Real-Time Face Recognition and Tracking in Matlab

Üsame Delihasan Student ID: 040230789 Email: delihasan23@itu.edu.tr
Samet Atak Student ID: 040220615 Email: atak22@itu.edu.tr

Instructor: Prof. Dr. Tufan Kumbasar

***Abstract*—In this paper, a real-time face recognition system is presented by way of transfer learning on a pre-trained GoogleNet model. Furthermore, face detection, IP camera video streaming, and classification for target recognition and eigenfeatures of faces are detected to establish a face-tracking mechanism. The system integrated these, which benefited from Matlab toolboxes. Face recognition applications have placed many areas in public services and rely on some critical learning algorithms during the process. The main purpose of the project is to provide and implement a comprehensive application for an Arduino or Raspberry PI-based vehicle. The algorithms used in this project aim to provide a clear solution for these vehicles to recognize and track the faces of people.**

***Index Terms*—Face Recognition, Transfer Learning, IP Camera, GoogleNet, Real-Time Detection.**

I. Introduction

Face recognition is one of the most widely used components in the facilitation of public services. Thanks to recent technological advancements, facial feature information can be extracted using algorithms such as Linear Discriminant Analysis (LDA) and Convolutional Neural Networks (CNNs). These features can be utilized for recognition, functioning as a form of identity. In the study, the proposed system is established using transfer learning for fine-tuning and GoogleNet for face recognition. In particular, it is aimed to obtain a less time-consuming method and make the system efficient in terms of working with fewer datasets as images compared to training.

In order to reach the goal, GoogleNet pre-trained model is used a face-recognizing-tuning for the sole reason of ensuring that the model which we adapted is working specifically on recognizing faces. Some of packages and toolboxes are installed in matlab to mix advancements of codes. For instance, Deep Learning Toolbox Model for GoogLeNet Network, Deep Learning Toolbox, etc.. what is more, the code implements real-time face detection and classification by using IP camera. Also, in order to increase generalization, data augmentation techniques are utilized. Furthermore, the created project is not only recognizing face of the people but also tracking them by detecting minimum eigenfeatures of the face. To explain deeply, Real-time tracking system.

In addition to this, the application, which is designed specifically for the purpose of face recognizing and tracking system, can be used as a multifunctional thanks to the integration of several functions such as counting the people that detected. Also, facial orientation and movements were analyzed in the detected face whenever it was tracked.

II. Literature Review

GoogleNet is a kind of deep learning model which is able to extract features from images and then provides useful information about the facial identities of humans. The architecture of GoogleNet offers a very efficient way to solve the problem of obtaining valuable details for image classification or face recognition based on Inception modules. [1] There are several methods that can enhance the performance of GoogleNet, and data augmentation is one of them. In addition to data augmentation, real-time face recognition can be achieved through IP camera integration. It is easier to connect using the DroidCam application. Furthermore, for face detection, one of the most widely used method is the Haar Cascade. In the project, Haar Cascade algorithm was not used for classifier but it is used to detected faces. The classifying process is conducted by means of GoogleNet. On the other hand, it is a well-known fact that a Haar algorithm is able to detect faces efficiently in real time by way of Haar-like features based on the Viola-Jones algorithm. [2] Additionally, eigenfeatures are one of the important components used in real-time tracking. Benefiting from the algorithm which is created by Shi and Tomasi and used to find minimum eigenvalues of a face, is one of the ways of extracting information from corner features. [3]

III. Problem Description

One of the primary tasks of the project is to improve the capability of robustness in real-time face detection and to prepare and establish a better dataset to get high accurate face recognition system while taking the real-time IP camera input streams. Enhancing the generalization in order to achieve a reliable face recognition process with high accuracy by augmenting the training dataset and fine-tuning the GoogleNet. The main challenges include the following. On the other hand, in tracking mode which is another mode of the application, the directions of the tracking face are stored for the vehicle that will be in need of directions meanwhile tracking and following it.

IV. Solution Strategy

The solution strategy is separated into different stages in order to show applied solutions explaining with details in every stages in the following stages:

A. Dataset Preparation

This is a well known fact that the the dataset preparation is intensely highlighted in order to train a well model. In dataset preparation, it is decided to prepare a dataset utilizing a large amount of dataset for using face recognition from the public sources.

Additionally, the implemented code is provided to take pictures from the target face and save them into the file which is assigned by the name. After collecting enough images to use the appropriate model, collected images are organized and each representing each file represents an identity. For picture taking, it benefited from Matlab Support Package for IP Cameras. By utilizing the function `imageDatastore` which was used for data loading. The dataset was separated into 80% training and 20% validation since it is restricted to not encountering an overfitting problem. This process is crucial to be avoided underfitting or overfitting issues by taking care of bias and variation balance. To elaborate further on this point, errors can increase either due to simplified assumptions, which is a high bias issue or due to excessive sensitivity to training data, which results in high variance [4].

In addition to this, the prepared dataset is augmented for the purpose of avoiding overfitting by increasing the data diversity which means extracting new data from a prepared dataset. For instance, horizontal flipping, scaling, and pixel translation methods were applied to augment data after re-sizing to 224x224x3 as required for GoogleNet input which 3 represents RGB. For optimizing the training process, hy-perparameters such as MiniBatchSize, MaxEpochs, and InitialLearnRate were set in certain fixed values. For the sole reason of ensuring that to make a balance in the learning process, InitialLearnRate was fixed in an appropriate value to not combatting with decelerating in the weight updating due to very small value or missing the demanded solution, also MiniBatchSize was used to obtain the results faster, and in an optimized way in fixed MaxEpochs. In the Table 1, the dataset division is demonstrated for model training and evaluating model performance.

TABLE I
DATASET SPLIT FOR TRAINING AND VALIDATION

Dataset Portion	Percentage (%)	Description
Training Set	80	Used for model training.
Validation Set	20	For evaluation model performance.

B. Loss and Accuracy functions

In the process of training the model. Utilizing Accuracy and loss functions is critical. The following mathematical equations are the key principles underlying the operations of the training model.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}), \tag{1}$$

Where:

- L: the calculated Total loss,
- N: Number of samples,
- C: Number of classes,
- y_{ij}: True label which is one-hot encoded,
- $\hat{y}_{ij}$: Predicted probability.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}. \tag{2}$$

C. Probability and Label Definitions

Labeling is also fundamental for face recognition and the following mathematical concepts are used to describe labels and probabilities that are illustrated above the rectangular box in face-recognizing:

1) Label (label): The model's output is calculated and its maximum probabilities are used as indexes for label representation.

$$label = \arg\max_{j} \hat{y}_{ij}, \quad j \in \{1, 2, \ldots, C\}, \tag{3}$$

where $\hat{y}_{ij}$ is the predicted probability for the j-th class of the i-th sample, and C is the total number of classes.

2) Probability (prob): The probability shows the confidence of the prediction for the selected class, calculated as:

$$prob = \max_{j} \hat{y}_{ij}, \quad j \in \{1, 2, \ldots, C\}. \tag{4}$$

These are essentially mathematical equations that were used in face recognition in the project.

D. Explanation of Haar Cascade

After the training process is finished and saved, some of face recognition algorithms will be used to be able to detect and recognize faces. There are diverse methods to detect face features and one of the important algorithms is that Haar Cascade. Haar Cascade is a method which is depending on the Viola-Jones algorithm for face recognition task and it provides detecting face features by way of measuring the correlation between white and black rectangles. This method was used to find features of the faces quickly while in a real-time process. Particularly, the mathematical operations on white and black rectangles provide identities of the face. Further-more, MATLAB's `vision.CascadeObjectDetector` uses Haar Cascade by default which is provided by the toolbox of Computer Vision Toolbox [5].

The Haar Cascade method can be considered into the following key steps which are highlighted:

- **Feature Extraction:** The process of feature extraction is can be described as identifying meaningful patterns and extracting the difference between the sums of pixel intensities with taking into account of adjacent rectangular regions.
- **Integral Image:** Integral image is a notable factor for computing the rectangular features. Therefore, it is created to enhance the speed of computation in real-time process.
- **Cascade of Classifiers:** Multiple classifiers are arranged in a cascade structure, where simpler classifiers quickly eliminate non-face regions and more complex classifiers focus on the remaining areas.

As an implementation the Haar Cascade method, the `vision.CascadeObjectDetector` function provides pre-trained models to detect faces using Haar-like features.

Specifically, it uses bounding boxes to mark detected faces in the video stream.

The mathematical computation of Haar-like features is demonstrated in the following equation:

$$H = \sum_{i \in R_1} I(i) - \sum_{j \in R_2} I(j), \tag{5}$$

where:

- H is representative of the Haar feature value,
- R_1 and R_2 stands for the white and black regions,
- $I(i)$ and $I(j)$ demonstrations of the pixel intensity values in R_1 and R_2.

It is important to highlight that the Haar Cascade approach offers ability to disitnguish between regions that are not part of faces by using simple classifiers. Therefore, this provides reduced computational complexity and it is appropriate to be used in real-time face detection. After a pretrained model, which is trained by using GoogleNet, is loaded, the detected images are resized to match the inputs of CNN model and the function of `classify()` is utilized to perform recognition, and labels and probabilistic scores are assigned in every the location of the rectangular bounding box is updated dynamically in each frame. In addition to this, and minimum eigenfeatures were detected utilizing `rgb2gray()` function, which converts the region of the face into grayscale. After converting is conducted, grayscale images are processed with the `detectMinEigenFeatures()` function for the purpose of visualizing key points on the face during the real-time streaming in a manner of enhanced face feature demonstration.

E. Utilization of Bounding Box Data

During the process of distance calculation by benefiting from the obtained position data which is measured from the bounding box plays crucial role for determining the position of the detected face.

Mathematical Representation:

$$\text{face\_center\_x} = x + \frac{\text{width}}{2}, \quad \text{face\_center\_y} = y + \frac{\text{height}}{2} \tag{6}$$

F. Distance Estimation Using Bounding Box Size

The distance of the face from the camera is estimated using the width and height of the bounding box. This calculation is inversely proportional to the bounding box area and is implemented in the `startCamera` function.

G. Analysis of Camera Tracking Status

It is essential to determine position of the face within the frame using a mathematical reference. In this project, it is determined to utilize the function of `calculateFollowStatus` which evaluates the position of the face by comparing its relative location to the center of the frame and by assigning status using the ”rule of thirds” principle. The rule of thirds principle is a well known photographic principle that is used to divide frame into three vertical and three horizontal pieces which in equal widths and heights. By taking the rule into consideration, the function assigns a status in order to indicate position of the face within the framework. The status showed in the following equation:

Mathematical Representation:

$$\text{status} = \begin{cases} \text{”Left”} & \text{if } x_{\text{center}} < \frac{\text{frame\_width}}{3} \\ \text{”Right”} & \text{if } x_{\text{center}} > \frac{2 \cdot \text{frame\_width}}{3} \\ \text{”Up”} & \text{if } y_{\text{center}} < \frac{\text{frame\_height}}{3} \\ \text{”Down”} & \text{if } y_{\text{center}} > \frac{2 \cdot \text{frame\_height}}{3} \\ \text{”Centered”} & \text{otherwise} \end{cases} \tag{7}$$

1) Normalized Coordinate Calculation: In the process of coordinate calculating, it is considered that normalized positions of the face is can be determined in order to get efficient solution.

$$\text{norm\_x} = \frac{(\text{face\_center\_x} - \text{center\_x})}{\text{frame\_width}/2} \tag{8}$$

$$\text{norm\_y} = \frac{(\text{face\_center\_y} - \text{center\_y})}{\text{frame\_height}/2} \tag{9}$$

H. Point Tracker in Tracking Process

There are two modes in the face recognition and tracking application. Second mode performs tracking by extracting minimum eigen vlaues which have meanings as identities based on gray scale image of the ROI. When it is initialized by way of using `vision.PointTracker`. This function can perform focusing on key points obtained from the detected faces and continues tracking during real-time video stream. Additionally, the number of detected faces are displayed on the bottom left of the screen.

I. GoogleNet Architecture and Transfer Learning

In the process of model training, a deep learning architecture of GoogleNet, which is gained advanced capabilities benefiting from Convolutional Neural Networks (CNN) in order to acquire skill of advanced feature extracting, were used for the purpose of configuring taking into consideration the dataset prepared for the face recognition.

In addition to this, GoogleNet is a pre-trained model that includes 1000 classes that are already classified and learned. By way of the method of transfer learning, GoogleNet provides opportunities for users to change the layers as a specified goals by way of flexible and adaptable layers. They have the capability of be adapted and replaced with a new layer that is constructed to the use of special tasks. The process of layer changing for special task is called Transfer Learning.

To begin with, the prepared dataset is splitted into two different subsets such as train and validation in the ratio of 80:20. This ratio ensures that some of the portion of data is used in training process while leaving enough samples to validation process. After the dataset preparation, A `layerGraph` function is invoked to graph the neural network to demonstrate layers in detailed modify the architecture of GoogleNet for face recognition tasks. Then, feature learner and output classifiers are replaced with two different modified layers which is placed in the end of the GoogleNet neural network. In order to achieve this integration, `fullyConnectedLayer` was called to establish a feature learner and `classificationLayer` is invoked to create a customized output layer that will demonstrate modified classes.

Fully Connected Layer for Sequence Data:

- To give an example for this [6], if the layer before the fully connected layer has $1 \times N$ sized input vectors (X) and with a weights matrix (W) of $N \times M$ with the bias vector (b) of $1 \times M$, this will lead to a $1 \times M$ sized output vector (Z) and according to the representation of affine transformation:

$$Z = X \cdot W + b \tag{10}$$

Where:

- X: Input vector (1×9).
- W: Weights matrix (9×4).
- b: Bias vector (1×4).
- Z: Output vector (1×4).

J. Parameters in Fully Connected Layer

1. Weights (W): The weight matrix is a fundamental concept for a neural networks to multiply every input values which means are features in a weight to improve learning performance. Therefore, in a fully connected layer each feature can connect from the input to every output class. Also, in the case of F amount of input size and C amount of output classes are computed. An example for this case, W is a matrix of size 5×1024, where:

- 1024:input feature size in the layer (F).
- 5: output class size in the target (C).

Each entry in the weight matrix is a learnable parameter that determines the contribution of each input feature to a specific output class.

Mathematical Representation:

$$W \in R^{C \times F}, \quad W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,1024} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,1024} \\ \vdots & \vdots & \ddots & \vdots \\ w_{5,1} & w_{5,2} & \cdots & w_{5,1024} \end{bmatrix} \tag{11}$$

2. Bias (b): Another notable factor in weighted sum is bias vector. The bias vector changes the offset of the weighted sum for each classes in the output layer. Moreover, it offers better fitting into the data by resulting shifting in the activation function.

For this case, b is a vector of size 5×1, corresponding to the number of classes.

Mathematical Representation:

$$b \in R^{outputSize \times 1}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} \tag{12}$$

Each b_i is a parameter associated with a class.

3. Output Computation: Expanding the equation:

$$Z = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,1024} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,1024} \\ \vdots & \vdots & \ddots & \vdots \\ w_{5,1} & w_{5,2} & \cdots & w_{5,1024} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{1024} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

Softmax Activation Function

There are wide diversity for hidden layers in the field of neural networks. These hidden layers provides networks to learn better with a simplified approach by way of using activation functions which means simpler networks are turned into complex networks by means of using non linear activation functions in every hidden neurons. Additonally, non-linear functions provide opportunity to back-propagation if their derivations can be taken. In spite of the fact that there are many diverse functions that can perform activation transformation, it is decided to use a softmax function instead of other functions such as ReLU or tanh, so forth. The `Softmax` activation function is applied in order to make gain capability of learning complex data at the output $Z$of the fully connected layer and the softmax function assigns class scores into probabilities.

Mathematical Representation of softmax function

According to given vector $Z = [z_1, z_2, \ldots, z_C]$, the Softmax function assigns the probabilities $P = [p_1, p_2, \ldots, p_C]$ for each class as:

$$p_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i \in \{1, 2, \ldots, C\}$$

Where:

- z_i: Raw score for class i (output of the fully connected layer).
- $\exp(z_i)$: Exponential of the raw score for class i.
- $\sum_{j=1}^{C} \exp(z_j)$: Sum of exponentials across all classes, used to normalize the scores.

Output of Softmax

The output of the Softmax function after assigning probability distribution across C classes the sum of the probabilities will be lead to 1. Therefore, it is easy to interpret and suitable for classification tasks :

$$P = [p_1, p_2, \ldots, p_C], \quad \text{where } \sum_{i=1}^{C} p_i = 1$$

It is worth noting that, according to the number of classes that the model is trained on, the fully connected layer output size is changed dynamically. To clarify this concept further, the number of classes is increased as in the case of capturing a new model from the camera of the Ip-connected tablet and saving it in the dataset. Therefore, the number of classes is configured dynamically adjustable whenever added or deleted a data class from the dataset.

After the fully connected layer integration is completed successfully by replacing feature learner with the one of the input layers of GoogleNet, which is the 142nd layer chosen for this task, classifier layer is implemented by replacing output classifier layer by classifier layer. In the end of the transfer learning, the `analyzeNetwork()` function, which is provided by the Deep Learning Toolbox in MATLAB and used for the purpose of illustrating the adjusted graph is performed. Before performing the training, the last options are adjusted by the function of trainingOptions() which takes some of important inputs to enable optimized neural network training. Some of additional hyperparameters were set such as MaxEpochs, InitialLearnRate, ValidationData, ValidationFrequency. So forth. It is notable to highlight that these parameters are instrumental in facilitating the training process to converge meaningful solution. Epochs can be considered as iterations a time passed in a training set and it can be adjusted to set how many times entire training process can be divided into same time differences and all training process can be divided in epochs, which is updating the weights in every end of epochs with backpropagation. In order to complete one feedforward and backward passes training to get weights and biases, 1 epoch should be passed and in every epoch there divided batches which can be thought as for loop iterations to pass one epoch by way of taking into account the batch-size hyperparameter. Additionally, minibatch size is another kind of hyperparameter. Minibatch helps improve the stability of the stochastic gradient descent with momentum by separating the training data into subsets. Max Epochs is determined to set 6 and minibatch size is configured as 5 the for the training [7]. Initial learning rate is also configured to adjust learning speed and balance the training process in order not to encounter with an slow learning, underfitting or with losing generalization, which means overfitting. In a similar manner, the shuffle hyperparameter is set to `'every-epoch'` to enhance the generalization capability of the model which means aided algorithm by rearranging the order of training data at the beginning of each epoch, and provided avoiding overfitting which means the risk of memorizing specific parameters are avoided.

In a similar manner, the shuffle hyperparameter is set to `'every-epoch'` to enhance the generalization ability of the model. By rearranging the order of training data at the beginning of each epoch, the risk of overfitting is reduced, and the training process becomes more robust to the sequence of the input data. This adjustment helps ensure that the model does not memorize specific patterns in the data sequence.

Moreover, data augmentation process is conducted by using training data in order to enhance generalization ability as in the case of shuffling in epochs but this time the data is not reordered it is used to create new diverse data to prevent overfitting during the process of training. It is utilized the `imageDataAugmenter` function in MATLAB for the purpose of randomly reflecting horizontally within the framework of pixel range as $[-30, 30]$, and random scaling is applied within a range of $[0.9, 1.1]$. Overall the augmentations, diverse variations were exposed to the set which means the model can be more accurate or get improved its robustness in the case of encountering unseen data. During the data augmentation, not only training data is augmented but also validation data is improved.

Finally, the cost function and confusion matrix are created to detect whether the process is conducted with imposing enough optimization or it still in need of optimizations. In the resulting section it is illustrated the accuracy and loss graphs of both validation and training progress in every epochs and confusion chart and accuracy is observed.

We applied data augmentation techniques to enhance the generalization ability of the model and prevent overfitting during training. Specifically, we utilized the `imageDataAugmenter` function in MATLAB to introduce randomness into the training data. This included horizontal reflection, random translation within a pixel range of $[-30, 30]$, and random scaling within a range of $[0.9, 1.1]$. These augmentations ensured the model was exposed to a diverse set of variations, improving its robustness to unseen data. The augmented training images were resized to match the input layer dimensions of GoogleNet, ensuring compatibility with the architecture. While data augmentation was applied to the training set, the validation set remained unchanged to provide an accurate evaluation of the model's performance.

V. Results

The proposed system was developed, and some of the key results are highlighted to see the impacts of the changes and enhancements were performed throughout the process. Confusion Matrix is demonstrated on figure 1 and The graphs of the loss and accuracy functions are illustrated on figure 2.

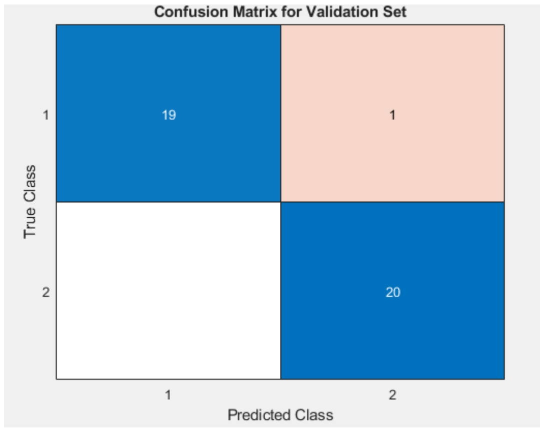
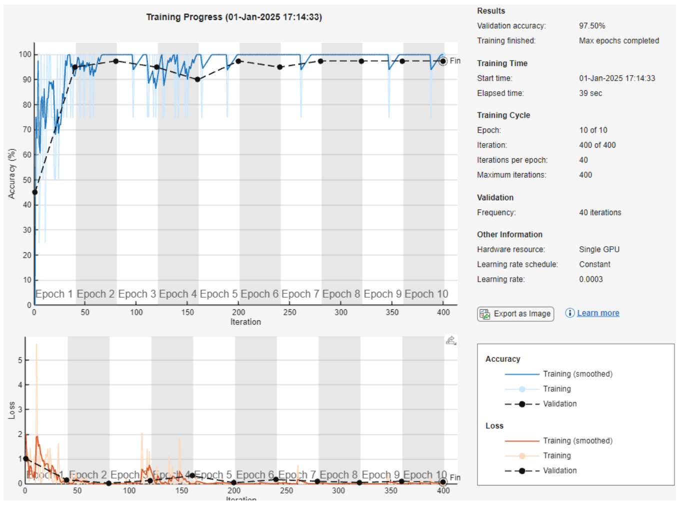


Fig. 1. Confusion Matrix for Validation Dataset

VI. Conclusion

In light of the findings and advancements presented in this project, the advancements offer a real-time face recognition and tracking system by conducting a training model process through transfer-learning and fine tuning of hyperparameters. During the implementation of adjustments on the application, some of critical algorithms and methods were systematically chosen to construct a robust and efficient model. The progress of integration Haar Cascade classifier for face detection, transfer learning on GoogleNet in order to perform recognition, and tracking algorithm for real-time tracking is enabled the system to achieve multi tasks by combination of them. Also, additional demonstrations are showed the interpret the accuracy and performance.



Results

Validation accuracy:	97.50%
Training finished:	Max epochs completed

Training Time

Start time:	01-Jan-2025 17:14:33
Elapsed time:	39 sec

Training Cycle

Epoch:	10 of 10
Iteration:	400 of 400
Iterations per epoch:	40
Maximum iterations:	400

Validation

Frequency:	40 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0003

Fig. 2. Accuracy and Loss Functions.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1–9, 2015.

[2] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I-I, IEEE, Dec. 2001.

[3] Shi, J. (1994, June). Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 593–600). IEEE.

[4] S. Fortmann-Roe, ”Understanding the bias-variance tradeoff,” 2012. Available: http://scott.fortmann-roe.com/docs/BiasVariance.html. Accessed: 2019-03-27.

[5] MathWorks. (n.d.). *vision.CascadeObjectDetector*. Retrieved December 20, 2024, from https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html.

[6] MathWorks. (n.d.). Fully connected layer. Retrieved December 27, 2024, from https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html

[7] J. Brownlee, ”What is the Difference Between a Batch and an Epoch in a Neural Network,” *Machine Learning Mastery*, vol. 20, pp. 1–5, 2018.