Low Light Gesture Recognition of RGB Images Using VGG16 Transfer Learning and S Channel Segmentation

UNIVERSITY OF
LINCOLN

Haani Rohaan Mahmood
MAH25270480

25270480@students.lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of BSc(Hons) Computer Science

Supervisor: Dr. Athanasios Polydoros

May 2023

Acknowledgements

Firstly, I’d like to thank my brother Humzah for visiting me last year and encouraging me to continue studying, as well as helping me organise myself to stay on track. I would also like to thank Dr. Athanasios Polydoros, for suggesting I work on hand gesture recognition and for being my project supervisor.

Abstract

A current area of interest for computer vision is the recognition of human hand gestures. This has many different applications, such as remote system control in manufacturing or sign language recognition for the hard of hearing. As an application of computer vision, un-optimal conditions like lighting can affect the ability of a HGR (hand gesture recognition) system. Many current methods focus on the use of depth cameras with light level invariant sensors to get around this issue; this paper instead focuses on a HGR system that can work on gestures in low light using a standard RGB camera. This paper proposes a method using image processing on the saturation channel and a VGG16 CNN classifier, trained using transfer learning. The effectiveness of this system is tested separately on high and low light images. Moreover, the effect of changing the ratio of low light to high light training inputs on the CNN is observed and evaluated.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Simply put, computer vision is the field of study with the goal of emulating humans' ability to view and recognise objects, environments, actions and etc., and through the use of computer logic, algorithms, and machine learning methods (known as image processing), to infer useful information which can then be used by humans and computer systems to take actions based upon what is recognised (Gonzalez and Wood, 2018, 18).

One common obstacle in computer vision is low light levels; low light levels reduce the available light that can reach the camera sensor, resulting in images that are more difficult to apply the image processing algorithms needed to find and recognise objects of interest, due to increased noise and loss of detail. Furthermore, subsequent machine learning classification methods are made more difficult by these issues (Ai and Kwon, 2020, 1). A current popular application of computer vision is static hand gesture recognition; humans make certain gestures, which correspond to different commands defined by a program. An advantage of such systems is the ability for users to intuitively use their body for inputs, rather than having to learn abstract commands, allowing for more efficiency, and a quicker rate of learning, due to the simplified input method (Tsarouchi et al., 2016, 5) (Mahmood, 2022, 1). Additionally, this application of computer vision is very useful for hearing impaired people who may want to communicate with a device through sign language, Alhafadee et al. (2022) provides a summary of multiple comprehensive reviews focusing on this sub-area.

Current research into hand gesture recognition can be distinguished into 2 main categories, vision based and depth based, depending on the sensor that is used, that being RGB or RGB-D respectively. The sensor being used largely affects the method of reducing the effects of low light, for maintaining a robust hand gesture recognition system.

RGB (Red Green Blue) cameras that are used in vision based research are identical to, the ones present in webcams and phones, as video input. They provide an advantage in their wide use and availability, as well as their low cost to implement into systems. 2D image frames from the video are acquired, they are pre-processed and analysed to extract the hand as a feature, which can then be classified using appropriate machine learning techniques (Mahmood, 2023, 1).

Depth based approaches utilising RGB-D (Red Green Blue Depth) cameras that have the ability to calculate the depth of objects. Generally, RGB-D cameras utilise either the time of flight principle, or light coding techniques to attribute depth values to RGB pixels, (Chen et al, 2013). This additional channel of information can facilitate detection techniques that are more resistant to adverse lighting conditions (Suarez et al, 2012 Previously, sensor based methods (utilising equipment such as gloves) were also heavily studied, however due to the cumbersome nature of the wear as well as cost, efforts have been concentrated elsewhere (Zhu and Sheng, 2011) (Nikam and Ambekar, 2016). As mentioned previously, the different methods used for HGR systems can be broken down into 2 major components: the feature extraction via image processing techniques and ML classification, trained upon the features which have been chosen to be extracted.

Different papers have proposed different methods to be implemented into both of these areas for the purpose of alleviating the effects of low light. For example, Luo et al. (2021) opts to use a YCbCr colour space filter to enhance the features of the gesture, this colour space separates luminance (the brightness) from

the chrominance (the colour) more effectively than the standard RGB colour space. A form of a Machine Learning classifier called a CNN (convolutional neural network) is used to classify the hand gesture images. This is done by training the CNN on a sample of hand gesture images, allowing it to learn the enhanced features in order to be able to distinguish new hand gesture images. Furthermore, Luo et al. include many variations of light data in the training images, so that the CNN classifier will be more resilient to lighting changes.

This paper focuses on a method developed using a standard phone RGB camera, as these sensors are far easier to access, therefore their potential use case is much wider; as Tsai et al. (2022) puts it- depth sensors' bulkiness, price and need for precise environments make them very restrictive in terms of daily life application scenarios. (Mahmood, 2023, 1).

The purpose of this paper is to produce a hand gesture recognition artefact, utilising a VGG16 CNN and HSV based image segmentation, directly addressing how it processes low light images, as well as the differences in efficiency and results of hand gesture recognition between low and high light images. This paper: reviews relevant literature in the area, to inform the knowledge needed for the project; details the aims and objectives for achieving the goal, as well as discussing their aspects in a SMART model; lays out and analyse the requirements for fulfilling these objects; discusses the design of the artefact and the methodology chosen for implementation; in depth, presents the actual artefact implemented; shows and discuss the evaluation of the artefact and finally a conclusion, reflecting on the project.

Chapter 2

Literature Review

2.1 Past papers

Chaudhary and Raheja (2018) propose a solution for light invariant hand gesture recognition system, in real time, achieving an accuracy of 92.86% in extreme light intensity changing environments. The regions of interest (ROI) are segmented from the background, pre-processing them for feature extraction. The orientation of the edges of the pre-processed images are extracted by finding the ROI's edges, calculating the gradient orientation vector of each pixel in the ROI and rearranging them as column matrices, so they can be used to plot an orientation histogram. This histogram is averaged locally to reduce noise. These histograms show to exhibit similar distributions for different skin tones and light conditions, for each gesture tested, resulting in a feature that can be used in the ANN classification. The ANN is trained using a back propagation algorithm, to systematically update the connection weights to produce a desired classification. This does produce a good accuracy of 92.86%, however the ANN takes a long time to decide the optimal number of hidden layers and number of nodes in each layer.

In the same year, Perimal et al. (2018) published a unique solution for the HGR of 14 different gestures for finger counts on a hand (i.e. recognising how many fingers are being held up). The images are acquired using a standard HD webcam, these images are converted into the YCbCr colour and are automatically thresholded using Otsu's method and converted into binary, resulting in a white hand with a black background. The image is pre-processed, small objects are removed and making sure only one object is remaining, before the algorithm can proceed further. In

order to detect the fingers, the centroid of the hand is found, which in tandem with the calculated maximum distance between one pixel to another on the contoured hand, used to remove the wrist and the palm of the image. Now the image is solely the segmented fingers, the finger count is obtained by counting the objects remaining in the image, then it is classified into its gesture by calculating the max distance between the centroids of the fingers using an equidistance equation. Results show >95% successful finger detection for all finger counts, but varying gesture recognition, with 4 fingers being as low as 56%- showing this method for gesture classification isn't reliable enough. Notably, however, when artificially changing the light intensity of the test image, the HGR success rate will not change until it is reduced to less than 0.1%, and the algorithm is not affected at all by increases in light intensity.

On the other hand, Huang et al. (2019) developed a more traditional approach for HGR using skin detection and deep learning classification method, with a successful classification rate of 98.41%. Skin is segmented from the background of images taken with a RGB camera by using RGB thresholds that should work for most humans' skin tone, this segmented image is converted into binary to produce a two tone image that can be pre-processed. Pre-processing reduces noise in the image, this paper uses a time efficient total variation algorithm, that implements simple morphological operations such as erosion and dilation. Now, with a binary image of a hand and face, VGGNet is used to classify the different point clusters, a deep CNN developed by the Visual Geometry Group and researchers at Google DeepMind, in order to contour and segment the hand. The VGGnet stacks 3x3 convolution cores and 2x2 maximum pool layer repeatedly, successfully constructing a 16 layer deep CNN. Finally, to perform gesture recognition, the original input image is passed through the convolution layer and maximum pooling layer to obtain a size of 1/2 the original image feature map- 4 different scale spatial pyramids are further pooled to obtain feature map sizes of 1/4 - 1/32 respectively so the different scale features

can be captured. Probability scores are calculated using fully connected layer and softmax, after the weights of global abstract feature are captured using global average pooling. In the abstract, illumination conditions are mentioned as a difficulty in the area of gesture recognition, however it is not further developed upon in the rest of the paper.

Sahoo et al. (2019) proposes a method for static hand gesture recognition using a simple RGB camera, which was chosen over a depth camera, due to noise arising in low illumination in Suarez and Murphy (2012)'s review paper on HGR using depth images. The pretrained AlexNet convolutional neural network is used for feature extraction on resized American Sign Language gesture images, developed using 5 convolution layers, 3 max pooling layers and three fully connected layers. Redundant features are discarded using PCA dimension reduction, to maintain desired information whilst removing those with low variance, this allows the training algorithm to become faster, as well as removing possible noise in the data and reducing the required storage size during training. The final features are normalised according to zero mean and unit variance, after which they are trained on a one-against-all (OAA) multi-class SVM classifier. The best mean accuracy was 99.32%, achieved from Holdout Cross-Validation, however this CV method was deemed biased, meaning the true mean performance was 87.83%, obtained from leave-one-out cross validation (LOO CV). Despite mentioning light intensity and illumination as issues in HGR, this paper fails to further develop proposed methods for alleviating these issues neither in the dataset nor the pre-processing stage, unlike Luo et al. (2021).

Luo et al. (2021) proposes an improved gesture segmentation method for gesture recognition based on CNN and YCbCr, also opting to use the AlexNet CNN model. The gesture data is first pre-processed, transforming the colour space from RGB into YCbCr, as mentioned previously this colour space separates the luminance from the chrominance more effectively than RGB does, making colour features more resistant to

lighting changes, after which the Cr channel is extracted. Gaussian filtering is performed, using a symmetric Gaussian kernel for filtering, providing a skin colour detection. A mark-based watershed algorithm is used to segment the hand, which does so without being affected by noise and irregular gradient by connecting the pixels of adjacent grey values into a contour, and an eight-connected seed filling algorithm is used to fill any internal pixels of the segmented area. These images are scaled and labelled and used as input data to train the AlexNet CNN through iterative forward propagation and backpropagation. In the analysis, half of all the testing gestures is recorded during day and the other half during the night, resulting in a reported average success rate of 94.27% “in natural light conditions”. However, details pertaining to the light levels during in lab testing are not given, nor is the accuracy of recognition of low light recognitions (Mahmood, 2023, 2).

In Khaleghi et al. (2022), a real-time HGR system is proposed for use in construction sites for control of heavy equipment, this system uses Mediapipe, an open source system developed by google, for hand detection and high quality hand landmarking- a very low cost system as it can be work in real time using only a CPU. Mediapipe segments the hand by finding the palm region and cropping the image, then it estimates 21 3D landmarks of this hand based upon this image by identifying different parts, points and joints of the hand. A multi-layer perceptron artificial neural network is used to classify the hand gesture, based upon the features estimated by Mediapipe. 26,000 data images were collected in various low light and high light conditions, by 5 different operators; the neural network was trained using the Adam optimiser for 500 iterations and an initial learning rate of 0.01, cross subject and cross lighting test protocols were used for training and evaluation of the models to ensure the classifiers robustness to different subjects and lighting conditions. The resulting ANN produces an accuracy of over 99% in both testing protocols, the best of

their tested models however, slightly slower than the Naïve Bayes and SVM models but still decidedly fast enough for real time processing.

2.2 Aims & Objectives

The aim of this project is to have produced a hand gesture recognition system that can correctly classify RGB images taken in high and low light environments, and to evaluate its performance on these types of images respectively.

Table 1 below outlines the objectives needing to be achieved to accomplish the aim set out above, breaking them down into the 5 components of the "SMART" model: Specific, a concise description of the objective; Measurable, quantifying the goal of the objective; Achievable, how the objective will be achieved; Relevant, why is the objective needed; and Timebound, when should the objective be completed by.

Table 1 SMART objectives.

	Specific	Measurable	Achievable	Relevant	Timebound
Objective 1	Collect a dataset of hand gestures.	Have compiled a dataset of 150 or more samples of 4 different hand gestures, in low and high light environments	Take videos of hand gestures using a phone or web camera and convert them to image frames. Alter the light conditions using lamps, curtains, blinds and etc.	A dataset is necessary for the training, validation and testing of the dataset.	Before 20/01/22

Objective 2	Implement a data pre-processing method and perform it on the acquired hand gesture images.	The hand gesture images are correctly segmented from the image and are labelled according to their class.	Research into image processing techniques and past papers, using knowledge gained in order to create a method for hand segmentation that works in high and low light. Implemented in python using the open-cv module.	In order to create an effective classifier, data must be pre-processed to standardise differences between images and make classification easier. Furthermore, training data must be labelled with a ground truth for training.	Before 02/02/23
Objective 3	Train a deep learning classifier on the pre-processed data.	Successful classification accuracy of classifier reaches 90% or higher.	Research into previous methods of hand gesture recognition to inform the options for implementing deep learning classifiers. Implemented in	Machine learning classifier is needed in order to recognise the pre-processed hand gesture data.	Before 20/03/23

			python using keras.		
Objective 4	Test and evaluate the classifier's ability to correctly classify low light and high light images.	Have evaluation metrics such as accuracy and F1 score of a foreign data set that the classifier has not been trained on.	Predict the trained model on a foreign data set and calculate these values based on their ground truth and predicted label.	Evaluation of the classifier is a necessary step in understanding the performance of the classifier. Producing evaluation data is necessary for this.	20/04/23
Objective 5	Complete dissertation	Achieve both aims, as well as the chapters of written content set out in the brief.	Perform research, create and abide by planning and methodology, detail implementation and evaluate results and paper as a whole	Required for partial fulfilment of the Degree of BSc(Hons) Computer Science.	11/05/23

Chapter 3

Requirements Analysis

For the aims and objectives set out in **Chapter 2**, additional considerations must be made to ensure that the desired project outcomes are achieved.

3.1 Functional and Non-functional Requirements

Functional requirements refer to the features that must be present in the artefact i.e., what the artefact can do, for example, image pre-processing. Whereas Non-functional requirements refer to standards that don't directly relate to a specific feature i.e., general properties of the artefact. Setting these requirements out are useful for breaking down and understanding the most important aspects of the artefact being developed.

Functional Requirements:

- Image reading, processing, and saving using open-cv
- Training, validation, and testing data allocation
- CNN model training on processed images using keras
- CNN classification of hand gesture images using keras
- Produce evaluation data on CNN classification predictions using scikit-learn

Non-Functional Requirements

- Can compile and run with no errors
- >90% accuracy on predicting hand gestures
- Accuracy, precision

3.2 Software and Hardware Requirements

In order to achieve the objectives of this paper, certain software and hardware requirements must be acknowledged.

Software Requirements:

- Python
- OpenCV library
- Tensorflow library
- Keras library
- Numpy library

Hardware Requirements

- RGB camera
- Up to 100Gb in storage for images, processed data and CNN model weights
- PC with powerful CPU and/or GPU for training CNN and plenty of RAM for training using keras

3.3 Risk Analysis

When developing the artefact, risks that may affect the development process must be considered. By considering possible risks that may arise, the degree and nature of their impact to the project can be evaluated and mitigation measures can be thought of and implemented. **Table 2** below presents these risks and their

assessment.

Table 2 Risk analysis

Risk	Likelihood	Impact	Mitigation Measure
Artefact corrupts.	Low.	Artefact would become unusable.	Using version control and backing up on OneDrive would ensure an uncorrupted version of the artefact is always available.
Image pre-processing fails to successfully segment low light hand gestures from their background.	Moderate.	Would be unable to train a classifier capable of low light hand gesture recognition.	Many human skin segmentation image processing methods are available and known. (Leite M, 2022)
Deep learning model fails to be able to predict hand gesture classes to an accuracy of 90%.	Low.	One of the main objectives would not be met, furthermore data would not be sufficient enough for necessary	There are pre trained CNNs available for the purpose of image classification transfer learning, which would make the possibility of not being able to

		evaluation of model.	classify very low Such as AlexNet (Krizhevsky et al., 2012) and VGG16 (Simonyan et al., 2015).

Chapter 4

Design & Methodology

4.1 Changes to Project Focus

As development on the project had progressed, the focus of the paper has slightly changed a couple of times. Initially, a project plan was proposed on the study on the weaknesses of the image recognition of human hand gestures. This proposals' scope was too large for the time frame and resources available, being a study on low light's effect on HGR, with multiple image processing and classification methods being created and compared, as well as studying into adversarial examples – investigating into their effects and danger to machine learning models. In a project meeting on the 9th of December 2022, it was agreed upon to reduce the scope to just the effects of light, with adversarial examples as a possible

stretch task if time allowed.

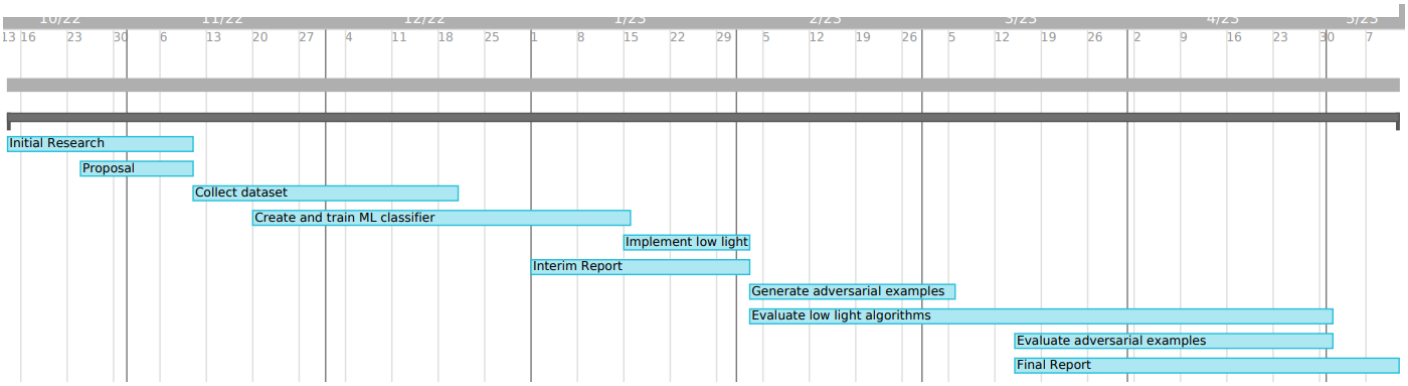
Secondly, as progress on developing the artefact began and more focused research was done, it was decided on the 31st of January to focus on RGB methods of low light hand gesture recognition, as methods implemented with depth cameras highly depend on the method in which the camera collects the data- which may make the detection of gestures in low light much more trivial, furthermore trying to implement multiple types of sensors would be costly and exceed the budget available as a student. Finally, it was decided to instead focus on a single method of low light hand gesture recognition, as implementing multiple methods, as well as discussing their design and evaluating their results appeared to be out of the scope of this project.

4.2 Project Management

This section covers the methods used to manage different aspects of project development, as well as a reflection on the methods chosen.

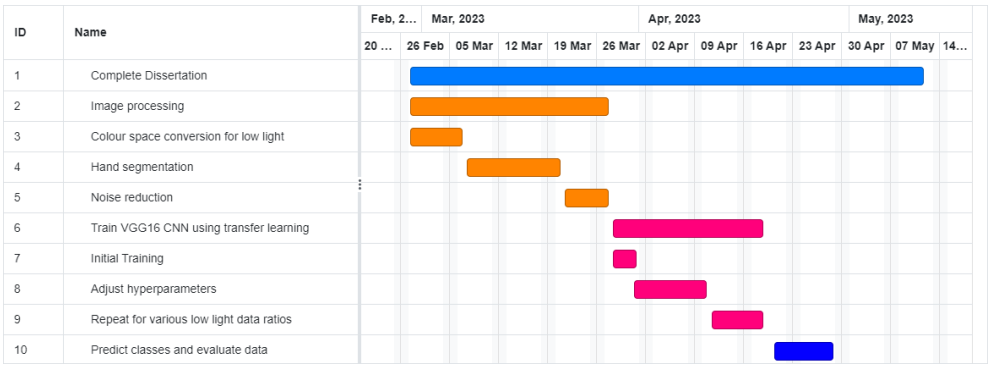
At the beginning of project development, an initial Gantt chart was produced, outlining the general tasks that needed to be completed for the old project focus. This can be seen below in **Figure 1.**

Figure 1 Old Gantt chart



Due to the project changes, as mentioned in **4.1**, this Gantt chart became outdated, additionally this Gantt chart did not provide enough detail for the tasks making it very unsuitable for use. Once planning the design had been completed for the final project plan, all remaining aims and objectives were broken down into a list of individual tasks that could be completed and marked as completed, these were made into a Gantt chart that can be seen in **Figure 2** below. A Gantt chart is useful for project management as it can provide a visual representation of a project timeline, making deadlines and the scope of certain tasks easier to grasp. Furthermore, utilising a Gantt chart helps with identifying task dependencies and the relationships between tasks, ensuring the pre-requisites for each task are understood and delays in completing a task do not affect the timespan of the entire project.

Figure 2 Final Gantt chart.



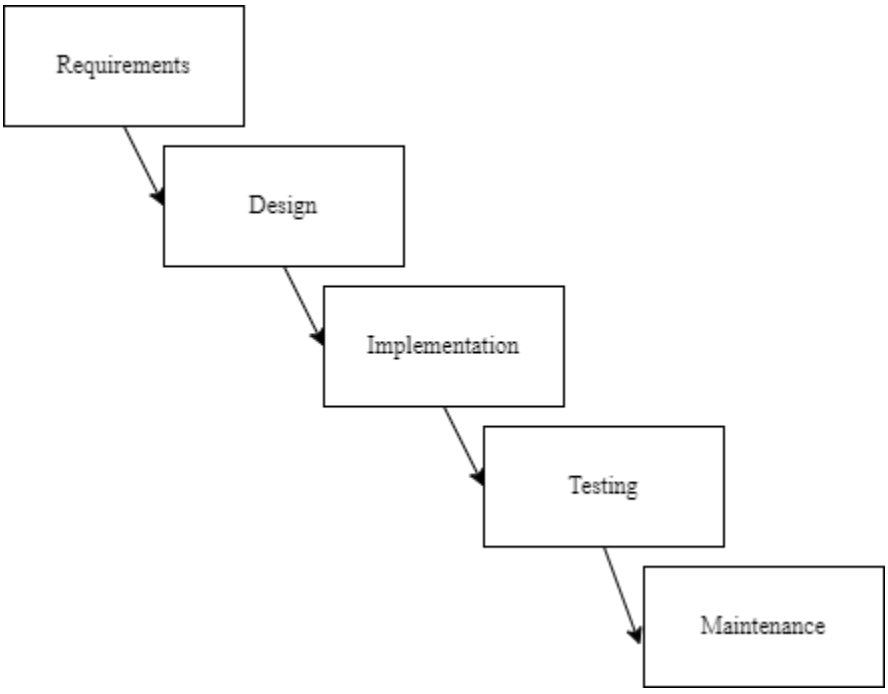
4.3 Software Development Methodology

This section discusses the software development methodology chosen for managing the development of the artefact.

The artefact creation of this project took place under a waterfall approach. This model is made up of a sequence of linear phases, where each phase's output is valuable information that is required to begin the following

step. This methodology suits the nature of this project, due to many of the objectives also following a similar structure, wherein they cannot be worked on until a previous objective has been accomplish. For example, the image processing methods cannot be worked on until the dataset has been gathered, the CNN cannot be trained until the image pre-processing method has been completed and etc. In **Figure 3** below is a diagram, representing each stage of the waterfall model.

Figure 3 Waterfall diagram



The contents of the requirements phase are set out in the requirements analysis section of the paper, these were first created during project proposal and have been changed since the interim report. As the old requirements encapsulated the nature of the new requirements, there was very little impact to the design stage of the methodology. The requirements of this project were informed from the initial research conducted into gesture recognition papers.

The design of the artefact is compiled into this section of the paper, the design refers to all aspects of the artefact e.g.: the coding language being

used, the segmentation methods chosen for pre-processing, the general CNN model structure and etc. Designing the artefact before implementing it can help with ensuring that it will perform as intended during implementation. The design is made up of information gathered from in depth literature reviews into the more relevant papers, as well as general research gathered from online resources.

The implementation consists of the actual software development of the artefact, features are implemented in code and integrated based upon the plans set out in the design phase. Certain parameters that were not yet decided upon in the design phase can also be worked out here, such as in this paper, the colour channel being used was determined during preliminary testing of the image pre-processing feature.

In this paper, the testing phase refers to the white and black box testing being performed, and the evaluation of the data produced. Finally, maintenance refers to additional work that could be done to the artefact, reflecting on its performance, that is covered in the conclusion of the paper.

4.4 Toolsets and Machine Environments

This section will discuss the different software tools used during the development of the art, as well as detailing the hardware used.

4.4.1 Software Toolsets

For all software development, Python 3 was chosen as the programming language. Python provides many well developed libraries for both image processing and deep learning applications, for example Pytorch, Keras, OpenCV and Tensorflow- all examples having capabilities for both image processing and deep learning.

Keras is used for training the CNN using transfer learning, and OpenCV is used for its image processing capabilities. Keras was chosen due to its in library implementation of VGG16 model, with pre-trained weights

on the ImageNet database. OpenCV was chosen due to having extensive amounts of image normalisation algorithms, as well as many segmentation algorithms.

4.4.2 Machine Environments

For capturing the images for the dataset, the rear 16 megapixel camera of a OnePlus 6T was used. All image processing, CNN training and predicting were performed on PCs in the University of Lincoln computer labs located in the Isaac Newton Building, the hardware specifications are listed below.

- Processor: 11th Gen Intel Core i5-11600K
- GPU: NVIDIA GeForce RTX 3070
- RAM: 16 GB
- Operating System: Windows 11 Enterprise

4.5 Dataset

This section will detail the process of choosing and producing the hand gesture dataset used in the project.

At first, a public dataset was sourced, containing 12 different static-hand gestures, from 5 different participants, including samples in varying backgrounds and light conditions (Nuzzi et al, 2021). The dataset only needed enriching with low light samples, however, in order to keep the scope of the project within the means of the timespan available, the dataset was dropped as it would require complex feature extraction methods to be implemented to segment the hand from the rest of the image, on top of needing to develop around low light conditions. Instead, a dataset for training, validation and testing was developed for this paper, by using a simple phone camera. Samples of hand gestures in low light can be seen in the below **Figures 4 & 5**. 300+ samples of four different hand gestures

were collected for the purpose of training, validation and testing. The gestures used are the American sign language symbols for the letters A, I, V and W. 4 symbols constitute as enough different classes to reasonably challenge the classifier, furthermore the pairs A-I & V-W are visually similar which should challenge the classifier's ability to learn low level features- especially in low light samples, due to the increased amount of noise. Varying light levels were achieved by taking the pictures indoors and adjusting the positioning of the curtains, altering the natural light in the room, and changing the brightness and positioning of lamps and ceiling lights in the room.

Figure 4 High light hand gesture image samples (Clockwise: A, I, W, V)

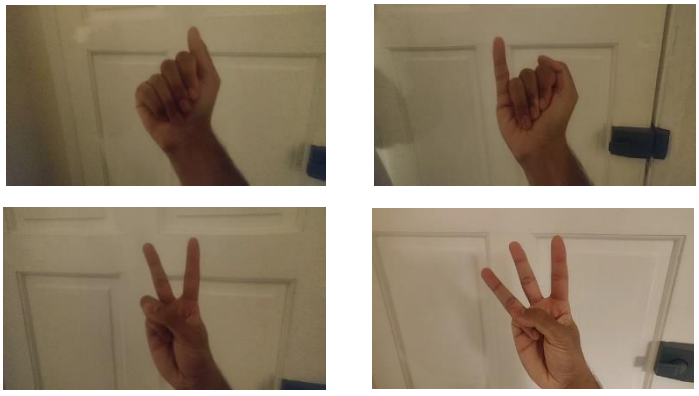


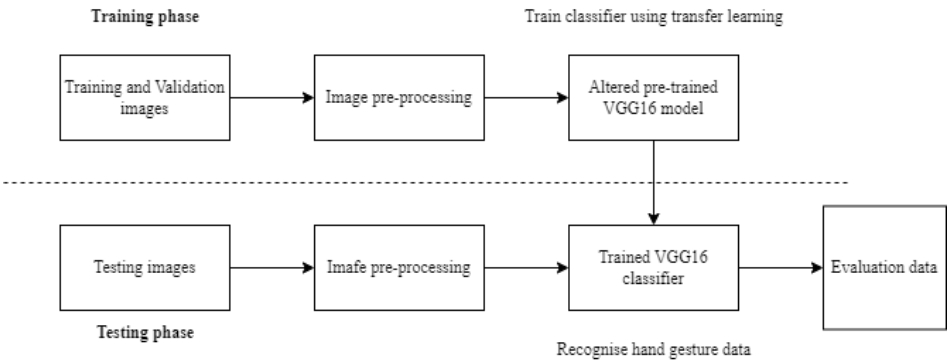
Figure 5 Low light hand gesture image samples (Clockwise: A, I, W, V)

4.6 Program Design

This section of the paper discusses the general program design, as well as the design for the features of the artefact, with discussion about the choices made for these aspects.

Below in **Figure 6** is a block diagram outlining the general processes that make up the hand gesture recognition system. The system can be split into 2 main features, the image pre-processing and the CNN model. The CNN model is trained on images that have been pre-processed to enhance relevant features, this same pre-processing method is used on the testing data, as it must match the features the model has been trained in order for it to be correctly classified.

Figure 6 Block diagram of the major processes



4.6.1 Image Processing and Hand Segmentation

Before any images are used as inputs for CNN training, they must first be processed to segment the hands and improve the visibility of features, especially those in low light.

The three primary colour spaces for recognising skin colour are RGB, YCbCr and HSV. In Luo et al (2022) the RGB images are converted into YCbCr, where the channel Cr is extracted, due to its robustness to

lighting changes, this is due to the luminance and chrominance components being explicitly separated into different channels. RGB's mixes these components, making detecting skin colour more difficult in sub-par lighting conditions. The HSV colour space describes perceptual features of colour, those being it's hue (H), saturation (S) and value (v), also known as intensity (I). HSV can work well in extreme high lighting conditions, but do not provide as much in the way of low light robustness. (Leite et al, 2022, 10).

For these reasons the same method will be used, in which the RGB images are transformed into the YCbCr colour, after which the Cr channel is extracted.

In order to segment the hand from the background, an adaptive thresholding algorithm will be used. Due to the simple background of the data images in, Otsu's method can be utilised to segment and binarize the hand (Perimal et al., 2018, 20) Otsu's method automatically thresholds the image by separating pixels into foreground and background classes and determining the threshold through maximising inter-class variance. Noise may still be an issue here, so morphological operations can deal with these issues, with very little processing power (Huang et al, 2019, 3).

4.6.2 CNN Model

CNNs are a very popular machine learning model used for hand gesture classification [Huang et al. (2019), Luo et al. (2021), Elliott et al. (2021)]. By learning high-level features built upon lower level features, they can still learn patterns in data with lots of noise, making them a good choice for the learning and classification of low light hand gesture images.

AlexNet (Krihevsky et al, 2012) and VGG16 (2015) provide CNN frameworks that have been optimised for image classification purposes, for this reason they are commonly used as the framework for creating and

training new classifiers. AlexNet and VGG16 were both trained on the ImageNet database (Deng et al., 2009), their pre-trained weights are available to allow the use of transfer learning. Transfer learning takes a pre-existing classification model, with weights and layers pre-defined, and trains the model on a new set of data, this is useful as it allows new image recognition systems to be trained with far fewer training samples and epochs. For these reasons, the artefacts model will be trained using transfer learning. VGG16 has been chosen over AlexNet due to its deeper architecture, allowing it to learn the features of the low light gestures more effectively than AlexNet. The Keras library provides the framework for creating, training and testing the model. For the loss function, categorical cross-entropy will be utilised, by measuring the difference between predicted probability distribution and actual probability distribution. At the end of each epoch the validation dataset is predicted on, the prediction data is used for the cross-entropy function, producing a gradient that can be used by the Adam optimiser (Kingma and Ba, 2015) in order to alter the hyperparameters. This is also done every batch during the epoch on the training data. The Adam optimiser was chosen due to its adaptive learning rate, allowing for quicker conversion during training, furthermore it implements regularization, which helps prevent overfitting when fitting noisy gradients.

4.7 Evaluation Methods

In order to evaluate the artefact, the effectiveness of the hand gesture classification model must be tested. This can be done by inputting a test set of data, with known labels, into the system and comparing the predicted labels with the actual labels. A simple accuracy score can be obtained by dividing the number of correct classifications by the total number of classifications. Accuracy can represent the ability of the classifier over the whole dataset, however in order to get a better understanding of the performance on each class, a confusion matrix will be also used.

A confusion matrix contains 4 data points for each class, true/false positive and true/false negative. An accuracy score and confusion matrix will be produced for a set of high light hand gesture images and low light gestures images, these figures will be used to perform a comprehensive evaluation of the performance of the artefact. Furthermore, these scores will also be calculated for varying ratios of low light data being present in the test set. By doing this, further insight into the CNNs training abilities can be made.

Chapter 5

Implementation

This chapter of the paper details the implementation of the artefact, following the design set out in the previous chapter. This chapter will be broken down into 2 sections, one covering the implementation of image processing and another covering the implementation of the VGG16 model. Much of the structure of the implementation is inspired and informed by the article Kommineni (2019).

5.1 Image processing implementation

All image processing for the train, validation and test images is located within the file 'HGR_preprocess.py'. Before the image processing takes place, the function "Proceessfile" is used to organise what data enters which data set. This function is needed, as different file paths containing high and low data must be distributed according to the ratio of high light to low light hand gesture images. Said function takes parameters for: the path, an array containing the number of images needed for each set, dictionaries that contain a count of each label for each data set and the arrays the data and data labels are being saved to. A partial screen shot of this function can be seen below in **Figure 7**.

Figure 7 Process file function

```
def processfile(path, param_array, train_dict, test_dict, val_dict, X, Y, X_test, Y_test, X_val, Y_val):
    # set variables for parameters
    train_num = param_array[0]
    val_num = param_array[1]
    test_num = param_array[2]

    #reference table
    class_ref = {"A":0,"I":1,"V":2, "W":3}

    files = os.listdir(path)
```

The file names from the path passed into the function are obtained and added to a list, these files are then shuffled- to ensure there are no patterns to which the data images are distributed to each set. A for loop is used to iterate through each file name in the now shuffled list, the label is obtained from the file name so a set of if statements can check whether the current file needs adding to any of the data sets. If so, the count in the relevant dictionary is increased, the image is processed, and its processed data and label are saved to the corresponding arrays.

The function “processimage” is where all the operations for hand segmentation are done- all of which are done using functions from the OpenCV library. After the image has been read, it is resized to 224x244 , to fit the requirements of the VGG16 model, then it’s colour space is transformed from RGB to HSV. The original plan set out earlier in **4.6.2** was to use the YCbCr colour space, with a Cr channel extraction, however preliminary tests on the dataset showed that the Otsu binarization using this channel did not give satisfactory results, and in fact converting to HSV and extracting the saturation channel S tended to show better results segmenting in low light. A comparison of these two methods is presented below in **Figure 8**, showing the S channel method to have better results in segmenting more of the hand. For some of the low light images, the background and foreground can be classified the wrong way (as can be seen in **Figure 8**) so a simple algorithm is used to flip the binary colours if the number of white pixels greatly exceeds the number of black pixels.

After the image is binarized, there is still some noise

present, which is reduced through the use of morphological operations. Morphological closing is first used remove holes within the segmented hand, reducing noise within the object, then morphological opening is used to reduce the noise around the image. Kernels sizes for opening and closing were obtained through trial and error, these can be seen in **Table 3** below. After that, a mask is applied to the image to remove all objects aside from the largest, to ensure that any remaining noise around the segmented hand is removed, and finally gaussian filter is used to blur the image slightly, to reduce noise on the edges of the segmented hand as well as to help reduce overfitting when training. Final, the image is converted back into a RGB format using OpenCV, to fit the input requirements of the CNN, the features of the image remain the same however. In **Figure 9 & 10** below are comparisons of a high and low light image before and after undergoing image processing. After all images have been processed and allocated to their respective data sets, they are saved to folders as numpy files so they can be accessed during training and prediction.

Figure 8 Cr channel and S channel Otsu binarization on low light image

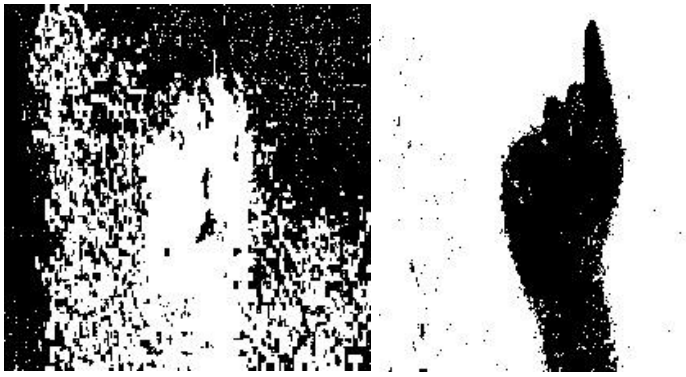


Table 3 Morphological Kernel sizes

Morphological operation	Kernel size	
	Erosion	Dilation
Opening	3x3	1x1
Closing	1x1	3x3

Figure 9 Low light image processing input and output

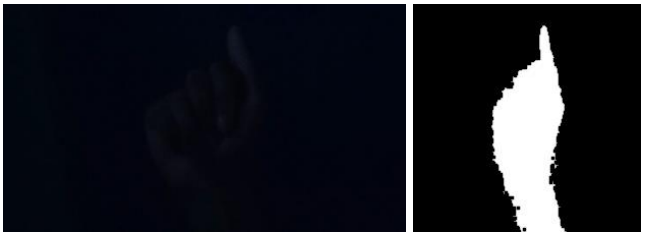
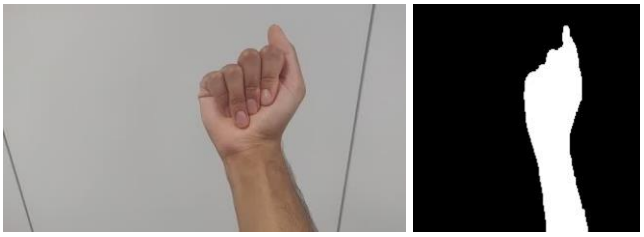


Figure 10 High light image processing input and output



5.2 CNN implementation

The CNN model implemented was a VGG16 based model, with 4 additional layers added, chosen during preliminary testing of the training function. The python file "HGR_train.py" contains the code pertaining to create the CNN model structure and training it.

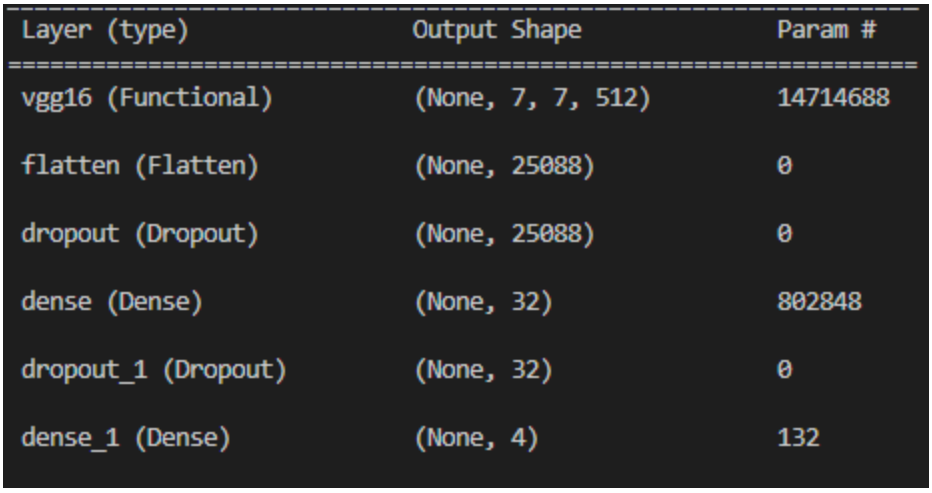
First the training and validation sets and label lists are loaded using numpy, the X sets are cast to float 32, so they match the same format as the Y set. After the relevant data has been loaded, the Keras library is used to build the model. Using the built in VGG16 function, the model can be

loaded in with its pre-trained ImageNet weights; the top 3 layers of the CNN have been removed, which keeps the feature extracting convolutional layers, but allows for the addition of different output layers to be added to fit the function they are being trained for. The first layer added is a dropout layer with a rate of 0.75, followed by a dense layer with a ReLU (Rectified Linear Unit) activation function, then a 0.5 dropout layer; the dropout layers have been implemented in order to reduce the effect of overtraining, as the dataset being seemed to be very susceptible to it during training, the ReLU dense layer contains 32 neurons, which are fed into the final 4 neuron sigmoid activation function, which provides the final classification output for the model. The model architecture for the added layers can be viewed in **Figure 11** below.

As set out in the design, a categorical cross-entropy loss function is used, in tandem with an Adam optimiser, with a learning rate of (0.0001), higher learning rates tended to make the model overfit the data after 1 epoch- alterations to the validation set were also made to reduce this. An SGD optimiser was also tested, however the optimiser tended to reach the same prediction rates as the Adam optimiser, but in more epochs and with swinging rates between them. A callback function is implemented into the training feature, that saves the weights of the model after every epoch as a json file, allowing users to view the training data and to pick which epochs' weights are used. 5 epochs were performed for each dataset, when the model has finished training it and the weights are saved as json files.

Prediction of test images is run in the file "HGR_predict.py". This is a very short file in which the low and high light test images are loaded, then the model weight jsons are loaded in as well. As mentioned, the saved epoch weights can be chosen, alongside the chosen light data ratio. After the relevant files are loaded, the model outputs the sigmoid function array output for each item of the test sets and saves them to an array. An argmax numpy function is used on this array to obtain the predicted class, by calculating which index in each of each embedded list contains the highest value, which in sigmoid activation refers to the predicted class. Finally, a classification report and confusion matrix are made using the scikit-learn library, the report is converted to a pandas dataframe so it can be saved as a CSV file, and the confusion matrix is outputted using matplotlib allowing it to be saved. With these data points, evaluation of the model can take place.

Figure 11 Model architecture



Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dropout (Dropout)	(None, 25088)	0
dense (Dense)	(None, 32)	802848
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 4)	132

Chapter 6

Results & Discussion

This chapter of the paper goes over the performance data produced by the artefact, that being the confusion matrices and the accuracy score of the classifiers- split by what ratio of low light data was used for the training set. Using this data, an evaluation of the artefacts training and testing capabilities can be made, as well as an evaluation of other characteristics of the artefact, such as the image pre-processing and the data set used.

6.1 Confusion Matrices and Model Accuracy

This section will show the confusion matrix of each different low light train set ratio, and the accuracy- with which an evaluation of the classifier can be made.

Table 4 Prediction accuracies

Dataset	Prediction accuracy (to 3 s.f.)			
	0%	10%	25%	50%
Low light	0.9017	0.915	0.882	0.884
High light	0.9107	0.933	0.882	0.898

6.1.1 0% Low light

The model trained on 0% low light training data passed the threshold of 90% accuracy for both low and high light images, with the high light having slightly better accuracy (0.9107 > 0.9017), showing the model to be a good classifier, as well as meaning SMART objective 3 (and by relation 1 and 2) have been successfully achieved. Neither dataset incorrectly classified the A images, showing the model could train on A's features very well, or perhaps suggesting the image pre-processing for gestures I, V and W is not as well suited compared to gesture A. This can be seen in gesture V, it was the most misclassified, with predicted labels in all labels. Overall, despite having only high light training data, it was able to classify the low light images to an accuracy only slightly under that of high light- thanks to the image pre-processing and validation data.

Table 5 0% low – low light confusion matrix

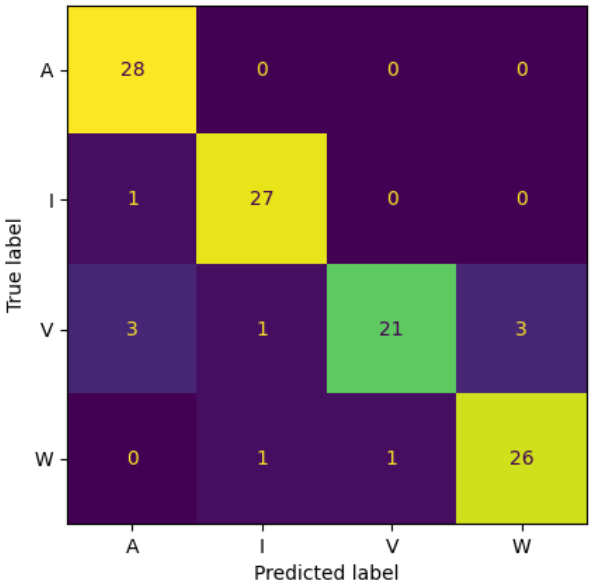
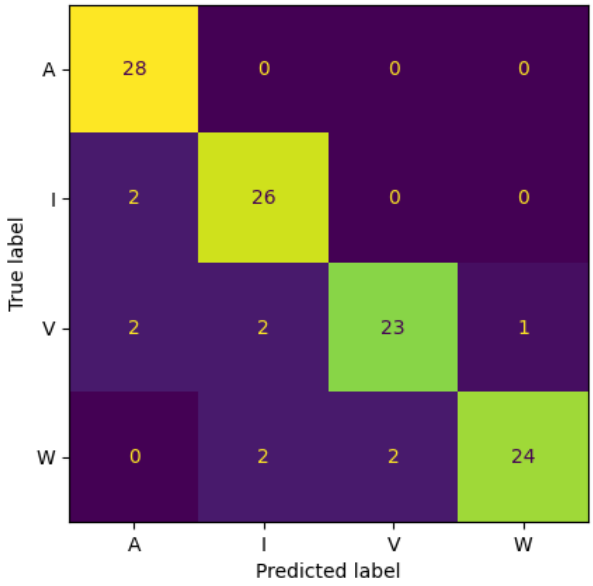


Table 6 0% low – high light confusion matrix



6.1.2 10% Low light

The model trained on 10% low light shows the best accuracy scores for both low and high light testing data- showing introducing some low light data into the train set does improve the effectiveness of the classifier. Once again, A is perfectly classified for both high and low light test sets, with high light V sets also being perfectly classified. However, low light V test images are occasionally mis-classified as W gesture, most likely due to the increased noise. Although the high classification rates look good, the perfect classifications of certain gestures could suggest the test data is too similar to the training and validation data.

Table 7 10% low – low light confusion matrix

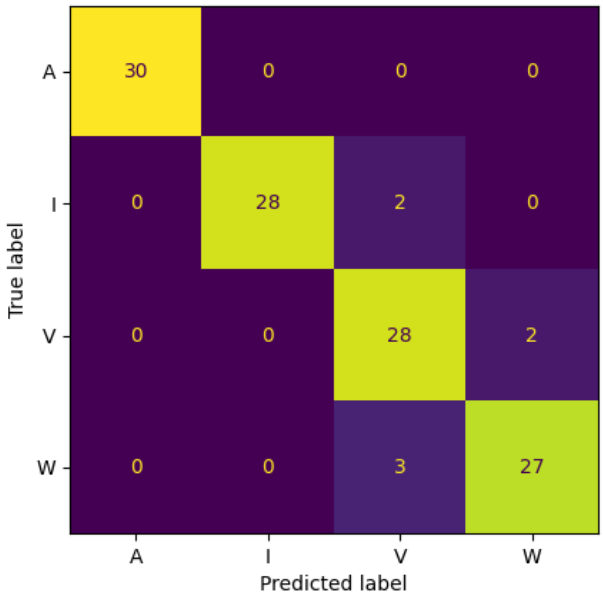
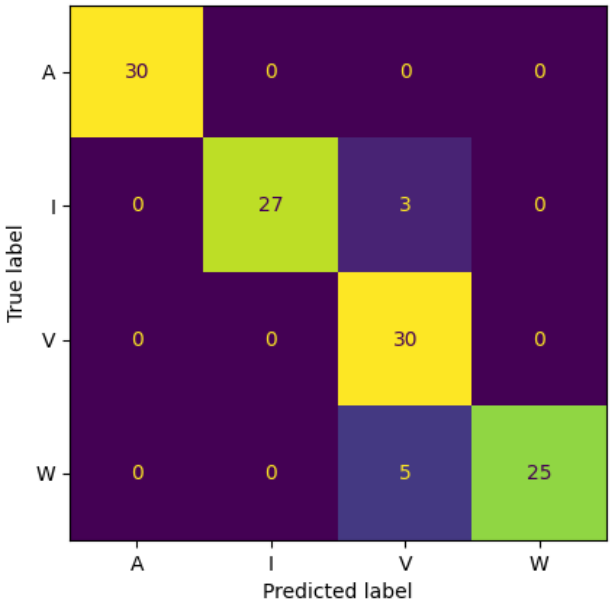


Table 8 10% low – high light confusion matrix



6.1.3 25% Low light

The model trained on 25% low light shows in dip in performance for low and high light accuracy, compared to all other datasets. This could be due to receiving a training and validation set with more noise than the other dataset ratios, causing that noise to be overfitted- causing performance of the classifier to drop. Alternatively, the increase in low light images may be a reason for an increase in noise and therefore worse training and performance. The vast majority of said misclassified images are the V gesture, making up 76% of low light misclassifications and 64% of high light.

Table 10 25% low – low light confusion matrix

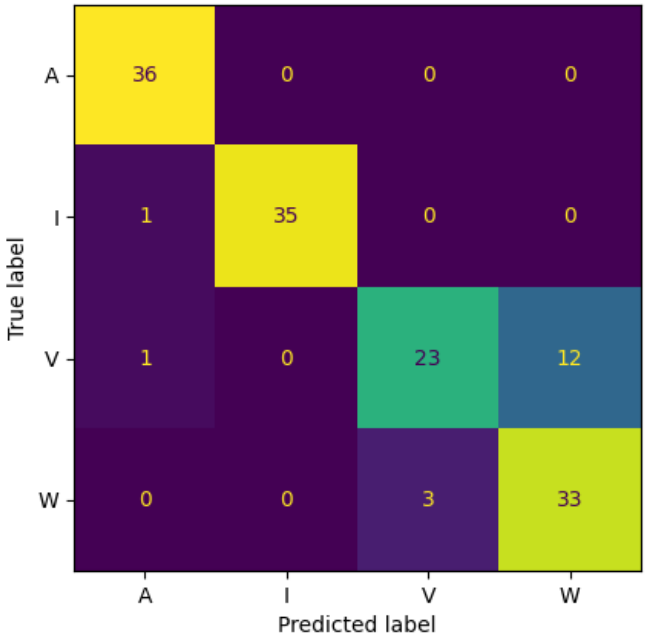
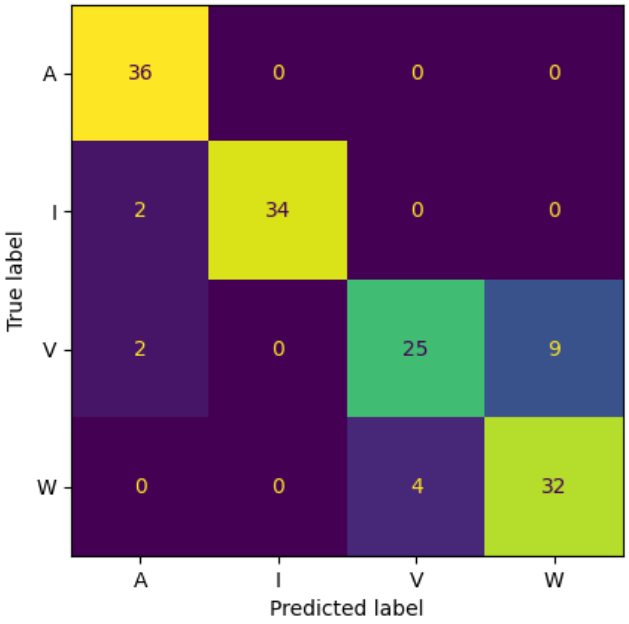


Table 11 25% low – high light confusion matrix



6.1.3 50% Low light

The model trained on a 50% low light train set provides an accuracy slightly higher than the 25% ratio, however still lower than the data splits with containing less low light. With this fact it is safe to assume that there isn't an inverse correlation between low light training data and a classifier accuracy, and that the slight differences in accuracy are most likely from the variations sets of images used for training, validation and testing This can further be seen in this classifier having many occurrences of classifying I images as A, whereas other classifiers only show slight issues in doing so.

Table 12 50% low – low light confusion matrix

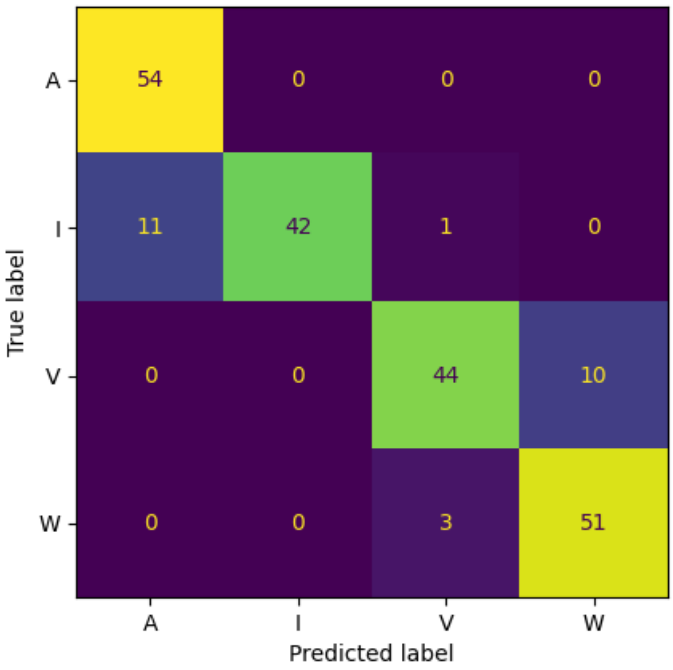
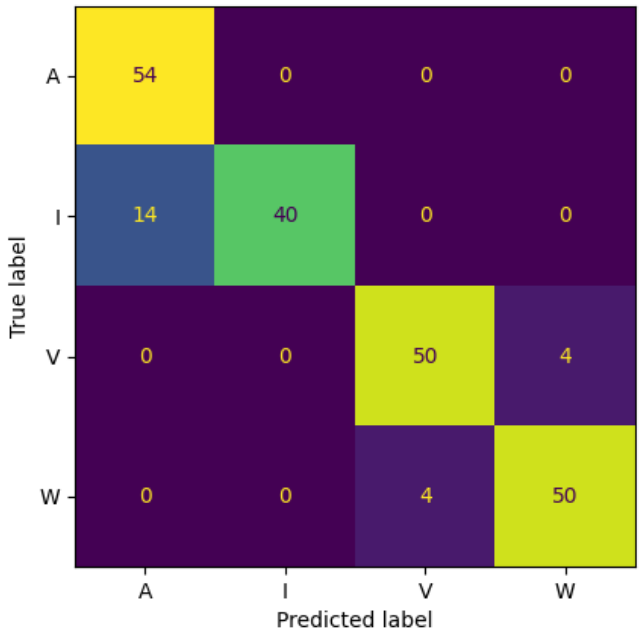


Table 13 50% low – high light confusion matrix



6.2 Aims & Objectives Met

Both aims have fully been achieved, a CNN has been produced that can classify hand gesture images in low and high lighting conditions, and the data has been evaluated in this section of the paper. Objective 1 was fully achieved, by creating the dataset using a phone camera. Objective 2 has been partially fulfilled, as high light images are fully segmented, however low light images tend to have enough noise that prevents a full image from being segmented – even with the techniques implemented to reduce its effects. As mentioned in **6.1.1**, a classifier with an accuracy 90% or higher has been successfully trained meaning objective 3 is fully achieved. Objective 4 has been partially achieved, as the classifier should be tested on further data that is more foreign in order to evaluate its effectiveness properly. Finally, objective 5 has been achieved fully, as both aims were achieved, and all sections of the brief have been satisfied.

Chapter 7

Conclusion

Reflecting on this paper, it has been relatively successful, despite the changes in project focus, a final artefact that achieves all the aims set out was still able to have been produced. If more in depth research into the area was done earlier, such as before the interim and before the project proposal, a more cohesive plan could have been made at an earlier point, which would allow for the topic of low light hand gesture recognition to be explored more thoroughly. This could be done, for example, by implementing a technique relying on purely vision and another relying on depth based sensor.

In terms of the methodology, an agile based approach would have served the project better, the ability to be flexible and adapt to changes would've been very useful with dealing with the changing project focus well. The project plan was followed almost exactly, with only slight changes to the image processing side of the artefact.

In terms of improvements that could be made onto the artefact in the future, the model would greatly benefit from a larger set of data, including differing volunteers for the dataset and more variations in lighting levels. Also, a technique that is robust to varying light angles and colours, on top of light levels could further the span of the project in a useful way. This could include a more advanced image processing technique that can also segment hands from larger images, that would further increase its application in real world scenarios.

In conclusion, hand gesture recognition system has successfully been produced, that can also classify images in low light through a HSV colour space method.

References

Gonzalez, R.C., Woods, R.E. (2018) *Digital Image Processing*. 4th edition. Pearson, New York, NY

Ai, S., Kwon, J. (2020) *Extreme low-light image enhancement for surveillance cameras using attention U-Net*. In: *MDPI*. Multidisciplinary Digital Publishing Institute. Available at: https://www.mdpi.com/1424-8220/20/2/495

Tsarouchi, P., Athanasatos, A., Makris, S., Chatzigeorgiou, X., Chryssolouris, G. (2016) *High Level Robot Programming Using Body and Hand Gestures*. In: *Procedia CIRP 2016*, Volume 55, Elsevier, Amsterdam, Netherlands

Mahmood, H.R. (2022) *Project Proposal: A Study on the Weaknesses of the Image recognition of Human Hand Gestures*, University of Lincoln, School of Computer Science

Alhafdee, A.H., Abbas, H., Shahadi, H.I. (2022). '=*Sign Language Recognition and Hand Gestures Review*. In: Kerbala Journal for Engineering Science, 2(4), pp. 209-234.

Mahmood H.R. (2023) *Interim Report*, University of Lincoln, School of Computer Science

Zhu, C. and Sheng, W. (2011*) Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living.* In: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 41(3), pp. 569–573

Nikam, A.S. and Ambekar, A.G. (2016) *Sign language recognition using image based hand gesture recognition techniques*. In*:* 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, India, 2016, pp. 1-5

Tsai, T.H., Ho, Y.C. and Chi, P.T. (2022) *A Deep Neural Network for Hand Gesture Recognition from Rgb Image in Complex Background*. [Preprint] Available at SSRN 4279994, pp. 1-8

Luo, Y., Cui, G. and Li, D. (2021) *An improved gesture segmentation method for gesture recognition based on CNN and YCbCr*. In: Journal of Electrical and Computer Engineering, 2021, pp. 1–9

Chaudhary, A. and Raheja, J.L. (2018) *Light invariant real-time robust hand gesture recognition*. In: Optik, 159, pp.283-294.

Perimal, M., Basah, S., Safar, M. and Yazid, H. (2018) *Hand-Gesture Recognition-Algorithm based on Finger Counting.* In: Journal of Telecommunication, Electronic and Computer Engineerin*g (JTEC)*, 10(1-13), pp. 19–24.

Huang, H., Chong, Y., Nie, C., Pan, S. (2019) *Hand Gesture Recognition with Skin Detection and Deep Learning Method*. In: Journal of Physics: Conference series, 1213(2), p.022001

Sahoo, J.P, Ari, S. and Patra, S.K. (2019) *Hand Gesture Recognition Using PCA Based Deep CNN Reduced Features and SVM Classifier*. In: *2019 IEEE International Symposium on Smart Electronic Systems (iSES)*, Rourkela, India, pp. 221-224

Suarez, J. and Murphy, R.R. (2012) *Hand gesture recognition with depth images: A review.* In: 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, IEEE

Khaleghi, L., Artan, U., Etemad, A. and Marshall, J.A. (2022) *Touchless Control of Heavy Equipment Using Low-Cost Hand Gesture Recognition.* In: IEEE Internet of Things Magazine, vol. 5, no. 1, pp. 54-57

Leite, M. & Parreira, W.D., Fernandes, A. and Leithardt, V. (2022) *Image Segmentation for Human Skin Detection*. In: Applied Sciences. 12. 12140. 10.3390/app122312140.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012) *ImageNet Classification with Deep Convolutional Neural Networks*. In: Advances in Neural Information Processing Systems 25, Curran Associates, Inc., pp. 1097--1105

Simonyan, K. and Zisserman, A. (2015) *Very deep convolutional networks for large-scale image recognition*. In: *arXiv.org*. Available at: https://arxiv.org/abs/1409.1556

Nuzzi, C., Pasinetti, S., Pagani, R., Coffetti, G., Sansoni, G. (2021) *HANDS: a dataset of static Hand-Gestures for Human-Robot Interaction*. In: Mendeley Data, V2

Elliott, G., Meehan, K. and Hyndman, J. (2021) *Using CNN and Tensorflow to recognise 'Signal for Help' Hand Gestures*. In: *2021* IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON*)*, New York, NY, USA, 2021, pp. 0515-521

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. (2009) *Imagenet: A large-scale hierarchical image database*. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255.

Kingma, D.P., Ba, J. (2015) *Adam: A Method for Stochastic Optimization.* In: 3rd International Conference for Learning Representations, San Diego

Kommineni, V. (2019) *How to use transfer learning for sign language recognition*. In: freeCodeCamp, Available at: https://medium.com/free-code-camp/asl-recognition-using-transfer-learning-918ba054c004