On Efficient Computer Vision Applications for Neural Networks

Rachel Mae Billings

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Alan J. Michaels, Chair
Guoqiang Yu
A. Lynn Abbott

March 17, 2021
Blacksburg, Virginia

Keywords: machine learning, neural networks, image processing, mask recognition,
COVID-19

On Efficient Computer Vision Applications for Neural Networks

Rachel Mae Billings

(ABSTRACT)

Since approximately the dawn of the new millennium, neural networks and other machine learning algorithms have become increasingly capable of adeptly performing difficult, dull, and dangerous work conventionally carried out by humans in times of old. As these algorithms become steadily more commonplace in everyday consumer and industry applications, the consideration of how they may be implemented on constrained hardware systems such as smartphones and Internet-of-Things (IoT) peripheral devices in a time- and power- efficient manner while also understanding the scenarios in which they fail is of increasing importance. This work investigates implementations of convolutional neural networks specifically in the context of image inference tasks. Three areas are analyzed: (1) a time- and power-efficient face recognition framework, (2) the development of a COVID-19-related mask classification system suitable for deployment on low-cost, low-power devices, and (3) an investigation into the implementation of spiking neural networks on mobile hardware and their conversion from traditional neural network architectures.

On Efficient Computer Vision Applications for Neural Networks

Rachel Mae Billings

(GENERAL AUDIENCE ABSTRACT)

The subject of *machine learning* and its associated jargon have become ubiquitous in the past decade as industries seek to develop automated tools and applications and researchers continue to develop new methods for artificial intelligence and improve upon existing ones. Neural networks are a type of machine learning algorithm that can make predictions in complex situations based on input data with human-like (or better) accuracy. Real-time, low-power, and low-cost systems using these algorithms are increasingly used in consumer and industry applications, often improving the efficiency of completing mundane and hazardous tasks traditionally performed by humans. The focus of this work is (1) to explore when and why neural networks may make incorrect decisions in the domain of image-based prediction tasks, (2) the demonstration of a low-power, low-cost machine learning use case using a mask recognition system intended to be suitable for deployment in support of COVID-19-related mask regulations, and (3) the investigation of how neural networks may be implemented on resource-limited technology in an efficient manner using an emerging form of computing.

To Luna and Laika

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Alan Michaels, for his consistent encouragement, saintly patience, and thorough guidance in conducting research throughout my degree and in writing this thesis. I would also like to thank Dr. Yu and Dr. Abbott for their time, counsel, and service on my committee. Additionally, I am grateful to Scott for his unwavering patience and for being a good influence.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Protocol Interface

BRISQUE Blind Referenceless Image Spatial Quality Evaluator

CNN Convolutional Neural Network

CPU Central Processing Unit

EER Equal Error Rate

FFT Fast Fourier Transform

FLOP Floating Point Operations Per Second

FPGA Field-Programmable Gate Array

FPS Frames Per Second

GLCM Gray-Level Co-occurence Matrix

GPU Graphics Processing Unit

ILSVRC ImageNet Large Scale Visual Recognition Challenge

IoT Internet-of-Things

IQA Image Quality Assessment

IRB Institutional Review Board

KDE Kernel Density Estimate

LFW Labeled Faces in the Wild

LIF Leaky Integrate-and-Fire

LLE Locally Linear Embedding

LoG Laplacian of Gaussian

MFDD Masked Face Detection Dataset

MLP Multilayer Perceptron

MSB Most Significant Bit

MSCN Mean-Subtracted Contrast Normalized

MTCNN Multitask Cascaded Convolutional Neural Networks

PCA Principal Components Analysis

RAM Random Access Memory

RGB Red Green Blue

RMFD Real-World Masked Face Dataset

RMFRD Real-World Masked Face Recognition Dataset

SIFT Scale-Invariant Feature Transform

SLLFW Similar-Looking Labeled Faces in the Wild

SMFRD Simulated Masked Face Recognition Dataset

SNN Spiking Neural Network

STDP Spike-Timing-Dependent Plasticity

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

As mobile technologies and Internet-of-Things (IoT) devices such as Amazon Alexa and Google Home, smartwatches, and smartphones become more prevalent, the importance of lightweight, low-power, and high-throughput computing has drastically increased. Applications for machine learning in regards to tasks such as natural language processing, computer vision, and biometrics for security applications have experienced a popularity boom, due in part to the increase in computational power afforded by modern processors and graphics processing units (GPUs) that makes training large neural networks feasible. Face detection and recognition is an oft-relied upon utility for modern society with a range of uses including unlocking smartphones [1], identifying possible criminals [2], enforcing common resource sharing [3], and improving accessibility for the impaired [4]. Consequently, maximizing the efficiency of its computation and our overall confidence in its results is pertinent as facial recognition becomes implemented within mobile devices. Additionally, the performance analysis of neural network-based object classification and detection models is well-established for several models and existing datasets [5]. However, pragmatic evaluations of real-life hardware and software implementations of these models in which unexpected situations may arise are seldom considered [6].

One example in which these unexpected situations may occur is illustrated by the following

scenario. Since early 2020, the coronavirus (COVID-19) pandemic [7] has impacted society on an unprecedented scale and contributing to economic uncertainty [8] and poor mental health [9]. The use of masks that cover the nose and mouth of an individual helps to mitigate the spread of the virus and is currently required in many settings such as grocery stores [10]. In early 2021, Dr. Anthony Fauci, the chief medical adviser to the United States president, stated that mask-wearing will potentially continue into 2022 [11]. The sometimes hazardous duty of manually enforcing mask-wearing compliance ([12], [13], [14]) and the possible long-term use of masks implicates the need for an automated 'gatekeeper', a prime candidate for convolutional neural network applications. A depiction of this system is shown in Figure 1.1, where an individual wearing a mask is given the go-ahead green light and an individual not wearing a mask is indicated with a red light; a camera is assumed to be placed opposite to the individual. The performance of this gatekeeper system is affected by many variables such as the environmental variability of its setting (e.g., weather), the capabilities of the hardware it is implemented on, and uncontrollable factors such as erratic human behavior.

An approach to mitigate these effects, as explored in Chapter 3 and Chapter 4, is to insert an additional step into the set of image processing procedures typically performed before inference that lessens the performance impact of images less likely to be classified correctly. One way to characterize how images may be likely to be classified incorrectly is through quality assessment. Neural networks utilized for image-related tasks are frequently trained on datasets largely comprised of high-quality images that lack distortions such as blur and noise, sometimes incurred during the image capture and file transfer process by camera hardware; such distortions can result in significant model performance degradation [15].

Additionally, traditional computer vision approaches such as used to calculate these metrics are often more efficient than deep learning-based approaches, which excel at solving complex

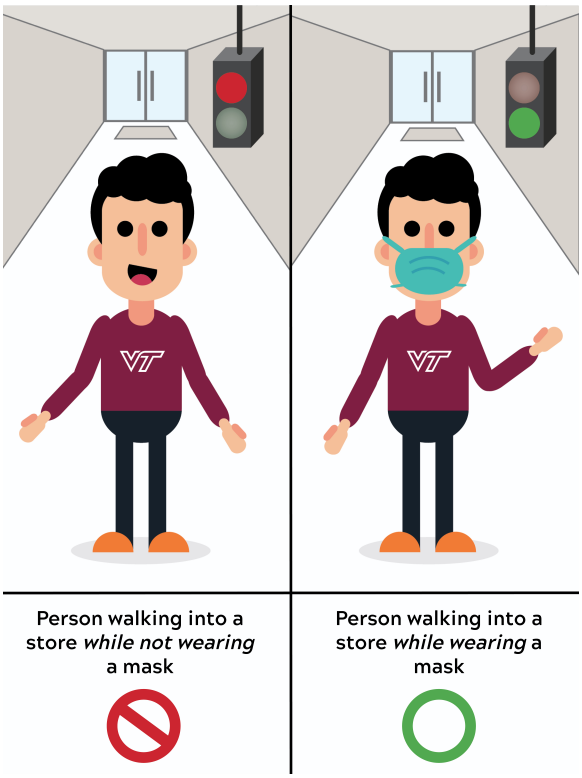

Figure 1.1: 'Gatekeeper' system for mask recognition system developed in Chapter 4.

problems with high-dimensional data inputs but are computationally expensive [16]. For example, a comparison of feature extraction using an image gradient-based algorithm versus a popular neural network model [17] showed that the former consumed about a tenth of the power. However, the neural network-based approach was over twice as accurate. The nearly-linear correlation between accuracy, computational complexity, inference time, and resource consumption has been established amongst several other neural network models [18]; this relationship communicates the importance of finding ways to implement emerging and existing deep learning methods in an efficient manner by supplementing inference with other algorithmic approaches, creating new frameworks and models, and by developing specialized software and hardware.

Shifting to a parallel topic of perhaps even greater importance, Moore's Law states that

the quantity of components that can be placed on an integrated circuit will increase by a factor of two every two years [19] - but nothing lasts forever. The speculated end of Moore's Law combined with the ever-growing need for more capable on-the-fly computing stipulates that further advancements in existing algorithms and hardware are required for continued deeper integration of machine learning in society. A few possible research directions for the future include analog-based application-specific integrated circuits (ASICs) as hardware accelerators [20], quantum computing [21], and neuromorphic hardware [22].

The work developed in this thesis shares an underlying question in common: how can subsets of machine learning algorithms be implemented on resource-limited systems for inference tasks in ways that provide results efficiently?

1.2 Research Contributions

Seeking to answer the prior question, the primary contributions of this research are threefold:

- A framework for improving the efficiency of face recognition models by conserving power and time through analysis of properties used to assess the quality of an input image prior to inference.
- The development of a binary classification system incorporating neural networks that aims to provide real-time output decisions as to if an individual is wearing or not wearing a mask, as well as the creation of a novel image dataset used to perform a preliminary evaluation of the classification system and identify biases. The dataset includes masked and unmasked individuals encouraged to 'break the machine' via presentation attacks with outlandish appearances.

  – R. Billings and A. Michaels, "Real-time mask recognition," *ACM Transactions*

on Internet of Things (TIOT), 2021. In review by ACM Transactions on Internet of Things (TIOT) journal

- An appraisal of the benefits of converting existing neural network models to spiking neural network models and of the feasibility and practicality of implementing the converted models on low-power, low-cost hardware.

1.3 Outline

The rest of the document will proceed as follows:

- An introduction to image processing techniques, object detection and classification, neural networks, and neuromorphic hardware and software is given in Chapter 2. Pivotal findings from past and present publications are summarized in the fields of face detection, face recognition, applied machine learning for efficient low-power inference tasks, and spiking neural network algorithms and implementations.

- Chapter 3 introduces a framework that preserves the time and power consumption used by a facial recognition model during inference through estimating the likelihood that a photo of a person, within the currently observed environment, will lead to a correct facial classification. Prior to inference, the performance estimation for images is acquired through a flexible set of low-cost metrics, which have been chosen to indicate whether the presented image is representative of the training data set and likely to produce a correct result. These input image qualifications estimate may then be used to accept or reject an image as being likely classifiable, or simply flag the facial recognition's result as being more prone to error. Furthermore, the proposed framework is believed extensible to identifying adversarial images.

- Motivated by the COVID-19 pandemic, in Chapter 4 a binary image classification system using convolutional neural network models is developed in order to determine whether or not an individual is wearing a mask. The assembly of a novel face dataset including presentation attacks is detailed. The performance of these models in both everyday and unique scenarios is evaluated as part of validating the practical next steps for guaranteeing sufficient performance and computation efficiency, ultimately leading to an implementation of a real-time, low-power mask recognition system ready for real-life deployment. A slightly modified version of this chapter was submitted for publication [23].

- Chapter 5 outlines strategies that could be used to deploy ANN-to-SNN conversion algorithms on neuromorphic hardware, in software simulator tools, and other hardware systems for the purpose of comparing the accuracy, time, and power consumption during training and inference across assorted neural network models.

- A summary of the pertinent findings of Chapter 3, Chapter 4, and Chapter 5, a discussion of future research directions, and the concluding notes of this document are composed in Chapter 6.

Chapter 2

Literature Review

Before progressing further, it is imperative to have some context for the content of the following chapters, the central ideas of which include subjects that sprawl across several fields of scientific and engineering research including machine learning, computer vision, and hardware. Although these topics are widely-ranging in both historic and modern academic literature, this chapter provides insight into the basic concepts and relevant research in their applicable subfields including neural networks, computer vision, face detection and recognition, and efficient hardware implementation of learning systems.

2.1 Neural Networks

Stated simply, machine learning is a term that encompasses several types of algorithms for solving problems with computers and math. Neural networks are a type of machine learning algorithm that can be applied to myriad tasks, such as image classification, audio processing, and object recognition, and can have many different architectures.

A simple neural network consists of an input layer, a hidden layer, and an output layer [24]. Each layer contains a number of nodes called neurons that each have weights and biases. Fully-connected neural networks, as one can guess by the name, have neurons that are all interconnected with one another from one layer to the next. Forward propagation and backpropagation are two of the central processes used to train neural networks. In

forward propagation, the sum of weighted connections and biases from neurons in a layer are input to an activation function, which computes the output to be passed to the next layer. Types of activation functions include sigmoid activation [25] and rectified linear unit (ReLU) activation [26]. Gradient-based optimization techniques are used with the backpropagation algorithm, which uses the current weights and biases at each layer to find the gradient of the cost function, to modify the weights in order to minimize the error between the expected and actual outputs, specified by the loss function ([27], [24]).

Deep learning is a phrase referring to machine learning model architectures composed of several layers of representations of information that allow for the decomposition of unwieldy, complex data into smaller, simpler data [27]. Deep learning breaks down high-dimensional data inputs similarly to how one could deconstruct a deep dish pizza if it were an input. At first, the pizza is a huge quantity of unrecognizable ingredients thrown on top of each other and baked. Once the pizza is sliced, you only have to work with a manageable fraction of the pie. If you peel apart the layers of a single slice, it's easy to see the pepperoni, cheese, sauce, spices, and crust as components of the whole pizza, now feasible to eat in bite-size pieces; this process of discovering how the ingredients of the pizza work together as a whole is akin to how a neural network learns the features of the data it is given.

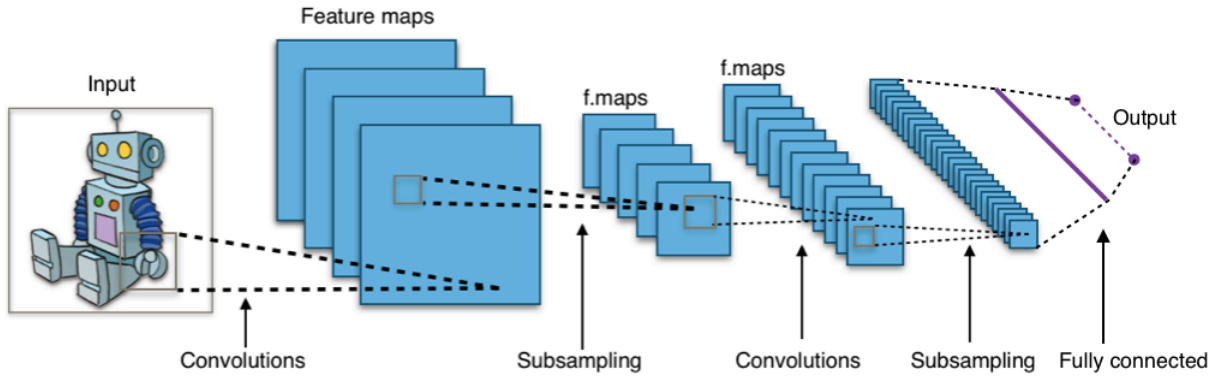

Figure 2.1: A visual depiction of the architecture of a convolutional neural network. 'f.maps' refers to a feature map and subsampling refers to the pooling operation. Source: [28]

A convolutional neural network (CNN) is a type of deep neural network that uses convolutions with filters (or kernels) of weights between layers to obtain feature maps at the output of each layer. Due to the use of convolution in CNNs, sparse matrices and weight and bias sharing make computations more efficient than in general fully-connected neural networks. Additionally, *pooling* layers are used to reduce the size of feature maps by taking the average, mean, or other statistic of output data, thus further reducing calculations. These features of CNNs make training and inference on large inputs (such as images with thousands of pixels) feasible. [29]. A visualization of a simple CNN architecture is shown in Figure 2.1.

A multitude of CNN architectures have been conceived for varying purposes; examples of these are described in subsection 2.1.1. Software libraries for training and inference using neural networks have been implemented in numerous libraries in several programming languages; however, the work of this thesis primarily used Python. A few common tools with versions written in Python include TensorFlow [30], a library by Google that represents neural network models as data flow graphs composed of nodes and edges; Keras [31], a Python API that allows for a more high-level interface to TensorFlow; and PyTorch [32] a library that uses data structures called *tensors* to work with neural network models.

2.1.1 Model Architectures

The evolution of neural networks is rather convoluted, but a pivotal milestone occurred at the beginning of their rise to omnipresence in 2012. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [33] is a competition for object detection and classification using subsets of images from ImageNet [34], a dataset comprised of over 14.1 million images and over 21000 object classes nuanced as to include categories difficult for the author to visually distinguish from one another such as ‘Border collie’, ‘collie’, and ‘Shetland sheepdog’ (these

are actually all unique dog breeds) or 'cucumber' and 'zucchini'. In the 2012 competition, a CNN now dubbed AlexNet [35] won the top-5 classification and localization contests with respective error rates of 16.4% and 34.2%. Although these numbers pale in comparison to modern neural network accuracy reports on benchmark datasets, they were groundbreaking at the time, helping to spark further deep learning research and development. In the past few years, remarkable models that have competed in ILSVRC include GoogLeNet [36], VGGNet [37], and ResNet [38]; the former two are utilized in Chapter 3 and Chapter 4, respectively.

GoogLeNet [36] competed alongside VGGNet in the 2014 ILSVRC using a CNN with 'Inception modules', in which multiple filters of varying sizes are placed in between layers; increasing the depth and width of the model resulted in high performance without adding significant computational complexity. The GoogLeNet model is also interesting due to its inspiration source and name: the internet meme "we need to go deeper", based off the movie *Inception* [39].

VGGNet [37] is comprised of a number of convolutional layers utilizing ReLU activations, max-pooling layers, and a softmax classification layer. Notably, VGGNet showed that good performance can be achieved with a 'deeper' architecture made of many layers that use small filters (each 3x3), but is otherwise comparable to a standard CNN. Common configurations of VGGNet are VGG-16 with 13 convolutional layers and VGG-19 with 16 convolutional layers, each appended with 3 fully-connected layers.

ResNet [38] is an architecture that incorporates residual blocks with 'shortcut' connections to skip layers. Inputs from the previous layers are preserved, then added to the outputs of the succeeding layers. Configurations of ResNet include 18, 34, 50, 101, or 152 convolutional layers. In each of the configurations, after the initial convolutional layer one max-pooling layer is used; before the final fully-connected layer, one average-pooling layer is used. ResNet's use of 'deep residual learning' improves performance whilst decreasing the overall network

complexity and the number of computations. The ResNet model is overall much smaller and more efficient than VGGNet - for example, ResNet-50 has approximately 25 million parameters and utilizes 3.8 billion FLOPs, whereas VGG-16 has 138 million parameters [37] and uses 15.3 billion FLOPs [38].

Another common model architecture central to the work of this writing is MobileNetv2. MobileNets [40] are a group of neural network models that were designed to be implemented on resource-constrained devices such as cellphones (hence, the name 'MobileNets'). The original model is referred to as *MobileNetv1*, succeeded by *MobileNetv2*. MobileNets decrease model size and reduce calculations by factoring convolutions between layers. The factoring process, referred to as *depth-wise separable convolutions*, first computes *depthwise* convolutions on each input channel with a filter. The outputs from the depthwise convolution are combined with a 1x1 *pointwise* convolution. The use of depth-wise separable convolutions in MobileNets reduces the computational cost to around 9 times as compared to models using regular convolutions. MobileNetv2 [41] also uses depth-wise separable convolutions, but also introduces bottleneck by appending an additional linear 1x1 convolution to a layer. Similarly to ResNet, shortcut residual connections are used between these layer blocks; however, MobileNetv2 'inverts' them by linking connections to the 1x1 bottlenecks as opposed to the deepest layer. Evaluation on ImageNet showed that the base MobileNetv2 model utilized 3.4 million parameters and obtained top-1 72% accuracy; MobileNetv1 utilized 4.2 million parameters and obtained 70.6% accuracy. Whilst describing neural networks, this section has made mention of image-related tasks such as object classification and referenced a well-established image dataset. To further this context, the subject of computer vision is described in Section 2.2.

2.2 Computer Vision

The fields of computer vision, computational photography, and image processing are often heavily intertwined with each other as well as with machine learning in past and current research. Visual information is complex, and although humans can typically make sense of the world around us with our eyes, the creation of machines which can do the same with cameras and sensors is a challenging notion. *Computer vision* involves taking into account the physics and statistical nature of images to formulate algorithms that dissect images into accurate representations of the actual scene, and to analyze them in an efficient manner [42]. *Computational photography* is a subject including many of the topics in computer vision with a focus on techniques used with cameras, image capture, and digital image processing digital photography, including sensors, denoising, and color correction [43].

To elucidate how the aforementioned web of research is weaved together, the basic process of a common supervised learning application is described from the ground up. In this context, an instance of input data is a photograph that was at some point in time taken by a digital camera developed using the principles of optics and including typically CMOS-based sensors to capture light information. Photographs aim to accurately represent visual information using a 2D array of pixels representing the height and width, or resolution, of an image and a third dimension to represent red, green, and blue color channels. However, this representation may be negatively impacted by factors such as blur and noise acquired during the image acquisition process and by environmental factors such as poor lighting. Such degradations may be improved via image processing techniques such as histogram equalization and filtering to reduce noise or blur. [44]

Given an image, object recognition is a task within computer vision of finding specific, known objects (such as a fingerprint scanner given a fingerprint previously entered into its

database) or multi-class classification, which seeks to sort images into labeled categories [42]. Object detection aims to locate the areas of an image where an object is most likely to exist [45]. Object recognition is essentially a pattern recognition problem as applied to image data, as both areas seek to classify objects into categories. As such, object recognition problems often are solved using the same fundamental steps as used in a general classification task [46] [47]: a preprocessing step to extract useful information from input data such as dimensionality reduction, the selection of relevant features that can be used by the classifier, the selection and development of an appropriate classifier, and performance evaluation to evaluate how well the classifier can predict the right solution. Depending on the complexity of the classification task, the features may be based on simple factors such as the distributions of gray levels in image histograms [46], well-known methods in computer vision such as Canny edge detection [48] or scale-invariant feature transform (SIFT) [49], or learned with a neural network model [50]. Traditional algorithms and deep learning-based approaches to object recognition can be used independently of one another or as a combination, and both approaches have their benefits and downsides; for example, neural networks are typically much more accurate, but also more computationally intensive. [16] The application of several detection and classification methods based on each of these approaches specifically to faces is described in Section 2.3.

2.3 Face Detection and Recognition

Our face is used to uniquely identify us in numerous ways, such as with the photos used in driver's licenses, passports, and social media. Consequently, one of the most popular subjects for which object recognition algorithms are used is the human face and is a highly active (and oft controversial) area of research due to its challenging nature and widespread relevance.

One significant difficulty within face detection and recognition systems is the presence of occlusions, which introduce large variations in typical features of the face. A wide variety of methods have been formulated and refined for improved accuracy on occluded faces in the past few decades, but the ongoing challenges they present are explored more in-depth in Chapter 4. On a tangential note, the intentional evasion of automatic face identification is an intriguing area of research, motivated by privacy concerns. Evasion approaches can be categorized into two main groups: (1) methods that alter previously-captured photos so that they cannot be accurately analyzed [51] and (2) methods that alter an individual's appearance so that the face cannot be detected and/or recognized [52].

There are several mechanisms for face *detection* [53] [54], which seeks to identify the presence and location of a face. Face *recognition* [55] seeks to find an identity for a face. Face recognition is the focus of Chapter 3, while face detection is central to the work of Chapter 4. Prior to recognition tasks, faces in images must be detected and bounded using methods including feature-based object detection [56], local binary patterns [57] [58], and deep learning-based approaches [59], [60], [61].

One of the most influential of these approaches is Viola-Jones face detection [56], which is an object detection algorithm for identifying faces in an image using a Haar feature-based cascade of classifiers. Haar features [62] are the difference of pixel intensity sums taken across adjacent rectangular subwindows in detection windows; these features are then used to find regions such as the eyes and nose of a human face based on their intensity levels. Integral images are used to represent the input images, which decreases the computation time of calculating Haar features. A modified Adaboost algorithm is used for classifier construction by selecting and utilizing the most relevant features, based on a threshold set by positive and negative image examples [63], [64]. The classifiers are then cascaded, which ensures a low false-positivity rate.

On the deep learning-based approach side, one method is MTCNN (multitask cascaded convolutional neural networks) [60], a framework that uses cascaded CNNs to infer face and landmark locations from images. Three stages of CNNs are used to find and calibrate bounding boxes as well as find coordinates for the nose, left and right mouth corners, and left and right eyes of a human face.

In regards to face recognition, OpenFace [65] is an open-source software library that incorporates ideas from architectures developed by two of the most notable monoliths in artificial intelligence: Facebook's DeepFace [66] and Google's FaceNet [67]. DeepFace [66] was trained on over 4 million images; FaceNet [67] was trained on over 200 million images. Although these numbers may seem hugely excessive (and they are atypically large), the training stage of model development typically requires considerable quantities of data.

2.3.1 Datasets

One of the primary obstacles to the development of deep learning models is the vast amount and characteristics of the data required for training, both of which are important to consider as they will impact the robustness and accuracy of the final model [68]. Consequently, hundreds of face datasets for evaluating model performance in general and specific ways, such as for diverse pose and illumination, have been assembled in recent years.

A few notable examples include VGGFace [69], which has 2.6 million images across 2,622 subjects (1000 images each), VGGFace2 [70], consisting of 3.31 million images across 9,131 subjects (approximately 300 images each), and IARPA Janus Benchmark-C [71], which contains 31,344 images of 3,531 diverse subjects with metadata and annotations.

Labeled Faces in the Wild (LFW) [72] contains 13,000 images of faces detected with the Viola Jones face detection algorithm from the web labeled with names; 1680 of the subjects

pictured have at least two distinct images. There are many variations of LFW that may produce better or worse results when used for evaluation model performance. Deep Funneled LFW [73] is a pre-aligned variant that has produced better results than the unaligned LFW dataset. As performance of models for face recognition is nearly saturated when tested on LFW, the Similar-Looking LFW (SLLFW) Database ([74], [75]) was curated to increase the difficulty of facial verification by replacing negative pairs of faces (images of two different people) with more similar looking faces. For example, a pair of images in the original LFW dataset consisting of a blonde woman and a man with a dark complexion may be replaced by two blonde women [74].

2.3.2 Mask 'Recognition'

An important clarification for understanding the subject matter of Chapter 4 is the difference between terms such as object recognition, detection, face recognition, and 'mask recognition'.

'Object recognition' is an ambiguously-defined term sometimes used interchangeably with other classification terminology in literature, and can best be understood as an umbrella for the process of finding and recognizing entities in an image as previously described in Section 2.2 [42]. These entities can be classified into assorted categories via labels. Object detection includes the classification and localization of potentially *multiple* entities within an image [76].

Face detection is typically a preliminary step in face recognition. Face recognition can be interpreted as a special type of object recognition; given an image of a person's face, face recognition may seek to classify the face as belonging to a single identity in a set of known identities (*1:N* matching or identification) or to classify the face as belonging or not belonging to a certain known identity (*1:1* matching, or verification) [55]. *Mask recognition* is a term

used in Chapter 4 to describe the binary classification ('yes' or 'no') question of whether or not a person is wearing a mask; the approach used relies on face detection, but does not address face recognition as specific identities are not considered.

2.4 Low-Power Hardware

Deep learning-based approaches to computer vision problems are a complex yet exciting area of research that may beneficially impact society in countless ways in the coming decades. Making them more power efficient is of vital importance to their incorporation into low-power systems such as IoT devices, spaceflight systems, and medical devices. Due to its accessibility, one notorious example of a low-power hardware device is the Raspberry Pi [77] is a miniature computer well-reputed for its active community of developers. Most Raspberry Pi models are between $30 to $50, making them useful for strictly-budgeted projects. All devices utilize a 5.1V supply and consume 100mA to 1.2A, depending on the model and use case. Additionally, numerous peripherals are produced for the Raspberry Pi, including 5-, 8-, and 12-megapixel camera modules consuming approximately 250mA. In literature, Raspberry Pi devices have been used for applications such as real-time emotion recognition [78] [79], face detection and recognition [80], and smart device systems [81].

For computer vision scenarios, event cameras may become an ideal peripheral, although their development is still in its infancy. Event cameras capture images asynchronously as they record variations in brightness as separate 'spikes'. Standalone, they typically use less than 10mW of power; with a processor, the entire system can use less than 100mW of power. [82]

Another category of hardware systems with significant potential to improve current system power consumption is neuromorphic computing. Examples of neuromorhpic hardware include the Intel Loihi [83], a digital ASIC that uses about 23.6pJ per operation, and the

IBM TrueNorth [84], a digital ASIC consuming only 70mW of power and about 26pJ per operation. Research into applications for neuromorphic hardware includes topics such as EEG analysis [85], emotion recognition [86], face recognition [87], occluded face recognition from video [88], smell classification, [89], security and defense use cases such as target detection and tracking [90], and speech recognition [91]. Further background on neuromorphic hardware and its benefits is provided in Chapter 5.

2.5 Open Research Needs

In brief, the preceding sections within this chapter have reviewed just a small portion of the elements of fields such as neural networks, computer vision, face detection and recognition, and low-power hardware implementations. It is clear that the fields discussed are areas of active research with significant potential for further future innovations. This section seeks to introduce a few areas for such innovations that are relevant to the ensuing chapters:

- The combination of established computer vision algorithms which are not deep learning-based with neural network architectures may allow each approach to augment the other's strengths, such as the high accuracy of neural networks and the relatively small complexity of algorithms for image analysis tasks that do not utilize neural networks.
- Face detection and recognition systems are increasingly used in a variety of situations, but the efficacy of their use in real-world situations, which can vary significantly from tests performed in a controlled environment, may be an area for further analysis and improvement.
- Emerging low-power hardware platforms are promising in regards to improving on the efficiency of existing platforms. The analysis of these platforms to establish their

practicality and comparison between platforms may serve to bolster their adoption in consumer applications.

Chapter 3

Architecture for Efficient Facial Recognition

Prior to classification, faces in images generally must be detected, aligned, and undergo feature extraction, which can be quite costly in terms of time. For example, the DeepFace recognition framework by Facebook as evaluated on a single core Intel 2.2GHz CPU in the publication utilizes 0.33 seconds total per image; around half of this time is used just for feature extraction [66]. The aim of this chapter is to introduce a framework for improving the efficiency of inference in deep learning models for object recognition, specifically targeting facial recognition and verification tasks. Determining whether or not an image is likely to be valid or recognizable by a neural network by evaluating it with less computationally-intensive methods prior to inference can conserve time and power; this is especially useful for applications on mobile devices, where time and power may be in short supply.

An architecture for the framework developed in this chapter is presented in Figure 3.1. With the following desired functions in mind, the design of the framework was shaped to:

- Use a performance estimate to determine how likely a decision made by a neural network model will be valid before the decision is actually made in order to save computational power and processing time.
- Compare the processing load of the performance metric output and the neural network.

Ideally, the time and power utilized by the former is be lower than that used by the latter.

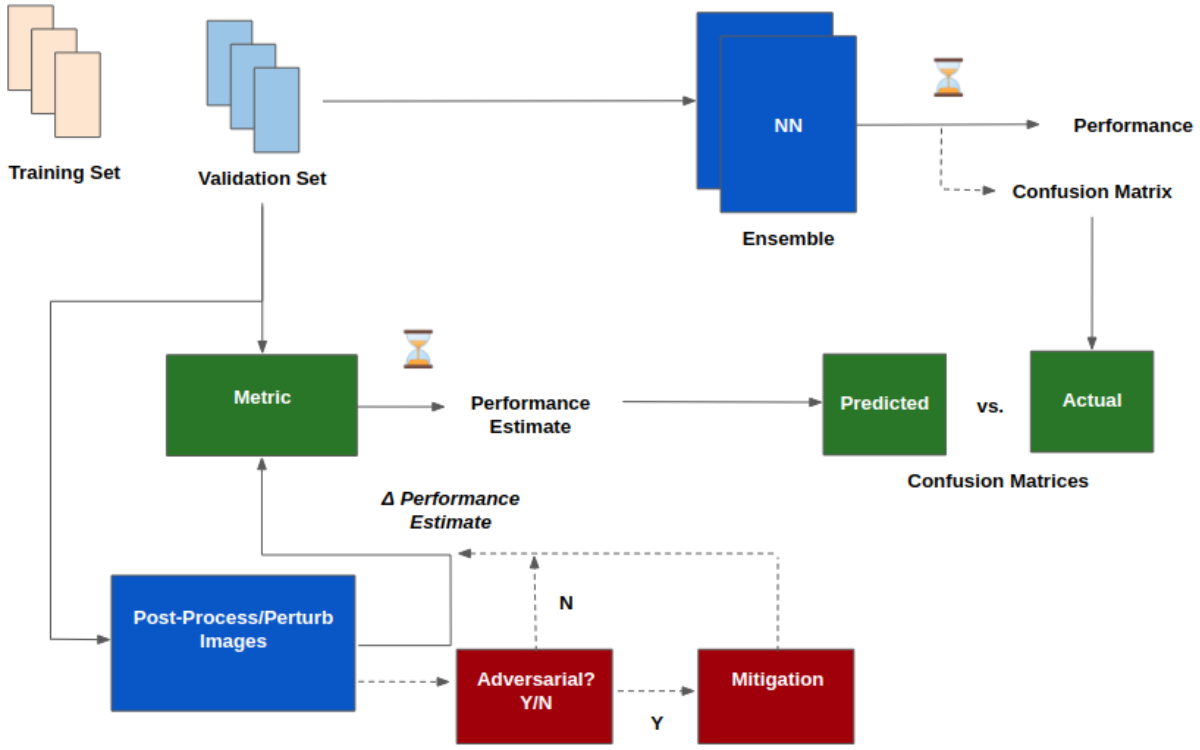

Figure 3.1: Diagram of the framework for image evaluation and inference developed in Chapter 3.

The performance estimate is obtained based on the distributions acquired from the validation dataset for a specific metric, a few of which are described in subsection 3.1.2. The performance estimate is then used to indicate whether or not an image of an individual is likely to be classified erroneously; additionally, inducing perturbations in images and establishing the difference in performance estimates with unperturbed images enables the framework to be augmented by implementing a step to determine whether or not an image is adversarial and if so, mitigating the attack. Adversarial image classification is considered in order to make the framework applicable to real-time, realistic environments in future developments. After ensuring an image is likely to be classified correctly, inference using a neural network model

is used to obtain the actual performance result. The actual and predicted performances can then be used to validate the efficacy of the system as a whole. The image dataset, performance estimate metric, and deep learning architecture are intended to be 'fill-in-the-blank' boxes so that the overall framework can be analyzed across variations of each.

3.1 Background

3.1.1 Datasets and Models

VGGNet [37] is comprised of a number of convolutional layers utilizing ReLU activations, max-pooling layers, and a softmax classification layer. Common configurations of VGGNet are VGG-16, which has 13 convolutional layers and 3 fully-connected layers, and VGG-19, which has 16 convolutional layers and 3 fully-connected layers.

Labeled Faces in the Wild (LFW) [72], described in subsection 2.3.1, is a face dataset containing over 13,000 labeled images; 1680 identities have at least two images each.

VGGFace [69] is a CNN similar to and based on the architecture of VGGNet [37]. VGGFace improves performance by incorporating triplet loss training in the last fully-connected layer in order to learn embedded feature descriptors, which use an input image with positive and negative image examples to learn the best face descriptors. For example, a VGGFace model with an architecture similar to VGG-16 and pretrained on a 2D-aligned LFW dataset [72] yielded a 99.13% Equal Error Rate (EER) with triplet-loss embeddings and a 97.27% EER without triplet-loss embeddings [69].

VGGFace2 [70] is a dataset composed of 3.31 million images across 9,131 total subjects, with on average over 300 images per subject and includes variations in illumination, pose, and occlusions. ResNet-50 and Squeeze-and-Excite (SE) ResNet-50 models pre-trained and

fine-tuned on this dataset are publicly available in Caffe, MatConvNet, and PyTorch. Due to this accessibility, VGGFace2 was utilized for testing the potential utility of the framework.

3.1.2 Metrics

The statistical properties of a given image containing a face may be informative when compared to the statistical distributions of faces from a given dataset and utilized as metric for assessing the image quality and soundness (and therefore likelihood of correct classification). Computations applied to an entire image or face necessitate less pre-processing time if they eliminate the need for face detection, alignment, and feature extraction; however, the described juxtaposition is anticipated to be more informative when applied to specific regions of a face or when incorporating texture features. For example, there are numerous well-known texture descriptors such Haralick features [92] that can be calculated; these properties and their statistical distributions across a test image could then be compared to corresponding distributions in the training dataset. One instance of a Haralick feature that has been applied to facial recognition is the gray-level co-occurrence matrix (GLCM), which can be analyzed for properties including but not limited to entropy, energy, and homogeneity [93], [94].

Because part of the utility of this framework is to make an efficient decision regarding the validity of an image, the time to generate a predicted performance estimate from the metric and the time taken by the neural network model to output the actual performance was to be measured. In order to maximize potential efficiency, this work focused on image-based metrics or metrics only necessitating low-level processing and feature extraction in order to decrease the computational time necessary to identify landmarks for a given face. However, this will obviously vary depending on the method used for feature extraction and alignment,

as well as the hardware utilized, and is a subject for further investigation. For example, DeepFace utilizes less than one-fifth of the total inference time for facial alignment and approximately half of the total inference time for feature extraction [66].

Several metrics for estimating the performance of the deep learning architecture were implemented and are described in subsection 3.2.2, incorporating the aforementioned concepts. The purpose of fault identification and metric analysis within this framework is to identify when and where a neural network model may fail to generalize on a certain input image, and quantify why it does so. Due to the lack of reliability of deep learning frameworks and the pertinent consequences of their failures when they are utilized for face recognition, there is an open need to define metrics to support the above notion. The performance estimates and their distributions across a dataset can be utilized to screen an input image using the metric calculation to sort the image into its best-fit distribution. Ideally, the distributions produced by the metric are clearly disparate from the distributions of the neural network, so that the disjoint subset may be used in comparison to an input image to be accepted or rejected.

In order to wholly analyze fault occurrence and identify the performance distributions of certain features and perturbations of an image, the framework may be tested on both facial identification (1:N matching) and facial verification (1:1 matching). These are similar applications, but may yield different insights regarding the impact that induced perturbations and/or variations in facial features may have. In the present work, 1:N matching was analyzed.

3.1.3 Adversarial Detection and Mitigation

Circumventing deep learning models for object recognition and classification has been a widely-approached topic that can range from simple and straightforward physical alterations like stickers and graffiti placed on road signs in a strategic manner [95] to ostentatious personal aesthetics [52] and relatively subtle decor, such as eyeglasses [96]. Another area of approach includes purposefully-induced flaws in a captured image such as universal adversarial perturbations, which are intended to generalize across broad varieties of neural network architectures and datasets [97], [98].

Other attack implementations focus on specifically thwarting facial recognition and verification models, such as attacks based on grid-based occlusion and MSB-based noise [99] or tactics based on morphing or replacing specific regions and attributes of a face whilst being recognizable by a human interpreter [100], [101]. By introducing such flaws in the input image, the impact of such perturbations can be quantified via the difference in performance estimates produced by a given metric. This result can be used to further analyze attacks that intend to confound deep learning models by modifying an image.

A simple metric for estimating how well a model will do based on input data is detecting whether or not the image is adversarial or not; if it is, it is more likely to be incorrectly classified. For example, [102] develops a trained binary classification detector that identifies real and adversarial data, to be used in conjunction with a network for classification. [103] uses a pixel-based flattening or PCA with a SVM classifier to determine whether or not an image is real or adversarial.

3.2 Experimental Design

3.2.1 Implementation Details

The framework in Figure 3.1 was built in Python using PyTorch [32], primarily using a computer equipped with 16GB RAM, a NVIDIA GeForce GTX 1050TI 4GB GPU, and an Intel Core i7-7700HQ 2.8GHz CPU. After MTCNN [60] was used to localize faces in VGGFace2 dataset, a ResNet-50 model pretrained on the MS-Celeb-1M dataset and fine-tuned on the VGGFace2 dataset [104] was used to evaluate images from VGGFace2.

The specifications of the computer used, as mentioned above, are typical of a mid-range consumer gaming laptop; the most significant hurdle incurred by using this hardware was the time consumed to calculate the metrics for the sheer amount of images contained in the evaluated dataset. Due to the large size of VGGFace2 and the limitations of the hardware used, in order to hasten development two sets of 10,000 images were randomly selected from the sets of images that were classified correctly or incorrectly by the model.

3.2.2 Selection of Metrics

The fundamental goal driving the selection of metrics used was efficiency, ideally in the form of a low-complexity, low-time consumption algorithm that can sufficiently be used prior to model inference in order to separate images as less likely or more likely to be classified correctly. The main limitation of this goal is whether or not the metric *works*, as there are a seemingly-infinite amount of computer vision approaches that fit the bill of 'relatively simple' and 'faster than a convolutional neural network'. A few of the tactics employed are detailed in this section.

Several of the metrics attempted include:

- Global entropy of grayscale images (Equation 3.2)
- Local entropy of grayscale images - 30 blocks, 25 pixels each (Equation 3.3)
- Global entropy of each color channel of RGB image (Equation 3.2)
- Local entropy of each color channel of RGB image - 30 blocks, 25 pixels each (Equation 3.3)
- Kurtosis of grayscale image histogram and of each RGB histogram
- Skewness of grayscale image histogram and of each RGB histogram
- Saliency
- Blur/sharpness via FFT
- Blur/sharpness via LoG

Other metrics that were considered include:

- GLCM entropy, contrast, contrast, correlation, energy
- All of the above applied to individual and/or combined features (eyes, mouth, etc.)

3.2.3 Entropy Measures

The Shannon entropy quantifies the amount of uncertainty of a random variable:

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \tag{3.1}$$

The local and global Shannon entropy levels were computed for grayscale images and individual color channels of color images via the methods given in [105]. For an image X of size

$T = M\ x\ N$ with n_l pixels of intensity level l (of L total intensity levels), the global Shannon entropy can be found via

$$H(X) = -\sum_{l=1}^{(L-1)} P_l \log_2 P_l = \sum_{l=0}^{(L-1)} \frac{n_l}{T} \log_2 \frac{T}{n_l} \tag{3.2}$$

where P_l is the probability of $X = l$ or $\frac{n_l}{T}$. The local Shannon entropy, taken over k image blocks, is a more accurate, consistent, and efficient measure of quantifying image randomness according to the authors, and is given by

$$\overline{H_{(K,T_b)}} = \sum_{i=1}^{k} \frac{H(S_i)}{k} \tag{3.3}$$

where $S_1 - S_k$ is a randomly-selected set of k image blocks with a specified number of T_B pixels.

3.2.4 BRISQUE Quality Assessment

BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) [106] is an algorithm used to assess the quality of an image based on its regular statistic properties. These properties are statistically regular amongst 'natural' images such as photos captured with a camera and are used in a process that results in a quality score ranging from 0-100, where smaller scores indicate higher image quality. No additional images to the image being assessed are required using BRISQUE; this type of image quality assessment (IQA) method is referred to as a 'no-reference' or 'referenceless' algorithm. Moreover, BRISQUE uses just a fraction of the time required by other referenceless IQA methods and a small number of extracted features to calculate the quality score of an image making it very computationally efficient, implying its potential to be an exceptionally good metric for this framework.

Nevertheless, the BRISQUE algorithm was not one of the metrics used in this chapter because it had not yet been discovered by the author. However, the general use of image quality as a metric for future work on this framework is explained in Section 3.3. Additionally, the BRISQUE method is revisited in Chapter 4, where the quantification of image quality is heavily relied upon to support the analysis of classification decisions.

Feature extraction is performed using the MSCN (Mean-Subtracted Contrast-Normalized) coefficient distributions and the pairwise products of the coefficients with neighboring pixels in four directions. For these inputs, the best-fit mean, shape, and variance parameters to an asymmetric or symmetric generalized Gaussian distribution are used in the feature vector, which is then mapped to the final quality score via regression. Furthermore, it is believed that BRISQUE is a reasonable method for real-time image quality assessment because it is highly computationally efficient.

3.3 Discussion

None of the metrics described in Section 3.2 resulted in a significant separation of distributions when tested. Table 3.1 contains the means and standard deviations of the global entropy and local entropy of grayscale and each color channel of randomly-selected images within correctly and incorrectly classified sets. By skimming the table, one can see the unfortunate similarity between the 'Correct' and 'Incorrect' columns.

The plots shown in Figure 3.2 of the distributions of global entropy across the color channels of correctly and incorrectly classified images are representative of the separation observed in plots obtained from other metrics tested (i.e., very little).

Table 3.1: Means and standard deviations of metrics from 10000 randomly-selected images that were classified correctly and incorrectly by the model. The local entropy measure utilized 30 blocks of 25 pixels each.

		Classification Result			
		Correct	Incorrect	Correct	Incorrect
Metric	Image Type	Mean		Std. Dev.	
Global Entropy	Gray	7.3805	7.3191	0.27950	0.33222
	R	7.50101	7.45237	0.27089	0.31499
	G	7.36662	7.31144	0.29616	0.35100
	B	7.26543	7.20221	0.35792	0.41686
Local Entropy	Gray	14.13428	13.88092	1.331409	1.45533
	R	14.37146	14.12354	1.22554	1.34868
	G	14.20646	13.94835	1.29030	1.42266
	B	14.26937	14.00644	1.26973	1.38784

3.3.1 Future Work

An investigation into how demographics such as race, gender, and other visual attributes may contribute to bias in image classification accuracy may offer different results than quantitative metrics used here. This topic is brushed upon in Chapter 4.

While searching for quantitative image quality assessment methods for the labelling of images in Chapter 4, an additional discovery was made: many of the publications found were highly relevant to the work of Chapter 3, as they included established methodologies to image analysis and details on the statistical distributions of images and scenes than those found whilst researching metrics for this chapter. Additionally discovered while developing the work of Chapter 4, the quality of images captured in realistic operating scenarios is substantially lower than that encountered in most published works. FaceQnet [107] is a model that uses image quality to estimate the expected performance of an image prior to face recognition.

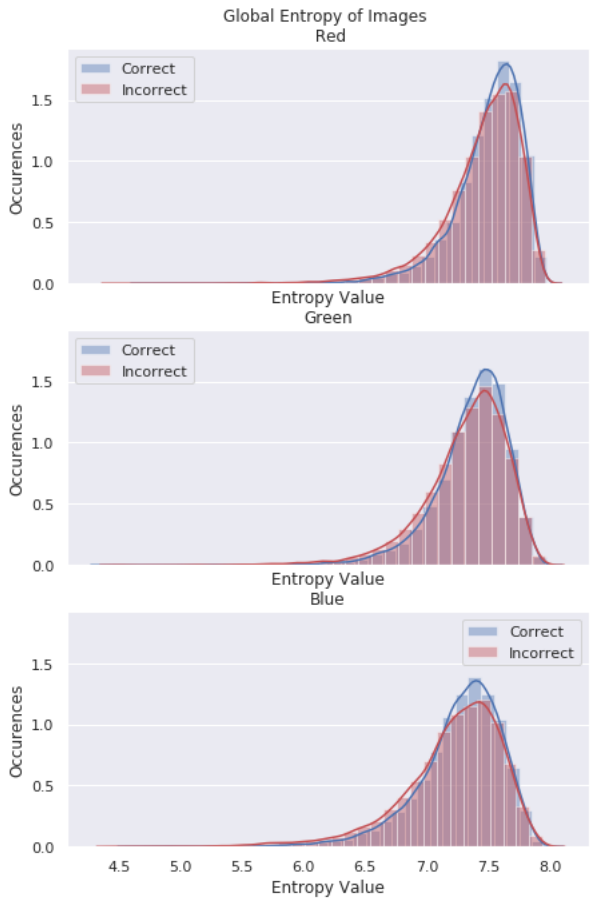

Figure 3.2: Plot of global entropy distributions across the red, blue, and green color channels for input images that were correctly and incorrectly classified.

Although it differs in that it does not aim to use algorithms with low computational complexity to compute the metric used for performance estimation (FaceQnet is a CNN), it is a highly analogous framework to the one presented in this chapter. Furthermore, FaceQnet successfully establishes a correlation between the performance estimate produced by a metric and the accuracy of a face recognition system. The inability to find a metric producing this correlation was a significant obstacle to the development of this chapter; therefore, it is obvious that a clear path for any future work on the framework in Figure 3.1 should start with the investigation of image quality as a metric, and refer to influential papers in image quality assessment such as [108] and [109] for other promising metrics.

The concept of *image quality* at first appears to be simple: does the image look good or bad? However, as learned throughout the research and development process of the projects in this thesis, image quality assessment (IQA) is a well-established field of research with many methods, such as the deep learning-based approach used in FaceQnet [107], and BRISQUE [106], the algorithm used in the succeeding chapter.

Chapter 4

Real-Time Mask Recognition

While a variety of image processing studies have been performed to quantify the potential performance of neural network-based models using high-quality still images, relatively few studies seek to apply those models to a real-time operational context. This chapter seeks to extend prior work in neural-network based mask detection algorithms to a real-time, low-power deployable context that is conducive to immediate installation and use. Particularly relevant in the COVID-19 era with varying rules on mask mandates, this work applies two neural network models to inference of mask detection in both live (mobile) and recorded scenarios. Furthermore, an experimental dataset was collected where individuals were encouraged to use presentation attacks against the algorithm to quantify how perturbations negatively impact model performance. The results from evaluation on the experimental dataset are further investigated to identify the degradation caused by poor lighting and image quality, as well as to test for biases within certain demographics such as gender and ethnicity. In aggregate, this work validates the immediate feasibility of a low-power and low-cost real-time mask recognition system.

4.1 Introduction

The coronavirus (COVID-19) pandemic continues to have a global impact on our society since its initial spread began in late 2019 [7]. One of the most visually perceptible ways

the coronavirus has affected society's ritual errands and routines is through the widespread use, and in many cases government or commercial mandates, of facial masks intended to slow the spread of the virus by impeding transfer of respiratory droplets via the nose and mouth [10]. Several US states now legally mandate that people wear masks before entering a public space where it is not feasibly practical to socially distance, such as a grocery store. As of early 2021, additional federal mandates [110], [111] have been put into place, further extending those requirements.

Independent of the questions associated with mask efficacy or policy, this chapter seeks to answer how effectively compliance with masking guidelines can be gauged, as it is uncertain as to how many people actually follow these mandates. Further, the chapter seeks to validate the technical capability and implementation feasibility of a sub-$100 Raspberry Pi-scale solution that is suitable for installation at the front door of a store or other congested passageway.

4.1.1 Motivation

A widespread problem since the inception of the pandemic has been not just the virus itself, but its airborne spread, which is believed to be directly correlated with mask usage [112]. One particular challenge in the present environment is that of mask adherence, which has largely been delegated to 'gatekeeper' store employees that can themselves get sick and/or lead to altercations when prompting incoming patrons to don their masks [12], [13], [14]. In lieu of these challenges, we want to test the feasibility of a more automated machine-based solution that is capable of giving a binary decision as to whether individuals are wearing masks, which may be converted to a visual green/red light that indicates permission to enter; a notional example of this process is depicted in Figure 4.1.

Lab-based mask recognition performance on pristine images routinely achieves over 99%

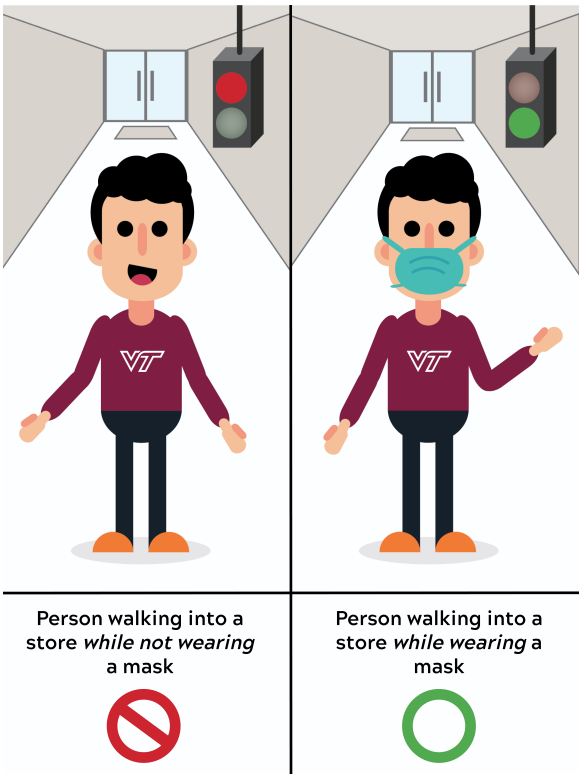

Figure 4.1: Conceptual depiction of how automated mask recognition might be used at a storefront, when a camera is placed opposite the hypothetical individual. A visual indicator turns red when an individual is not wearing a mask and green when an individual is wearing a mask, giving immediate feedback about permission to enter.

accuracy using these machine learning techniques [113]. However, in order to be acceptable in a practical context, that performance level must be retained when implemented in a real-world context, using non-pristine images, and in a cost / form factor that is realistic for a store to deploy. Further, knowing that many machine-learning solutions are susceptible to degradation resulting from training dataset mismatch with respect to ethnicity [114], gender [115], image quality [116], lighting conditions [117], and combinations of these parameters with other characteristics [118] [119] [120], it was chosen to intentionally bombard the neural net model with different presentation attacks to quantify how quickly performance degrades.

4.1.2 Research Questions & Goals

A few of the questions that the experimental design and analysis presented in this chapter seek to answer include:

- What are the primary features that influence the decision(s) a binary mask recognition classifier will make?
- How many images are necessary for the model to ultimately identify whether an individual is wearing a mask (i.e., select the correct label)?
- How and why do 'real-world' scenarios affect performance, versus tests in controlled environments?
- Can a sufficient machine-based performance be achieved and scaled to that of a low-cost real-time platform?

4.1.3 Prior Work

As mentioned in Section 2.3, occlusions present a challenge for face detection and recognition algorithms. In [121], an adversarial face detector is trained to detect and segment occluded faces and areas on the face. Three varieties of facial datasets, including masks, were assembled in [122], including images scraped from the internet and images that were photo-edited to show faces with masks on. A method for facial recognition of masked faces is also introduced, which improves accuracy over traditional architectures for facial recognition by training a multi-granularity model on the aforementioned datasets in combination with existing face datasets. In [123], a masked face dataset is created and locally linear embedding (LLE) CNNs are introduced for detection of masked faces.

Despite these works, a large masked face dataset with sufficient image feature diversity to be similar to real-world images was difficult to find. Dataset augmentation can be used to supplement the amount of feature variation seen by a model during training as well as the overall size of a dataset.

In [124], the quantity and quality of simulated, captured (collected), and augmented training data is analyzed for their impacts on a radio frequency classification task. Simulated datasets are created via random generation based on set characteristic distributions. Dataset augmentation is carried out in two ways: by sampling from existing collected capture data and modifying it to obtain uniform signal characteristic distributions in the resulting dataset, and by random selection from a Gaussian joint kernel density estimate (KDE) on captured data. One key result shows that augmentation using the KDE of the captured data increases test performance compared to data augmentation without taking into account the distributions of known data parameters, but that in both cases augmentation may increase generalization and improve performance, compared to training with solely the original dataset. One caveat is that as the dataset quality decreases and difficulty increases to a certain degree (the exact threshold was not computed by the authors), and the sample quantity remains fixed, model generalization and performance tend to decrease especially when using augmented data that was collected without consideration of the existing dataset distribution.

Additionally, the quantity of data samples required for 100% and 95% accuracy based on linear trend and logistic fit data is found to be lower for both of the datasets acquired via the previously mentioned augmentation techniques only with less difficult datasets. For example, the KDE-based augmented dataset necessitated approximately 58% and the non-KDE augmented dataset required 167% of the number of samples per class than the original dataset required to obtain 95% accuracy on the least difficult data analyzed in the chapter; on the most difficult data, the KDE-based augmented dataset required 180% and the non-

KDE augmented dataset required 330% of the number of samples per class than the original dataset required to obtain 95% accuracy. Although more data is needed for high performance, collecting data for the context of the referenced paper is a time-conserving procedure - generation of augmented data significantly decreases the time required to obtain sufficient samples for 95% performance on the order of years.

Essentially, the results relevant to the work presented in the following sections of this paper are that (1) utilizing the statistical properties of the original dataset when assembling an augmented dataset is crucial to ensuring model generalization is improved and (2) the augmented dataset will have an overall lower quality of data than the original dataset and should include a greater number of samples from each class than the original dataset to ensure high performance on more difficult test data.

Independent measures of quality and other properties may cross-check and/or improve the confidence of a neural network's classification decision. These measures also may improve performance as human-assigned categorical labels are prone to human error and consequently mislabelled data, which has a negative impact on classification performance [125]. BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) [106] is a spatial image quality assessment (IQA) method that does not use a reference image. The statistical regularity of natural images (i.e., images captured by a camera and not images created on a computer) are altered when distortions are present. Based on this, BRISQUE extracts features from an image based on its MSCN (Mean-Subtracted Contrast-Normalized) coefficient distributions and the pairwise products of the coefficients with neighboring pixels in four directions. For these inputs, the best-fit mean, shape, and variance parameters to an asymmetric or symmetric generalized Gaussian distribution are used in the feature vector, which is then mapped to the final quality score via regression. Furthermore, it is believed that BRISQUE is a reasonable method for real-time image quality assessment because it is highly compu-

tationally efficient. In the paper, using a 1.8Ghz computer with 2GB of RAM, a 512x768 image took only 1 second to process. This is much faster than two other IQA algorithms, DIIVINE [126] and BLIINDS-II [127], which took 149 and 70 seconds, respectively.

4.1.4 Chapter Outline

The organization of this chapter is as follows: Section 4.2 begins with an overview of the experimental framework for testing the feasibility of a real-time mask recognition system. The selection and modification of the dataset used is described in subsection 4.2.1. In subsection 4.2.2, scenarios for characterizing imperfect image capture conditions are described, including how performance is affected by head pose variations and a variety of presentation attacks. Section 4.3 outlines an experiment where video-recorded participants were encouraged to attempt presentation attacks; although relatively small in dataset size, these results offer insight into how quickly categorical data labelling and unexpected/untrained stimuli degrade overall system performance. Section 4.4 describes the process of capturing and analyzing real-world video recordings of pedestrians using a laptop and a webcam; both results of these real-time evaluations and extrapolations to expected number and frame rate of images required for a correct inference are presented. Finally, Section 4.5 offers overall conclusions from our experiment and suggestions supporting near-term implementation of a real-time low-cost platform capable of performing mask recognition on a commercial scale.

4.2 Experimental Design & Methodology

Implementation of real-time inference for binary classification (i.e., whether or not a person is wearing a mask) on a webcam-equipped PC or a low-cost device could help bolster local

health administration statistics and evaluation of the efficacy of current policies. A graphical summary for our experimental model is presented in Figure 4.2.

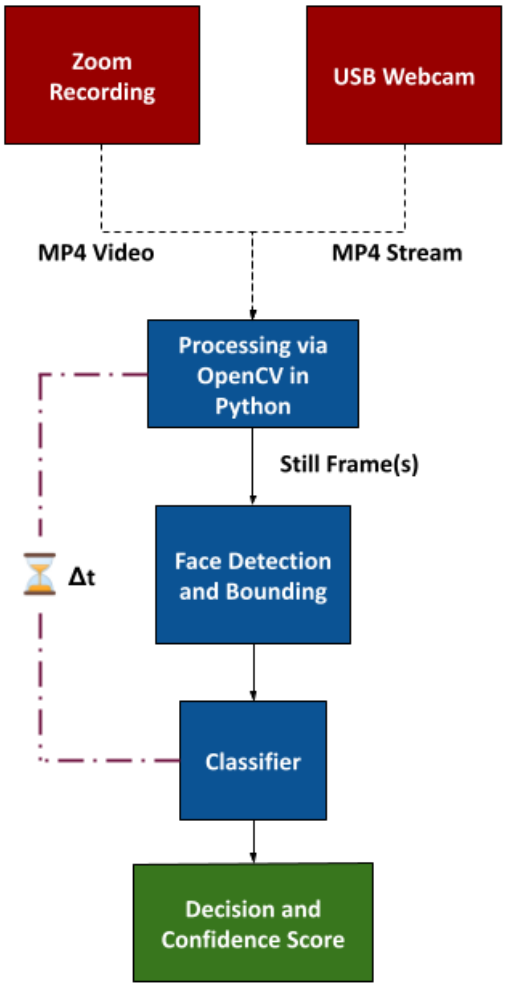

Figure 4.2: Preliminary architecture of mask recognition system.

To examine this setup further, two neural network models [41] [38] were given a binary classification task with experimental images collected from a recorded video experiment in which participating individuals were encouraged to alter their appearance in ways that may be somewhat unexpected, but not entirely uncommon, in real-life implementations. A few examples of edge cases that are shown to impact classification performance include people wearing masks with animal faces (such as a dog nose), human faces, or other distracting

patterns printed on them, and people with beards or mustaches that extend past the mask. The architecture was implemented in Python using PyTorch [32], an open source machine learning software framework, and OpenCV [128], a computer vision library written in C++ with bindings for Python. Prior to the execution of any experiments involving human subjects, the Virginia Tech IRB (Institutional Review Board) reviewed the protocol used for this research (IRB 20-736) on September 24th, 2020. The protocol was approved and was determined to meet the exemption criteria under 45 CFR 46.104(d) category(ies) 2(ii) [129].

4.2.1 Dataset Selection & Augmentation

One of the largest published datasets of masked faces is the Real-World Masked Face Recognition Dataset (RMFRD) [122], which contains over 2,000 photos of individuals wearing masks and 90,000 photos of individuals without masks, across a total of 525 identities. [1] RMFRD, the Simulated Masked Face Recognition Dataset(SMFRD) (comprised of images of masks digitally imposed onto faces), and the Masked Face Detection Dataset (MFDD) (similar to RMFRD) are three separate datasets assembled in the same paper under the parenting Real-World Masked Face Dataset (RMFD).

Based on the distribution discrepancy across the 'mask' and 'no mask' classes, the dataset is highly biased towards the 'no mask' label. However, this does not seem to have significantly negatively affected model performance towards 'mask'-labelled images (at least in controlled environments), as shown in Section 4.3. In this chapter (in contrast to the process described in subsection 4.1.3 from [124]), *augmentation* refers to supplementing the original RMFRD dataset with additional images of people wearing and not wearing a mask. These images were captured from the internet, specifically Instagram (it is worth noting that the set of

[1] Although RMFRD is reported to have 5,000 images of individuals wearing a mask, less than half of such images could be successfully obtained from the download links provided in the publication.

Table 4.1: Dataset image quantities for classification into 'mask' or 'no mask' labeled images.

Dataset	Label	Training	Validation	Test
Original	*'mask'*	1573	236	309
	'no mask'	67287	9602	13579
Augmented	*'mask'*	2108	307	416
	'no mask'	67499	9630	13621

images from Instagram may be biased towards demographics that tend to use Instagram more frequently than other demographics)

RMFRD consists of typically average-quality images (as described further in this section) with closely cropped faces and homogeneous demographic diversity. To mitigate this, which may cause poor performance in real-world implementations, several hundred photos were scraped from Instagram using the hashtags #maskon and #maskup. Approximately 2,800 images were gathered. MTCNN [60] was used to quickly determine which images contained faces, resulting in 993 images. These images were then manually labeled into two categories, based upon whether or not an individual was correctly wearing a mask. In total, 711 images of individuals with masks and 282 images of individuals without masks were gathered. The resulting split used for training is shown in Table 4.1. Table 4.2 displays the accuracy for training, validation, and test phases of two CNN architectures on the original RMFRD dataset and on the augmented RMFRD dataset.

Two models, MobileNetv2 [41] and ResNet18 [38], were evaluated on the RMFRD training set. Both models were pre-trained on ImageNet. Transfer learning was then utilized to retrain the last layer for classification on masked and unmasked faces. The performance is shown in Table 4.2. The performance of the original MobileNetv2 model and the augmented MobileNetv2 model on solely the Instagram test set was only 83.28% and 83.59%, respectively. This concept is also displayed on a minute level in Table 4.2, where the augmented

Table 4.2: Performance results of MobileNetv2 and ResNet18 on the RMFRD dataset and on the augmented RMFRD dataset.

		Accuracy		
Dataset	**Model**	**Training**	**Validation**	**Test**
Original	MobileNetv2	0.9888	0.9933	0.9971
	ResNet18	0.9902	0.9946	0.9974
Augmented	MobileNetv2	0.9926	0.9833	0.9933
	ResNet18	0.9876	0.9884	0.9934

models have a slightly diminished performance compared to the original models. Similarly to the age-old *'jack-of-all-trades, master of none'* adage, generalization of training data makes network models more broadly applicable to new data, but reduces overall performance due to broadened learned behavior(s). This degradation in observed performance also suggests that there is room from improvement in baseline performance simply by improving the robustness of the dataset(s) during training. As described in subsection 4.1.3 and shown in [124], despite the reduced overall performance, this generalization may also result in better performance on previously unseen data, so no attempts were made to further augment the dataset to prevent over-training.

4.2.2 Realistic Image Capture

The variety of ways in which everyday individuals walk, dress themselves, and interact with each other necessitates characterizing a system meant for real-world implementation in ways that help encompass the environmental factors a neural network might encounter when tasked to make decisions as well as the hardware on which it is deployed. In order to determine the realistic performance of the mask recognition system during real-time operations, algorithm performance was tested via still images derived from video recordings at different

frames-per-second (FPS) rates. Such comparisons help scope the cost and quality of the deployment system. This is described more in-depth in subsection 4.4.2

Angle Variations & Performance

To estimate the performance of the CNNs in situations where faces appeared off axis, a controlled experiment was devised to evaluate images of an individual (a) wearing a mask and (b) not wearing a mask while moving their head pose across different azimuth and elevation angles. The *img2pose* PyTorch package for facial detection and alignment was used to obtain the angles [130]. Figure 4.3 and Figure 4.4 display the side-to-side and up-down angles of the head pose, plotted by performance from the frame of reference of the camera. A coordinate of 0 azimuth and 0 elevation corresponds to an individual facing the camera along its boresight without moving their head to the left, right, up, or down. A coordinate of negative elevation means the individual was looking upwards. A coordinate of negative azimuth means the individual was looking to their left. Each point in Figure 4.3 and Figure 4.4 represents a recorded image in terms of head pose orientation and the output probability of the correct label ('mask' or 'no mask'). The performance of the augmented MobileNetv2 model was 78.57% and 84.50% for the 'no mask' and 'mask' image sets consisting of 42 and 71 images, respectively.

Similarly to the results in Section 4.3, the 'no mask' performance is worse than that of the 'mask' performance. However, due to the wider variation of angles in the 'no mask' test, as well as the smaller number of images, it is inconclusive to say that 'no mask' performance is always worse than 'mask' performance. The results of both trials show that, in general, classification performance decreases at greater tilts of the individual's head, yet the performance is reasonably stable over the ranges that a person walking towards the entrance of a building would normally exhibit. It was inferred, based on Figure 4.3 and Figure 4.4 that for

an individual wearing a mask, that the model is particularly sensitive to changes in azimuth (side-to-side) head pose variations, yet reasonably robust to elevation.

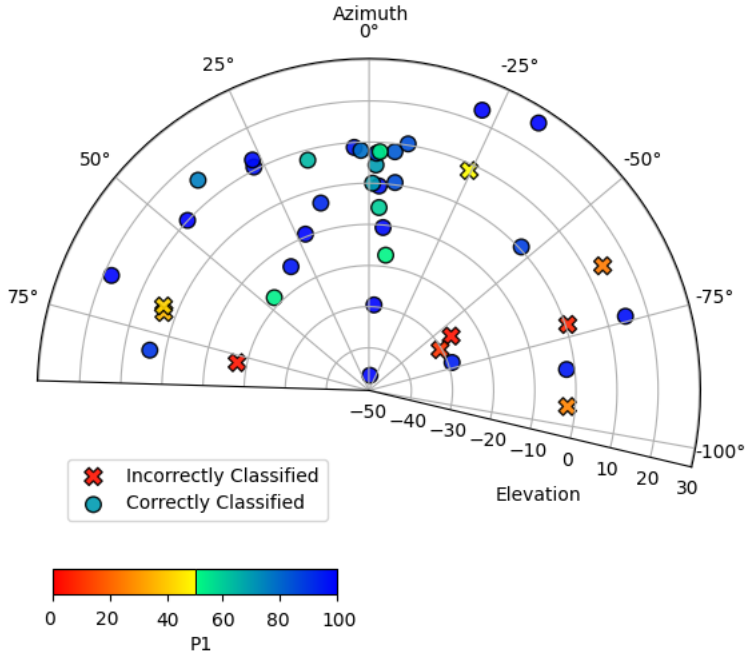

Figure 4.3: Scatter plot of azimuth and elevation head angle and performance for an individual not wearing a mask.

Presentation Attacks

A major consideration in the design of the experiment covered in Section 4.3 is the inclusion of *presentation attacks* [131] [132] that would intentionally try to degrade the performance of the neural networks used. A few examples of these attacks include:

- Masks of different shapes, sizes, colors, or textures between recordings (e.g. neck scarves, respirators)
- Facial or hair styles that can be easily put on or taken off, such as eye makeup, glasses, ponytails, etc.

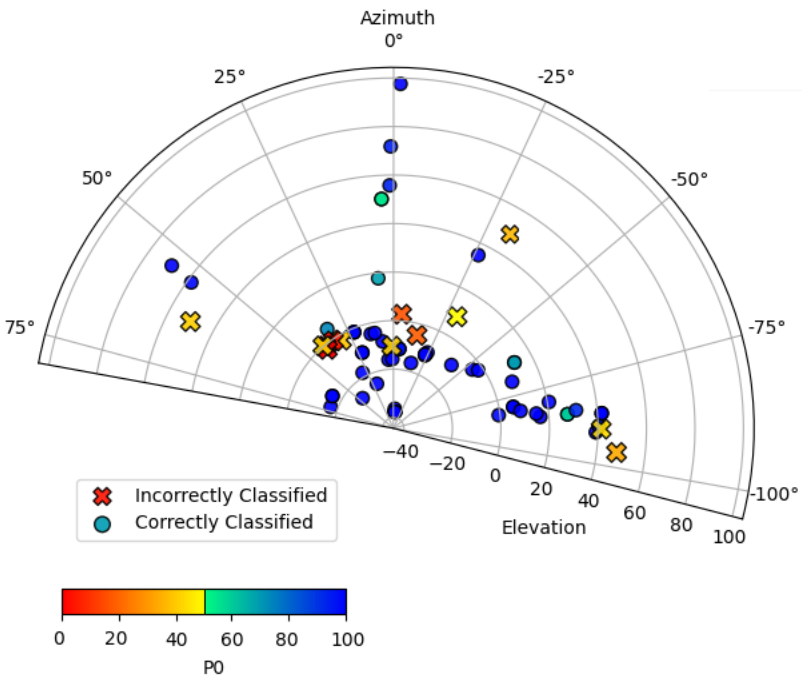

Figure 4.4: Scatter plot of azimuth and elevation head angle and performance for an individual wearing a mask.

- Hats, scarves, or other partial head coverings worn at different angles and/or worn to cover the face in different ways (such as occluding eyebrows)
- Wearing Halloween masks, wigs, and/or costumes
- Tilting of webcam off-axis
- Using facial augmentation filters such as those from Facebook Messenger, Tik Tok, Snapchat, or other apps
- Having pictures of other people in the foreground or background, such as with posters
- Placing pets wearing masks or variations (e.g., dog wearing hat) in front of webcam
- Holding a dog or cat while you are in front of the webcam
- Using a variety of Zoom backgrounds

- Positioning the webcam so that the lighting varies, such as facing the sun or in a dark room
- Changing facial expressions in odd ways (e.g., intentionally closing or crossing eyes)

An important note of these presentation attack scenarios is that participants maintained their chosen state of *'mask'* or *'no mask'* during that single instance of time on camera, while changing variables between times on camera; as such, the mask recognition algorithm was not presented with dynamic scenarios of an individual donning or doffing their mask.

4.3 Recorded Video

Two experimental sessions were recorded over Zoom, a software application for remote video conferencing [133]. This enabled data collection from participants without needing to interact with them at a close distance. Furthermore, this remote setup allowed for recordings that spanned more environmental surroundings, behaviors, facial expressions, hair styles, mask designs, etc. than could be collected via a public data collection system (such as on a street), since each participant had complete access to their personal wardrobe and effects.

Following confirmation of Institutional Review Board (IRB) disclaimers and that all participants were in a location where they may have safely chosen not to wear a mask, individuals were prompted to randomly appear on screen via an audio cue so that the Zoom client recorded the individual. Individuals were instructed to correctly or incorrectly wear masks, to not wear masks, or to follow some randomly chosen variation of mask-wearing listed in Section Section 4.2.2. The variations performed by the individual were recorded for approximately 5-15 seconds each.

Following the sessions, the recorded MP4 files were imported and processed using OpenCV

Table 4.3: Results of recorded Zoom sessions across 8 categories for describing attributes of participating individuals' appearances.

Category	**Category Counts**		
Mask	**Mask**	**No Mask**	
	2143	3025	
Ethnicity	**Caucasian**	**Black**	**Asian**
	3528	389	1251
Gender	**Female**	**Male**	**Other**
	1234	3896	38
Image Quality	**Poor**	**Average**	**Good**
	1087	3114	967
Light	**Dark**	**Neutral**	**Bright**
	82	4232	854
Glasses	**No**	**Normal**	**Sunglasses**
	3798	1044	326
Zoom Background	**No**	**Yes**	
	4246	922	
Other	**No**	**Yes**	
	3518	1650	

[128] to convert the videos into still images at approximately 1FPS. The first session was recorded for 45 minutes and 50 seconds, resulting in a total of 2,749 frames. The second session was recorded for 40 minutes, 20 seconds, for a total of 2,419 frames. In total, 5,168 still frames from 86 minutes, 10 seconds of recorded video were obtained from the experimental sessions, which involved 13 individuals. The authors were present in both sessions and are therefore a larger percentage of the final dataset.

The still frames were manually labeled for the presence of a mask, no mask, or an incorrectly worn mask. This process was aided by the development of a Jupyter Notebook [134] that allowed a human to manually label the frames and check for the presence of certain attributes, such as glasses. An example of this labelling acceleration tool is displayed in Figure 4.5. One suggestion for future attempts in labelling a large number of images in a limited time frame is to utilize methods for ensuring image labels are reliable and accurate, such as in [135].

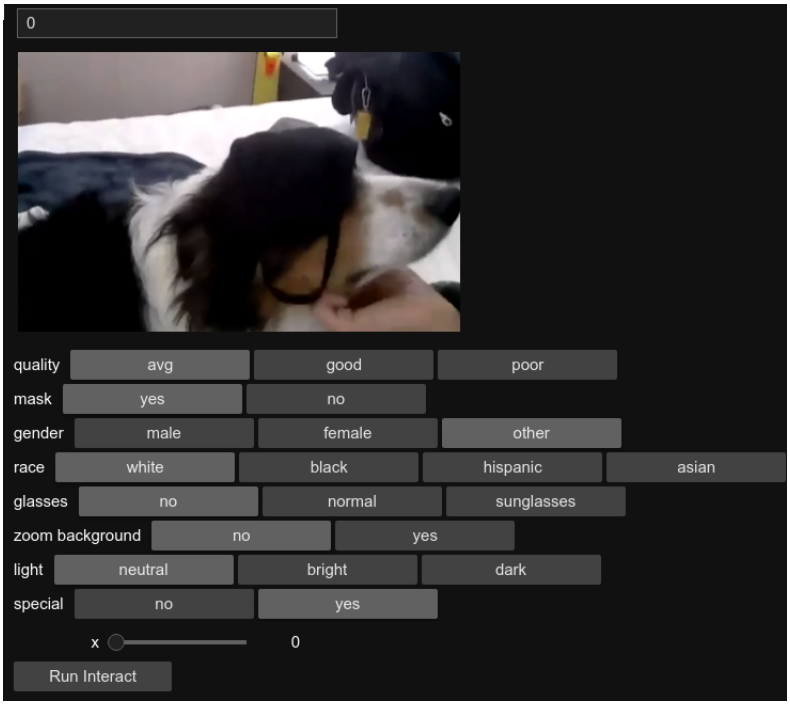

Figure 4.5: Example of custom Jupyter Notebook labelling process.

4.3.1 Results of Recording Analysis

Due to the nature of the training dataset images, the performance of both the MobileNetv2 and ResNet18 model was negatively impacted by the initial images saved from the recordings, which were zoomed out from a face and included the background details, such as a desk or house plant. To mitigate this, MTCNN [60] was used to first identify faces in the images with a 10 pixel bounding box margin. The images were then cropped to only include the face and resized to 240x240. One significant problem presented by this method was that in the case of images containing facial occlusions, such as a mask or pet cat, MTCNN did not bound a face; this issue occurred in approximately 1000 images. These images were separated from the correctly bound faces and manually cropped. A total of 162 images that did not contain faces were removed from the original dataset, resulting in a total of 5007 images.

The resulting performance of the original models and of the augmented models on the edited dataset is given in Table 4.4. The augmented MobileNetv2 model achieved the highest accuracy at 72.91%, closely followed by the original MobileNetv2 model with an accuracy of 72.59% and the augmented ResNet18 model with an accuracy of 71.73%. The original ResNet18 model had the poorest performance at 69.1%. While these figures are substantially lower than that of the >99% accuracies obtained with pristine images, they are representative of (1) the incorporation of real-world time-varying environments around the faces of interest, (2) the use of partially tuned open-source image pre-processing tools to convert that real-world scenario into still images sized for the neural net, and (3) intentional manipulations of the environments and/or anticipated characteristics affecting learned behaviors of the neural net algorithm in an attempt to fool classification. Further, these accuracies are representative of individual images taken at 1FPS, leaving the potential for further decision accuracy improvement by combining results of multiple images at a higher frame rate as

Table 4.4: Performance results of original and augmented MobileNetv2 and ResNet18 models on total dataset from recorded video.

Description	Model	Accuracy
Trained on Augmented Dataset	MobileNetv2	0.7291
	ResNet18	0.7173
Original Model	MobileNetv2	0.7259
	ResNet18	0.6910

discussed in subsection 4.4.2.

The relatively poor results in Table 4.4, Table 4.5, and Table 4.6 were anticipated due to the odd nature of the recorded video sessions. Relatively few 'normal' images of individuals wearing or not wearing masks were taken, as compared to a typical setting encountered at an office or supermarket. To further refine the above analytical results, the images were thinned to exclude irrelevant images and images of extremely poor quality, as described in the following section.

4.3.2 Notes on Qualitative Assessment

An 'other' label category was included in the manual labelling of the recorded video frames due to the number of oddities making certain images unable to be classified. This definition was a qualitative assessment of whether the image represented a scenario where a human user would make a correct decision and/or represented too great of a diversion from a normal face (and therefore an aberration in behavior that itself would stick out) to expect the algorithm to make a correct decision. An example of such an 'other' image is shown in Figure 4.6. The 1100 'other' images were set aside for portions of the performance analysis. For the augmented MobileNetv2 model, removing 'other' images increased mask classification performance from 63.72% to 77.21%, as shown in Table 4.5, confirming the expectation the

Table 4.5: Performance results of MobileNetv2 model on Zoom dataset.

		Classification Result		
Category	**Description**	**Y (#)**	**N (#)**	**Y (%)**
Mask	**No**	2140	787	*73.11*
	Yes	1511	569	*72.64*
Ethnicity	**White**	2602	823	*75.97*
	Asian	772	428	*64.33*
	Black	277	105	*72.51*
Gender	**Male**	2785	1005	*73.48*
	Female	855	330	*72.15*
	Other*	11	21	*34.38*
Quality	**Average**	2246	798	*73.78*
	Poor	704	299	*70.19*
	Good	701	259	*73.02*
Light	**Neutral**	3018	1102	*73.25*
	Bright	568	237	*70.56*
	Dark	65	17	*79.27*
Glasses	**No**	2704	956	*73.88*
	Normal	719	309	*69.94*
	Sunglasses	228	91	*71.47*
Zoom Background	**No**	3041	1057	*74.21*
	Yes	610	299	*67.11*
Special	**No**	2636	778	*77.21*
	Yes	1015	578	*63.72*

Table 4.6: Performance results of ResNet18 model on Zoom dataset.

		Classification Result		
Category	**Description**	**Y (#)**	**N (#)**	**Y (%)**
Mask	**No**	2357	845	*73.61*
	Yes	1235	570	*68.42*
Ethnicity	**White**	2442	983	*71.30*
	Asian	849	351	*70.75*
	Black	301	81	*78.80*
Gender	**Male**	2625	1165	*69.26*
	Female	953	232	*80.42*
	Other*	14	18	*43.75*
Quality	**Average**	2202	842	*72.34*
	Poor	699	312	*69.14*
	Good	691	261	*72.58*
Light	**Neutral**	2967	1153	*72.01*
	Bright	569	236	*70.68*
	Dark	56	26	*68.29*
Glasses	**No**	2715	945	*74.18*
	Normal	681	347	*66.25*
	Sunglasses	196	123	*61.44*
Zoom Background	**No**	2920	1178	*71.25*
	Yes	672	237	*73.93*
Special	**No**	2492	922	*72.99*
	Yes	1100	493	*69.05*

performance degrades when presented with untrained stimuli. An important note, however, is that the failures appeared to be more random than systematic, suggesting the opportunity for a higher level decision agent to arbitrate streams of decisions.

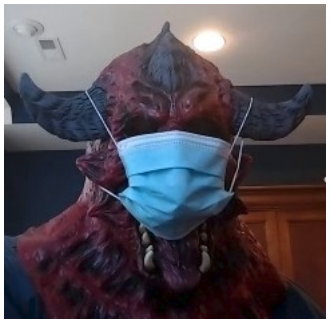

Figure 4.6: Example of 'other'-category image. The individual is definitely wearing a mask, but are they wearing it properly, and what would their gender and race be labeled as?

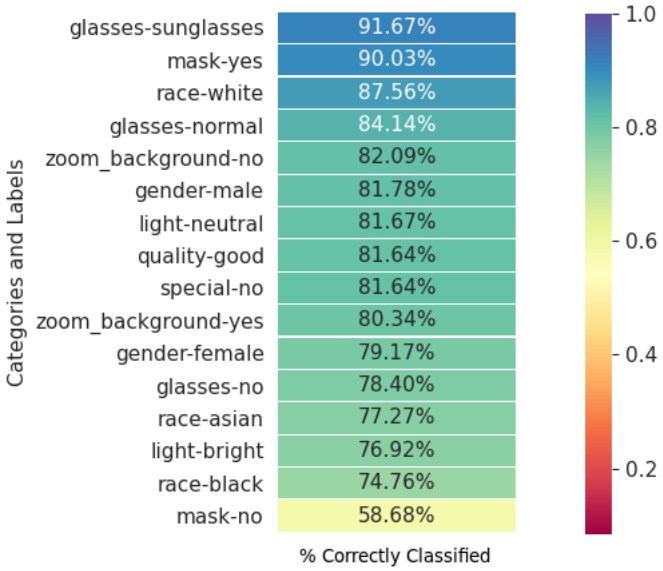

Figure 4.7: Ranked order map of performance results on data filtered to remove 'other' category and 'poor' or 'average' quality images.

Furthermore, image quality was assessed qualitatively during the manual labelling process. The 'quality' parameter proved to be of significant importance when evaluating the performance results of the experiments in this chapter - as shown in Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13, excluding images marked as 'poor' and 'average' quality in-

creases overall performance and will assist with requirements definition of the camera and/or image capture setup in a deployed solution.

Quantitatively describing the quality of an image is a more complex task than it initially seemed; in fact, the overall existence and nature of a 'quality measure' has come into question. These questions, partially asked and answered in [136], have straightforward solutions; a good quality metric for facial images separates images that improve recognition performance from those that impair recognition performance. The 'meaning of existence of the quality metric' remains vague in the scope of this chapter, given that mask presence is being evaluated rather than generic facial characteristics, but is a subject of investigation for future work. A rank-order comparison of the underlying parameter labels, and their corresponding differences when only qualitatively-assessed 'high'-quality images are retained, are shown in Figure 4.8 and Figure 4.9.

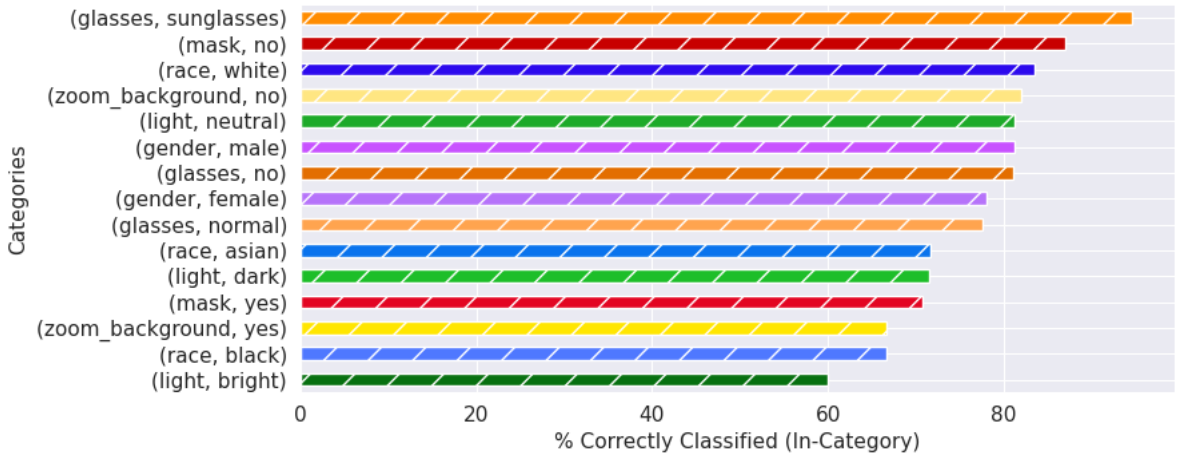

Figure 4.8: 'Good' (qualitative) quality image test performance with augmented MobileNetv2 model. The plot shows the categorical labels and classification performance.

The BRISQUE method introduced in [106] was selected to quantitatively evaluate the Zoom dataset, because it is a 'no-reference' IQA algorithm (i.e., it does not require an image of good quality to assess a new image) and because of its speed compared to other no-reference IQA algorithms, thus making it feasible for real-time implementation of the mask recognition

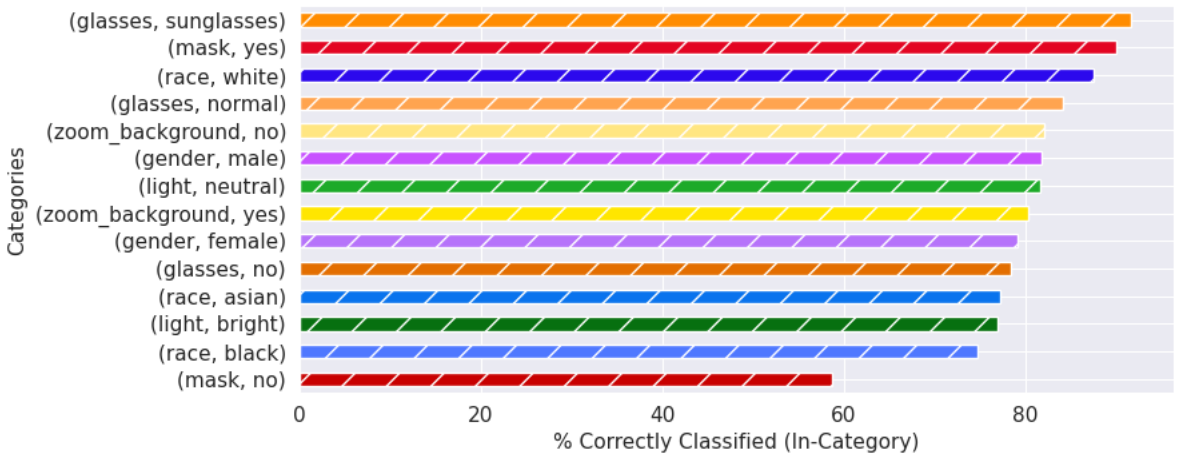

Figure 4.9: 'Good' (BRISQUE) quality image test performance with augmented MobileNetv2 model. The plot shows the categorical labels and classification performance.

system. The BRISQUE quality measure was separated into three categories, similar to the original subjective quality measure, based on thresholds one standard deviation below and above the mean score for the training dataset. Using a quantitative measure both decreased the chance of a mislabelled image being categorized incorrectly and decreased any human bias present in the quality category. On both the original RMFRD and RMFRD-Instagram augmented test sets, a roughly 5:90:5 {'poor':'average':'good'} breakdown was observed in terms of BRISQUE quality, using the same threshold levels to assign the categorical labels as was used on the Zoom dataset. The Zoom dataset had a 15.3% 'poor', 71.4% 'average', and 13.2% 'good' label split.

Figure 4.8 and Figure 4.9 display the classification performance across each category (i.e., about 80% of all of the images with individuals labeled as 'male' were classified correctly). The primary difference (that will be explained further in the chapter) to note between these two figures is within the 'mask' and 'no mask' labels, which respectively swap places as the second-highest performing category between Figure 4.8 and Figure 4.9. The metadata labels were chosen to include standard demographic data such as gender and race, as well as common variations observed in individual appearances during the experimental record-

ing sessions. The poor performance within the 'race' category for individuals categorized as 'asian' or 'black' can be attributed to the lack of data across multiple participants for those categories; the majority of participating individuals were categorized as 'white'. The small sample size also contributes to extreme performance for female participants, individuals wearing sunglasses, and for dark illumination conditions. This is further visualized in Figure 4.13.

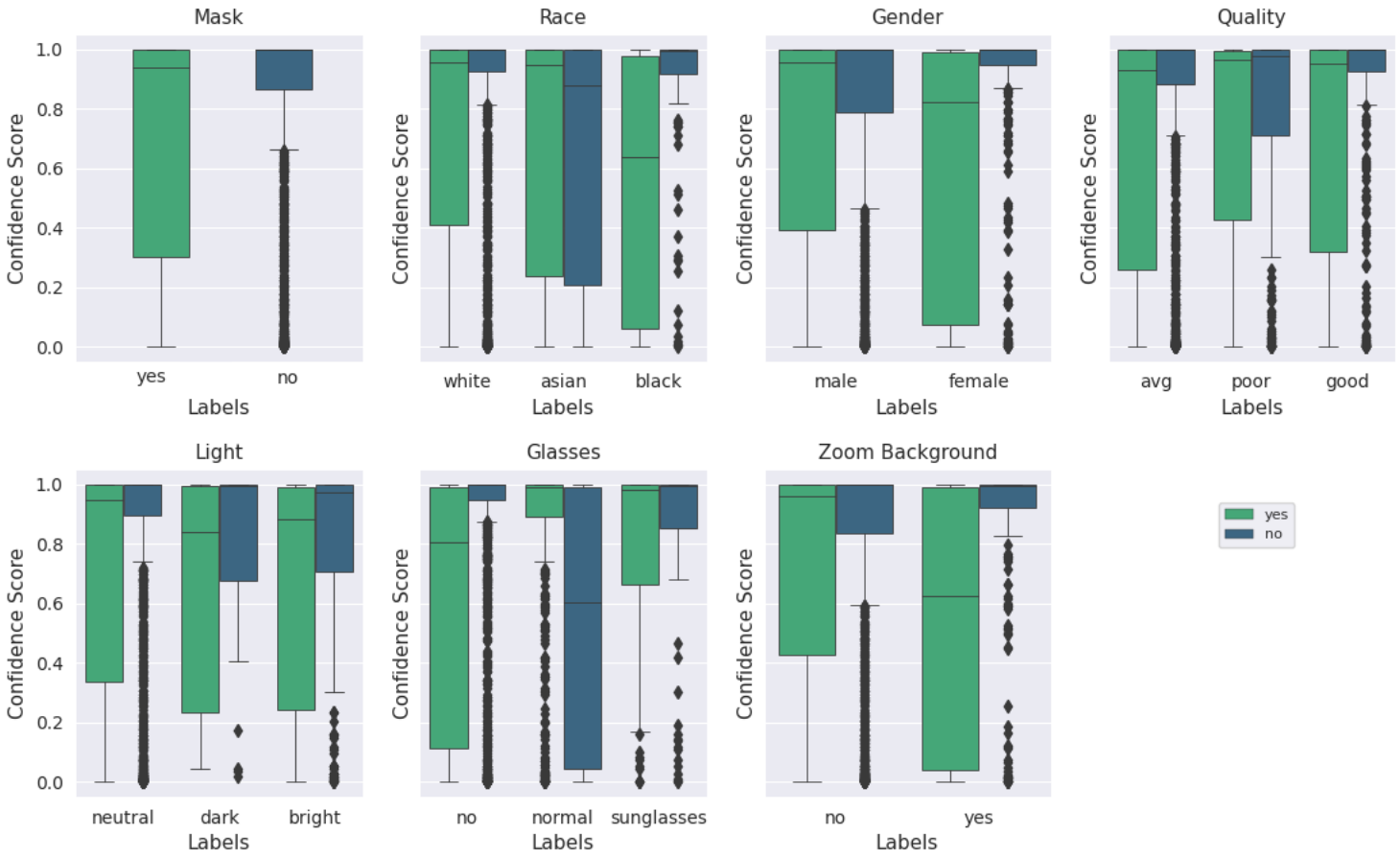

Figure 4.10: Box plot of categorical labels vs. confidence scores for all image qualities, as qualitatively evaluated, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' scores are plotted beside each other. The median is represented by the center line of each box while the upper and lower bounds represent the respective quartiles. Outliers are represented by diamond symbols.

Box plots are a useful way to illustrate basic statistics in a set of data, thus simplifying

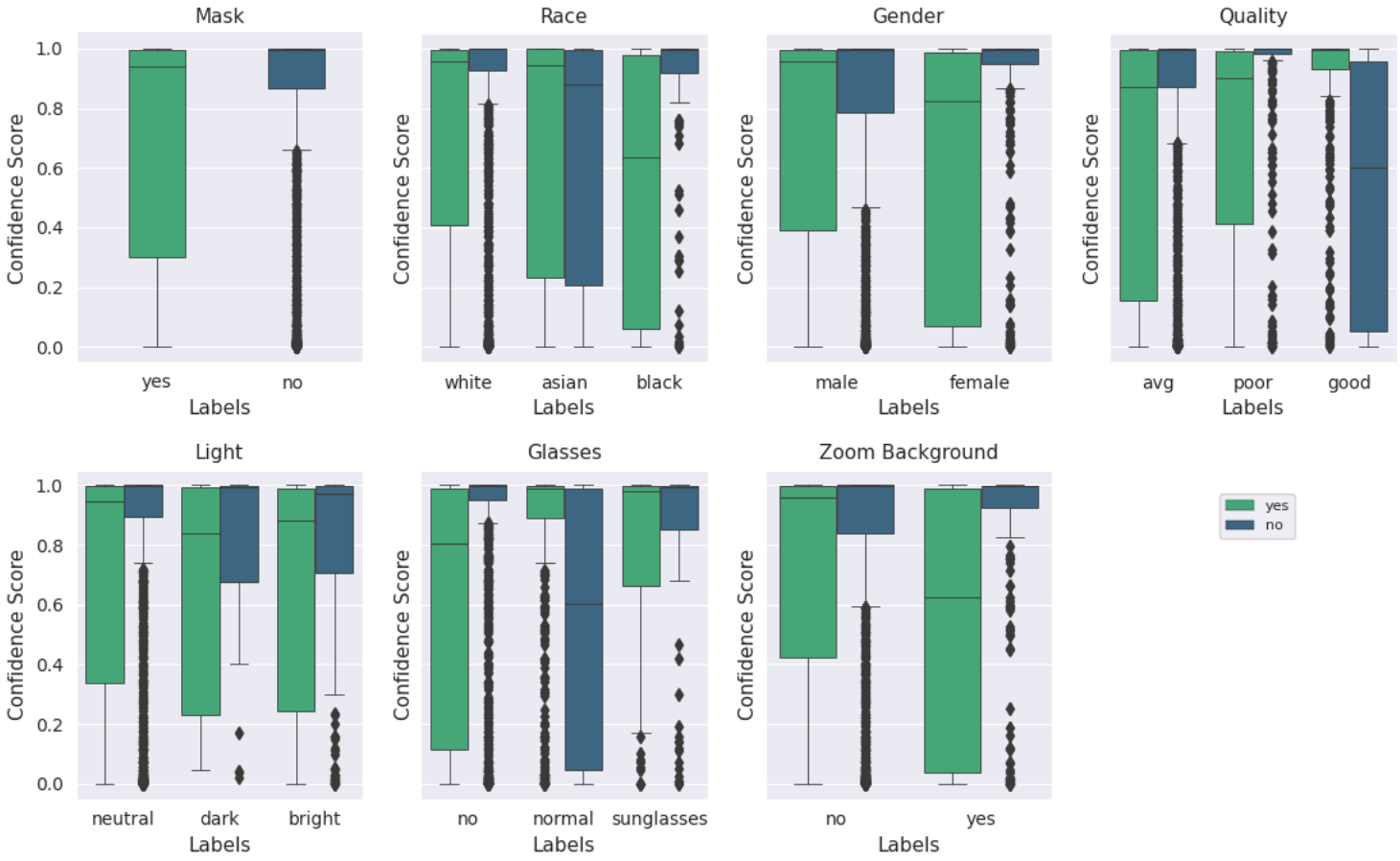

Figure 4.11: Box plot of categorical labels vs. confidence scores for all image qualities, as evaluated via BRISQUE, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' scores are plotted beside each other. The median is represented by the center line of each box while the upper and lower bounds represent the respective quartiles. Outliers are represented by diamond symbols. The primary difference between this plot and Figure 4.10 is the 'Quality' subplot, which shows the difference in the original subjective and the BRISQUE quantitative quality categories.

the identification of relationships and distributions that may lie within the data [137]. The upper and lower extrema are represented by the top and bottom lines, respectively; similarly, the upper and lower quartiles are represented by the top and bottom of the box. The median of the data is represented by the center line. For this application, box plots help visualize how each category and label are related to the confidence score output by the model during classification. Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13 consist

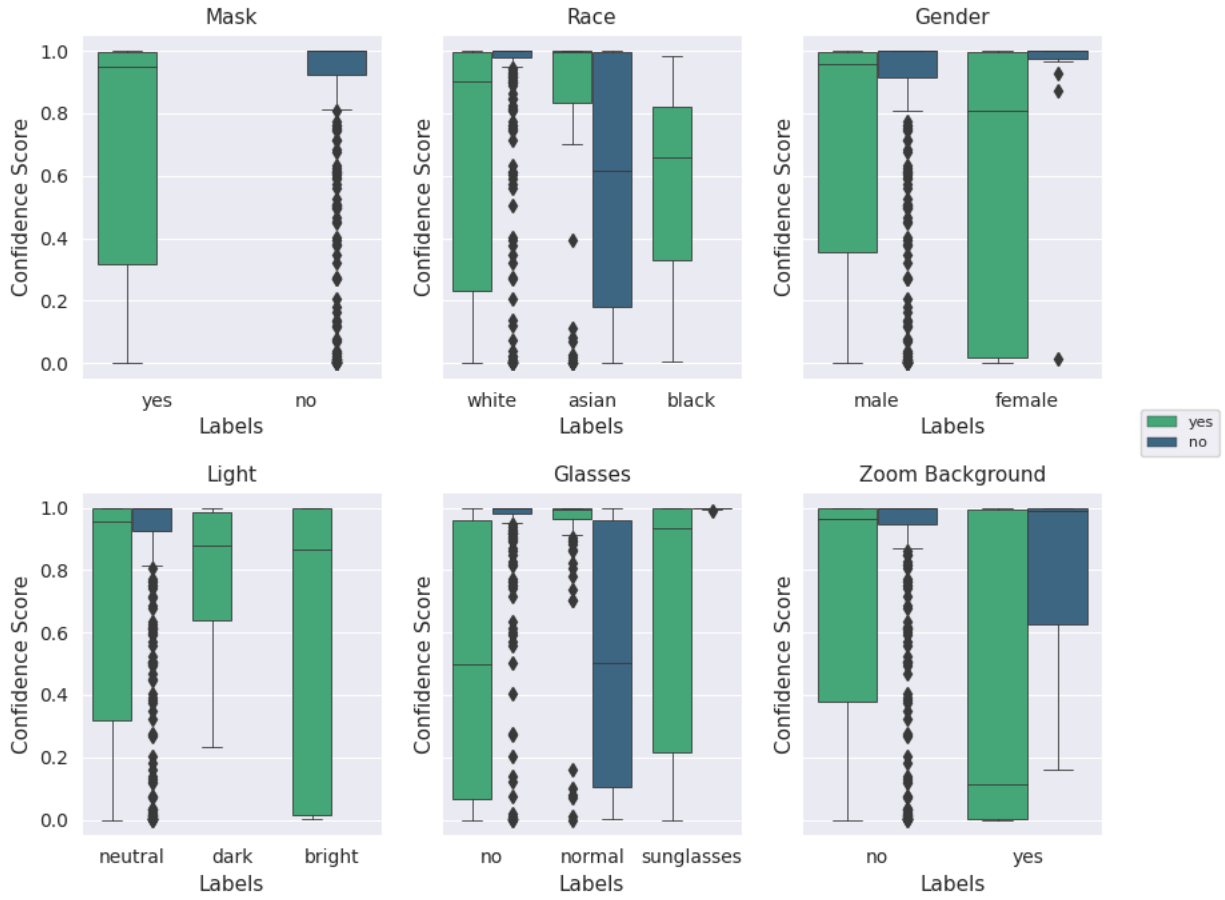

Figure 4.12: Box plot of categorical labels vs. confidence scores for only good quality images, as qualitatively evaluated, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' scores are plotted beside each other. The median is represented by the center line of each box while the upper and lower bounds represent the respective quartiles. Outliers are represented by diamond symbols.

of box plots displaying the categorical label distributions for 'mask' and 'no mask' images. On the vertical axes, *confidence* scale refers to the outputs from the model for the correct classification label: during evaluation, a softmax function is used to scale the two model outputs between 0 and 1 so that the sum is 1. P_0 corresponds to the 'mask' label and P_1 corresponds to the 'no mask' label. For instance, an image of an individual wearing a mask with output confidence scores $P_0, P_1 = [0.7290, 0.271]$ would be a correctly classified 'mask' sample with 72.9% confidence. If the confidence value for the correct label is less than 0.5, the

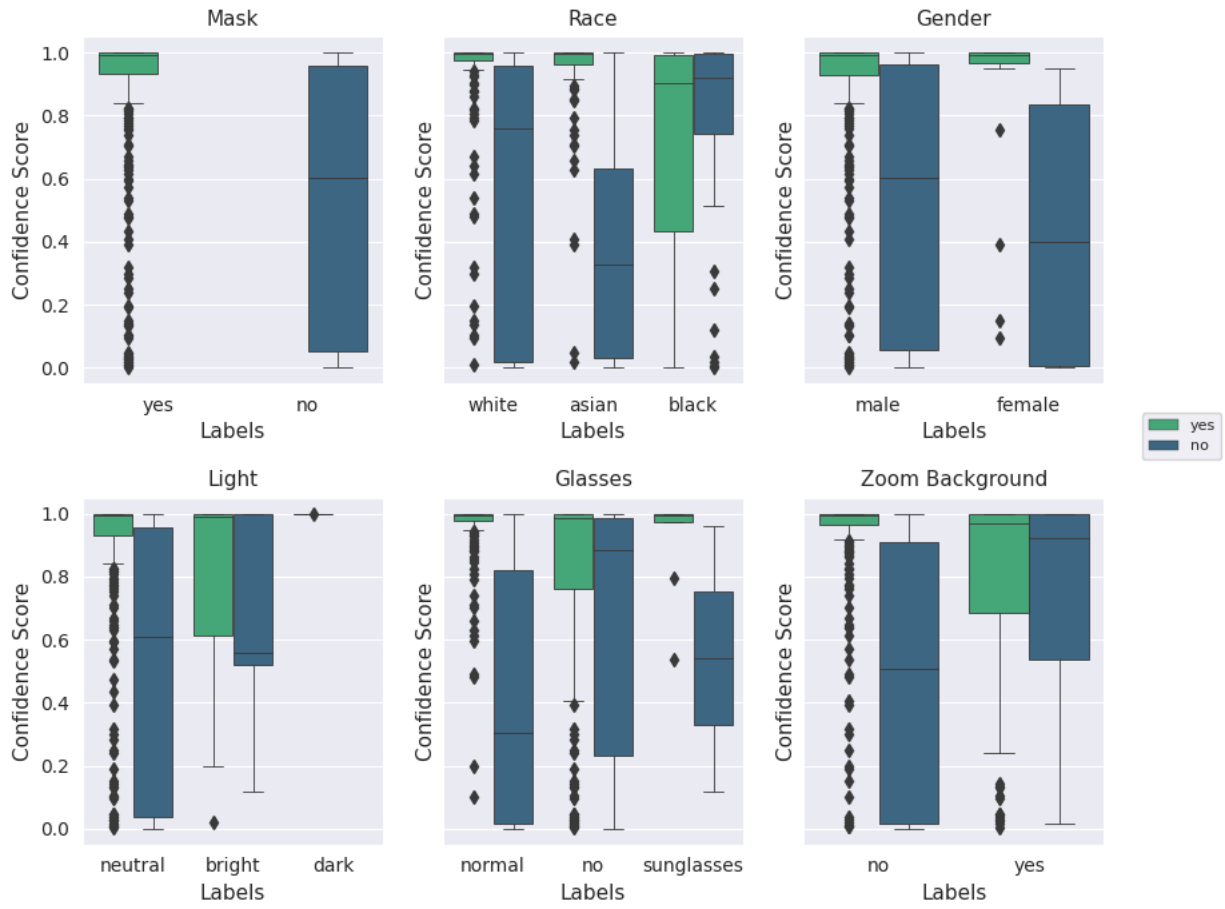

Figure 4.13: Box plot of categorical labels vs. confidence scores for only good quality images, as evaluated via BRISQUE, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' scores are plotted beside each other. The median is represented by the center line of each box while the upper and lower bounds represent the respective quartiles. Outliers are represented by diamond symbols.

image was labeled incorrectly by the neural network. The horizontal axis for each plot lists the categorical label names. For each of these labels, the boxes are plotted directly alongside each other for 'mask' and 'no mask' cases. For example, referring to Figure 4.13, the 'glasses' category can be 'no', 'normal', or 'sunglasses' for each variation of glasses-wearing individual. Figure 4.14, Figure 4.15, Figure 4.16, and Figure 4.17 are stacked probability histograms that complement the aforementioned box plots by illustrating the probability distributions on the vertical axis across each category and its labels across the classification confidence scores on

the horizontal axis. The kernel density estimate (KDE) is also displayed as a visual aid for comparisons. In Figure 4.16 and Figure 4.17, the small sample size for high quality images is apparent for images of black individuals, individuals wearing sunglasses, and in dim and bright illumination settings.

There are relatively few examples of individuals wearing sunglasses, compared to individuals wearing normal eyeglasses or no glasses at all. Images with an individual wearing normal glasses and no mask had a lower typical confidence score than images with an individual wearing normal glasses and a mask.

Figure 4.12 and Figure 4.13 show that the 'mask' images of 'good' quality were generally classified with higher accuracy and a higher confidence score. Moreover, the importance of quantitative quality labelling is apparent based on Figure 4.12 and Figure 4.13 that the distribution of the confidence score for the 'mask' label is significantly different for the qualitative quality label and the BRISQUE quality label. The performance on images filtered for 'good' quality via the BRISQUE metric is also notable as it is much poorer than that for the qualitative metric for the 'no mask' category, which is believed to be due to the greater number of 'no mask' images in the qualitatively labeled set (320) compared to the number of 'no mask' images in the BRISQUE quality set. Across all images (excluding the 'other' category), 'no mask' labelled images make up 46.28% of the data. In the 'good' (BRISQUE) quality category, they make up only 26.76% of the data. In the 'good' (qualitative) quality category, 'no mask' images make up 62.69% of the data. Therefore, the discrepancy can likely be attributed to the generally poorer performance of 'no mask' images and the lower number of images after filtering for quality. Although this is counter-intuitive as the training dataset was largely composed of photos of individuals not wearing masks, and one would expect higher performance on the same type of image, the images captured in the Zoom experiments are still of poorer quality and vary more in characteristics such as race than

the training dataset, even when taking into account the Instagram data inclusion. In the hypothetical end application where images are being taken in a fixed location, much better consistency is anticipated in the image quality (fixed or slowly varying ambient conditions as well as a single camera of interest) and other controllable environmental factors.

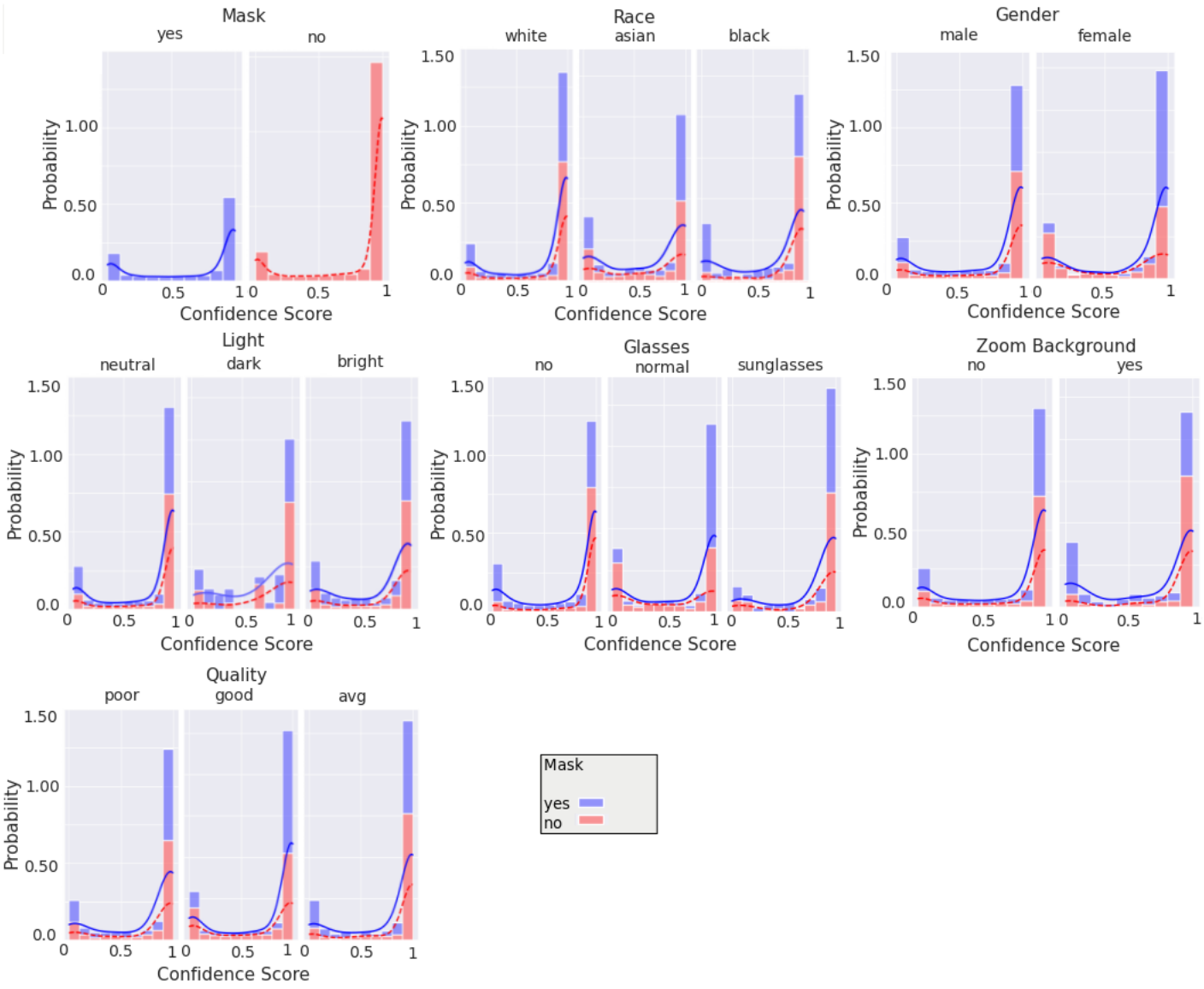

Figure 4.14: Histogram of categorical labels vs. confidence scores for all image qualities, as qualitatively evaluated, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' data bins are stacked. The solid and dashed lines represent the KDE for 'mask' and 'no mask' image scores, respectively.

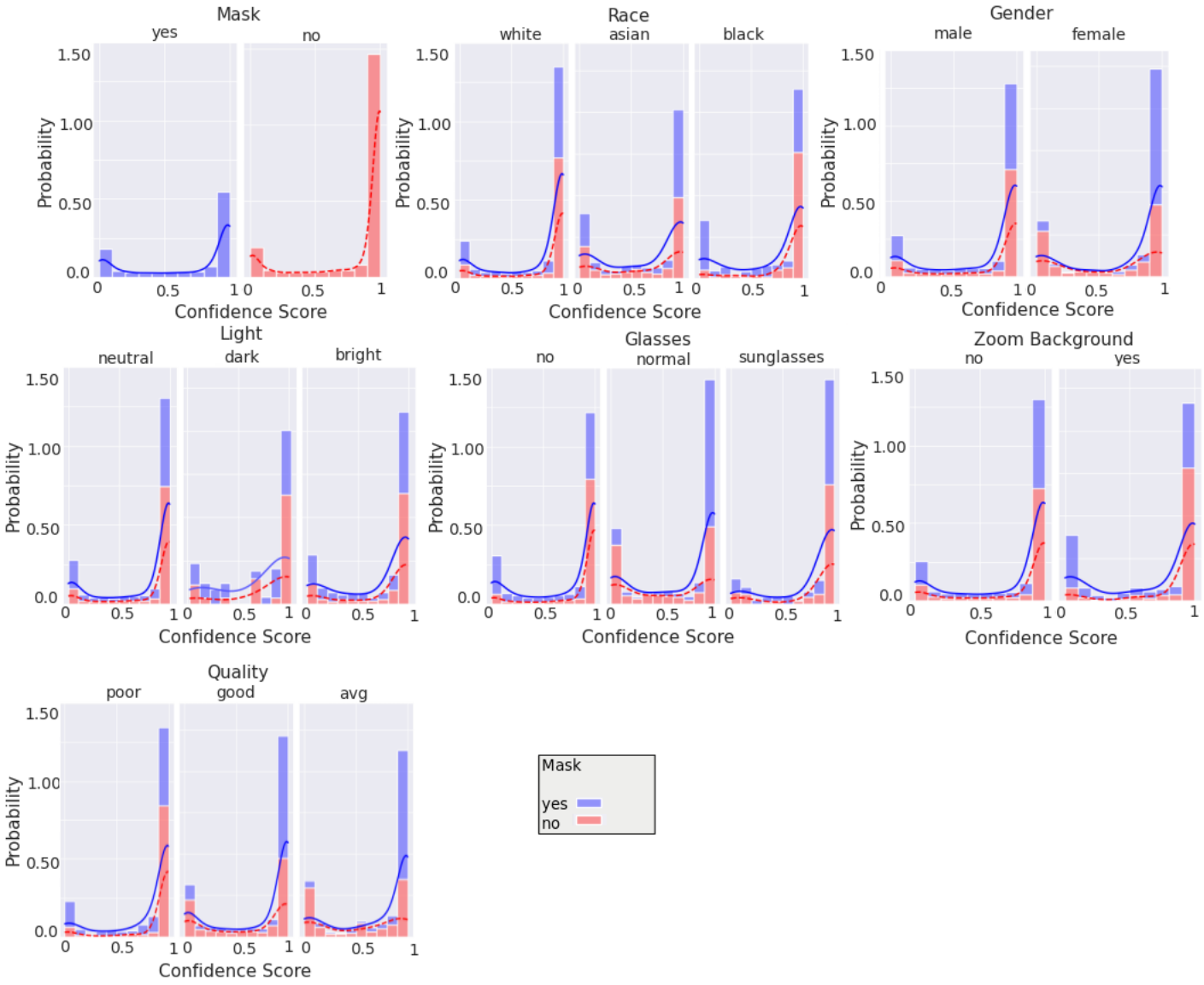

Figure 4.15: Histogram of categorical labels vs. confidence scores for all image qualities, as evaluated via BRISQUE, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' data bins are stacked. The solid and dashed lines represent the KDE for 'mask' and 'no mask' image scores, respectively.

4.3.3 Uncategorized Data

Images from the Zoom dataset labeled as 'other' contain an assortment of different conditions that may confound results from the dataset as a whole. Many of these images were difficult

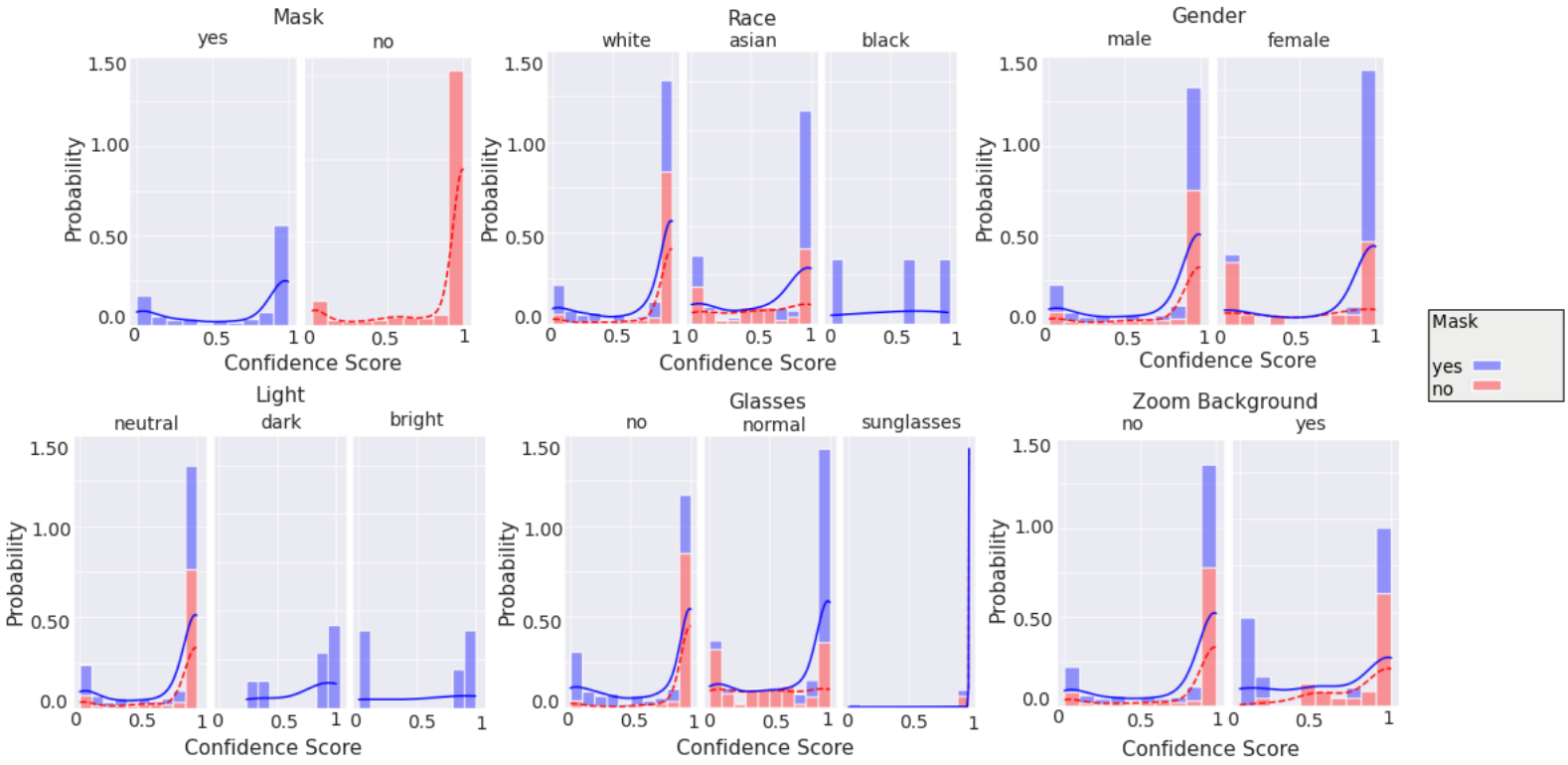

Figure 4.16: Histogram of categorical labels vs. confidence scores for only good quality images, as qualitatively evaluated, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' data bins are stacked. The solid and dashed lines represent the KDE for 'mask' and 'no mask' image scores, respectively.

if not impossible for a human to classify as having an individual wearing a mask or not. For example, Figure 4.5 is an image of a dog wearing a mask. Other examples of images in this category contain people wearing Halloween masks, substantial occlusions caused by objects held in front of the face, etc. A richer dataset, generated in a more controlled context, would be necessary to quantitatively measure the degradations of each of these factors.

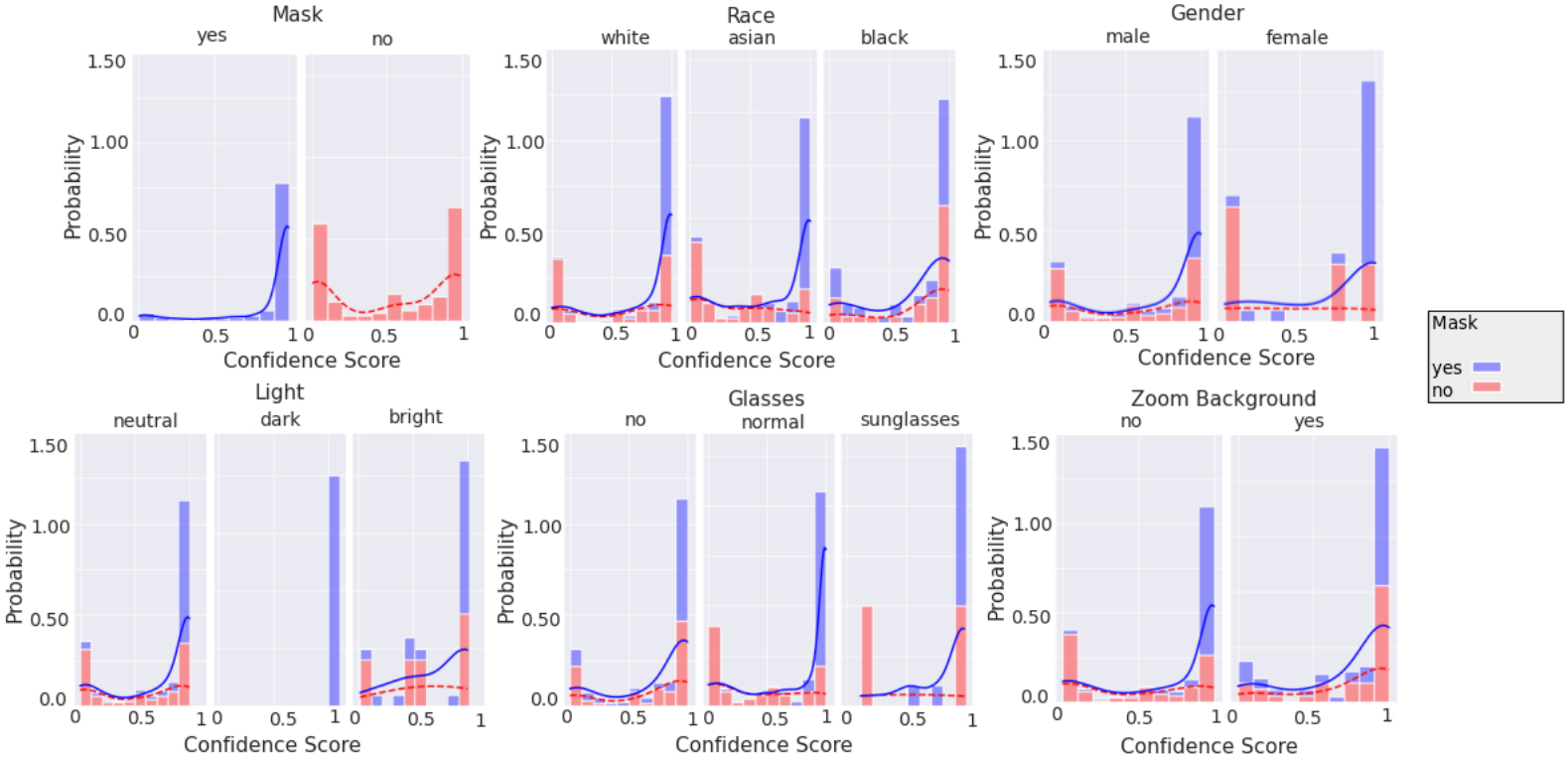

Figure 4.17: Histogram of categorical labels vs. confidence scores for only good quality images, as evaluated via BRISQUE, using the augmented MobileNetv2 model. For each category and its labels, 'mask' and 'no mask' data bins are stacked. The solid and dashed lines represent the KDE for 'mask' and 'no mask' image scores, respectively.

4.4 Live Recording

To extend our experiment one step closer towards real-world deployment, a camera was setup to capture recordings of individuals walking in public with masks on. The recordings were transferred via USB cable to a laptop computer, where they were then evaluated in near-real-time as the video was being captured and as still frames extracted from the video data. The emphasis of this trial was to evaluate the system as implemented in real-world scenarios with variations in illumination, demographics, and angles.

A Logitech C922 Pro Stream web camera [138] with 720P recording at 10FPS, 1280x720 resolution was used to test near real-time performance. A JupyterLab notebook was used

to capture input from the webcam. Haar face detection was used to identify and bound faces in a frame. For testing purposes, frames containing detected faces were first saved to a PC, then input to the CNN. The label confidence values were averaged over 15 frames to improve performance accuracy. Widgets [139] were used to display the confidence of the label predictions as they were calculated. The total cost of this experiment, not including the laptop computer used for running the model, was approximately $150USD. Recordings were captured in Blacksburg, Virginia on the Virginia Tech campus in locations where there are frequently many people walking, such as near libraries, dining halls, and parking garages. All of the video recordings were taken during the day, but weather variations contributed to data capture of individuals in sunny (bright) and cloudy (normal to dark) illumination conditions. The time recorded per session varied between 10 to 25 minutes. In two attempts to record data, areas where crowds were expected - a typically populated open field and a scenic pond with ducks on the Virginia Tech campus - human encounters were few and far between, so the session was cut short. The most fruitful recording was captured at

Figure 4.18: The Virginia Tech drillfield, a typically popular spot for students to gather, serves as an example of a potentially ideal spot to record live data for analysis.

the Blacksburg Farmer's Market, located adjacent to the university campus. This session consisted of 22:01 minutes of video capture including camera setup, readjustment of the lens, and time in which no people walked closely past the lens. In total, images across over twenty individuals were evaluated; individuals with an insufficient number of suitable images, at too far a distance, or ones not facing the camera were not included.

4.4.1 Results of Live Recordings

The live experimental video captures proved to be more challenging than anticipated from both a practical and technical perspective. The session at the Blacksburg Farmer's Market was affected by numerous factors: a poor choice of camera setup in which too many people and too many faces would be recorded at once, from too far away (approximately 20ft). The weather also made some faces difficult to detect even when people walked directly towards the camera, due to sun glare and reflections. From a subset containing 173 images that were manually labeled, after face detection and bounding, 18%, 68%, and 13% of the images were respectively categorized as 'poor', 'average', and 'good' via BRISQUE quality evaluation. When input to the model, overall accuracy was only 67.05%; however, when filtered to contain only 'good' quality images, 90.47% accuracy was achieved.

Initially, the live recording system was programmed and tested using a webcam placed directly in front of a test subject. From a detected face, 15 consecutive frames were taken. In a controlled environment, each frame capture and inference on average took an average of 120ms. Inference post-frame capture and processing took on average 60ms. The total time for a decision was on average 2 seconds. In contrast to this design, individuals in the live recording testing frequently did not face the camera, even when it was placed in an area that would seemingly encourage facing forward, such as a library entrance. This was especially

Table 4.7: Results of live recordings across 5 categories for describing attributes of individuals' appearances.

Category	Category Counts (Individuals)		
Mask	**Mask**	**No Mask**	
	36	1	
Ethnicity	**Caucasian**	**Asian**	
	33	4	
Gender	**Female**	**Male**	
	20	17	
Light	**Neutral**	**Bright**	
	7	30	
Glasses	**No**	**Normal**	**Sunglasses**
	28	3	6

problematic for Haar facial detection; if a face was not detected, it could not be bounded and input to the model, essentially adding more total time per individual to make a decision. Furthermore, individuals often walked in groups of two or more people. This made it difficult to obtain consistent performance results, as the system would often recognize different faces every few frames. However, the data flow could easily be reworked to display confidence results for each face recognized as opposed to a display of a single result and visualization plot for an individual.

The images gathered from these sessions consisted of typically young to middle-aged Caucasian adults. Additionally, no individual was seen without a mask on. One way to remedy this would be to record data in a setting where less people wear masks, but in a safe way for those around them: e.g., on a running trail or a dog park, where it has been observed by the authors that people adhere to social distancing principles while not wearing masks. Table 4.7 contains the categorical result counts. In contrast to Table 4.3, which contains the

results counted by number of occurrences over all of the image frames, Table 4.7 contains the unique individual counts over the entire video recording (e.g., a Caucasian woman wearing a mask and occurring in multiple frames was only tallied once for Table 4.7).

Further implementations of this setup would be improved by incorporating image pre-processing to correct for exposure and contrast before attempting to detect a face. An additional observation was that many people avoided the camera, which was intentionally placed in a conspicuous location. Additional experiments may be benefited by camouflaging the camera and tripod, or placing LED lights or other attention-grabbing mechanisms around the camera so that individuals are more likely to willingly look approximately into the boresight of the camera.

4.4.2 FPS Analysis

In order to determine the optimal performance of the system for a given FPS, a two-minute recording was selected and processed at 1FPS, 2FPS, 3FPS, 10FPS, and 30FPS. This process was used to find the effective decision accuracy of the system; i.e., how many 'good' decisions are needed based on the number of images to be evaluated in order to make a correct inference, and how long (how many frames) are necessary? Through bifurcation of a set of images to be evaluated based on quality, three separate approaches to this analysis are explored:

- *Discard*: Toss the 'bad' images based on BRISQUE quality assessment and classify k images based on the remaining images.
- *Consensus*: Implement a voting system so that in a set of images of quantity k, classification decisions made on higher quality images are weighed more heavily than those made on poor quality images.

- *Discard, then find consensus*: Combine the 'discard' and 'consensus' mechanisms by first throwing away poor quality images, and making the final classification decision by weighing the individual inference outputs from the remaining images according to their quality score.

The *Discard* method is based on the BRISQUE quality score of the images. Images with a score above the second quartile of the quality scores from the image set were not considered for classification decisions. The mean confidence output of the remaining images was calculated and used as the final output classification. The *Consensus* method uses all images, but weighs each decision based on the quality score of the image and the confidence score of the inference decision. Scores below the first quartile are given a 1.25x 'vote', scores above the first quartile but below the second quartile are given a 1x vote, and so on and so forth for the remaining quartiles. The *Discard + Consensus* method simply consists of the aforementioned methods combined sequentially.

Table 4.8 displays the performance results for video of an individual wearing a mask sampled at 1, 2, 3, 15, and 30FPS. The set of {N = 3, 5, or 10} images were selected in consecutive order as to simulate a real-life scenario. Overall, first discarding images of poor quality, then weighting votes for classification generally improves the probability P of a correct classification score. However, it is important to note that these results are from a single video; additional images and testing of these methods is an effort to be undertaken in future work. Furthermore, the analysis of binomial reduction of multinomial classification decisions across multiple real-time images is being separately prepared for publication.

4.5 Conclusions & Future Work

Summarily, the main contributions of this work are twofold:

Table 4.8: Results from video sampled at varying FPS rates. N is the number of images used. *P* is the overall probability of correct classification based on N images. *M1*, *M2*, and *M3* refer to the 'Discard', 'Consensus', and 'Discard + Consensus' methods described in subsection 4.4.2 for image selection.

t (s)	FPS	N	P	M1	M2	M3
3	**1**	**3**	32.66	48.88	49.37	86.70
5	**1**	**5**	31.92	65.80	56.00	90.87
1.5	**2**	**3**	49.85	97.86	56.52	89.22
2.5	**2**	**5**	69.23	97.86	78.56	100
5	**2**	**10**	73.84	98.53	82.80	100
1	**3**	**3**	70.34	56.34	72.94	88.46
1.66	**3**	**5**	79.45	94.22	88.84	100
3.33	**3**	**10**	58.69	94.22	74.39	100
0.3	**10**	**3**	78.72	67.06	78.72	91.42
0.5	**10**	**5**	74.99	63.18	85.97	93.29
1	**10**	**10**	75.67	60.80	74.62	81.13
0.2	**15**	**3**	49.05	65.72	36.91	18.88
0.33	**15**	**5**	53.12	53.82	50.72	69.81
0.66	**15**	**10**	65.22	67.44	63.80	74.74
0.1	**30**	**3**	69.49	56.27	72.78	88.45
0.16	**30**	**5**	62.53	54.17	75.34	86.58
0.33	**30**	**10**	59.95	57.35	55.21	76.59

- A unique experimental dataset in which participants of varying demographics were encouraged to try presentation attacks was recorded, manually labeled, and evaluated. The primary finding from the analysis was that the model performs reasonably well on individuals wearing and not wearing a mask provided that the image quality is sound. Furthermore, quantitative image quality was found to impact the distributions of the results as compared to qualitative image quality, which is subject to the typical

problems associated with human behavior (i.e., bias and errors.)

- The characterization of the feasibility of real-world implementation of a mask recognition system, including consideration of hardware resources and time in terms of frames captured. This analysis also took into account how real-world conditions such as lighting variations due to weather and head angle changes may impact performance of the system, and how to mitigate this impact by recording for the necessary number of frames necessary for a high-confidence classification.

Ultimately, the fundamental goal of this work was to prove the feasibility of low-cost deployment of a mask recognition system with acceptably high confidence outputs. In lieu of Jupyter Widgets, a red/green traffic-light style indicator could easily be integrated with the setup to provide feedback. Furthermore, implementation on a low-cost device such as a Raspberry Pi or Arduino could be used to implement the mask recognition algorithm in real-world scenarios. One consideration for this candidate system is ensuring that the processor adjusts to the frame rate of video recordings in order to optimize performance in regards to the capabilities of both the camera and the computing system. Additionally, re-training the model for each location could be accomplished with relative ease: training data could be gathered by having a store associate manually label captured images of customers walking into the establishment, via a clicker button or Jupyter Notebook (similarly to the process used for manual labelling in this chapter); this new data could then be utilized for re-training a model to be deployed. This would result in a uniquely trained model for each location that the mask recognition system would be used in. Although the performance impact caused by differences in illumination, camera position, and background objects in a new setting versus the typical training data setting is unclear, the aforementioned re-training process would help remedy any negative affects due to a change in location.

There remains a fundamental question as to whether or not a neural network is 'overkill' for this application. This could be revealed by testing other methods for classification. For example, a Haar cascade classifier could be trained to bound the eyes, nose, and mouth of a human face; the absence of a nose or mouth could be used to classify whether or not a mask is present without the use of a neural net. The results found in the experimental sessions show that performance is primarily affected by image quality and head pose. Image quality is a broad area of research; this work used the BRISQUE algorithm, which does not utilize distortion-based metrics (such as blur). A wider exploration of evaluation methods may reveal different impacts on classification performance than shown in this chapter. One significant issue found across the architecture in Figure 4.2 was the importance of face bounding prior to inference. Investigation into faster, smaller, and more accurate methods than Haar cascade classifiers and MTCNN for this would be helpful for real-world implementations.

Additionally, no specific distribution of image properties (such as file format (e.g., PNG, JPG), pixelation, and intensity), nor demographic characteristics (such as race and gender) were targeted when the images were collected other than whether or not the subject of the image is wearing a mask. As detailed in [124], it is straightforward to speculate that analyzing the parameter distributions of the RMFRD training set images and then augmenting the training dataset with additional images either collected from other sources or modified after collection such that the resulting dataset has a more uniform range of parameter distributions (as compared to the original dataset) may enhance the model's ability to generalize and improve test and implementation performance compared to the results in this chapter. Further, it is believed that model performance will also improve if the training dataset is independently augmented with images in the specific environment of interest (e.g., each independent installation of the mask recognition device is updated during installation), though that expectation was not sufficiently tested.

In conclusion, this is an easily-marketable experimental setup that could be produced inexpensively. The authors encourage those with an interest to replicate and improve the work in this chapter based on the suggestions given above and lessons learned about implementation.

Chapter 5

Neuromorphic Computing

The framework of Chapter 3 was designed to boost the efficiency of face recognition inference tasks through the conservation of time and power. Continuing to emphasize the efficiency of predictions, Chapter 4 applies the metric-based categorization of images (via quality using the BRISQUE algorithm) in the development and performance evaluation of a mask recognition system. Both of the systems presented in Chapter 3 and Chapter 4 were developed with the consideration of real-time deployment on resource-limited hardware kept in mind. This chapter introduces a candidate for deployment of these systems and other real-world deep neural network-based applications: extreme low-power hardware devices in the cutting-edge field of neuromorphic computing.

Neuromorphic computing is a research field that broadly encompasses the development of hardware architectures and algorithms that function and utilize structures in a manner similar to brains, which are naturally low-power parallel systems that (usually) learn, respond, and adapt to real-time stimuli with relative ease [140]. The neural networks referred to in earlier chapters borrow terminology found in neuroscience, but are not typically developed to mimic the same concepts as brains use to process information. One group of algorithms developed specifically for deployment on neuromorphic hardware is the spiking neural network (SNN). SNNs are similar to the aforementioned networks, but typically seek to align with biological principles found in neurons such as 'spike' output events from a node.

This chapter presents the steps undertaken in an effort to evaluate selected approaches to the

transformation of artificial neural network (ANN) models into SNN models. By deploying converted SNN models to hardware platforms, then analyzing the power consumption and performance of the models during inference, the overall value of the SNN-to-ANN conversion algorithms tested can be assessed. The software tools leveraged for this analysis are Whetstone [141] and the SNN Conversion Toolbox (SNN-TB), [142]. Both of these tools are based on publications ([143], [144]) by research institutions such as Sandia National Laboratories and the Institute of Neuroinformatics at the University of Zurich.

5.1 Background

5.1.1 Spiking Neural Networks

Spiking neural networks (SNNs) are a type of model that mimics biological information processing through the incorporation of spiking (or 'integrate-and-fire' neurons to perform computations. Similarly to how neurons operate based on membrane voltages in the brain, the output of the spiking neuron is a set of spikes in time that represent when the activation threshold of the neuron was reached (i.e., when the neuron 'fired') [145]. SNNs can be more power-efficient than traditional neural networks due to their event-driven nature; they also operate in parallel and process information asynchronously. Data inputs are encoded as either *rate codes*, which average the number of spikes over windows of time, or *pulse codes*, or temporal codes, which are based on the time changes between spikes [146].

One of the archaic biological neuron models is the Hodgkin-Huxley model of 1952, which describes neurons and their behaviors based on chemicals in the brain such as potassium using differential equations [147]. Although their roots come from the Hodgkin-Huxley model, the architecture of modern SNNs is structured in a still biologically-plausible, yet less complex

way; in this context, the words 'neuron' and 'synapse' oft do not literally mean neurons and synapses in the brain.

One of the basic structures used in SNNs is the basic LIF (leaky 'integrate-and-fire') neuron is modeled as an RC circuit given by:

$$\tau \frac{du}{dt} = -u(t) + RI(t) \tag{5.1}$$

where τ is the time constant RC, R is resistance, C is capacitance, u is voltage/membrane potential, and I is current. A spike $\delta(t-t^f)$ is fired when $u(t)$ equals or exceeds the threshold voltage θ, where $t^(f)$ is the firing time. $u(t)$ is set to its resting potential for a time determined by the refractory period [148].

The outputs of the neurons used in SNNs are generally discontinuous as they are composed of discrete spike 'trains'; this is problematic for the backpropagation algorithm typically used to train neural networks as they are non-differentiable [149]. However, this issue has been circumvented in various ways including training ANNs with binarized activation functions, converting trained ANNs to spiking equivalents, and the use of modified backpropagation algorithms and local learning rules [150].

One well-known example of an approach to training SNNs with modified backpropagation is developed in [151], which equivocates discrete spiking outputs as noise to allow for differentiation of the potential and spiking signals of a neuron. Another notable example is SpikeProp [152], which instantiates a learning rule for error-backpropagation and approximates the spiking outputs from the neuron when its potential meets threshold value θ to account for the aforementioned discontinuity problem. Additionally, modifications have been developed for SpikeProp, which improve factors such as performance accuracy [153] and speed [154].

Learning rules can be introduced to implement additional features to SNNs. For exam-

ple, STDP (spike-timing-dependent plasticity) modifies the weight of connections between neurons, or synapses, based on the difference between input (pre-synaptic) and output (post-synaptic) spike occurrences over a window of time [146].

The conversion of an artificial neural network to a spiking neural network provides a way to take advantage of current architectures while also enabling them to be used on neuromorphic hardware. Artificial neural nets converted to spiking CNNs and fully-connected feedforward SNNs have shown up to $\geq 99\%$ accuracy for classifying MNIST data, which is comparable to pre-conversion performance [149]. In [155], a method for sharing weights between ANN and SNN models with a tandem learning rule is presented. The method was tested using the TensorFlow Tensorpack toolbox on the CIFAR-10 dataset with a 6-layer CNN, and the ImageNet-2012 dataset with AlexNet [35].

5.1.2 Approaches to Implementation

The development of neuromorphic hardware for SNN deployment has resulted in several platforms such as in [83], [84], and [156]. One of the most utilized and easily-accessible hardware platforms is SpiNNaker [157], the name being an acronym for 'Spiking Neural Network Architecture'. SpiNNaker is a massively-parallel low-power computing system intended for SNN simulations. To communicate spike event location and timing information from an individual neuron model within a single processing core (of which there are over one million), data packets are routed to other cores via parent nodes. Although SpiNNaker is a digital architecture, it uses an event-driven software model to ensure processors are in a low-power state until an event such as a spike occurs. SpiNNaker aims to model the human brain, which can perform computations at a rate of approximately 10^18 FLOPs while consuming only around 20W [158]; the SpiNNaker system is stated to be capable of around 10^14 (non-

floating) operations per second, and a system with one million cores was estimated to use 90kW of power [159]. Additionally, the HBP Neuromorphic Computing Platform can be used to remotely submit and run simulations on SpiNNaker, which is physically located in the United Kingdom, and to run emulations on a system called BrainScaleS [160], located in Germany. Access to the platform is available for research purposes.

Software Simulations

SNN models are often simulated in software due to the additional flexibility and readily accessible development process as compared to implementation on neuromorphic hardware. Additionally, software simulation of SNNs allows for more rapid prototyping of new architectures than allowed for in hardware implementations. Numerous software packages have been built for simulating artificial and spiking neural nets, often in C++ and Python. In the case of SNNs, several packages focus on simulations for neurological applications (as in, they attempt to model the human brain from a neuroscience and medical perspective) as opposed to performance in machine learning applications.

pyNN [161] is an open-source, Python-based API for simulating neural nets and supports models for standard neuron, synapse and synaptic plasticity. The models are capable of being run on SNN simulators such as NEST [162]. sPyNNaker [163] is a software package based on pyNN for simulating neural nets on the SpiNNaker platform [157]. Many other Python-based simulators exist. Spike [164] is a Python-based tool optimized to simulate SNNs on GPUs. Evaluated using the Vogels-Abbott model benchmark simulation [165] and compared to GeNN [166], Auryn [167], Brian2 [168], ANNarchy [169], and NEST [162] simulators, Spike achieved the fastest normalized simulation run time with a synaptic delay of 0.8ms, and a run time similar to GeNN with a 0.1ms synaptic delay. BindsNET [170] is a SNN library based on PyTorch. Nengo [171] is a simulator based on the Neural Engineering Framework

[172], which provides quantitative guidelines for the creation of neural frameworks. There are several backends for running Nengo models. For example, Nengo-Loihi runs Nengo models on Loihi and also simulates Loihi; Nengo-SpiNNaker simulates Nengo models for SpiNNaker and always executes in real time; Nengo-OpenCL runs Nengo models on GPUs using OpenCL; Nengo-FPGA runs Nengo models on FPGAs.

5.1.3 The SNN Toolbox

A method to convert ANNs to SNNs with various modifications is developed in [144]. A Python library implementing the modifications is available at [142].

In this method, the weights of an ANN are replaced with integrate-and-fire spiking neurons in order to represent the activation values of the ANN as a rate in terms of spike firing. Conventional ANN operations are replaced with SNN-compatible implementations of biases, parameter and batch normalization, analog input to 'constant currents', softmax, and max-pooling layers. A recent work similar to the methods and analysis in [144], but with better performance on CIFAR-10 and small modifications is [173].

5.1.4 The Whetstone Algorithm

Whetstone [143] is an algorithm for training a non-spiking neural net model while progressively converting the model to a SNN through iterations of 'sharpening'. Sharpening refers to the layer-by-layer discretization of activation levels. After training, the model is converted to a SNN-compatible model.

In Whetstone models, bounded rectified linear units (bRELUs) are used for activation func-

tions due to their bounded outputs. The general bRELU [174] is given by:

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x \leq A \\ A, & x > A \end{cases} \tag{5.2}$$

where A is the maximum output value.

For the Whetstone bRELU, A = β is set to 1 and is given by (2), where h is the Heaviside function and α, β are parameters defined by $|\beta - 0.5| = |\alpha - 0.5|$ Initially $\alpha = 0$ and $\beta = 1$; as $\alpha \longrightarrow 0.5$ and $h(\alpha, \beta) \longrightarrow h$, the difference between α, β is reduced during each period of sharpening.

$$h(\alpha, \beta) = \begin{cases} 0, & x_i \leq \alpha \\ (x_i - \alpha)/(\beta - \alpha), & \alpha \leq x_i \leq \beta \\ 1, & x_i \geq \beta \end{cases} \tag{5.3}$$

In [175], the Whetstone algorithm was applied to convert MLPs and CNNs to the SpiNNaker platform and achieved high performance during inference on the Binary MNIST dataset. LIF neurons and a modified LIF neuron for instant decay were utilized for faster firing rates and denser neuron packing; training was done without a bias or with using a relayed bias via weights, using a separate neuron. It is also noted that Whetstone preserves the floating-point values of ANNs; in [175], reduced fixed-point precision weights even lower than what SpiNNaker specifies (Q4.8) were used to enable portability on hardware.

5.2 Approach

5.2.1 Algorithms and Conversion Methodologies

Supervised learning models are easier to conceptually understand and have been shown to have high performance in several tasks; ANN-to-SNN conversions have shown high performance on various data sets, as mentioned above.

Whetstone was developed for Keras models; since Keras based on TensorFlow, it was anticipated that the availability of documentation and support for various architectures will make any benchmarking and deployment on hardware such as SpiNNaker and the TPU a simpler process. Furthermore, in [175], sPyNNaker is used and the method to do so is described in relatively high detail, which makes the learning curve of this endeavour more approachable.

5.2.2 Datasets

CIFAR-10, MNIST, and Binary MNIST have been used extensively for comparison of machine learning models. Other variations of MNIST used to benchmark SNNs include MNIST-DVS, N-MNIST, MNIST Intensity-to-Latency Encoding, and MNIST Poisson Encoding.

Novel datasets that may show useful applications for the methods presented above include the THz and thermal video data set, FERET, SpaceNet, and Indoor User Movement Prediction from RSS Data. Images could be pixel-wise converted to Poisson spike trains similarly to the modified MNIST dataset.

In [149], it is mentioned that 'static frame-based benchmarks' might not be the best measure for SNNs.

5.2.3 Edge TPU

The Google Coral Edge Tensor Processing Unit (TPU) is an ASIC optimized for multiply-and-accumulate (MAC) operations [176].

The TPU utilizes TensorFlowLite models for inference and training on-board is limited. TensorFlow Lite is a lightweight version of TensorFlow optimized for low-memory, low-power devices such as cellphones and microcontrollers. A model trained in Keras or TensorFlow can be made into a TensorFlow Lite model using the TFLiteConverter tool. TensorFlow Lite models are used only for inference/prediction tasks, must be quantized, and support only a limited amount of operations. TensorFlow Lite models compiled for the Edge TPU have more restrictions. Deploying a SNN model trained via Whetstone to the Edge TPU requires that the model is first converted to TensorFlow Lite. TensorFlowLite models must use quantization-aware training or undergo post-training quantization. With MobileNetV1, MobileNetV2, and Inception V3, performance loss was minimal when floating point weights and activation values were quantized to 8-bit integers.

5.3 Summary

In [143], it is mentioned that the Whetstone method for conversion may improve the performance of other accelerators in addition to neuromorphic platforms. Additionally, Whetstone has not been as thoroughly analyzed or applied to a variety of applications for SNNs that Whetstone SNNs may show enhanced performance in. Despite the counter-intuitive application, those sentiments, combined with the readily-available hardware, suggested that evaluating traditional neural net architectures converted to spiking models and deployed on the Edge TPU may be an informative venture.

The initial goal for the implementation of converted neural network models on hardware platforms was to successfully convert a Whetstone network model to TensorFlow Lite. Afterwards, the model was to be compiled with the Edge TPU compiler and deployed onto the TPU and analyzed in terms of power consumption and inference accuracy. A repository of code for Whetstone published under GNU General Public License was utilized to compile and train SNN models [141].

Initial attempts were made to convert a Keras model to TensorflowLite and assorted difficulties with the API, package versions, and repository code were encountered. Whetstone is currently written in Python 2 with an older version of the Keras API. The process of converting a trained Whetstone model resulted in problems primarily involving Python package version incompatibilities and navigating software documentation across the resources available for different versions of TensorFlow, TensorFlow Lite, and the Edge TPU device. The Whetstone repository was written for Python 2.7, Keras 2.1.2, TensorFlow 1.3.0, Numpy 1.14.5, opencv-python 3.4.1.15, and Keras-rl 0.4. Although a model will fully build and train in an environment with those package versions, TensorFlow 1.3.0 does not include a way to convert models to TensorFlow Lite. Additionally, changes in the Whetstone library were required for newer versions of TensorFlow and Python 3. Ongoing issues included conversion of the custom layers in the repository to TensorFlow Lite.

As stated in communications with the developers, Whetstone will be ported to Python 3 and newer versions of Tensorflow and/or Keras in future updates, thereby mitigating the aforementioned software issues. Moreover, MobileNetV1 is fully compatible with and will be ported to Whetstone. MobileNetV2 will require significant modifications to Whetstone due to its output layer architecture in order to be ported, and may be infeasible.

These revelations and previously failed attempts led to to the investigation of *why* they failed, ultimately leading to the reevaluation of the approaches used in this chapter.

It was presumed that it was feasible and/or semi-practical to put an SNN model on the TPU. Additionally, the TPU is optimized for large matrix operations, which clashes with the intended structure and flow of information in spiking models - SNNs do not use large matrix operations, so using SNNs on the TPU is counter-intuitive. A further confounding factor was the fact that inference models deployed on the TPU must be in TensorFlowLite, which is quite limited as compared to TensorFlow.

5.3.1 Future Work

The modification of the systems used in Chapter 3 and Chapter 4 as to be compatible with neuromorphic hardware is a worthwhile endeavour for future work as both of these frameworks seek to be as efficient as possible. Although the results of this chapter are inconclusive, the findings support a further investigation into the utilization of SNNs and neuromorphic hardware for image recognition applications that target high efficiency.

As previously speculated, the TPU will most likely not benefit from Whetstone as it lacks support for sparse operations - resources for operations will be wasted as they cannot support less than 8 bits (speculatively). However, non-neuromorphic hardware platforms that support binary operations are expected to benefit via reduced latency and power. However, the exploration of other acceleration platforms is a worthwhile research effort and may prove more simple to execute than the ventures attempted in this chapter.

Chapter 6

Conclusions

To review, this thesis has presented fundamental information on the state of modern machine learning architectures and uses, computer vision for object recognition, and low-power hardware (primarily in the context of neuromorphic computing). Three central research ideas based on these broad fields were planned and undertaken:

- The creation of an architecture to reduce time and power consumption by neural networks used for face recognition, based on thresholds established by independently-calculated metrics meant to separate images likely to be correctly or incorrectly classified based upon observable quality features.
- A COVID-19 mask detector with a straightforward path to real-time, real-life implementation and the assembly and analysis of an interesting dataset composed of people wearing face masks and holding cats.
- The study of neuromorphic computing systems and the algorithms that could be used to make them as good or better than current hardware systems, as well as analyzing the feasibility of implementing them on non-supported hardware.

The development of the system in Chapter 4 was greatly aided by the initial research conducted beforehand for the efforts of Chapter 3 and Chapter 5. As shown by Chapter 2, the intersection of deep learning- and computer vision-based approaches to the numerous

variations of object recognition, including those applied to human faces, is a vast area of research that includes seemingly-infinite amounts of influential literature. The sheer quantity of research performed in the fields can make developing new ideas within them seem unfathomable, but the trial-and-error process of Chapter 3 contributed to a more comprehensive understanding of the current state of research on deep neural networks, image processing algorithms, and the practical application of each as compared to the basic theory known previously.

On a high level, this thesis contributes to current research by proving, albeit in a limited fashion, the concept of using independent quality metrics to bifurcate datasets into low- or high- confidence decision classes through the work of Chapter 4. This contribution is further bolstered through the framework presented in Chapter 3 and the claim is supported by the system in [107] as described in Section 3.3. Moreover, Chapter 4 proves the feasibility of a low-cost hardware platform for implementing real-time mask recognition, which is unfortunately still a relevant tool at the time this thesis was written due to the COVID-19 pandemic. An open area for future work is the improvement of characterizing the FPS necessary for high-confidence predictions, as is described in subsection 4.4.2.

Of equal importance to the technical discussion of these contributions is the consideration of demographic bias in deep learning systems and the implications of unrestrained deployment of technologies integrating these systems ([177], [178], [179]). It should be noted that the work presented does not represent a viewpoint that condones unrestricted development or use of these algorithms in ways that contribute to discriminatory or otherwise harmful practices. Although a full review of the ethical concerns raised by widespread adoption of face detection and recognition is outside of the scope of this thesis, this subject is more thoroughly established in [180] and [181]; it is left as an area of future work to analyze the systems of Chapter 3 and Chapter 4 in the context of ethical AI.

An additional area of future work for Chapter 4 is to map the mask recognition system to low-power platforms such as a Raspberry Pi [77] or to a system compatible with spiking models in order to deploy it onto neuromorphic hardware. In Chapter 5, limited results were acquired towards the mapping of MobileNetV2 to the extreme low-power hardware platforms developed with the principles of neuromorphic computing.

Overall, Chapter 5 establishes that neuromorphic hardware and the spiking models assembled for it can be perplexing subjects, as the field uses similar terminology for the structures and functions utilized within its systems as is used for both typical artificial neural networks and in biomedical neuroscience. This terminology may have *entirely different connotations* for the actual functionality of a term depending on whether the research was performed from an engineering-based or a computational neuroscience-based perspective. This 'complaint' should be taken very optimistically: the muddled-together definitions and approaches imply that the foundations of the field are still being cemented into place, and their ultimate contributions to the research community as a whole are just beginning to be constructed.

The question presented in Chapter 1 was *"How can subsets of machine learning algorithms be implemented on resource-limited systems for inference tasks in ways that provide results efficiently?"* This thesis brushes on the very top of that question - the burgeoning development of neuromorphic computing and the prevalence of CNNs for image-related inference tasks such as face recognition are thriving areas of exploration with new innovations being developed on a regular basis.

Although the projects in Chapter 3 and Chapter 5 were not as successful as initially planned, the attempts will hopefully serve to help any readers avoid similar mistakes and encourage them to explore the groundbreaking research and its implications for new technologies in the areas described in Chapter 2. The research presented in this thesis points to the continuing challenges and possible directions to solutions for problems in these fields.

Bibliography

[1] D. Crouse, H. Han, D. Chandra, B. Barbello, and A. K. Jain, "Continuous authentication of mobile user: Fusion of face image and inertial measurement unit data," in *2015 International Conference on Biometrics (ICB)*, pp. 135–142, IEEE, 2015.

[2] R. Bhatia, "Biometrics and face recognition techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, 2013.

[3] J. C. Hernández, "China's high-tech tool to fight toilet paper bandits," *The New York Times*, March 2017.

[4] Y. Zhao, S. Wu, L. Reynolds, and S. Azenkot, "A face recognition application for people with visual impairments: Understanding use beyond the lab," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2018.

[5] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[6] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[7] World Health Organization, "Listings of WHO's response to COVID-19." https://www.who.int/news/item/29-06-2020-covidtimeline, June 2020.

[8] A. Sharif, C. Aloui, and L. Yarovaya, "COVID-19 pandemic, oil prices, stock market, geopolitical risk and policy uncertainty nexus in the us economy: Fresh evidence

from the wavelet-based approach," *International Review of Financial Analysis*, vol. 70, p. 101496, 2020.

[9] G. Serafini, B. Parmigiani, A. Amerio, A. Aguglia, L. Sher, and M. Amore, "The psychological impact of COVID-19 on the mental health in the general population," *QJM: An International Journal of Medicine*, vol. 113, no. 8, pp. 531–537, 2020.

[10] T. Parker-Pope, "The scientist, the air and the virus," *The New York Times*, 2020.

[11] A. Goodnough, "Fauci expects americans could still need to wear face masks in 2022.," *The New York Times*, February 2021.

[12] J. E. Bromwich, "Fighting over masks in public is the new American pastime," *The New York Times*, 2020.

[13] C. Staff, "Ventura county sheriff: Face mask confrontation caused melee at Thousand Oaks mall," *CBS Los Angeles*, 2020.

[14] KTRK-TV ABC13, "Bar attack suspect captured, charged with assault," *ABC 13 News*, 2020.

[15] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6, IEEE, 2016.

[16] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Science and Information Conference*, pp. 128–144, Springer, 2019.

[17] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, "Towards closing the energy gap between hog and cnn features for embedded vision," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, IEEE, 2017.

[18] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[19] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[20] R. S. Williams, "What's next?[the end of Moore's law]," *Computing in Science & Engineering*, vol. 19, no. 2, pp. 7–13, 2017.

[21] M. M. Waldrop, "The chips are down for Moore's law," *Nature News*, vol. 530, no. 7589, p. 144, 2016.

[22] B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, "Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 97–110, 2019.

[23] R. Billings and A. Michaels, "Real-time mask recognition," *ACM Transactions on Internet of Things (TIOT)*, 2021. In review by ACM Transactions on Internet of Things (TIOT) journal.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[25] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks*, pp. 195–201, Springer, 1995.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] Aphex34, "File:typical_cnn.png," December 2015. CC BY 4.0 via Wikimedia Commons.

[29] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[31] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, Computational and Biological Learning Society, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[39] C. Nolan, "Inception," Warner Bros. Pictures, 2010.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[42] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[43] R. Lukac, *Computational Photography: Methods and Applications*. USA: CRC Press, Inc., 1st ed., 2010.

[44] T. Acharya and A. K. Ray, *Image processing: principles and applications*. John Wiley & Sons, 2005.

[45] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pp. 555–562, IEEE, 1998.

[46] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. USA: Academic Press, Inc., 4th ed., 2008.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[48] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[49] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[50] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 science and information conference*, pp. 372–378, IEEE, 2014.

[51] M. J. Wilber, V. Shmatikov, and S. Belongie, "Can we still avoid automatic face detection?," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9, 3 2016.

[52] A. Harvey, "CV Dazzle," 2017.

[53] S. Zafeiriou, C. Zhang, and Z. Zhang, "A survey on face detection in the wild: past, present and future," *Computer Vision and Image Understanding*, vol. 138, pp. 1–24, 2015.

[54] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[55] M. Taskiran, N. Kahraman, and C. E. Erdem, "Face recognition: Past, present and future (a review)," *Digital Signal Processing*, vol. 106, p. 102809, 2020.

[56] P. Viola, M. Jones, *et al.*, "Robust real-time object detection," *International journal of computer vision*, vol. 4, no. 34-47, p. 4, 2001.

[57] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li, "Face detection based on multi-block LBP representation," in *International conference on biometrics*, pp. 11–18, Springer, 2007.

[58] K. Kadir, M. K. Kamaruddin, H. Nasir, S. I. Safie, and Z. A. K. Bakti, "A comparative study between LBP and Haar-like features for face detection using OpenCV," in *2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)*, pp. 335–339, IEEE, 2014.

[59] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015.

[60] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[61] H. Jiang and E. Learned-Miller, "Face detection with the faster R-CNN," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pp. 650–657, IEEE, 2017.

[62] 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, 2001.

[63] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[64] K. Tieu and P. Viola, "Boosting image retrieval," *International Journal of Computer Vision*, vol. 56, no. 1, pp. 17–36, 2004.

[65] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "OpenFace: A general-purpose face recognition library with mobile applications," tech. rep., CMU-CS-16-118, CMU School of Computer Science, 2016.

[66] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[67] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2015.

[68] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, pp. 1521–1528, IEEE, 2011.

[69] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference (BMVC)* (X. Xie, M. W. Jones, and G. K. L. Tam, eds.), pp. 41.1–41.12, BMVA Press, September 2015.

[70] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pp. 67–74, IEEE, 2018.

[71] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney, and P. Grother, "IARPA Janus benchmark - C: Face dataset and protocol," in *2018 International Conference on Biometrics (ICB)*, pp. 158–165, 2 2018.

[72] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep. 07-49, University of Massachusetts, Amherst, 10 2007.

[73] G. Huang, M. Mattar, H. Lee, and E. G. Learned-Miller, "Learning to align from scratch," in *Advances in neural information processing systems*, pp. 764–772, 2012.

[74] W. Deng, J. Hu, N. Zhang, B. Chen, and J. Guo, "Fine-grained face verification: FGLFW database, baselines, and human-DCMN partnership," *Pattern Recognition*, vol. 66, pp. 63–73, 2017.

[75] N. Zhang and W. Deng, "Fine-grained LFW database," in *International Conference on Biometrics*, pp. 1–6, 2016.

[76] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2553–2561, 2013.

[77] T. R. P. Foundation, "Raspberry Pi hardware." https://www.raspberrypi.org/documentation/hardware/, 2021.

[78] P. Suja, S. Tripathi, *et al.*, "Real-time emotion recognition from facial images using Raspberry Pi II," in *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 666–670, IEEE, 2016.

[79] A. A. Wazwaz, A. O. Herbawi, M. J. Teeti, and S. Y. Hmeed, "Raspberry Pi and computers-based face detection and recognition system," in *2018 4th International Conference on Computer and Technology Applications (ICCTA)*, pp. 171–174, IEEE, 2018.

[80] I. Gupta, V. Patil, C. Kadam, and S. Dumbre, "Face detection and recognition using Raspberry Pi," in *2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pp. 83–86, IEEE, 2016.

[81] F. Leccese, M. Cagnetti, and D. Trinca, "A smart city application: A fully controlled street lighting isle based on Raspberry-Pi card, a ZigBee sensor network and WiMAX," *Sensors*, vol. 14, no. 12, pp. 24408–24424, 2014.

[82] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[83] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[84] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[85] E. Nurse, B. S. Mashford, A. J. Yepes, I. Kiral-Kornek, S. Harrer, and D. R. Freestone, "Decoding EEG and LFP signals using deep learning: heading TrueNorth," in *Proceedings of the ACM international conference on computing frontiers*, pp. 259–266, 2016.

[86] P. U. Diehl, B. U. Pedroni, A. Cassidy, P. Merolla, E. Neftci, and G. Zarrella, "Truehappiness: Neuromorphic emotion recognition on TrueNorth," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4278–4285, IEEE, 2016.

[87] Q. Liu, O. Richter, C. Nielsen, S. Sheik, G. Indiveri, and N. Qiao, "Live demonstration: Face recognition on an ultra-low power event-driven convolutional neural network asic," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1680–1681, IEEE, 2019.

[88] D. Liu, N. Bellotto, and S. Yue, "Deep spiking neural network for video-based disguise face recognition based on dynamic facial movements," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 6, pp. 1843–1855, 2019.

[89] N. Imam and T. A. Cleland, "Rapid online learning and robust recall in a neuromorphic olfactory circuit," *Nature Machine Intelligence*, vol. 2, no. 3, pp. 181–191, 2020.

[90] P. Kirkland, G. Di Caterina, J. Soraghan, and G. Matich, "Neuromorphic technologies for defence and security," in *Emerging Imaging and Sensing Technologies for Security and Defence V; and Advanced Manufacturing Technologies for Micro-and Nanosystems in Security and Defence III*, vol. 11540, p. 115400V, International Society for Optics and Photonics, 2020.

[91] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Frontiers in Neuroscience*, vol. 14, p. 199, 2020.

[92] R. M. Haralick, K. S. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, pp. 610–621, 1973.

[93] A. Eleyan and H. Demirel, "Co-occurrence matrix and its statistical features as a new approach for face recognition," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 19, no. 1, pp. 97–107, 2011.

[94] S. A. Alazawi, N. M. Shati, and A. H. Abbas, "Texture features extraction based on GLCM for face retrieval system," *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 3, pp. 1459–1467, 2019.

[95] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2018.

[96] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (New York, NY, USA), p. 1528–1540, Association for Computing Machinery, 2016.

[97] U. G. Konda Reddy Reddy and V. B. Radhakrishnan, "Fast Feature Fool: A data independent approach to universal adversarial perturbations," in *Proceedings of the British Machine Vision Conference (BMVC)* (G. B. Tae-Kyun Kim, Stefanos Zafeiriou and K. Mikolajczyk, eds.), pp. 30.1–30.12, BMVA Press, 9 2017.

[98] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2016.

[99] G. Goswami, A. Agarwal, N. K. Ratha, R. Singh, and M. Vatsa, "Detecting and mitigating adversarial perturbations for robust face recognition," *International Journal of Computer Vision*, vol. 127, pp. 719–742, 2019.

[100] P. Majumdar, A. Agarwal, R. Singh, and M. Vatsa, "Evading face recognition via partial tampering of faces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 6 2019.

[101] S. Chhabra, R. Singh, M. Vatsa, and G. Gupta, "Anonymizing k-facial attributes via adversarial perturbations," *arXiv preprint arXiv:1805.09380*, 2018.

[102] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *International Conference on Learning Representations*, 2017.

[103] A. Agarwal, R. Singh, M. Vatsa, and N. Ratha, "Are image-agnostic universal adversarial perturbations for face recognition difficult to detect?," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–7, 10 2018.

[104] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2 dataset for face recognition." https://github.com/ox-vgg/vgg_face2, 2018.

[105] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323–342, 2013.

[106] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[107] J. Hernandez-Ortega, J. Galbally, J. Fierrez, R. Haraksim, and L. Beslay, "FaceQnet: Quality assessment for face recognition based on deep learning," in *2019 International Conference on Biometrics (ICB)*, pp. 1–8, IEEE, 2019.

[108] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[109] Z. Wang and Q. Li, "Information content weighting for perceptual image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1185–1198, 2010.

[110] J. Biden, "Executive order on promoting COVID-19 safety in domestic and international travel," *The White House*, January 2021.

[111] J. Biden, "Executive order on protecting the federal workforce and requiring mask-wearing," *The White House*, January 2021.

[112] W. Lyu and G. L. Wehby, "Community use of face masks and COVID-19: Evidence from a natural experiment of state mandates in the us: Study examines impact on COVID-19 growth rates associated with state government mandates requiring face mask use in public.," *Health affairs*, vol. 39, no. 8, pp. 1419–1425, 2020.

[113] C. Gallagher, "Masks no obstacle for new NEC facial recognition system," *Reuters*, 2021.

[114] S. Nagpal, M. Singh, R. Singh, and M. Vatsa, "Deep learning for face recognition: Pride or prejudiced?," *arXiv preprint arXiv:1904.01219*, 2019.

[115] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, pp. 77–91, PMLR, 2018.

[116] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6, IEEE, 2016.

[117] S. W. Cho, N. R. Baek, M. C. Kim, J. H. Koo, J. H. Kim, and K. R. Park, "Face detection in nighttime images using visible-light camera sensors with two-step faster region-based convolutional neural network," *Sensors*, vol. 18, no. 9, p. 2995, 2018.

[118] B. F. Klare, M. J. Burge, J. C. Klontz, R. W. Vorder Bruegge, and A. K. Jain, "Face recognition performance: Role of demographic information," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1789–1801, 2012.

[119] A. Acien, A. Morales, R. Vera-Rodriguez, I. Bartolome, and J. Fierrez, "Measuring the gender and ethnicity bias in deep models for face recognition," in *Iberoamerican Congress on Pattern Recognition*, pp. 584–593, Springer, 2018.

[120] N. Srinivas, K. Ricanek, D. Michalski, D. S. Bolme, and M. King, "Face recognition algorithm bias: Performance differences on images of children and adults," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[121] Y. Chen, L. Song, and R. He, "Masquer hunter: Adversarial occlusion-aware face detection," *arXiv preprint arXiv:1709.05188*, 2017.

[122] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, *et al.*, "Masked face recognition dataset and application," *arXiv preprint arXiv:2003.09093*, 2020.

[123] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with LLE-CNNs,"

in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2682–2690, 2017.

[124] W. H. Clark IV, S. Hauser, W. C. Headley, and A. J. Michaels, "Augmentation of real-world data for neural network based automatic modulation classification," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology special issue, pp. (accepted)*, 2021.

[125] D. Hao, L. Zhang, J. Sumkin, A. Mohamed, and S. Wu, "Inaccurate labels in weakly-supervised deep learning: Automatic identification and correction and their impact on classification performance," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 9, pp. 2701–2710, 2020.

[126] A. K. Moorthy and A. C. Bovik, "Blind image quality assessment: From natural scene statistics to perceptual quality," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3350–3364, 2011.

[127] M. A. Saad, A. C. Bovik, and C. Charrier, "Blind image quality assessment: A natural scene statistics approach in the DCT domain," *IEEE transactions on Image Processing*, vol. 21, no. 8, pp. 3339–3352, 2012.

[128] OpenCV, "Open source computer vision library," 2015.

[129] The Department of Health and Human Services, *45 CFR 46 - PROTECTION OF HUMAN SUBJECTS - Subpart A - Basic HHS Policy for Protection of Human Research Subjects*, October 2019. Code of Federal Regulations.

[130] V. Albiero, X. Chen, X. Yin, G. Pang, and T. Hassner, "img2pose: Face alignment and detection via 6DoF, face pose estimation," *arXiv preprint arXiv:2012.07791*, 2020.

[131] S. Schuckers, "Presentations and attacks, and spoofs, oh my," *Image and Vision Computing*, vol. 55, pp. 26–30, 2016. Recognizing future hot topics and hard problems in biometrics research.

[132] R. Ramachandra and C. Busch, "Presentation attack detection methods for face recognition systems: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 1, pp. 1–37, 2017.

[133] I. Zoom Video Communications, "Video conferencing, web conferencing, webinars, screen sharing - zoom." https://zoom.us/, 2021. (Accessed on 02/03/2021).

[134] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (F. Loizides and B. Schmidt, eds.), pp. 87 – 90, IOS Press, 2016.

[135] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2691–2699, 2015.

[136] P. J. Phillips, J. R. Beveridge, D. S. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, S. Cheng, M. N. Teli, and H. Zhang, "On the existence of face quality measures," in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp. 1–8, IEEE, 2013.

[137] D. F. Williamson, R. A. Parker, and J. S. Kendrick, "The box plot: a simple visual method to interpret data," *Annals of internal medicine*, vol. 110, no. 11, pp. 916–921, 1989.

[138] Logitech, "Logitech c922 pro stream 1080p webcam + capture software." https://www.logitech.com/en-us/products/webcams/c922-pro-stream-webcam.960-001087.html, 2021.

[139] F. Perez and B. E. Granger, "Project Jupyter: Computational narratives as the engine of collaborative data science," *Retrieved September*, vol. 11, no. 207, p. 108, 2015.

[140] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[141] SNL-NERL, "Whetstone." https://github.com/SNL-NERL/Whetstone, October 2018.

[142] B. Rueckauer, "Spiking neural network conversion toolbox." https://github.com/NeuromorphicProcessorProject/snn_toolbox, 2020.

[143] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the whetstone method," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 86–94, 2019.

[144] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[145] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[146] N. K. Kasabov, *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Springer, 2019.

[147] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[148] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[149] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47–63, 2019.

[150] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.

[151] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[152] S. M. Bohte, J. N. Kok, and J. A. La Poutré, "SpikeProp: backpropagation for networks of spiking neurons.," in *ESANN*, pp. 419–424, 2000.

[153] B. Schrauwen and J. Van Campenhout, "Extending SpikeProp," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 1, pp. 471–475, IEEE, 2004.

[154] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 3970–3977, IEEE, 2006.

[155] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for efficient and rapid inference on deep spiking neural networks," *arXiv preprint arXiv:1907.01167*, 2019.

[156] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, "A 0.086-mm 212.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 145–158, 2018.

[157] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[158] H. Markram, "The human brain project," *Scientific American*, vol. 306, no. 6, pp. 50–55, 2012.

[159] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[160] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1947–1950, IEEE, 2010.

[161] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: a common interface for neuronal network simulators," *Frontiers in neuroinformatics*, vol. 2, p. 11, 2009.

[162] M.-O. Gewaltig and M. Diesmann, "NEST (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[163] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. Rowley, *et al.*, "sPyNNaker: a software pack-

age for running PyNN simulations on SpiNNaker," *Frontiers in neuroscience*, vol. 12, p. 816, 2018.

[164] N. Ahmad, J. B. Isbister, T. S. C. Smithe, and S. M. Stringer, "Spike: A gpu optimised spiking neural network simulator," *bioRxiv*, p. 461160, 2018.

[165] T. P. Vogels, K. Rajan, and L. F. Abbott, "Neural network dynamics," *Annu. Rev. Neurosci.*, vol. 28, pp. 357–376, 2005.

[166] E. Yavuz, J. Turner, and T. Nowotny, "GeNN: a code generation framework for accelerated brain simulations," *Scientific reports*, vol. 6, no. 1, pp. 1–14, 2016.

[167] F. Zenke and W. Gerstner, "Limits to high-speed simulations of spiking neural networks using general-purpose computers," *Frontiers in neuroinformatics*, vol. 8, p. 76, 2014.

[168] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *Elife*, vol. 8, p. e47314, 2019.

[169] J. Vitay, H. Ü. Dinkelbach, and F. H. Hamker, "Annarchy: a code generation approach to neural simulations on parallel hardware," *Frontiers in neuroinformatics*, vol. 9, p. 19, 2015.

[170] H. Hazan, D. J. Saunders, H. Khan, D. Patel, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, "Bindsnet: A machine learning-oriented spiking neural networks library in Python," *Frontiers in neuroinformatics*, vol. 12, p. 89, 2018.

[171] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in neuroinformatics*, vol. 7, p. 48, 2014.

[172] T. C. Stewart, "A technical overview of the neural engineering framework," *University of Waterloo*, 2012.

[173] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[174] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems," *Neurocomputing*, vol. 216, pp. 718–734, 2016.

[175] C. M. Vineyard, R. Dellana, J. B. Aimone, F. Rothganger, and W. M. Severa, "Low-power deep learning inference using the SpiNNaker neuromorphic platform," in *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, pp. 1–7, 2019.

[176] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

[177] K. Crawford and R. Calo, "There is a blind spot in ai research," *Nature News*, vol. 538, no. 7625, p. 311, 2016.

[178] T. Gebru, "Oxford handbook on ai ethics book chapter on race and gender," *arXiv preprint arXiv:1908.06165*, 2019.

[179] R. Van Noorden, "The ethical questions that haunt facial-recognition research," *Nature*, vol. 587, no. 7834, pp. 354–358, 2020.

[180] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of ai ethics guidelines," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 389–399, 2019.

[181] M. D. Dubber, F. Pasquale, and S. Das, *The Oxford Handbook of Ethics of AI*. Oxford University Press, USA, 2020.