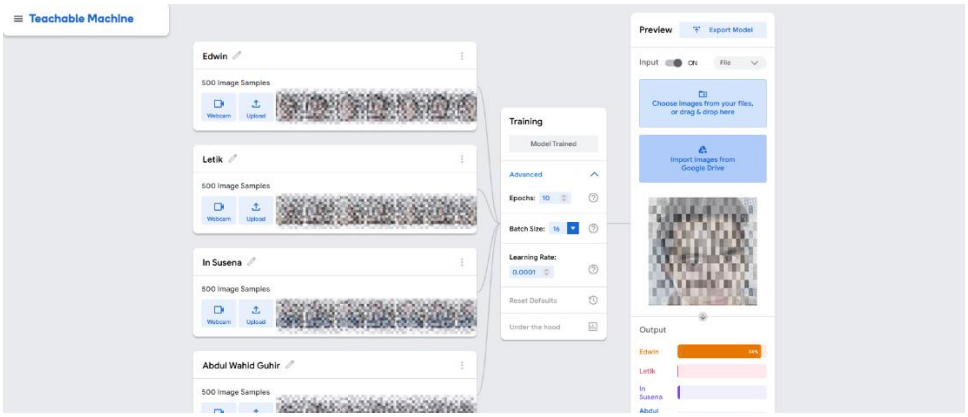


Figure 3. Teachable Machine interface

In figure 3, the image classification process using TM starts with the collection of face images for each category that the model wants to recognize, ensuring sufficient variety for accurate representation. After that, the user opens TM on the website www.teachablemachine.withgoogle.com, creating new category classes to build the image classification model. The next step is to upload the collected images into TM, grouping them by category with the labels that have been created. After the grouping is complete set the desired parameter values (epoch, batch size and learning rate), the model training process can be started with one click on the “Train Model” button, then wait until the classification process is complete. After completion, the next step is for the model performance to be evaluated using test images, if the results are not optimal, the user can add data or adjust settings and retrain the model. Once satisfied with the accuracy of the model, the next step is to export or implement the model for use in an application or website. The Object data testing architecture to export models in TM can be seen in figure 4.

 ISSN 2723-6471

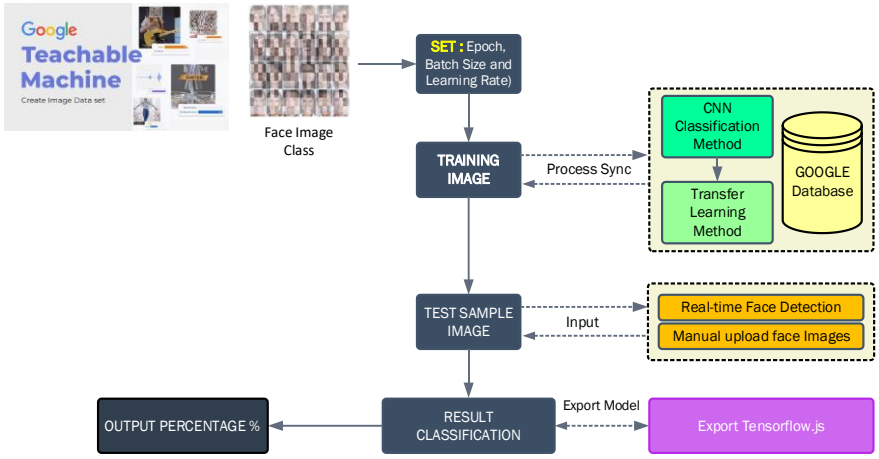


Figure 4. Object data testing architecture to export models in TM

The first process in figure 4, starts when (1) the user inputs image data through TM to be stored in the model, then it will be divided into classes in TM to be easily labeled for easy classification, (2) then setting the parameters of epoch value, batch size and learning rate for the process of searching and producing the best accuracy, (3) Next, the training sample will be trained, this process will be managed by the convolutional neural network algorithm and the transfer learning model which will then be stored in the Google database, (4) after training, users can test the image either in real-time or manually (uploading images) into TM, (5) if the identification process is complete, it will produce an image classification recognition based on the accuracy of the percentage value, the results of the classification process can also be exported to the Tensorflow. js file to be developed into an application or website (6) and display text output of the image name and percentage value.

2.4. Hyperparameter with Epoch, Batch Size and Learning Rate

In deep learning, there are three parameters that are very important in measuring the training dataset, namely epoch, learning rate, and batch size [22]. In TM, an epoch is where the model has handled all the available training datasets and then updates the parameters based on the loss calculation results. Then, batch size in TM controls the number of training samples that must be processed before the internal parameters of the model get updated. Meanwhile, the learning rate in TM regulates how often the neural network updates its knowledge in learning about the object [23]. It is very important for developers to find the optimization level of detection in TM in order to achieve effective dataset training from all three parameters [10].

The value of epochs in TM can vary, but it should be kept in mind that increasing the number can lead to overfitting, where the model becomes overly customized and specific to the training data and its performance degrades on new data. A larger number of epochs generally improves machine learning accuracy, but lengthens training time. Conversely, a smaller number of epochs can speed up learning, but may decrease accuracy.

Meanwhile, the batch size values in the provided TM range from 16, 32, 64, 128, 256 and 512. The selection of batch size affects the training time, time per epoch, and model quality. Therefore, it is necessary to experiment with various batch sizes to find the best one. Batch size determines the number of data samples processed in one training iteration, where a smaller batch size can improve accuracy but lengthen training time.

Previous studies have used the above parameters as test values for epochs from 10 to 1,000 [24], [25], [26], then for batch sizes that have been determined by TM range from 16, 32, 64, 128, 256 and 512 which do not need to be changed again. Meanwhile, the learning rate ranges from 0.000001, 0.00001; 0.0001; 0.001; 0.01; 0.1 and 1 [27], [28], [29]. These three parameters affect the training of the machine learning model, and the quality of the image data also determines the results to be achieved. So that through this research can test the number of samples and the value of these parameters

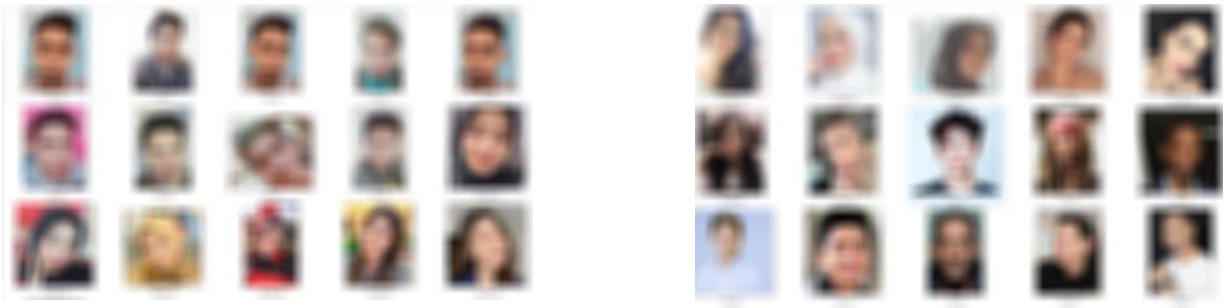
 ISSN 2723-6471

2.5. Facial Object Sample

Face recognition is one of the characteristics of humans that can be identified and recognized through the recognition of expressions, health and emotions [30], [31], [32]. The research case study conducted by the researcher uses the face sample to be tested as the object to be studied to find the accuracy value and the optimal value of the classification testing parameters using epoch, batch size and learning rate in TM. The following sample training images per class category with dimensions of 224 × 224 px can be seen in figure 5.

Figure 5. Sample training images per class category with dimensions 224 × 224 px

In figure 5, the training dataset consists of images categorized into six test groups, with each group including a varying number of samples per face category. The analysis starts with the use of the number of image samples per category, covering four face categories. Subsequent testing involved increasing the number of image samples per category to 10, 25, 50, 100, 250 and 500, incrementally until reaching 500 image samples per category. This evaluation process utilized a variety of predefined parameters, including epoch value, learning rate, and batch size, as outlined in the test table (see table 1). Implementation and testing were conducted using the TM platform to assess the effect of changing the number of image samples on the effectiveness of the model in image recognition. The following sample test images for real faces and faces of different people can be seen in figure 6.



(a) (b)

Figure 6. Sample testing: (a) original face test data image, and (b) test image with a different person's face.

In figure 6, the original face test data and different face test data are provided which will be used in the identification and testing process manually or gradually according to the amount of image test data provided. After the training test is carried out, TM will display the results of the accuracy value level in percent (%) which is used as a reference value for object detection results, and the test will see the results of the accuracy per epoch and loss per epoch values, to review the extent to which the graph provides a visual representation of the level of object detection classification perfection. The following is a flowchart image of testing sample test images can be seen in figure 7.

 ISSN 2723-6471

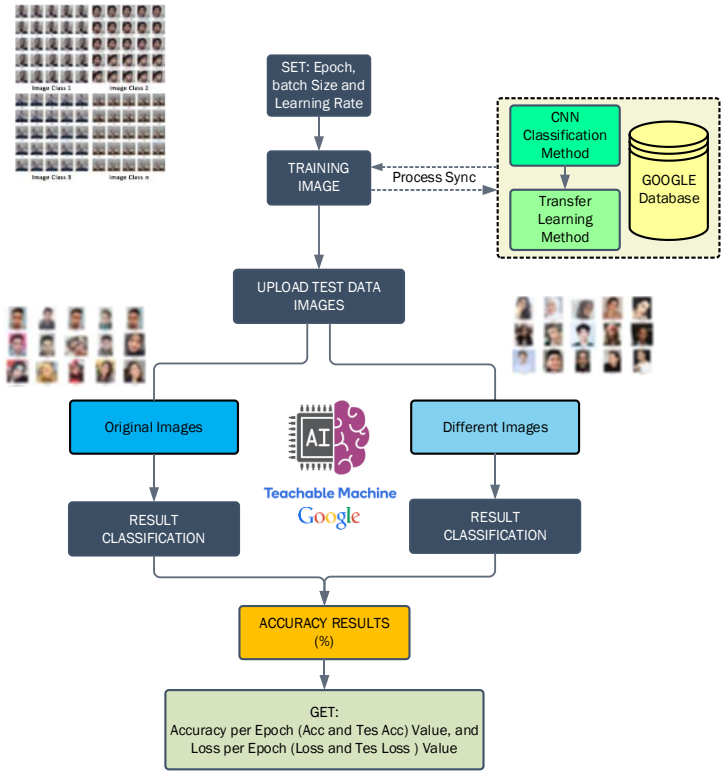


Figure 7. Flowchart of testing sample test image

In figure 7, is a flowchart of the object classification testing mechanism based on variations in the number of data samples and also the hyperparameter values tested. Where pre-processing is done by collecting data variations, resizing, normalizing and labelling objects into each class of each face object. After all face classes are entered, the hyperparameter value settings are set where the order of epoch value, batch size and learning rate will be tested based on the number of data sample variations (see table 1). After the hyperparameter value is entered, the next step is data training, this process will take time depending on the amount of data and also the epoch value to produce the percentage. After the training results are complete, the next step is to test the detection by uploading one by one test images starting from the original face image and different people's face images, where the accuracy value results are recorded manually both the accuracy value per epoch (acc and acc test) and the loss value per epoch (loss and loss test), this method will be carried out continuously until the combination of hyperparameter value testing and variation in the number of samples is tested and recorded.

3. Methodology

The grid-search approach is a systematic search method used to find the optimal combination of hyperparameters in a machine learning model [33]. In the context of TM, grid-search is applied to optimize the epoch, batch size, and learning rate parameters based on the varying number of image model samples for classification. This research defines a hyperparameter space with predefined values and creates all possible combinations of these hyperparameters. Each combination is tested by training the TM model and recording the performance results. Evaluation is done to determine the hyperparameter combination that yields the best performance. The grid-search implementation ensures that all combinations are thoroughly tested, thereby identifying the configuration that is most effective in improving TM model performance.

In the data pre-processing process, there are important steps to ensure the quality and consistency of the data used in model training. The pre-processing steps are carried out through the following process; The data collection process involves collecting face image datasets from various sources with 10, 25, 50, 100, 250, and 500 samples per class. This step is important to ensure variation in the data, which helps the model learn better and reduce bias. Having various sample sizes per class (10, 25, 50, 100, 250, and 500) helps in understanding how the size of the dataset affects the

performance of the model. After that, TM will automatically resize the images to a uniform dimension of 224 × 224 px to ensure consistency of input to the model. Uniform image size ensures that all inputs to the model have the same shape, which is important for consistency and efficiency in data processing by the CNN model. It also reduces computational complexity. After the resizing process, the normalization process continues, data normalization is an important step to ensure that the features in the data are of a similar scale, which helps in model convergence during training and prevents problems such as vanishing/exploding gradients. TM will automatically convert the image pixel values to the 0-1 range by dividing each pixel value by 255. Then the last step is labeling, where the user labels the image according to the respective class to be used in model training. Labeling is essential for supervised learning, where the model learns to associate the input image with the correct label. This allows the model to make accurate predictions on new, unseen data.

Meanwhile, data augmentation techniques are used to increase the diversity of the dataset and help the model learn from a wider variety of images. The augmentation techniques used go through the following process; In the data preprocessing process, expression variation, rotation, and image scaling are important steps applied to improve the diversity and quality of the dataset. Expression variation involves adding a variety of facial shapes in order to obtain a range of expressions under certain conditions, such as smile, sad, moody, angry, shocked, laughing, silent, and eyes closed. This step ensures that the model can accurately recognize and classify faces with various expressions. In addition, image rotation in the range of -15 to 15 degrees is performed to make the model more robust to variations in face orientation in the image. Scaling, i.e. enlarging or reducing the image size in the range of 0.8 to 1.2 times the original size, is also applied to provide a variety of image sizes that can help the model in recognizing faces of different sizes. The combination of these steps helps in enhancing the generalization ability of the model and improving the accuracy in face recognition.

To implement the grid-search method in TM, the following rules can be followed:

1) Determines the hyperparameter value

This study, we used the parameters of epoch value, batch size, and learning rate to find the optimal value for each category of the number of image samples to be tested. The epoch value determines how many times the entire dataset is used to train the model, the batch size regulates the number of samples processed before the model is updated, and the learning rate controls the speed at which the model learns. Testing different combinations of these parameters is essential to find the most effective and efficient configuration. Table 1 details the parameter values used, covering variations in the number of image samples such as 10, 25, 50, 100, 250, and 500 per class. A grid-search approach was used to systematically test all combinations of these values, so as to determine the combination that provided the best performance in terms of model accuracy and efficiency. The parameter values can be seen in table 1.

Table 1. Testing epoch parameters, batch size and learning rate based on the total number of images on TM

Number of Sample Images	Epoch (*e*)	Batch Size (*b*)	Learning rate (*l*)
10	10	16	0.1
25	25	32	0.01
50	50	64	0.001
100	100	128	0.0001
250	250	256	0.00001
500	500	512	0.000001

2) Combination of hyperparameter values

The second step is to create all hyperparameter combinations from the predefined value space. This process involves creating a Cartesian product of the hyperparameter value sets, i.e. by combining each value of each parameter (epoch, batch size, and learning rate) to form each possible combination. For example, if the epoch values used are 10, 25, and 50, the batch sizes are 16, 32, and 64, and the learning rates are 0.01, 0.001, and 0.0001, then all possible combinations of these values must be calculated and tested. This process results in a large number of combinations that must be evaluated, but it is important to ensure that every potential configuration is tested to find the most optimal hyperparameter settings. In this way, the grid-search approach ensures that no combinations are missed, which could

lead to the identification of better hyperparameter settings and more optimal model performance. These combinations will then be used in the model training process to evaluate the performance of each and determine which combination yields the best performance in terms of accuracy, training speed, and generalization ability on new data. The following hyperparameter combination formula model:

$$C = e \times b \times l \quad (1)$$

So as to get the total number of combinations to be tested, where each combination gets the amount of e (6) × b (6) × l (6) = 216 combination. With 216 combinations, will test each combination of epoch value, batch size, and learning rate thoroughly to find the best hyperparameter value.

3) Train the model for each combination

After determining all the hyperparameter combinations, the next step is to train the TM model with each combination. The process of training involves the following steps: (1) Prepare the data; ensure the dataset is ready for use in training. The data should be uploaded and processed according to TM requirements, (2) Hyperparameter set; for each hyperparameter combination (e, b, l) ∈ C, set the epoch value, batch size, and learning rate in the TM model, and (3) Train the model; run the model training with the predefined hyperparameters. This process will be repeated for each combination of (e, b, l).

4) Recording model performance results

During or after training for each hyperparameter combination, it is required to record the model performance. TM only provides the commonly used performance metrics Accuracy per epoch and Loss per epoch to record the accuracy of the model under test. Because TM avoids technical and mathematical issues (recall, precision, and F1-score) to make it easier for researchers from different backgrounds to understand, it focuses on the final test results that are specifically designed to be easily understood directly.

5) Evaluation of results

The last step is to evaluate all the recorded performance results to find the hyperparameter combination that gives the best performance. Performance analysis process; where compare the performance metrics M (e, b, l) for each combination of (e, b, l). Identify the combination that gives the highest accuracy value or the lowest loss, and then select the optimal combination; where the optimal hyperparameter combination is the one that maximizes or minimizes the selected performance metric.

The following is the formula for the maximum equation, if the performance metric used is the accuracy matrix in equation (2) below:

$$(e^*, b^*, l^*) = \arg \max_{(e,b,l) \in C} M(e, b, l) \quad (2)$$

Then, the minimal equation formula, if the loss metric that needs to be minimized in equation (3) follows:

$$(e^*, b^*, l^*) = \arg \min_{(e,b,l) \in C} M(e, b, l) \quad (3)$$

Formula description:
(e, b, l): is a combination of hyperparameters consisting of epoch (*e*), batch size (*b*), and learning rate (*l*).
C: is the space of all possible combinations of hyperparameters.
M (e, b, l): is the performance metric that we want to minimize or maximize (e.g., loss or accuracy).
arg min: denotes the argument (combination of hyperparameters) that produces the minimum value of the *M* metric.
arg max: denotes the argument (combination of hyperparameters) that yields the maximum value of the *M* metric.

By finding this optimal combination, this research can ensure that the TM model performs at its best based on the explored hyperparameter space

4. Result and Discussion

This section presents findings from the optimization of hyperparameter values using the grid-search method approach in TM. The analysis includes optimal hyperparameter combinations for various sample sizes and their impact on model performance. The main factors affecting classification accuracy, such as epoch, batch size, and learning rate, are

discussed. Results demonstrate the effectiveness of the grid-search approach in improving the accuracy and efficiency of TM, providing insights into how to minimize overfitting and improve model accuracy.

4.1. Result

The following discussion will explain the results of the classification test process based on the number of faces of 10, 25, 50, 100, 250 and 500 in each class (there are four test classes) and the epoch parameter value with a variation of test values of 10, 25, 50, 100, 250 and 500, batch size with a variation of test values of 16, 32, 64, 128, 256 and 512 and learning rate with a variation of test values of 0.1, 0.01, 0.001, 0.0001, 0.00001 and 0.000001 which all of these test results are carried out on the TM service site manually. The following are five test tables that researchers have summarized from the many data that have been tested and taken the five best value sequences from each category of the number of images. The results of testing the selection of the best accuracy level on Teachable Machine can be seen in table 2, table 3, table 4, table 5, table 6 and table 7.

Table 2. Test results of 10 image samples with a combination of epoch parameters, batch size and learning rate in Teachable Machine (ten tests selected highest to lowest)

Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image
100	256	0.0010	1	1	0.00	0.00	90-100	72-99
500	128	0.0010	1	1	0.00	0.01	89-100	60-99
25	64	0.0010	1	1	0.00	0.00	86-98	66-99
25	128	0.0010	1	1	0.00	0.02	84-99	55-99
50	32	0.0010	1	1	0.00	0.01	81-100	53-99
50	512	0.0010	1	1	0.66	0.66	80-99	56-98
25	256	0.0010	1	1	0.00	0.01	78-99	67-99
25	16	0.0010	1	1	0.00	0.00	75-99	58-96
250	16	0.0001	1	1	0.00	0.00	72-95	49-88
25	32	0.0010	1	1	0.00	0.02	67-99	56-96

Table 3. Test results of 25 image samples with a combination of epoch, batch size and learning rate parameters in Teachable Machine (ten highest to lowest selected tests)

Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image
500	16	0.001	1	1	0.00	0.00	97-100	67-99
250	256	0.001	1	1	0.00	0.00	94-100	63-99
500	64	0.001	1	1	0.00	0.00	93-100	79-97
500	128	0.001	1	1	0.00	0.00	93-100	76-99
50	16	0.001	1	1	0.00	0.00	91-100	63-98
50	64	0.001	1	1	0.00	0.00	91-100	76-98
100	256	0.001	1	1	0.00	0.00	91-100	75-98
500	512	0.001	1	1	0.00	0.00	91-100	62-99
250	32	0.001	1	1	0.00	0.00	90-100	78-98
250	64	0.001	1	1	0.00	0.00	90-100	70-98

Table 4. Test results of 50 image samples with a combination of epoch, batch size and learning rate parameters in Teachable Machine (ten highest to lowest selected tests)

Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image

 ISSN 2723-6471

100	512	0.001	1	1	0.00	0.00	88-100	85-97
100	16	0.001	1	1	0.00	0.00	86-100	80-98
100	32	0.001	1	1	0.00	0.00	85-100	77-96
250	512	0.001	1	1	0.00	0.00	85-100	87-99
50	32	0.001	1	1	0.00	0.00	84-100	83-98
500	64	0.001	1	1	0.00	0.00	83-100	84-99
250	64	0.001	1	1	0.00	0.00	81-100	87-99
250	128	0.001	1	1	0.00	0.00	80-100	81-98
250	256	0.001	1	1	0.00	0.00	80-100	87-99
500	32	0.001	1	1	0.00	0.00	80-100	88-99

Table 5. Test results of 100 image samples with a combination of epoch, batch size and learning rate parameters in Teachable Machine (ten highest to lowest selected tests)

Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image
500	32	0.00100	1	0.99	0.00	0.00	88-100	53-99
250	16	0.00100	1	1.00	0.00	0.00	87-100	55-99
500	16	0.00010	1	1.00	0.00	0.00	87-100	52-98
500	32	0.00001	1	0.99	0.00	0.00	86-98	53-84
100	16	0.00100	1	1.00	0.00	0.00	86-100	55-99
250	32	0.00100	1	0.99	0.00	0.00	86-100	53-98
500	64	0.00100	1	0.99	0.00	0.00	84-100	63-99
500	256	0.00100	1	0.99	0.00	0.00	84-100	59-96
500	512	0.00100	1	0.99	0.00	0.00	84-100	51-98
50	16	0.00100	1	1.00	0.00	0.00	83-100	62-96

Table 6. Test results of 250 image samples with a combination of epoch, batch size and learning rate parameters in Teachable Machine (ten highest to lowest selected tests)

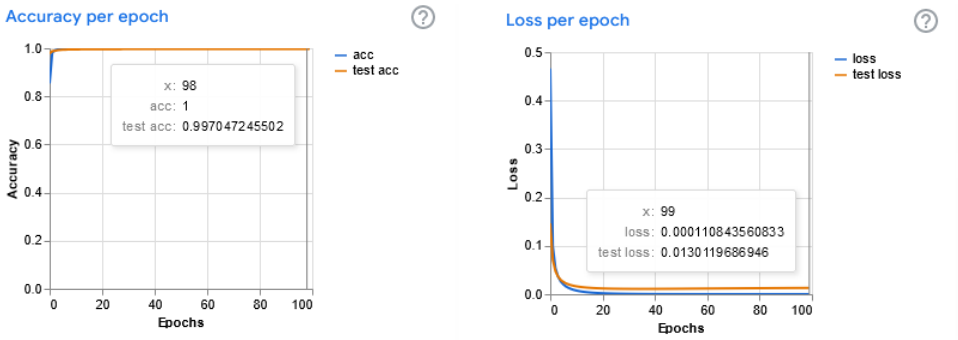
Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image
50	16	0.0010	1	1	0.00	0.00	92-100	51-99
500	16	0.0001	1	1	0.00	0.00	92-100	59-98
500	128	0.0010	1	1	0.00	0.00	89-100	54-96
50	32	0.0010	1	1	0.00	0.00	88-100	57-97
500	16	0.0010	1	1	0.00	0.00	88-100	50-94
100	128	0.0010	1	1	0.00	0.00	87-100	65-94
100	16	0.0010	1	1	0.00	0.00	86-100	59-93
100	32	0.0010	1	1	0.00	0.00	86-100	62-92
500	512	0.0010	1	1	0.00	0.00	86-100	56-93
10	16	0.0010	1	1	0.00	0.00	85-100	50-98

 ISSN 2723-6471

Table 7. Test results of 500 image samples with a combination of epoch, batch size and learning rate parameters in Teachable Machine (ten highest to lowest selected tests)

Epoch	Batch Size	Learning Rate	Accuracy Per Epoch		Loss Per Epoch		Accuracy range (%)	
			Acc	Test Acc	Loss	Test Loss	Original Image	Different Image
500	256	0.00100	1	1.00	0.00	0.00	96-100	75-99
10	32	0.00100	1	1.00	0.00	0.00	95-100	79-98
500	512	0.00100	1	0.99	0.00	0.00	95-100	60-99
25	32	0.00100	1	1.00	0.00	0.00	94-100	84-99
250	256	0.00010	1	1.00	0.00	0.00	94-100	56-97
500	256	0.00010	1	1.00	0.00	0.00	94-100	56-98
250	64	0.00010	1	1.00	0.00	0.00	93-100	54-99
500	64	0.00001	1	1.00	0.00	0.00	93-100	53-98
100	128	0.00100	1	1.00	0.00	0.00	91-100	71-99
500	512	0.00010	1	0.99	0.00	0.00	91-100	62-94

The test results in table 2, table 3, table 4, table 5, table 6 and table 7 how that in the “original image” column, the ten best data have a maximum accuracy of 100%, proving that certain hyperparameter values can produce optimal accuracy. While in the “different image” column, the test results for different people's face images have a maximum average value of 99% if there is an image identical to the original face of the face object in the test data. Previously, the test to test each variable had 216 combinations for each class and the number of test samples was 6 categories, so it had a total of 1,296 variations. Then the 10 best tests were taken based on the number of test samples (10, 25, 50, 100, 250, and 500 samples). The following is a graphical image of accuracy per epoch and loss per epoch for the number of samples 250 at an epoch value of 500, batch size 16, and learning rate 0.0001 which can be seen in figure 8.



(a) (b)

Figure 8. (a) Graphic accuracy per epoch, (b) Graphic loss per epoch

In figure 8, is a graphical display of accuracy and loss in testing on the number of samples 250 at an epoch value of 500, batch size 16, and learning rate 0.0001, with a percentage of detection accuracy in the range of 92% - 100% which will be used as the best accuracy value (see table 6). This value variation was chosen because the sample size to be used will certainly be adjusted by the developer based on the dataset owned, so that in this study the face object is not difficult enough to have so take 250 samples because the more the number of samples the better CNNs recognize objects. While the hyperparameter values chosen are epoch 500, batch size 16, and learning rate 0.0001 because it gives very good accuracy results in TM testing. With a high epoch value (500), the model has enough time to learn from the training data, although this also increases the risk of overfitting, but the small batch size (16) helps in

 ISSN 2723-6471

overcoming overfitting by increasing the variation in each learning iteration. The very small learning rate (0.0001) allows the model to update its weights in very small steps, preventing the model from jumping to the optimal solution and ensuring stable convergence. Test results show accuracies in the range of 92% to 100%, both on the original images, while images with different face variations get accuracies in the range of 59% to 98%, indicating that this combination is optimal to produce an accurate and robust model in recognizing good faces on the original test data.

Thus, developers can also select hyperparameter variables below 100% in the "different image" column in the previous test table (see table 6 second row) to be used in the next software development process, because in user development must maximize detection accuracy up to 100% which can be seen in the "original image" column (see table 6) to ensure the user's face object can be detected correctly, not as someone else's face. after that the next step is developed into a mobile web system that will be used for the software testing process. The following is the flow process of developing a mobile face detection system with Tensorflow.js framework which can be seen in figure 9.

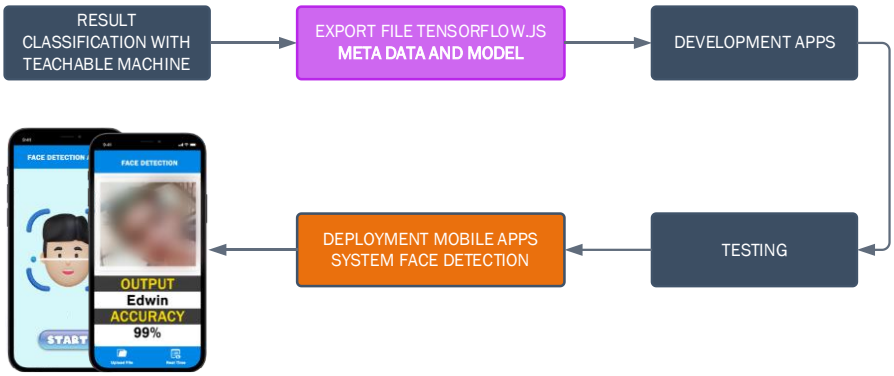


Figure 9. Development flow of mobile face detection system with Tensorflow.js framework

In figure 9, we can see the system development process which starts when the user classifies the data until finding the optimal value for the previous epoch, batch size and learning rate (see figure 7). Then the development process into software will go through the export stage in the form of source code test results, here researchers use the Tensorflow.js file to be developed into a mobile web application, which goes through the stages of development and improvement of line code, UI design, testing results and up to the publish stage. The following face detection mobile web application display page can be seen in figure 10.

(a) (b)

Figure 10. (a) The first display page, (b) The face detection page

In figure 10, where in figure 10 (a) is the initial page when the user first opens the developed mobile web application, and there is a "start" button if clicked it will direct the user to the face detection page. Then, in figure 10 (b) is the face detection page, this page has information in the form of the name of the detected face and also the percentage level of accuracy. If the user wants to upload an image file manually, they can click on the tab menu in the footer, namely "upload file", and if they want to detect faces directly, they can select the tab menu in the footer, namely "real time".

4.2. Discussion

From the results of the tests carried out, there are several factors that affect image classification during the process of analyzing and testing data and parameters carried out by researchers to be evaluated in further research using TM, as follows:

This research shows that the grid-search approach is effective in evaluating and optimizing hyperparameters in TM, resulting in optimal hyperparameter combinations for various numbers of samples tested. Although in-service TM has proven effective in object classification by utilizing CNNs, recent developments have resulted in new models such as ResNet, VGGNet, AlexNet, and GoogleNet. These models offer unique advantages and efficiencies in feature extraction and classification that need to be evaluated to distinguish the various new algorithms. Whereas, ResNet overcomes the vanishing gradient problem with residual layers, enabling very deep network training. VGGNet, with its simple architecture and 3 × 3 convolution layers, is easy to implement and highly effective in feature extraction. AlexNet popularized the use of GPUs for training and uses dropout layers to reduce overfitting. GoogleNet, with its Inception architecture, combines various filter sizes in one layer, improving efficiency and accuracy while reducing the number of parameters and overfitting.

Based on the test results and tabulation analysis of the results of classification with TM with six class variation models and parameter values, it has been concluded that the best value evaluation of each sample size is; for a sample size of 10 images, the best value is with an epoch value of 100, batch size 256 and learning rate 0.001 with an average percentage of 90% - 100%. For 25 sample images the best value is with an epoch value of 500, batch size 16 and learning rate 0.001 with an average percentage of 97% - 100%. For 50 image samples, the best value is with an epoch value of 100, batch size 512 and learning rate 0.001 with an average percentage of 88% - 100%. For 100 sample images the best value is with an epoch value of 500, batch size 32 and learning rate 0.001 with an average percentage of 88% - 100%. For 250 sample images the best value is with epoch value 50, batch size 16 and learning rate 0.001 with an average percentage of 92% - 100%, and for 500 sample images the best value is with epoch value 500, batch size 256 and learning rate 0.001 with an average percentage of 96% - 100%.

The high epoch value will be good for accuracy but will be too specific in image classification, causing overfitting of the tested image.

The worst learning rate values are 0.1 and 0.01 because the learning process is too fast so that the model does not recognize the details of the trained and tested images. So that in the testing experience in TM getting accuracy up to 100% (overfitting), because it is not able to recognize the tested image even though the different face images will be detected 100%, and the learning rate values of 0.00001 and 0.000001 also give very bad values because of the slow convergence so that the time to train the model is very long and inefficient from computing resources. So, we would recommend a learning rate value between 0.001 or 0.0001.

The batch size value has a role in improving classification, where the batch size value is very relative in this study both from the value of 16, 32, 64, 128, 256 and 512 where the model in the classification process will learn patterns more efficiently but also requires a relatively fast and long process time in the process of achieving the optimal level of accuracy.

The more the number of training image data samples for each class, the better the model will be in recognizing and learning the patterns and characteristics of the features in the image. The model will also be more accurate in recognizing image objects in newly recognized image data.

Images that are identical or have similarities will affect the process and percentage of object recognition, so face images that are close to similarity in the testing process will be recognized as real faces in the training data.

However, the large number of image samples will also affect the classification process time so that it takes time (on average 15 - 30 minutes per-training data), in this study the testing process by trying various combinations of parameters will take a long time coupled with an increase in the higher epoch value.

Using manual pre-processing techniques outside of the TM system by cropping or capturing facial objects from the top of the head to the neck can improve the accuracy of image object detection. Focusing on this area reduces the influence of noise, improves consistency, and ensures that the features of the facial area are in the main focus of detection, which is especially important for facial images.

In object detection, especially for recognition or identity verification purposes, using up-to-date facial images is essential as a person's appearance may change over time. Factors such as aging, changes in hairstyle, state of health, or weight changes can affect a person's facial features. By using the most recent facial images, detection systems can work with the most accurate and relevant data, maximizing the likelihood of successful recognition and reducing the risk of misidentification. Updated images ensure that the system reflects recent changes to facial features, thereby increasing the reliability and effectiveness of the object detection process.

Sometimes, a small percentage of tests on TM show instability in accuracy at certain test values (epoch, learning rate or batch size). Therefore, it is necessary to repeatedly train the training data again (changing the value or not changing the value of the hyperparameters in TM). After test and get the result, so that the image classification get better accuracy, increased or otherwise.

In the same experiment, the TM sometimes showed instability in detecting image objects. For example, in the test, the face of Student A, the system will mistakenly identify it as Student B. To solve this problem, it is necessary to return to training the training data repeatedly (changing the value or not changing the value of the hyperparameter in TM), so as to get better accuracy, increase or vice versa.

The data from the test and evaluation results in this study are the results of exploratory tests with data that has been determined by the researchers themselves with test parameter values (epoch, batch size and learning rate) that are available and can be modified in TM services, so it is possible that these results can be a reference, or will be different from other researchers' tests if they have different topics, different tools, different algorithms or methods, different image data quality and different test parameter values. So, it is necessary to conduct in-depth studies according to other similar research topics.

5. Conclusion

This research has shown that the grid-search method approach is effective in evaluating and optimizing hyperparameters in TM, generating and finding consistent and optimal hyperparameter combinations for various number of samples. The results show that by utilizing the grid-search method that has been used to test 216 hyperparameter combinations at various numbers of sample variations between 10, 25, 50, 100, 250, and 500 samples per class. The results show that the optimal combination is obtained based on the best testing hyperparameters with the grid-search method approach, i.e.,; 10 samples using *e:*100, *b:*256, *l:*0.001 get an accuracy range of ≥ 90%; 25 samples using *e:*500, *b:*16, *l:*0.001 get an accuracy range ranging from ≥ 97%; for 50 samples using *e:*100, *b:*512, *l:*0.001 get an accuracy range ranging from ≥ 88%; for 100 samples using *e:*500, *b:*32, *l:*0. 001 gets an accuracy range between ≥ 88%; for 250 samples using *e:*50, *b:*16, *l:*0.001 gets an accuracy range between ≥ 92%, and 500 samples using *e:*500, *b:*256, *l:*0.001 gets an accuracy range between ≥ 96%, and was on average able to achieve 100% accuracy. These findings conclude how the number of samples affects the effectiveness of different hyperparameter combinations. As such, this research contributes to finding improved model optimization in TM, minimizing overfitting, as well as providing better hyperparameter setting methods with a grid-search approach. As for the use of certain hyperparameter values and the number of samples, the author uses the number of samples 250 at the epoch value of 500, batch size 16, and learning rate 0.0001, with a percentage of detection accuracy in the percentage range of variation of this value chosen because for the sample size to be used, of course, it will be adjusted by the developer based on the dataset owned, so that in this study the face object is not difficult enough to have so take 250 samples because the more the number of samples the better CNNs recognize objects. While the hyperparameter values chosen are epoch 500, batch size 16, and learning rate 0.0001 because it gives very good accuracy results in TM testing. With a high epoch value (500), the model has enough time to learn from the training data, although this also increases the risk of overfitting, however the small batch size (16) helps in overcoming overfitting by increasing the variation in each learning iteration. The very small learning rate (0.0001) allows the model to update its weights in very small steps, preventing the model from jumping to the optimal solution and ensuring stable convergence. The test results show accuracy in the range of 92% to 100%, both on the original images, while images with different facial variations get accuracy in the range of 59% to 98% (not reaching 100%). Therefore, in future software development, the detection results can be limited to a maximum value of 100% to ensure higher accuracy and avoid detection of faces that do not belong to the user, thus demonstrating that this combination is optimal to produce an accurate and robust model in recognizing good faces on the original test data.

The practical implications of the evaluation results of this research show that the Grid-search method approach will be very helpful in finding the accuracy of the model in various other image object detection studies. By knowing the configuration that gives the best performance, future researchers can save time and resources in model development. As for future research, it can test new models such as ResNet, VGGNet, AlexNet, and GoogleNet. Whereas, ResNet overcomes the vanishing gradient problem with residual layers, enabling very deep network training. VGGNet, with its simple architecture and 3 × 3 convolution layers, is easy to implement and very effective in feature extraction. AlexNet popularized the use of GPUs for training and uses dropout layers to reduce overfitting. GoogleNet, with its Inception architecture, combines various filter sizes in one layer, improving efficiency and accuracy while reducing the number of parameters and overfitting.

The research that has been conducted, however, has raised a number of concerns for future attention in providing better results in the TM system, including; image similarity significantly affects object recognition accuracy, with similar face images tending to be recognized as the original in the training data. Image capture techniques, which target the area from the upper border of the head to the neck, are shown to improve detection accuracy by reducing noise and ensuring focus on facial object features. Using recent facial images is also crucial to reflect changes in facial features as age changes, improving detection reliability. However, testing on TM revealed instability in accuracy, emphasizing the importance of repeated training to optimize the accuracy of image classification again.

6. Declarations

6.1. Author Contributions

Conceptualization: E.A.U.M., G.R.I., Y.S.B., P.K., and J.A.; Methodology: G.R.I., P.K., and J.A.; Software: E.A.U.M., G.R.I.; Validation: E.A.U.M., Y.S.B., P.K.; Formal Analysis: E.A.U.M., G.R.I., Y.S.B.; Investigation: E.A.U.M., G.R.I.; Resources: P.K., J.A.; Data Curation: P.K., J.A.; Writing Original Draft Preparation: E.A.U.M., G.R.I., Y.S.B.; Writing Review and Editing: E.A.U.M., G.R.I., Y.S.B., P.K., and J.A.; Visualization: E.A.U.M., G.R.I.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

Source codes and research datasets are available on the author's Kaggle profile site, https://www.kaggle.com/edwinariesto. These data will always be updated by researchers in accordance with the development of data and technology from Teachable Machine.

6.3. Funding

This research was supported by funds from the LP3M STIKOM Uyelindo Kupang institution.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. A. U. Malahina, M. Saitakela, S. J. Bulan, M. I. J. Lamabelawa, and Y. S. Belutowe, "Teachable Machine: Optimization of Herbal Plant Image Classification Based on Epoch Value, Batch Size and Learning Rate," *Journal of Applied Data Sciences*, vol. 5, no. 2, pp. 532–545, May 2024, doi: 10.47738/jads.v5i2.206.

[2] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, and A. Chen, "Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification," *in Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, New York, NY, USA: ACM*, vol. 2020, no. Apr., pp. 1–8, 2020.

doi: 10.1145/3334480.3382839.

[3] E. A. U. Malahina, R. P. Hadjon, and F. Y. Bisilisin, "Teachable Machine: Real-Time Attendance of Students Based on Open Source System," *The IJICS (International Journal of Informatics and Computer Science),* vol. 6, no. 3, pp. 140–146, Nov. 2022, doi: 10.30865/ijics.v6i3.4928.

[4] D. Agustian, P. P. G. P. Pertama, P. N. Crisnapati, and P. D. Novayanti, "Implementation of Machine Learning Using Google's Teachable Machine Based on Android," *in 2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS), Makasar: IEEE,* vol. 2021, no. Oct., pp. 1–7, 2021. doi: 10.1109/ICORIS52787.2021.9649528.

[5] T. Bagby, K. Rao, and K. C. Sim, "Efficient Implementation of Recurrent Neural Network Transducer in Tensorflow," *in 2018 IEEE Spoken Language Technology Workshop (SLT), Athens: IEEE,* vol. 2018, no. Dec., pp. 506–512, 2018. doi: 10.1109/SLT.2018.8639690.

[6] P. Borisagar, Y. Agrawal, and R. Parekh, "Efficient Vehicle Accident Detection System using Tensorflow and Transfer Learning," *in 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore: IEEE,* Dec. 2018, pp. 1–6. doi: 10.1109/ICNEWS.2018.8903938

[7] H. Selcuk Nogay and H. Adeli, "Diagnostic of autism spectrum disorder based on structural brain MRI images using, grid search optimization, and convolutional neural networks," *Biomed Signal Process Control,* vol. 79, no. 2, pp. 104234–104254, Jan. 2023, doi: 10.1016/j.bspc.2022.104234.

[8] D. M. Belete and M. D. Huchaiah, "Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results," *International Journal of Computers and Applications,* vol. 44, no. 9, pp. 875–886, Sep. 2022, doi: 10.1080/1206212X.2021.1974663.

[9] J. J. N. Wong and N. Fadzly, "Development of species recognition models using Google teachable machine on shorebirds and waterbirds," *Journal of Taibah University for Science,* vol. 16, no. 1, pp. 1096–1111, Dec. 2022, doi: 10.1080/16583655.2022.2143627.

[10] S. Salim, M. M. A. Jamil, R. Ambar, W. S. W. Zaki, and S. Mohammad, "Learning Rate Optimization for Enhanced Hand Gesture Recognition using Google Teachable Machine," *in 2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE), Penang: IEEE,* vol. 2023, no. Aug., pp. 332–337, 2023. doi: 10.1109/ICCSCE58721.2023.10237148.

[11] M. K. Jha, S. Shukla, A. P. Singh, and V. Shukla, "Advancing Image Classification Through Self-teachable Machine Models and Transfer Learning," *Switzerland: Springer,* vol. 2024, no. 1, pp. 361–373, 2024. doi: 10.1007/978-3-031-56700-1_29.

[12] P. Kaur, S. Harnal, V. Gautam, M. P. Singh, and S. P. Singh, "A novel transfer deep learning method for detection and classification of plant leaf disease," *J Ambient Intell Humaniz Comput,* vol. 14, no. 9, pp. 12407–12424, Sep. 2023, doi: 10.1007/s12652-022-04331-9.

[13] J. Wang, Z. Zhang, Z. Liu, B. Han, H. Bao, and S. Ji, "Digital twin aided adversarial transfer learning method for domain adaptation fault diagnosis," *Reliab Eng Syst Saf,* vol. 234, no. 15, pp. 109152-109166, Jun. 2023, doi: 10.1016/j.ress.2023.109152.

[14] N. Chockwanich and V. Visoottiviseth, "Intrusion Detection by Deep Learning with TensorFlow," *in 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang: IEEE,* vol. 2019, no. Feb., pp. 654–659, 2019. doi: 10.23919/ICACT.2019.8701969

[15] S. Asif, M. Zhao, F. Tang, and Y. Zhu, "An enhanced deep learning method for multi-class brain tumor classification using deep transfer learning," *Multimed Tools Appl,* vol. 82, no. 20, pp. 31709–31736, Aug. 2023, doi: 10.1007/s11042-023-14828-w.

[16] F. Lingua, N. C. Coops, and V. C. Griess, "Valuing cultural ecosystem services combining deep learning and benefit transfer approach," *Ecosyst Serv,* vol. 58, no. 20, pp. 101487–101489, Dec. 2022, doi: 10.1016/j.ecoser.2022.101487.

[17] P. Liu et al., "A CNN-based transfer learning method for leakage detection of pipeline under multiple working conditions with AE signals," *Process Safety and Environmental Protection,* vol. 170, no. Feb., pp. 1161–1172, Feb. 2023, doi: 10.1016/j.psep.2022.12.070.

[18] T. Toivonen, I. Jormanainen, J. Kahila, M. Tedre, T. Valtonen, and H. Vartiainen, "Co-Designing Machine Learning Apps in K–12 With Primary School Children," *in 2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT), Tartu: IEEE*, vol. 2020, no. Jul., pp. 308–310, 2020. doi: 10.1109/ICALT49669.2020.00099.

[19] S. I. Nafisah and G. Muhammad, "Tuberculosis detection in chest radiograph using convolutional neural network architecture and explainable artificial intelligence," *Neural Comput Appl*, vol. 36, no. 1, pp. 111–131, Jan. 2024, doi: 10.1007/s00521-022-07258-6.

[20] T. L. Kurz, S. Jayasuriya, K. Swisher, J. Mativo, R. Pidaparti, and D. T. Robinson, "The Impact of Teachable Machine on Middle School Teachers' Perceptions of Science Lessons after Professional Development," *Educ Sci (Basel)*, vol. 14, no. 4, pp. 417-433, Apr. 2024, doi: 10.3390/educsci14040417.

[21] S. Mishra, M. Ryerkerk, Y. Lockerman, D. Eis, and J. M. Rzeszotarski, "Teachable Facets: A Framework of Interactive Machine Teaching for Information Filtering," *in Proceedings of the 2024 ACM SIGIR Conference on Human Information Interaction and Retrieval, New York, NY, USA: ACM*, vol. 2024, no. Mar., pp. 178–188, 2024. doi: 10.1145/3627508.3638289.

[22] S. Shafi and A. Assad, "Exploring the Relationship Between Learning Rate, Batch Size, and Epochs in Deep Learning: An Experimental Study," *in Soft Computing for Problem Solving, Singapore: Springer*, vol. 2023, no. 1, pp. 201–209. doi: 10.1007/978-981-19-6525-8_16.

[23] R. Lin, "Analysis on the Selection of the Appropriate Batch Size in CNN Neural Network," *in 2022 International Conference on Machine Learning and Knowledge Engineering (MLKE), Guilin: IEEE*, vol. 2022, no. Feb., pp. 106–109, 2022. doi: 10.1109/MLKE55170.2022.00026.

[24] R. D. Nurfita and G. Ariyanto, "Implementasi Deep Learning berbasis Tensorflow untuk Pengenalan Sidik Jari," *Emitor: Jurnal Teknik Elektro*, vol. 18, no. 1, pp. 22–27, Jun. 2018, doi: 10.23917/emitor.v18i01.6236.

[25] Y. Sari, Y. F. Arifin, Novitasari, and M. R. Faisal, "The Effect of Batch Size and Epoch on Performance of ShuffleNet-CNN Architecture for Vegetation Density Classification," *in 7th International Conference on Sustainable Information Engineering and Technology 2022, New York, NY, USA: ACM*, vol. 7, no. Nov., pp. 39–46, 2022. doi: 10.1145/3568231.3568239.

[26] N. Das and S. Das, "Epoch and accuracy based empirical study for cardiac MRI segmentation using deep learning technique," *PeerJ*, vol. 11, no. 3, p. 14939-14953, Mar. 2023, doi: 10.7717/peerj.14939.

[27] N. K. R. Mirayanti, S. Sariyasa, and I. G. A. Gunadi, "Batch size and learning rate effect in covid-19 classification using CNN," *Jurnal Mantik*, vol. 7, no. 3, pp. 1752–1765, Nov. 2023, doi: 10.35335/mantik.v7i3.417.

[28] J. Pan, "The impact of learning rate and data size on CNN for skin cancer detection," *in Second International Conference on Medical Imaging and Additive Manufacturing (ICMIAM 2022), Y. Yusof, Ed., Xiamen: SPIE*, vol. 2022, no. Jun., pp. 28-41, 2022. doi: 10.1117/12.2636723.

[29] A. Johny and K. N. Madhusoodanan, "Dynamic Learning Rate in Deep CNN Model for Metastasis Detection and Classification of Histopathology Images," *Comput Math Methods Med*, vol. 2021, no. 2, pp. 1–13, Oct. 2021, doi: 10.1155/2021/5557168.

[30] M. Kuehne, L. Polotzek, A. Haghikia, T. Zaehle, and J. S. Lobmaier, "I spy with my little eye: The detection of changes in emotional faces and the influence of facial feedback in Parkinson disease," *Eur J Neurol*, vol. 30, no. 3, pp. 622–630, Mar. 2023, doi: 10.1111/ene.15647.

[31] I. Vukovic, P. Cisar, K. Kuk, M. Bandjur, and B. Popovic, "Influence of Image Enhancement Techniques on Effectiveness of Unconstrained Face Detection and Identification," *Elektronika ir Elektrotechnika*, vol. 27, no. 5, pp. 49–58, Oct. 2021.

[32] S. Venkatesh, K. Raja, R. Ramachandra, and C. Busch, "On the Influence of Ageing on Face Morph Attacks: Vulnerability and Detection," *in 2020 IEEE International Joint Conference on Biometrics (IJCB), USA: IEEE*, vol. 2020, no. Sep., pp. 1–10, 2020. doi: 10.1109/IJCB48548.2020.9304856.

[33] X. Jiang and C. Xu, "Deep Learning and Machine Learning with Grid Search to Predict Later Occurrence of Breast Cancer Metastasis Using Clinical Data," *J Clin Med*, vol. 11, no. 19, pp. 5772–5791, Sep. 2022, doi: 10.3390/jcm11195772.