MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

RestauAI – The Application of Facial Expression Recognition in Food Business

by

Dang H. Nguyen

A thesis submitted in conformity with the requirements for the degree of

Master of Software Engineering (MSE)

MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

RestauAI – The Application of Facial Expression Recognition in Food Business

by

Dang H. Nguyen

A thesis submitted in conformity with the requirements for the degree of

Master of Software Engineering (MSE)

Supervisor:

1. Dr. Minh, Nguyen Ngoc Truong

List of Abbreviations

2D ..... Two-dimensional

ANN ..... Artificial Neural Network

API ..... Application Programming Interface

AI ..... Artificial Intelligence

CNN ..... Convolutional Neural Network

CV ..... Computer vision

FACS ..... Facial Action Coding System

FER ..... Facial Expression Recognition

FnB ..... Food and Beverage Business

GPU ..... Graphic Processing Unit

HOG ..... Histogram of Oriented Gradients

POS ..... Point of Service

ReLU ..... Rectified Linear Unit

RNN ..... Recurrent Neural Network

ABSTRACT

RestauAI – The Application of Facial Expression Recognition in Food Business

Dang H. Nguyen

Degree Master of Software Engineering

FPT University

2023

Amidst, and even after, the global pandemic of Coronavirus disease, human has never cared more about sanitary and respiratory transmitted diseases. Thus, the popularity of automated and unmanned restaurants has increased. Due to the absence of staff, Computer Vision and Artificial Intelligence become the most promising methods to enhance the ordering and dining experience for customers. For this purpose, this thesis presents a system based on facial recognition and facial expression recognition using a pre-trained Convolutional Neural Network (CNN) model, a back-end to stream and analyze video data streamed from the camera (using said model).

For now, the system will have 2 main features: Detecting returned customers via facial recognition and Detecting customer's emotion expression to begin order process at Point of Service (POS)/Kiosk. There might be new features in the future using collected facial data, like payment, menu recommendation, small games etc...

This could be a game-changer for fast food chains or even small food store owners. It provides the customers a novel, exciting and state-of-the-art experience. Obviously, we are not aiming to remove 100 percent human factor from the process, on the contrary, a POS guider is always necessary to explain and encourage customer to interact with the system.

Keywords: automated restaurants; facial expression recognition; food and beverage business; convolutional neural network

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1 Context

In 2020 and 2021, the Corona Pandemic was at its peak, lockdowns and social distancing causing huge damage to all aspects of life, from society to economics, from culture to human, including Food and Beverage (FnB) businesses. Ho Chi Minh City has been the largest epicenter in Vietnam since the fourth virus wave began on April 27, with over 438,219 local infections and nearly 17,000 deaths - Tuoi Tre News, 2021. Store's closed, no staff, customer's behaviour shifted, avoid direct contact and respiratory transmitted disease (Tuoi Tre News, 2021), governments are forcing restaurant to ensure social distancing and proper sanitation per Sardar et al., 2022. That situation forced us to think of a new way to enhance customer's contact-less ordering and dining experience but also help business owners to analyze and service their customers in an automatic manner.

Computer vision (CV) offers truly impressive advances for the FnB industry, it is in the top ten industries using CV for quality control (Zou and Zhao, 2015). As computer-based visual inspection systems become more commercially adopted, we can see how image analysis technologies are changing the food production process here and now (Sun, 2012). CV deals with simple identification, measuring and counting better than human vision. However, to achieve this high level of performance and accuracy, the inspection conditions should be strictly standardized. For example, object identification becomes much more difficult in a blur background. Efficient edge detection and segmentation require a homogenous background, if the background is very fragmented, or the object is being covered partially by something, the detection may not perform well comparing to human effort. The same for light conditions: insufficient light, obscures shapes of inspected objects... can all cause false edge detection.

Luckily, CV is evolving rapidly, offering more accurate and reliable techniques for image enhancement, edge detection and segmentation. Also, advances in machine learning (CNN, support vector machines, etc...) allow creating the software able to perform classification of features and objects detected in an image or video more intelligently. With further refinement of image

analysis techniques, the food industry can expect that even more visual inspection tasks will be automated in the future.

Furthermore, **Artificial Intelligence** (AI) is making significant waves in the FnB industry (Fesabas et al., 2021). As a result of rising competition and demand in the Covid situation, the FnB industry has begun to adopt AI technology in an effort to maximize revenues and investigate new methods of reaching and serving consumers. With improved efficiency and cost savings more and more each day, AI has been effectively used for applications like supply chain management (Handley, 2023), sorting foods, crop monitoring, livestock monitoring, new product development, manufacturing and processing to packaging (Vincent, 2023), to even totally transform the way customer shops (Safaryan, 2023). However, to adopt AI in a FnB business, the cost, cultural shifts, the need for experts' skills, transparency issues, and a clear, well-defined tech product vision are some of the difficulties. Due to the high cost of AI research and adoption, only big players in the food industry can afford it.

The use of AI is still being researched despite these difficulties, but it's important to remember that the advantages of its application far exceed the drawbacks in the food business. Consumer food sales are seeing disruptions similar to those not witnessed since the last pandemic, which occurred more than 100 years ago. AI is being used more and more in all business sectors and is being merged with technology all around the world due to its supposedly endless possibilities (McKendrick, 2021).

Hence, our core idea is applying **Computer Vision** and **Artificial Intelligence** techniques to identify customer via facial recognition, thus automate the ordering process and count returning customers. Also recognize customer's facial expression to analyze customer's feedback, and give them a better dining experience.

1.2 Goals and Objectives

1.2.1 Goals

Within the context of this thesis, we will create a proof of concept system which is able to recognize a human face, then try to infer the object's current emotion state. In other words, we will create a proof of concept program that has **Facial Recognition** and **Facial Expression**

Recognition (FER) ability. Those are two major features that would greatly benefit the FnB industry in particular, the whole hospitality industry in general.

1.2.1.1 Facial Recognition

Facial recognition is a biometric method of identifying an individual by comparing a live image capture with a previously stored record for that person, also known as a faceprint. The faceprint can then be used to compare data captured from faces in an image or video.

The concept of contact-less ordering is not new in 2021; it has been used for years in many countries. KFC collaborated with Baidu to create a restaurant that uses Facial Recognition technology to infer customer moods and guess other information such as gender and age to inform their recommendations (Zacks, 2016). Customers can use the faceprint to make payments by **smiling** after placing their order at one of the fast food restaurant's self-service screens. For added security, an additional phone number verification option is available. This technology not only speeds up the payment process but also eliminates the need for physical contact, making it a convenient and safe option during the ongoing pandemic.

1.2.1.2 Facial Expression Recognition

After spoken language, facial expression is one of the most commonly used human modes of communication. It is caused by facial muscle movements. While different people have different types of facial expressions as a result of their own expressive styles or personalities, many studies show that there are several types of basic expressions shared by people from various cultural and ethnic backgrounds (Fasel and Luettin, 2003).

Specifically within the FnB industry and hospitality industry as a whole, the FER feature proves to be highly relevant and useful (Fesabas et al., 2021). By accurately recognizing facial expressions of individuals, the model can assist in several key areas within the industry. For instance, it can be employed in customer feedback analysis, allowing businesses to analyze customers' expressions of satisfaction, delight, or surprise while experiencing different FnB offerings. This valuable insight can inform decision-making processes, product development, and marketing strategies to enhance customer satisfaction and drive business growth.

Moreover, the ability to accurately recognize facial expressions can be leveraged for market

research within the FnB industry. By analyzing expressions of individuals during taste tests or product trials, companies can gain deeper insights into consumer preferences, likes, dislikes, and emotional responses. This information can guide product innovation, refine marketing campaigns, and improve overall customer experiences. (de Wijk et al., 2021)

During the last few decades, there has been a lot of interest in research on automatic recognition of such basic facial expressions. Many studies show that subtracting neutral faces from their corresponding expressive faces can help the algorithms to emphasize on the facial moving areas, and significantly improve the expression recognition rate. Traditional approaches tend to conduct facial expression recognition by using Gabor Wavelets (Bazzo and Lamar, 2004), sparse coding, and so on. In recent years, Convolutional Neural Networks (CNNs) have been widely used in FER. CNNs can achieve good performance by learning powerful high-level features that outperform hand-crafted features. While research on FER has expanded over the past decades, it remains a challenge due to the many varied facial expressions in various situations. Thus, new algorithms are always needed for FER.

The ability to extract facial expression could benefit human-machine interactions greatly, bring it to a new height. Because of its significance, several large databases of images of faces have been created and emotionally annotated (labeled), for example EmotioNet (Benitez-Quiroz et al., 2016), AffecNet (Mollahosseini et al., 2017), SFEW (Dhall et al., 2011), FER-13 (Goodfellow et al., 2013) and Google facial expression comparison dataset (Agarwala and Vemulapalli, 2018)... Notably, quite different from other datasets that focus mainly on discrete emotion classification or action unit detection, Google facial expression comparison dataset is intended to assist researchers working on facial expression analysis topics such as expression-based image retrieval, expression-based photo album summarization, emotion classification, expression synthesis, and so on.

In the following chapters, we aim to create a program that can detect human face and an AI model that can identify facial expressions, of detected face, using a dataset. We will look at various machine learning algorithms and techniques for training the AI model and incrementally improve its accuracy.

1.2.2 Objectives

This thesis aims to accomplish several objectives:

1. Face Detection and Segmentation: The first objective is to detect human faces in an image and segment them for further processing. This step lays the foundation for subsequent facial expression recognition.

2. CNN Model Design for FER:

    - Dataset Selection and Split: The thesis will select a suitable dataset and split it into training and testing sets to train and evaluate the CNN model.

    - CNN Model Design: A CNN architecture will be designed to accurately detect facial expressions.

    - Model Testing: The trained model will be evaluated using the testing set to measure its accuracy in facial expression recognition.

    - Model Fine-Tuning: To further improve accuracy, the model will undergo fine-tuning, adjusting hyperparameters and optimizing its performance.

    - API Development: An API will be built to serve the best-performing facial expression recognition model. This API will provide a convenient and accessible interface for utilizing the model's capabilities.

3. Proof-of-Concept Program: A proof-of-concept program will be developed to demonstrate the practical application of the FER system. This program will read images from videos or cameras, process them, segment the faces, and call the API to detect and classify facial expressions.

4. Real Data Testing: The developed system will be tested with real-world data, including images and videos containing diverse facial expressions. This step ensures the system's effectiveness and performance in real-world scenarios.

5. Reporting and Conclusion: The report will present the findings, results, and insights gained from the experiments and evaluations. A comprehensive report will summarize the achievements, challenges, and potential areas for further improvement. The conclusion will provide an overall assessment of the developed facial expression recognition system.

CHAPTER 2. LITERATURE REVIEW

2.1 Basic Knowledge

Before moving on further into this thesis, one should know the basic about Deep Leaning, some of its technique and how it is being applied in FER.

2.1.1 Deep Leaning

Deep Learning is a subfield of Machine Learning that focuses on training artificial neural networks (ANNs) to learn and make predictions from large amounts of data. It is inspired by the structure and function of the human brain and is known for its ability to automatically learn hierarchical representations of data.

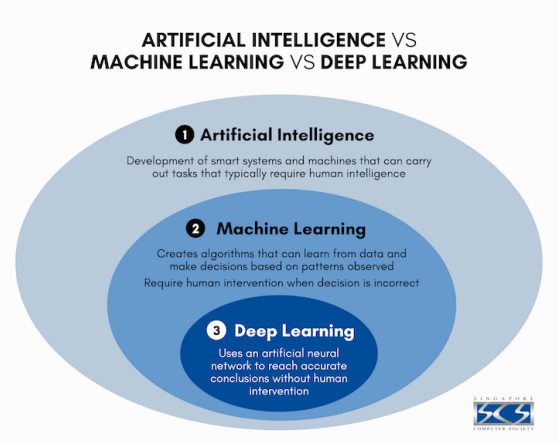


Figure 2.1: Overview of AI, Machine and Deep Learning - Singapore Computer Society, 2020

Here are some key points about deep learning:

- Neural Networks: Deep learning models are built using ANNs, which consist of interconnected nodes or neurons. These neurons are organized into layers, with each layer responsible for extracting and transforming specific features from the input data.
- Deep Neural Networks: Deep learning models are characterized by their depth, meaning they have multiple hidden layers between the input and output layers. These deep architectures enable the models to learn complex representations and capture intricate patterns in the data.
- Training: Deep learning models are trained using a large labeled dataset. During the training process, the model adjusts the weights and biases of its neurons to minimize the difference

between its predictions and the true labels. This is typically done using an optimization algorithm, such as stochastic gradient descent, to iteratively update the model's parameters.

- Activation Functions: Activation functions introduce non-linearities into the neural network, enabling it to learn complex relationships in the data. Common activation functions include the sigmoid and rectified linear unit (ReLU) functions.

- CNN: CNNs are a specialized type of deep neural network commonly used for image and video analysis. They utilize convolutional layers to automatically extract features from input data and pooling layers to reduce spatial dimensions.

- Recurrent Neural Networks (RNNs): RNNs are another type of deep neural network that can model sequential data. They have connections between nodes that form directed cycles, allowing them to capture temporal dependencies in the data.

- Unsupervised Learning: Deep learning can also be applied to unsupervised learning tasks, where the model learns patterns and structures in the data without explicit labels. Autoencoders and generative adversarial networks (GANs) are examples of unsupervised deep learning techniques.

Deep learning has achieved remarkable success in various fields (Dargan et al., 2019), including computer vision, natural language processing, speech recognition, recommendation systems, and medical diagnostics. It has enabled advancements in areas such as autonomous driving, image and speech recognition, and language translation. These are just the fundamentals of deep learning. As a rapidly evolving field, there are ongoing research and advancements happening in various aspects of deep learning, leading to new techniques and improved performance on challenging tasks.

2.1.2 Deep Learning in Facial Expression Recognition

By leveraging deep neural networks, researchers have achieved significant advancements in accurately detecting and classifying facial expressions. Here are some key applications of deep learning in FER:

- CNN for FER: CNNs have been widely utilized in FER tasks due to their ability to automat-

ically learn relevant features from facial images. CNN architectures, such as VGGNet (Jun et al., 2018), ResNet (Zhong et al., 2020), have been adapted and trained to recognize facial expressions with high accuracy.

- Facial Landmark Detection: Deep learning models have been employed for accurate facial landmark detection, which involves identifying key points on the face such as the eyes, nose, and mouth (Fan and Zhou, 2016). Precise landmark detection helps in aligning and normalizing facial images, enhancing the performance of subsequent FER models.

- Facial Expression Generation: GANs have been utilized to generate realistic facial expressions (Xia et al., 2022). These models can synthesize facial images depicting specific emotions, allowing for data augmentation and the creation of diverse training datasets for FER.

- Multi-Modal Fusion: Deep learning techniques have been applied to fuse information from multiple modalities, such as facial images, audio, and textual data, to improve the accuracy of FER models (Abdullah et al., 2021). Multi-modal fusion enables a more comprehensive analysis of facial expressions by incorporating different sources of information.

- Transfer Learning: Transfer learning has been leveraged in FER to utilize pre-trained deep learning models (Purkayastha and Malathi, 2022), such as CNNs trained on large-scale datasets like ImageNet. By fine-tuning these models on smaller FER datasets, researchers have achieved impressive results, even with limited labeled data.

And many more. The application of deep learning in FER has significantly improved the accuracy and robustness of emotion recognition systems, opening up opportunities for practical applications in various domains, including human-computer interaction and psychological research.

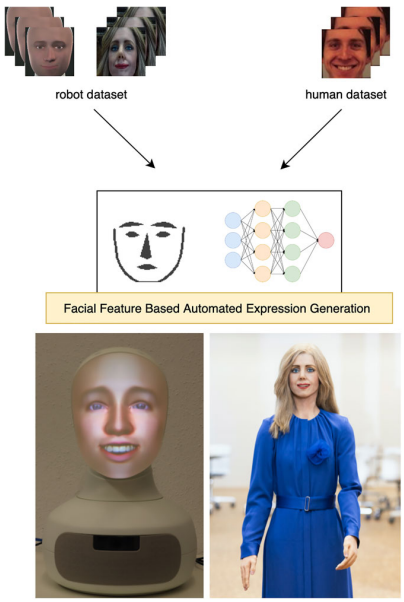


Figure 2.2: Facial Expression Generation (Rawal et al., 2022)

2.2 Previous Researches

Facial recognition and FER technology has become an increasingly important tool in many fields, not just in FnB sector. The ability to accurately and efficiently identify individuals and their emotional state based on their facial features has led to a proliferation of research into the development of sophisticated algorithms and methods for them. In this literature review, we examine three recent articles that provide comprehensive analyses of various facial recognition and FER techniques and their performance.

2.2.1 "A comparative study of facial recognition techniques"

In the first article, **"A comparative study of facial recognition techniques"** - Schenkel et al., 2019 the authors present a comprehensive study of different face recognition methods and compare their performance on a dataset of face images.

Facial recognition is a challenging task due to the variability of human faces in terms of pose, expression, and illumination. The authors first provide an overview of the face recognition pipeline, which consists of face detection, feature extraction, and matching. They then describe the different face recognition methods, including Eigenfaces, Fisherfaces and CNNs which will be implemeted using OpenCV, scikit-learn and FaceNet. For each method, the authors explain the underlying principles and implementation details.

The authors then evaluate the performance of each method on the Labeled Faces in the Wild (LFW) dataset using different evaluation metrics, including accuracy, fallout, recall, precision and F-score. They also analyze the impact of various parameters, such as the number of components in eigenfaces and Fisherfaces. As for CNN method implemented in FaceNet, there are no presented settings because there were no configurable parameters.

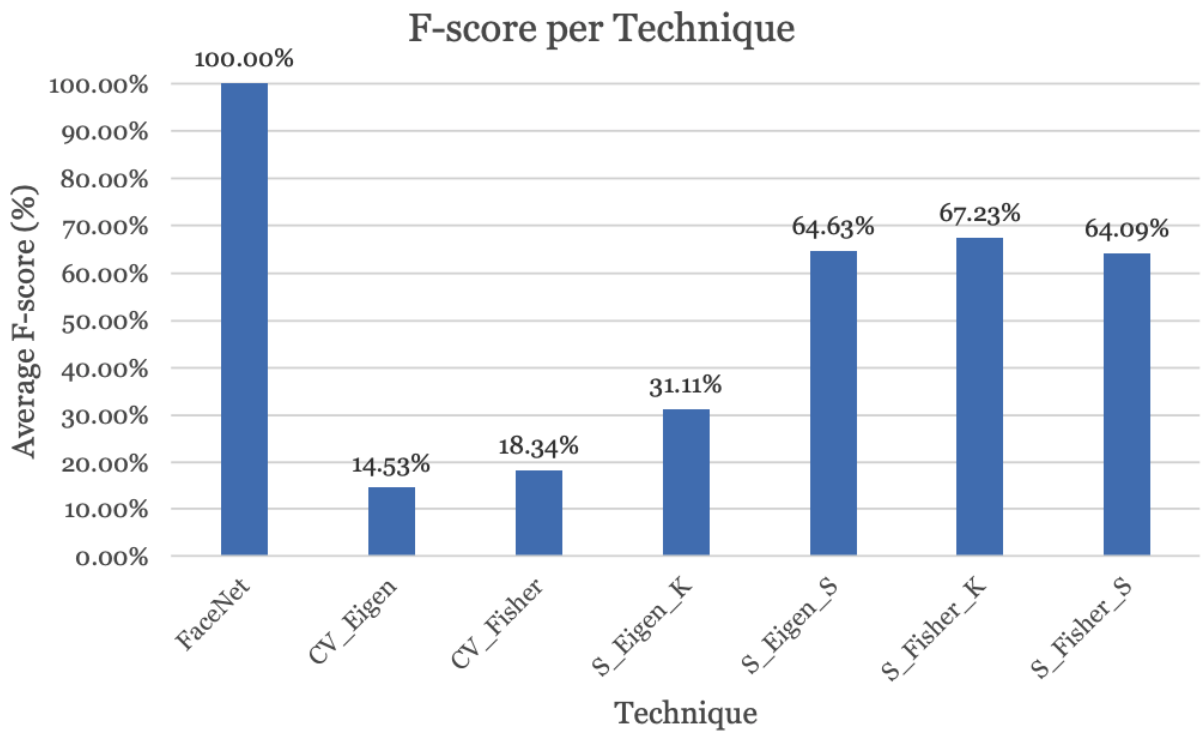


Figure 2.3: Graph over the average F-score (Schenkel et al., 2019)

The graph in 2.3 shows the average F-score of the different techniques. F-score is arguably the best metric to use to see which technique performs the best, both in doing correct predictions and not doing incorrect predictions. The experimental results show that CNNs (FaceNet) outperform the other methods in terms of accuracy and robustness to variations in pose, expression, and illumination. However, the authors note that CNNs require a large amount of training data and computational resources, making them less practical for some applications. Eigenfaces and Fisherfaces, on the other hand, are fast and require less training data but are less accurate and sensitive to variations in lighting.

The authors conclude that there is no one-size-fits-all solution for facial recognition and that the choice of method depends on the specific application requirements and constraints. They also highlight the importance of data preprocessing and quality control in achieving accurate and reliable results.

Overall, this paper provides a comprehensive study of different face recognition methods and their performance comparison, highlighting the strengths and weaknesses of each method and their suitability for different applications. The key take away here for this research is we should

apply CNN facial recognition method for much better accuracy in uneven lighting, real-world conditions. The two other options are best for lab condition only.

2.2.2 "Facial emotion recognition using deep learning: review and insights"

The next step after we had the faceprint (facial) recognized is to infer emotion from it. In order to have a better, more general view of the subject of FER, the second article we would like to explore is: **"Facial emotion recognition using deep learning: review and insights"** - Mellouka and Handouzi, 2020

FER is an important task in computer vision and human-computer interaction, with applications in areas such as affective computing, gaming, and virtual reality. In this paper, the authors provide a comprehensive review and analysis of state-of-the-art FER deep learning architectures. Beside, one of the reason why we pick this paper to review is it's quite recent to-date within our context.

The authors then first list out several publicly available datasets, which we summarized into Table 2.1, each one different from the others in term of the number and size of images and videos, variations of the illumination, population and facial expressions. From the list the author provides, we noticed AffecNet (Mollahosseini et al., 2017) and FER-13 (Goodfellow et al., 2013) for their large number of images and emotions they convey. These datasets can be used to train and evaluate FER models. It is important to have diverse datasets to ensure that the models are robust and generalizable to different populations and environments.

The author continues with a review of recent developments in FER using various deep learning architectures, including results from 2016 to 2019 and an explanation of the issues. Extracting features from one face to another is a difficult and sensitive task in order to have a good classification system. In recent years, deep learning has been very successful and efficient approach thanks to the result obtained with its architectures which allow the **automatic** extraction of features and classification such as the CNN and the recurrent neural network (RNN). This is what led researchers to begin applying this method widely, researchers put a lot of effort into creating deep neural architectures, and the results of them are also very satisfactory in this area.

The article Agrawal and Mittal, 2019, which examined the impact of changing the CNN

parameters on the recognition rate using the FER2013 database, caught our attention the most. The images vary in size and the number of filters they use, and they are all defined at 64x64 pixels. On a basic CNN with two successive convolution layers, the second layer plays the role of the max pooling, followed by a softmax function for classification, the type of optimizer (Adam, SGD,...) was also chosen. These studies claim that researchers developed two novel CNN models that, on average, achieve **65.23%** and **65.77%** accuracy. These models are unique in that they do not contain fully connected layer dropout and maintain the same filter size throughout the network. We find this is an interesting architecture and can be developed further. Other architectures can also see in Figure 2.4 below.

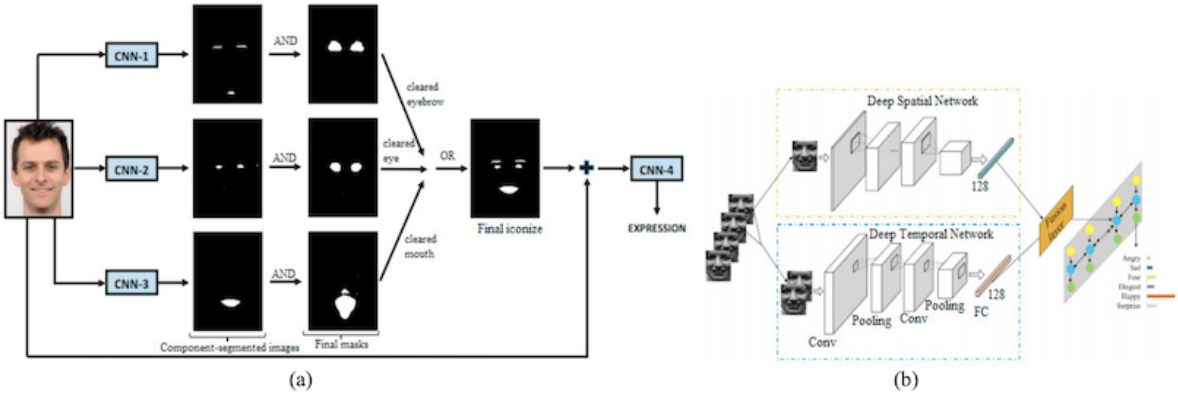


Figure 2.4: Deep learning methods proposed by Gozde et al., 2019 and Liang et al., 2020

Preprocessing is a crucial step that was included in all the papers cited in this review. It involves a number of techniques, including cropping and resizing images to shorten training times, normalizing spatial and intensity pixels, and data augmentation to increase image diversity and solve the over-fitting issue. Lopes et al., 2017 do a good job of presenting all of these methods.

The authors then list the performance of different FER deep learning methods on several publicly available datasets using their claimed recognition rate. They also analyze the impact of various factors, such as dataset size, number of expressions, and inter-subject variability.

The authors conclude that FER is a challenging task due to the high variability of facial expressions (facial emotion classes) and the limited availability of annotated datasets. They also highlight the importance of data preprocessing, feature selection, and ensemble methods in achieving accurate and reliable results. Overall, this paper provides a comprehensive survey and analysis of state-of-the-art FER deep learning techniques, highlighting the strengths and weaknesses of

different methods and their suitability for different applications.

2.2.3 *"Convolutional Neural Networks for Facial Expression Recognition"*

Let's now look more closely at how we can create a CNN Model for FER. The study **"Convolutional Neural Networks for Facial Expression Recognition"** by Alizadeh and Fazel, 2017, is the one we choose to review, in hope to take a look on the answer of that question. The paper proposes a CNN Model for a FER task, the goal is to classify each facial image into one of the seven facial emotion categories: Anger, Happiness, Fear, Sadness, Disgust and Neutral

The authors first introduce some researches in FER field, including but not limited to Facial Action Coding System (FACS) (Ekman and Friesen, 1978), Bayesian Networks, Neural Networks and the multilevel Hidden Markov Model. They also note that some of them have issues with timing or recognition rate. Typically, two or more techniques can be combined to achieve accurate recognition; then, features are extracted as needed. The success of each technique is dependent on pre-processing of the images, dataset because of illumination and feature extraction.

The authors proposed and developed multiple CNNs with variable depth to evaluate the performance of them against FER task. They design the following network architecture for their experiment 2.5:

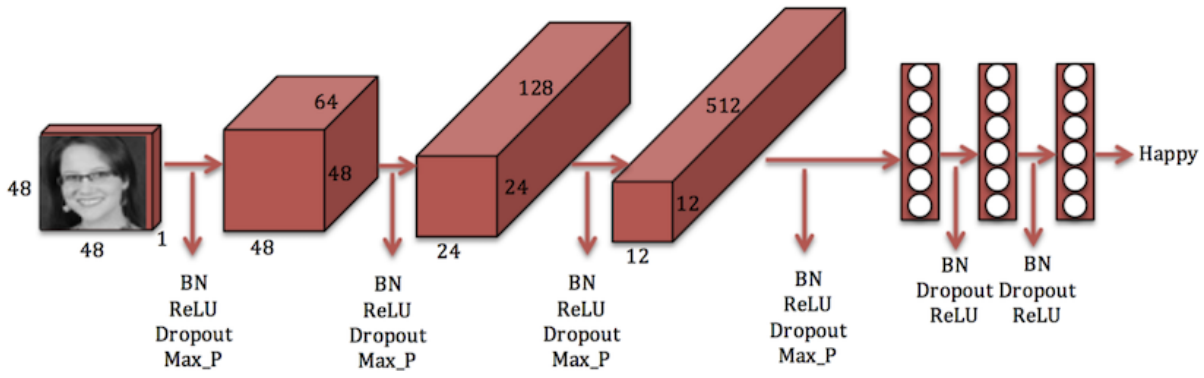


Figure 2.5: The architecture of the deep network (Alizadeh and Fazel, 2017)

The first part of the network refers to M convolutional layers that can have spatial batch normalization, dropout, and max-pooling. Each of these layer will have a convolution layer and

ReLU nonlinearity. The network is then led to N fully connected layers, which can include batch normalization, dropout, and always have Affine operation and ReLU nonlinearity. Finally, the network is followed by the affine layer that computes the scores and softmax loss function. The created model gives the user the freedom to choose the number of fully connected layers and convolutional layers, as well as whether batch normalization, dropout, and max-pooling layers are applied. Additionally, the user can specify the quantity of filters, strides, and zero-padding; otherwise, the default values are used. The authors also experiment with HOG feature by attaching it in the last convolution layer. They implemented the architecture in 2.5 in Torch to took advantage of GPU accelerated deep learning features.

About the dataset, the authors also use the FER2013 dataset just like us will. It consists of around 35000 48 × 48 pixel gray-scale images of faces. The images are processed in such a way that the faces are almost centered and each face occupies about the same amount of space in each image. Each image has to be categorized into one of the seven classes: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Neutral, 5=Sad, and 6=Surprise.

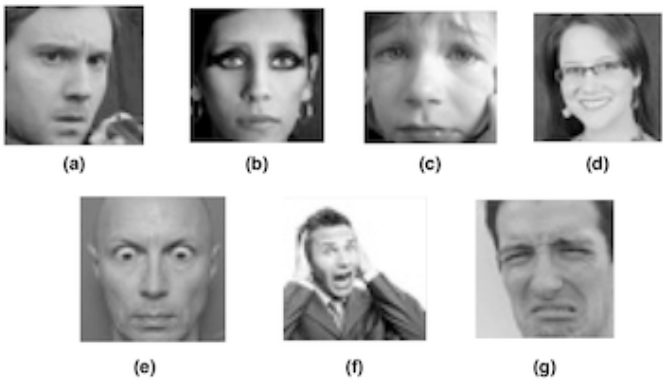


Figure 2.6: Examples of seven facial emotion classes: (a) angry, (b) neutral, (c) sad, (d) happy, (e) surprise, (f) fear, (g) disgust (Alizadeh and Fazel, 2017)

After reading the raw pixel data, they normalized it by subtracting the mean of the training images from each image including those in the validation and test sets. They produced mirrored images by flipping images in the training set horizontally for image augmentation.

For the purpose of experimenting, the authors first build a shallow model. It has 2 convolutional layers and 1 fully connected layer:

- First convolutional layer has 32 3x3 filters, stride size 1, batch normalization and dropout. Without max-pooling
- Second convolutional layer has 64 3x3 filters, stride size 1, batch normalization, dropout and max-pooling with a filter size 2x2
- Fully connected layer has a hidden layer with 512 neurons and Softmax for the loss function

In all layers, authors also use ReLU as the activation function. They took advantage of Torch's GPU-accelerated deep learning capabilities to speed up the model training process. They used the validation set to verify our model in each iteration and the test set to measure the model's accuracy. They achieved accuracy of **55%** on the validation set and **54%** on the test set using the best shallow model.

Next, the authors decided to train a deeper CNN with four convolutional layers and two FC layers to observe the impact of including more convolutional layers and FC layers to the network. The first four convolutional layers have a stride of size 1, batch normalization, dropout, max-pooling and ReLU as the activation function, however

- First convolutional layer has 64 3×3 filters
- Second convolutional layer has 128 5×5 filters
- Third convolutional layer has 512 3×3 filters
- Fourth convolutional layer has 512 3×3 filters

About two Fully Connected layers, both has batch normalization, dropoutm ReLU and Softmax as loss function:

- The hidden layer in the first FC layers had 256 neurons
- The second FC layer had 512 neurons

After setting the hyper-parameters, using same method of validation, the best deep model with four convolutional layers and two fully connected layers achived accuracy of **65%** on validation set and **64%** on test set.

The authors also trained with more, 5 and 6, convolutional layers to investigate the deeper networks, but these networks **did not** improve classification accuracy. The model with four convolutional layers and two Fully Connected layers was therefore deemed to be the **best network** for the dataset.

To explore if there is any way to apply HOG features along with raw pixels to our network and observe the performance of the model when it has a combination of two different features, the authors built another model containing two neural networks:

- First one contains convolutional layers
- Second one has only fully connected layers
- Features that are detected by the first network are combined with the HOG features and the resulted features will be fed to the second network

They trained two networks, one shallow, one deep with the same configuration as the previous experiment. Now, the resulted accuracy of the shallow model was very close to the accuracy we got from the shallow model that did not use HOG. The accuracy of the deep HOG model was also similar to the accuracy they got from the without HOG one. This means that CNN is **strong enough** to extract sufficient information including those coming from HOG features by using only raw pixel data.

Overall, this paper enlightened us about how to build a CNN model for a FER problem. They also experimented and proved that CNN deep models can be quite effectively to learn facial characteristics and facial emotion detect. Also the authors tested out and concluded that the hybrid feature set with HOG did not help with the accuracy of the CNN model.

Databases	Descriptions	Emotions
MultiPie (Gross et al., 2009)	More than 750,000 images captured by 15 view and 19 illumination conditions	Anger, Disgust, Neutral, Happy, Squint, Scream, Surprise
MMI (Pantic et al., 2005)	2900 videos, indicate the neutral, onset, apex and offset	Six basic emotions and neutral
GEMEP FERA (Valstar et al., 2011)	289 images sequences	Anger, Fear, Sadness, Relief, Happy
SFEW (Dhall et al., 2011)	700 images with different ages, occlusion, illumination and head pose	Six basic emotions and neutral
CK+ (Lucey et al., 2010)	593 videos for posed and non-posed expressions	Six basic emotions, contempt and neutral
FER2013 (Goodfellow et al., 2013)	35,887 grayscale images collect from google image search	Six basic emotions and neutral
JAFFE (Lyons et al., 1998)	213 grayscale images posed by 10 Japanese females	Six basic emotions and neutral
BU-3DFE (Yin et al., 2006)	2500 3D facial images captured on two view -45°, +45°	Six basic emotions and neutral
CASME II (et al, 2014)	247 micro-expressions sequences	Happy, Disgust, Surprise, Regression and others
Oulu-CASIA (Zhao et al., 2011)	2880 videos captured in three different illumination conditions	Six basic emotions
AffectNet (Mollahosseini et al., 2017)	More than 440.000 images collected from the internet	Six basic emotions and neutral
RAFD-DB (Li et al., 2017)	30000 images from real world	Six basic emotions and neutral
RaFD (Langner et al., 2010)	8040 images with different face poses, age, gender, sexes	Six basic emotions, contempt and neutral

Table 2.1: A summary of some FER databases. (Mellouka and Handouzi, 2020)

CHAPTER 3. METHODOLOGY

3.1 Facial Detection

3.1.1 Methodology

OpenCV (Bradski, 2000) is an open-source CV and machine learning library that has various algorithms for image processing, including face detection. Haar Cascade is one such algorithm, which can detect faces in an image or a video stream. In this section, we will discuss the working principle of Haar Cascade and how it detects facial recognition.

Haar Cascade is based on the Haar Wavelet Technique (Haar, 1910), which is a mathematical tool used for detecting edges in an image. It is a machine learning-based algorithm that uses a supervised learning approach. This means that the algorithm requires a set of positive and negative samples to learn from, which are provided during the training phase. The positive samples are images that contain the object we want to detect, in this case, faces. The negative samples are images that do not contain the object we want to detect.

During the training phase, Haar Cascade creates a classifier by analyzing the features of both positive and negative samples. The classifier is a machine learning model that can identify faces based on their features. The features used by the classifier are called Haar-like features. These features are rectangular patterns of pixels that differ in brightness levels.

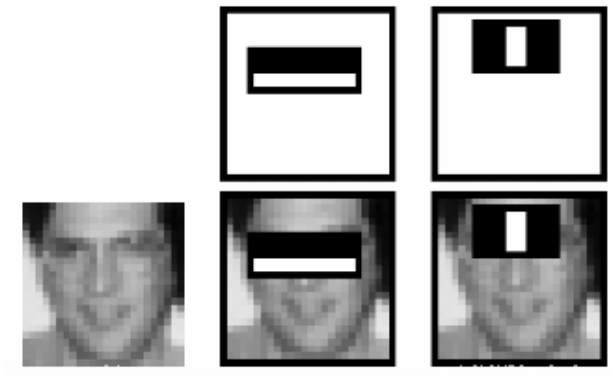


Figure 3.1: Haar-like features are being calculated

Haar-like features are calculated by moving a rectangular window over the image and comparing the sum of pixel intensities inside the rectangular area with the sum of pixel intensities outside the rectangular area. If the difference between these two sums is greater than a certain threshold, the feature is considered to be present. There are many types of Haar-like features, and they vary in shape, size, and position.

Once the classifier is created, it can be used to detect faces in an image or a video stream. The detection process involves sliding the classifier over the image in different scales and positions. At each position, the classifier applies the Haar-like features to the image and calculates a score. If the score exceeds a certain threshold, the classifier considers that the region contains a face.

However, the detection process is computationally expensive since the classifier has to slide over the entire image at multiple scales and positions. To optimize the detection process, Haar Cascade uses the Integral Image Technique. This technique involves creating a sum of the pixel intensities in a rectangular area of the image, which allows for the quick calculation of Haar-like features. The Integral Image Technique reduces the computation time significantly and speeds up the detection process.

In short, Haar Cascade is an efficient algorithm for facial recognition that uses a supervised learning approach. The algorithm learns from a set of positive and negative samples to create a classifier that can identify faces based on their features. The detection process involves sliding the classifier over the image in different scales and positions, and the Integral Image Technique is used to optimize the computation time. Haar Cascade is widely used in various applications, including security, surveillance, and human-computer interaction.

3.1.2 Experiment

As mentioned in Chapter 3.1, our first task is to separate the faceprint out from the context it is in. To achieve this goal, we decided to use Haar Cascade in OpenCV (Bradski, 2000), the programming language we used is Python. Below is our implementation:

```
import cv2

face_detector1 = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_dectector1 = cv2.CascadeClassifier('haarcascade_eye.xml')

# Segment the faceprint out of video stream
cap = cv2.VideoCapture(0)
while cap.isOpened():
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    faces = face_detector1.detectMultiScale(gray, scaleFactor=1.1,
    minNeighbors=6)
    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        eyes = eye_dectector1.detectMultiScale(roi_gray, scaleFactor=1.1,
    minNeighbors=6)
        # Only accept faceprint has at least 2 eyes
        if (len(eyes) > 1):
            # Found faceprint!
            cv2.rectangle(frame, pt1=(x, y), pt2=(x+w, y+h),
                          color=(255, 0, 0), thickness=3)

        cv2.imshow("window", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
frame.release()
```

Listing 3.1: Faceprint detection in OpenCV.

We implemented it with the input is the laptop camera video stream directly. After captured the image from the video, we convert it to gray scale, and then use ***haarcascade_frontalface_default.xml*** and ***haarcascade_eye.xml*** to detect and make sure it is an human faceprint. Through experimentation, we found that we could still detect faces while removing the false-positive cases by updating the ***minNeighbors*** to 6, and ***scaleFactor*** to 1.1. Notice we added ***eye_detector1*** here, not just face detector classifier only, as a touch to reduce the rate of false-positive cases. This method of combining 2 classifiers at the same time has shown significant improvement in that aspect.

3.2 Facial Expression Recognition

3.2.1 Methodology

After we had the faceprint recognized, cropped out and centered by using OpenCV as mentioned in 3.1, we would need to make an AI model to infer emotions within 7 categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral from it so that FnB businesses can utilize the result.

The framework applied in this model is Keras. Keras is an open-source neural network

library written in Python that is widely used for building deep learning models. It is especially useful for building FER models due to its ease of use, flexibility, and scalability.

First of all, before we dive in and build the model, we should do some Image augmentation. It is an essential step in training an AI model, especially for CV tasks such as image classification, object detection, and segmentation. It involves applying various transformations to the input images, such as cropping, rotating, flipping, zooming, and adjusting brightness and contrast. The aim of image augmentation is to increase the variability and diversity of the training data, which helps the model to learn more robust and generalizable features, and reduces overfitting.

In this section, we will discuss the importance of image augmentation, the different techniques used, and how to apply them in practice.

Why is image augmentation important? In CV tasks, the performance of a model heavily depends on the quality and quantity of training data. However, collecting and labeling large datasets can be expensive and time-consuming, especially for tasks that require expert knowledge or involve sensitive data. Moreover, even large datasets may not capture all the possible variations and scenarios that the model may encounter in real-world applications. Therefore, it is crucial to augment the training data to increase its diversity and variability, and make the model more robust and reliable.

Another benefit of image augmentation is that it helps to reduce **overfitting**, which occurs when a model memorizes the training data and fails to generalize to new examples. By applying random transformations to the training images, image augmentation introduces noise and variability into the data, which makes it harder for the model to memorize and easier to generalize. As a result, the model can achieve better performance on the test set, which consists of unseen examples.

There are many techniques for image augmentation, each with its advantages and limitations. Here are some of the most commonly used techniques:

1. Random Cropping: This involves selecting a random subregion of the input image and resizing it to the desired size. Random cropping helps the model to learn to recognize objects at different scales and positions.

2. Rotation: This involves rotating the input image by a random angle within a certain range. Rotation helps the model to learn to recognize objects from different viewpoints and orientations.

3. Flipping: This involves horizontally or vertically flipping the input image with a certain probability. Flipping helps the model to learn to recognize objects from different directions and angles.

4. Zooming: This involves zooming in or out of the input image by a certain factor. Zooming helps the model to learn to recognize objects at different distances and sizes.

5. Brightness and Contrast Adjustment: This involves adjusting the brightness and contrast of the input image by a random factor. Brightness and contrast adjustment helps the model to learn to recognize objects under different lighting conditions.

To apply image augmentation in practice, we can use Keras' built-in functions for applying different image augmentation techniques and allow us to specify the parameters and settings of each technique. In order to perform image augmentation while training the model, we utilized the Keras class *ImageDataGenerator* (Chollet, n.d.) with just some simple image augmentations:

- Convert to grayscale
- Crop to 48x48 pixel
- Batch size is 128
- Set *shuffle=True* for train set, *shuffle=False* for test set to save transformation effort on test set

With Keras, it would be just as simple as 3.2

```
batch_size  = 128
picture_size = 48
datagen_train  = ImageDataGenerator()
datagen_val = ImageDataGenerator()

train_set = datagen_train.flow_from_directory(
    folder_path+"train",
```

```
    target_size = (picture_size,picture_size),
    color_mode = "grayscale",
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

test_set = datagen_val.flow_from_directory(
    folder_path+"validation",
    target_size = (picture_size,picture_size),
    color_mode = "grayscale",
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Listing 3.2: Image augmentation in Keras sample code in Python.

Next, we go ahead and design the CNN Model architecture. CNN is a good choice because they were created to work with 2D data, which is all we have to work with in our problem. Considering that they can automatically learn and extract pertinent features from the input images, CNNs are particularly helpful for image classification tasks. They can also accommodate changes in the input data, such as rotations or various lighting conditions.

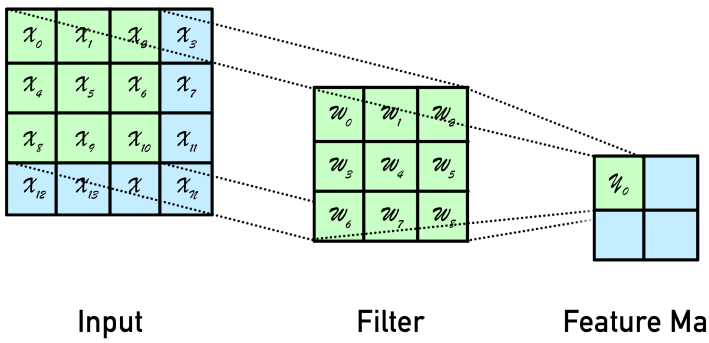


Figure 3.2: Calculate the feature map in a 2D convo neural network

Basically how CNN works is it applies filters to an input to create a feature map that summarizes the presence of features in the input. Filters can be customized like edge, lines, circle

detectors... but the innovation of CNN is it can learn the filters during training, within the context of the prediction problem.

Even further, multiple convolution layers can be stacked onto the output of each other, the stacking of convolutional layers allows a hierarchical decomposition of the input. Consider that the filters that operate directly on the raw pixel values will learn to extract low-level features, such as "lines". The filters that operate on the output of the first "line" layers may extract features that are combinations of lower-level features, such as features that comprise "multiple lines" to express "shapes". This process continues until very deep layers are extracting faces, animals, houses, and even what we want, facial expressions.

CNNs can learn multiple features in parallel for a given input. For example, it is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input. This gives the model different ways of extracting features from an input. This diversity allows **specialization**, e.g. not just lines, but the specific lines seen in your specific training data, that could possibly describe a human facial expression.

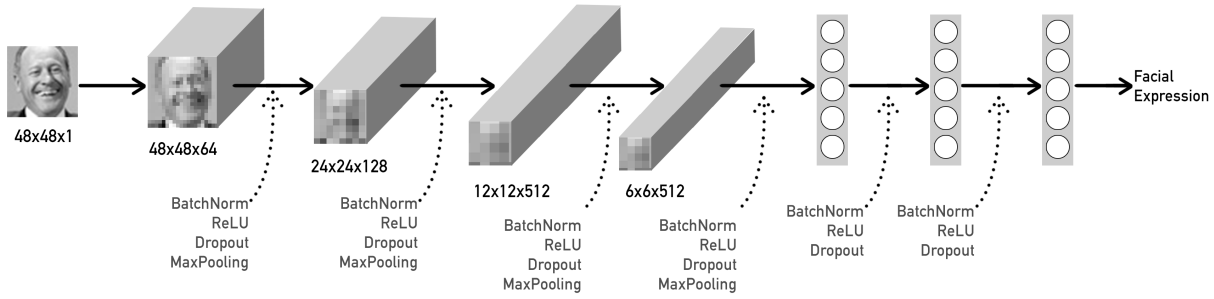


Figure 3.3: The CNN Model Architecture

As we can see from Figure 3.3, the first part of the network refers to 4 convolutional layers that have batch normalization (Brownlee, 2019a) to help speed up, dropout (Brownlee, 2018) to prevent over-fitting, and max-pooling (Brownlee, 2019b) in addition to the convolution layer and ReLU (Brownlee, 2020). After 4 convolution layers, the network is led to 2 fully connected (dense) layers (Unzueta, 2022) that always have ReLU, batch normalization and dropout. Finally, the network is followed by the layer that computes the scores and softmax (SuperDataScience

Team, 2018) loss function. Along with dropout and batch normalization techniques, we included Adam optimizer (Kingma and Ba, 2014) in our implementation. This deep model design is inspired by the study "Convolutional Neural Networks for Facial Expression Recognition" (Alizadeh and Fazel, 2017).

Batch normalization (Brownlee, 2019a) is a technique used in deep learning models to normalize the inputs of each layer by centering and scaling the values. It is commonly used in CNNs and RNNs to improve the training and generalization performance of the models. In Keras, batch normalization can be easily applied by adding a ***BatchNormalization*** layer after the activation layer of a model.

```
from keras.layers.normalization import BatchNormalization

# instantiate model
model = Sequential()

# we can think of this chunk as the input layer
model.add(Dense(64, input_dim=14, init='uniform'))
model.add(BatchNormalization())
```

Listing 3.3: Batch Normalization in Keras sample code in Python.

The ***BatchNormalization*** layer has two main parameters that need to be set during the training phase: the momentum and the epsilon. The momentum controls the moving average of the mean and variance of each feature dimension across batches, while the epsilon is used to avoid division by zero.

Dropout (Brownlee, 2018) is a regularization technique used in deep learning models to prevent overfitting. It works by randomly dropping out some neurons in a layer during training, which forces the network to learn more robust and generalizable features.

Max-pooling (Brownlee, 2019b) is a commonly used operation in CNNs for downsampling and reducing the dimensions of feature maps. In Keras, max-pooling can be easily applied using the *MaxPooling2D* layer.

The ***MaxPooling2D*** layer takes as input a 4D tensor of shape (***batch_size, height, width, channels***) and applies max-pooling to the height and width dimensions. The layer has two param-

eters: ***pool_size*** and ***strides***. The ***pool_size*** parameter determines the size of the pooling window (usually a 2x2 or 3x3 window), while the ***strides*** parameter determines the stride of the pooling window (usually equal to the ***pool_size*** or half the ***pool_size***).

Here's an example of how we add a max-pooling layer in Keras 3.4, in this example, we add a ***Conv2D*** layer with 32 filters and a 3x3 kernel size. We then add a ***MaxPooling2D*** layer with a pool size of 2x2 and a stride of 2x2 to downsample the feature maps.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential()

# Add a convolutional layer
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
    input_shape=(28, 28, 1)))

# Add a max-pooling layer
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
```

Listing 3.4: MaxPooling2D in Keras sample code in Python.

Rectified Linear Unit (Brownlee, 2020) is a commonly used activation function in neural networks, including those implemented using Keras. The ReLU function is defined as simple as follows:

```
f(x) = max(0, x)
```

Listing 3.5: ReLU function.

In other words, the output of the function is equal to the input **x** if **x** is greater than or equal to 0, and 0 otherwise.

ReLU activation function is popular in deep learning because it is computationally efficient, easy to implement, and does not suffer from the vanishing gradient problem. The vanishing gradient problem occurs when the derivative of the activation function approaches zero, making it difficult to update the weights of the neural network during training. ReLU, on the other hand, has a derivative of 1 for positive inputs and 0 for negative inputs, making it simple to compute and

avoiding the vanishing gradient problem for positive inputs.

In Keras, ReLU activation can be applied to any layer using the **relu** argument in the ***Activation*** layer. Here is an example of how to apply ReLU to a dense layer in Keras:

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()

# Add a dense layer with ReLU activation
model.add(Dense(units=64, input_dim=100))
model.add(Activation('relu'))
```

Listing 3.6: ReLU in Keras.

In this example 3.6, we add a dense layer with 64 units and 100 inputs. We then apply ReLU activation using the ***Activation*** layer and the **relu** argument.

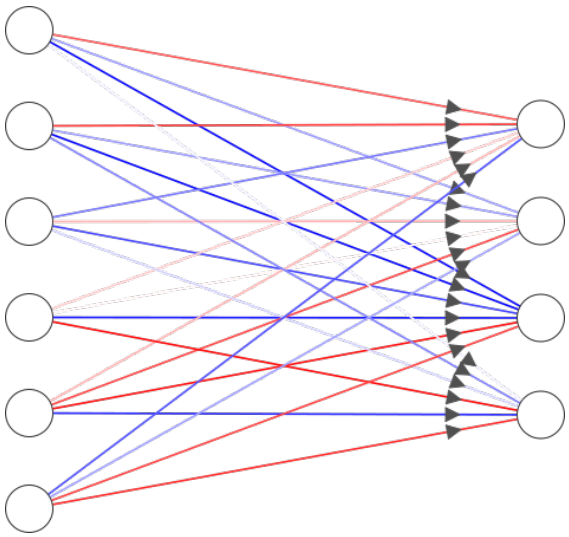


Figure 3.4: Fully-connected layer in a Neural network

A fully connected layer (Unzueta, 2022) (also known as a dense layer) is a layer in a neural network where every neuron is connected to every neuron in the previous layer. This layer is commonly used in the output layer of classification and regression models.

A fully connected layer in Keras can be created using the ***Dense*** layer. The ***Dense*** layer has a number of units (neurons) that can be specified as a parameter. For example, to add a fully connected layer with 64 units to a Keras model, you can do the following:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(64, activation='relu', input_dim=100))
```

Listing 3.7: A fully connected layer in Keras.

In this example, we create a Keras model and add a fully connected layer with 64 units using the ***Dense*** layer. We also specify the activation function as ReLU and the input shape as 100.

The output of a fully connected layer is computed as follows:

```
output = activation(dot(input, weights) + bias)
```

Listing 3.8: A fully connected layer output.

Where **input** is the **input** to the layer, **weights** are the weights associated with each neuron in the layer, **bias** is a bias term associated with each neuron, and activation is the **activation** function applied to the output.

The Softmax function and its corresponding loss function, Softmax loss (or Categorical Cross-entropy loss) (SuperDataScience Team, 2018), are commonly used in classification problems.

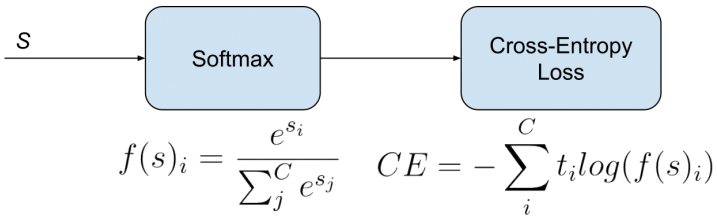


Figure 3.5: In short, Softmax Loss is actually just a Softmax Activation plus a Cross-Entropy Loss (Rashad, 2020)

The softmax function takes an input vector and produces a probability distribution over a set of classes. The output of the softmax function is a set of probabilities, where each probability represents the likelihood that the input belongs to a particular class. The softmax loss function is used to measure the difference between the predicted probabilities and the actual probabilities.

In Keras, the softmax function and softmax loss can be used in the output layer of a neural network for multi-class classification problems. Here is an example of how to create a Keras model with a softmax output layer and softmax loss function:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(64, activation='relu', input_dim=100))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['
    accuracy'])
```

Listing 3.9: Softmax in Keras.

In this example, we create a Keras model with two dense layers, the first one having 64 units and ReLU activation, and the second one having 10 units (equal to the number of classes in the classification problem) and **softmax** activation. We also compile the model with Stochastic Gradient Descent optimizer and **softmax loss** function, and accuracy as the evaluation metric.

Last technique we used to design our model is Adam (Adaptive Moment Estimation) optimizer (Kingma and Ba, 2014), it is a popular gradient descent optimization algorithm used in deep learning and implemented in Keras.

Adam optimizer uses a moving average of the gradient and the squared gradient to update the weights of the neural network. It also uses adaptive learning rates for each weight parameter, which helps to converge faster and avoid oscillation.

In Keras, Adam optimizer can be used by specifying the Adam keyword in the compile() method. Here is an example:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential()
model.add(Dense(64, activation='relu', input_dim=100))
model.add(Dense(7, activation='softmax'))
```

```
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy',
    metrics=['accuracy'])
```

Listing 3.10: Adam optimizer in Keras.

In this example, we create a Keras model with two dense layers, the first one having 64 units and **ReLU** activation, and the second one having 7 units (equal to the number of emotion classes in the classification problem we are facing) and **softmax** activation. We also compile the model with **Adam** optimizer, a learning rate of 0.001, and **softmax loss** function and accuracy as the evaluation metric.

3.2.2 Experiment

3.2.2.1 Dataset

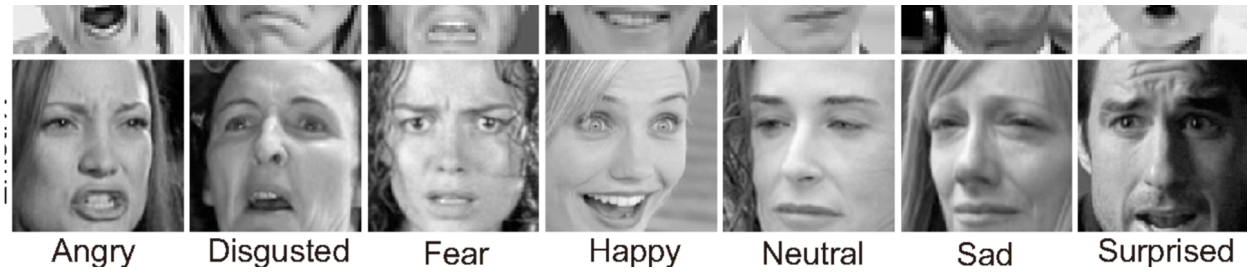


Figure 3.6: Dataset's seven categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral

[H]

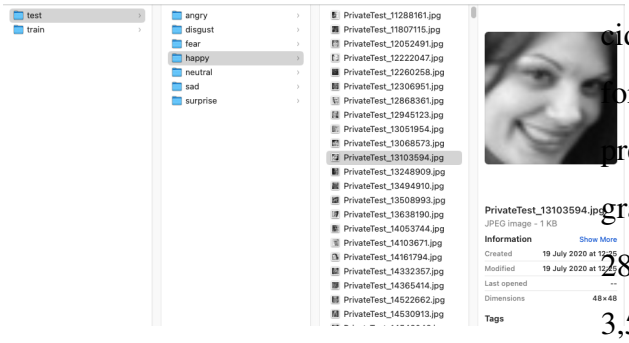


Figure 3.7: Dataset directory structure is divided into seven categories

Within the context of this thesis, we decided to use FER-13 (Goodfellow et al., 2013) for the purpose of detecting human facial expression. The dataset consists of 48x48 pixel grayscale images of faces. Train set covers 28,709 examples and the test set consists of 3,589 examples, each divided into seven categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. Those seven categories of emotion are sufficient for the food business if they can use it to infer customer moods and also provide a better, cutting-edge dining experience with FER features such as: Smile to begin payment automatically.

3.2.2.2 *Building Model*

Listing 3.2 provides an explanation of the image augmentation component. After augmented the input, we use Python and Keras to implement the first draft model of the CNN design shown in Figure 3.3. Currently, training is performed directly on the CPU in my local environment, which runs Mac OS 12 and has a 4 core CPU and 16 GB of memory. One training job takes about 3 hours to complete. Our code for such a design is shown below:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten,
    BatchNormalization, Conv2D, MaxPooling2D
from keras.optimizers import Adam,SGD,RMSprop

no_of_classes = 7

model = Sequential()

#1st wiath layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

```

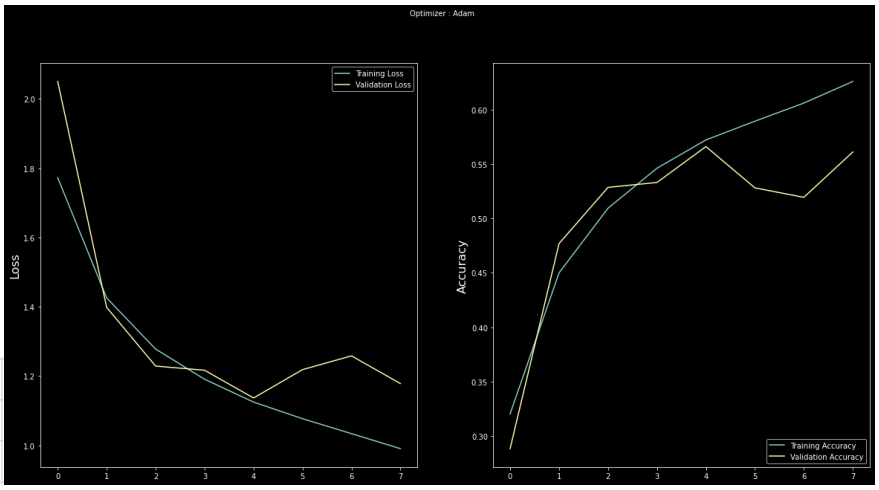
```
#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes, activation='softmax'))

opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['
    accuracy'])
```

Listing 3.11: Model implementation.

After finishing the draft model, we compile it and fit the input data to it. Plot 3.8b shows the resulted accuracy and loss. The hyperparameters for this draft model can be checked at 3.8a.

Epochs	48
Batch size	128
Learning rate	0.0001

(a) Draft Model's Hyperparameters.



(b) Draft Model's Accuracy and Loss.

Figure 3.8: Draft Model Implementation.

The best accuracy we got with this draft model was **0.5609**, and the best loss was **1.1785**. Let's figure out a way to raise those numbers and shorten the training time. To improve the robustness and accuracy of the model training process, we implement the aforementioned model in Tensorflow and make use of GPU accelerated deep learning features. The model can then be improved using SageMaker Automatic Model Tuning (Inc, 2022). This will enable us to achieve better results in a shorter amount of time, making the model more efficient and effective. Additionally, we will also experiment with different architectures and hyperparameters to further improve the model's accuracy. Finally, we will deploy the trained model on a cloud-based platform such as AWS Lambda for real-time inference on new data. By implementing these strategies, we believe we can significantly enhance our model's performance and provide more accurate predictions for clients.

3.2.2.3 Training Model with TensorFlow

Let's start by using TensorFlow to train the model on remote AWS instances in order to reduce the training time. The code to train the model is below. The content of the entry point ***4_layers_keras_tf.py*** defines the model using Keras as in 3.11, as well as data augmentation and saving the compiled model for serving. We just use 1 instance, as that is the limitation of the AWS free account. The instance has 16 vCPUs and 64 GB of RAM. We also specify some hyperparam-

eters, but we will fine tune them later on.

```
from sagemaker.tensorflow import TensorFlow

# GPU instance
tf_estimator = TensorFlow(entry_point='4_layers_keras_tf.py',
                          role=role,
                          instance_count=1,
                          instance_type='ml.m5.4xlarge',
                          framework_version='1.12',
                          py_version='py3',
                          script_mode=True,
                          hyperparameters={
                              'epochs': 20,
                              'batch-size': 512,
                              'learning-rate': 0.01}
                         )
tf_estimator.fit()
```

Listing 3.12: Using TensorFlow to speed up training time.

```
Training Env:
{
    "additional_framework_parameters": {},
    "channel_input_dirs": {},
    "current_host": "algo-1",
    "framework_module": "sagemaker_tensorflow_container.training:main",
    "hosts": [
        "algo-1"
    ],
    "hyperparameters": {
        "batch-size": 512,
        "epochs": 20,
        "learning-rate": 0.01,
        "model_dir": "s3://sagemaker-ap-southeast-1-321977351608/sagemaker-ter
    },
...
2022-03-06 08:13:43 Completed - Training job completed
ProfilerReport-1646552451: NoIssuesFound
Training seconds: 1883
Billable seconds: 1883
```

Figure 3.9: Training on remote instance

The training duration is now only about **1883 seconds**, or **31 minutes and 23 seconds**

3.9. That is acceptable training time under the given circumstances, but in the future, we would like to experiment with GPU instances because training on the GPU moves much more quickly than on the CPU. Any graphic operation you perform simply involves applying a mathematical transformation to a matrix because in computer memory, graphics are stored as matrices. For the most part, deep learning accomplishes exactly that.

3.2.2.4 Model Hyperparameters Auto Tuning with SageMaker

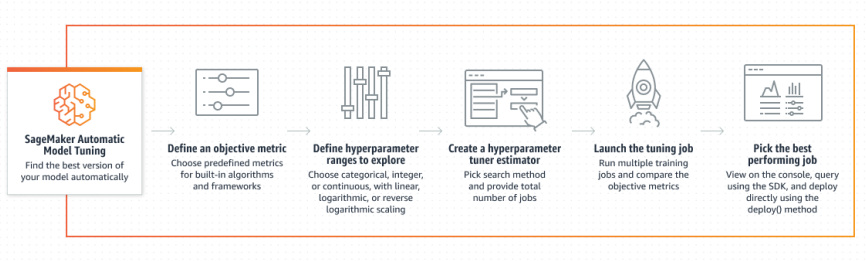


Figure 3.10: How SageMaker Automatic Model Tuning works.

Now to improve the accuracy of the model, I use SageMaker Automatic Model Tuning (Inc, 2022). Amazon SageMaker Automatic Model Tuning is a feature of Amazon SageMaker that automates the process of hyperparameter tuning for machine learning models. It offers several benefits that can greatly improve your model training process, by leveraging Amazon SageMaker Automatic Model Tuning, you can streamline and optimize your model training process, saving time, effort, and resources. It automates the tedious and challenging task of hyperparameter tuning, enabling you to focus more on model development and achieving better results with your machine learning models.

We launch multiple remote instances, iterate training jobs with each pre-configured hyper-parameter range, log the resulted Validation Accuracy and Validation Loss to find the best perform model.

```
from sagemaker.tuner import IntegerParameter, CategoricalParameter,
    ContinuousParameter, HyperparameterTuner

tf_estimator = TensorFlow(entry_point='4_layers_keras_tf.py',
                          role=role,
                          instance_count=1,
                          instance_type='ml.m4.xlarge',
```

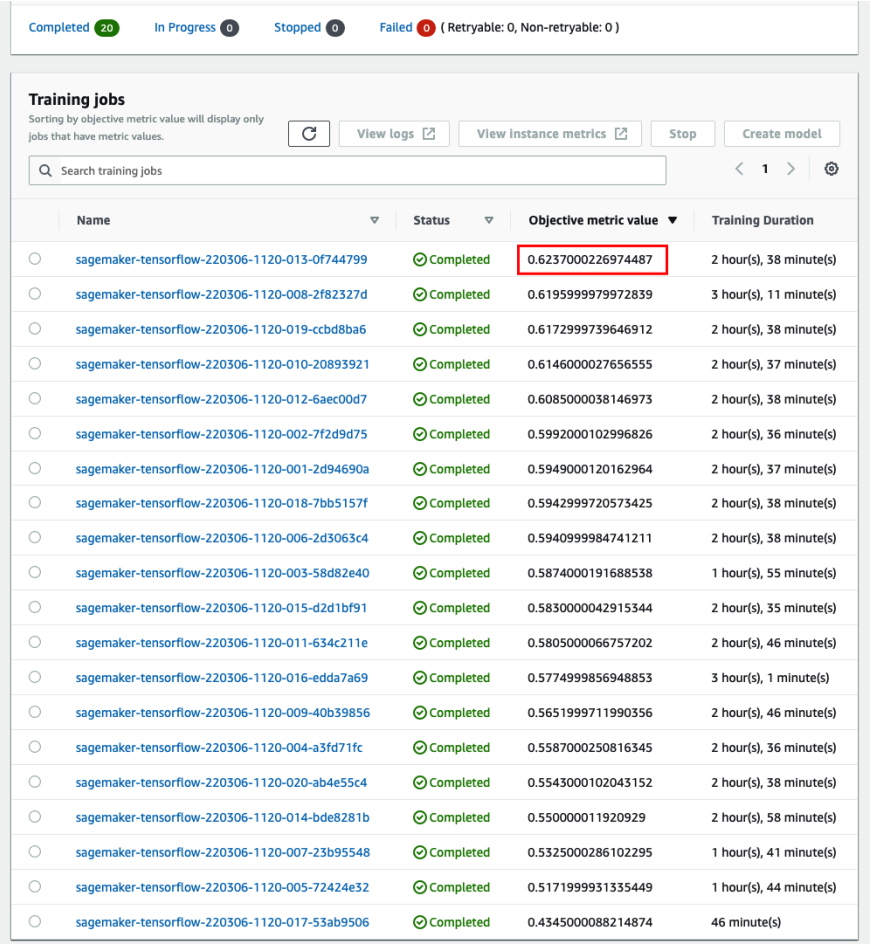
```
                            framework_version='1.12',
                            py_version='py3',
                            script_mode=True,
                           )

hyperparameter_ranges = {
    'epochs':        IntegerParameter(10, 100),
    'learning-rate': ContinuousParameter(0.001, 0.1, scaling_type='
   Logarithmic'),
    'batch-size':    IntegerParameter(32, 1024),
    'dropout':       ContinuousParameter(0.2, 0.6)
}

objective_metric_name = 'val_acc'
objective_type = 'Maximize'
metric_definitions = [{'Name': 'val_acc', 'Regex': 'val_acc: ([0-9\\.]+)'}]

tuner = HyperparameterTuner(tf_estimator,
                            objective_metric_name,
                            hyperparameter_ranges,
                            metric_definitions,
                            max_jobs=12,
                            max_parallel_jobs=4,
                            objective_type=objective_type)
tuner.fit()
```

Listing 3.13: Using SageMaker Automatic Model Tuning.

Here in 3.13 we experimented with some hyperparameters such as: Epochs range from 10 to 100, learning rate ranges from 0.001 to 0.1, batch size varies from 32 to 1024, Drop out value ranges from 0.2 to 0.6. The power of SageMaker hyperparameter tuning comes most is when we can run many parallel jobs on many remote instances to save training time even more significantly. I utilized max 12 training jobs, with 4 jobs run concurrenly at the same time.

Batch size	214
Dense layers	628
Dropout	0.47883682399925637
Epochs	34
Learning rate	0.011064697608126084

(a) Model's Hyperparameters.



(b) Hyperparameter Tuning jobs.

Figure 3.11: Model Hyperparameter Tuning.

As a result, on Figure 3.11, we can see the Validation Accuracy has improved from **0.5609** to **0.6237**. Validation Loss has decreased from **1.1785** to **1.0711**. Training took 2 hours 28 minutes. We also have the hyperparameter values used for this best performing model, so that we can replicate the result again to explore more if needed.

3.2.2.5 Deploy Best Model to an API endpoint

The generated model artifact will be saved to an Amazon S3 bucket, we will use SageMaker console deploy the model right from there. Like below:

```
from sagemaker.tensorflow import TensorFlowModel

model = TensorFlowModel(
```

```
    model_data='s3://sagemaker-ap-southeast-1-321977351608/sagemaker-tensorflow-scriptmode-2022-03-05-17-16-22-538/output/model.tar.gz',
    role=role,
    framework_version='1.12'
)

tf_predictor = model.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
tf_endpoint_name = tf_predictor.endpoint_name
```

Listing 3.14: Deploy directly from model artifact.

After deployed, we have an API endpoint (Figure 3.12) that we can call to predict. The input data would be an grayscaled, 48x48 pixel image, the output of the endpoint would be the facial expression category (0 = Angry, 1 = Disgust, 2 = Fear, 3 = Happy, 4 = Neutral, 5 = Sad, 6 = Surprise).

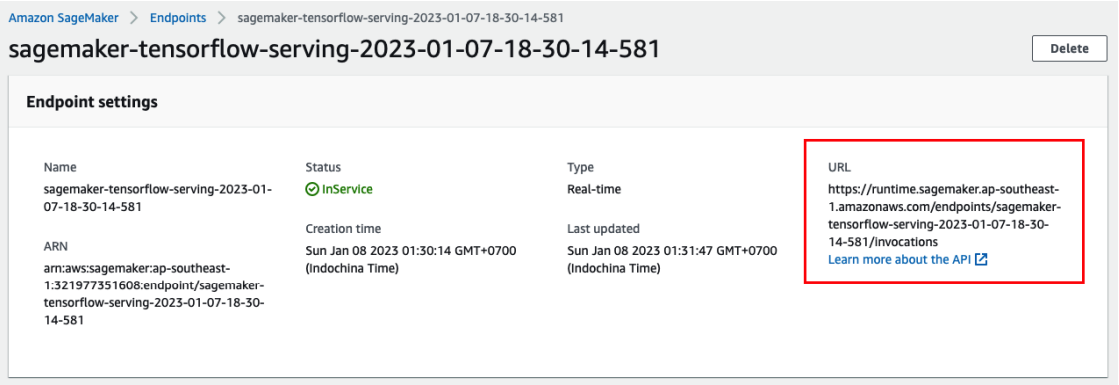


Figure 3.12: Deployed endpoint

CHAPTER 4. SIMULATION RESULTS

4.1 Facial Detection

In most use-cases, like smiling to start an order or getting the customer's feedback, we just set the camera nicely to look straight and close at the object; proper lighting will also be required. The distance from the camera to the object's face is around 40–80 cm. So that the conditions for facial recognition are optimal and could be happened as precisely as possible.

First, we test it using laptop camera (720P, 0.922 Megapixels), with a clean background and direct lighting at the object. The object is 50cm away from the camera.

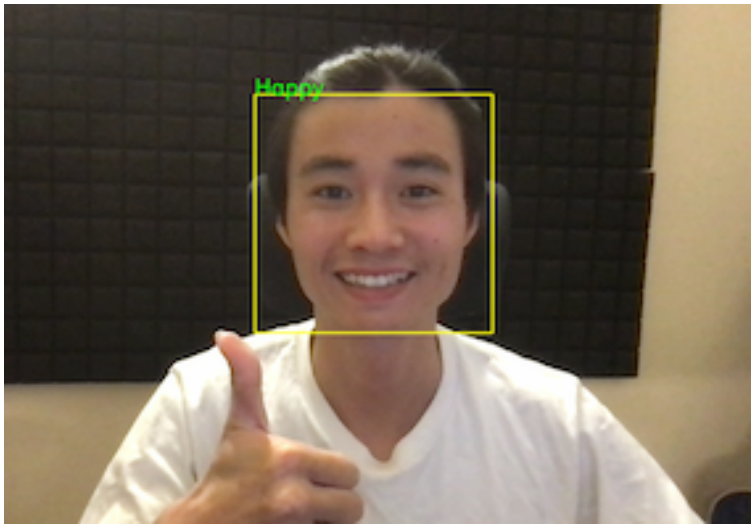


Figure 4.1: Facial Recognition with Haar-Cascade test with laptop camera setting.

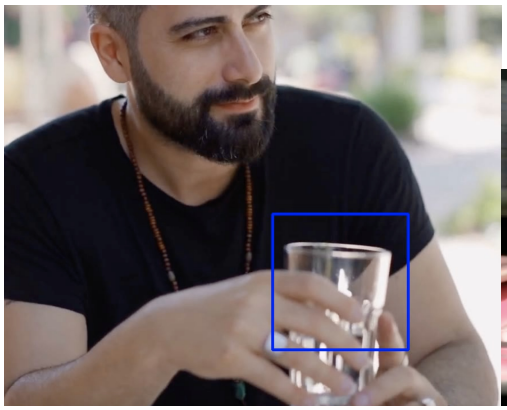
As expected, the result is very good, **0 case** of false-positive within 5 minutes of interactive with the test object. This successful test using a laptop camera with direct lighting and a clean background proves the reliability of the system. The lack of false-positives during five minutes of interaction with the test object is a strong indication that the system is accurate and effective. With the object positioned 50cm away from the camera, it's clear that this technology can be used in most use-cases for FnB business, with premise conditions, as we have mentioned in above chapters. Facial Recognition rate here is 100%, with recognition time less than 0.5 second with given configuration.

We can't completely control the lighting or the position and posture of the object, though. We therefore used a no copyright video (PexBell, 2021) with various restaurant backgrounds to test these conditions. In order to read from a video, we need to change the code a bit to read from a video, detect faces, then write the output video to a location:

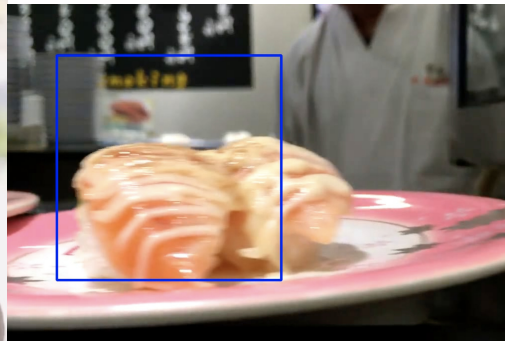
```
...
cap = cv2.VideoCapture('stock/Restaurant Stock Footage - Restaurant Free
    Stock Videos - Restaurant No Copyright Videos.mp4')

# Default resolutions of the frame are obtained.
# We convert the resolutions from float to integer.
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# Define the codec and create VideoWriter object.
out = cv2.VideoWriter('output.avi',cv2.VideoWriter_fourcc('M','J','P','G'),
    10, (frame_width,frame_height))
while cap.isOpened():
        ...
        # Write the frame into the file 'output.avi'
        out.write(frame)
        # cv2.imshow("window", frame)
        ...
```

Listing 4.1: Detect faceprint from a video.

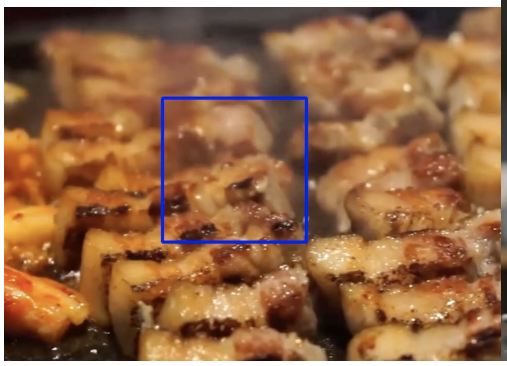
This time, based on the output video analysis, we detect **4 false-positive cases** and also some cases where the method failed to recognize faces, including when an object too far away, facing side way or object's face is not fully visible in the frame. However, four false-positive cases is a very small number for this longer than five minute, complex background conditions video, so this is still a very good outcome. 4.2.



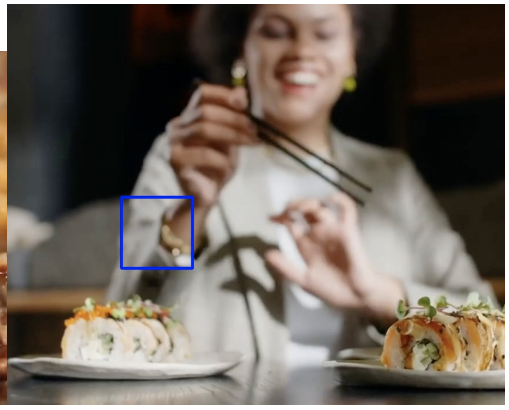
(a) False-positive case and the face is not fully shown in the frame



(b) False-positive case



(c) False-positive case



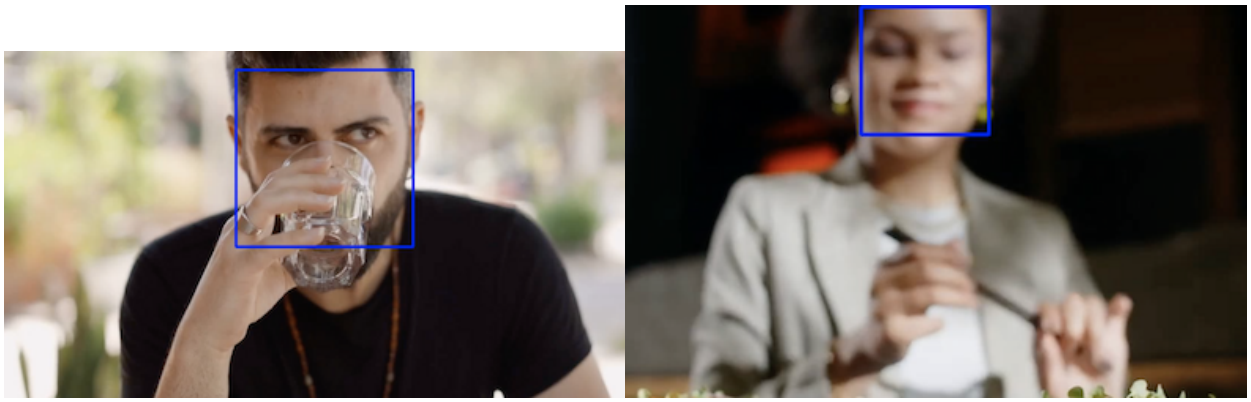
(d) False-positive case

(e) Object is facing side way

(f) Object is too far

Figure 4.2: Faceprint can not be recognized correctly cases.

The advantage of this method is it can recognize the face even if it was covered except the eyes. If the distance is in the optimal range and object is facing front, it recognizes very accurately, even when the face is blurred 4.3.



(a) Facial Recognition even if it was covered except the eyes.

(b) Facial Recognition even if blurred.

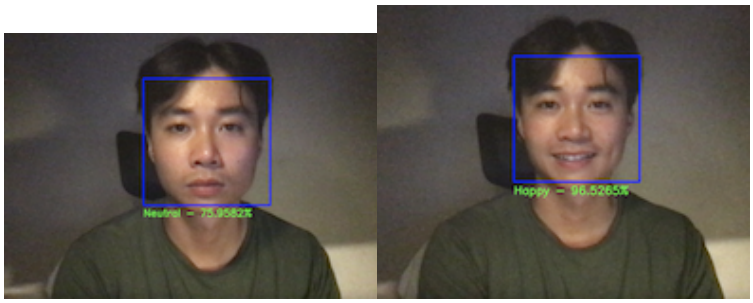
Figure 4.3: Faceprint recognized correctly cases.

Overall, using Python and OpenCV, we developed a Facial Recognition system proof of concept. which has been shown to be advantageous in our use cases for the FnB industry. The next step is to use a FER Deep Learning model to infer the object's emotion from the segmented faceprint.

4.2 Facial Expression Recognition

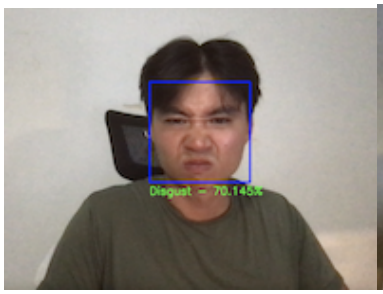
4.2.1 Simulation 1 - Using laptop camera with direct lighting

To assess the capabilities of our AI model, we conducted testing using a laptop camera (720P, 0.922 Megapixels) in an environment with a clean background and direct lighting. The object of interest was positioned at a distance of 50cm from the camera, representing the ideal operating condition for our model. During the testing phase, our AI model consistently exhibited exceptional performance, surpassing our expectations It successfully detected and accurately classified facial expressions with remarkable precision, especially on detecting **Happy** or **Neutral** 4.4, and reliability under these configurations, making it effective for applications in the FnB industry as mentioned in section 1.2.

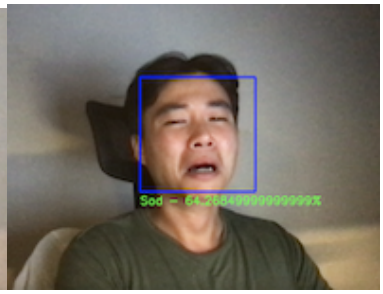


(a) Neutral

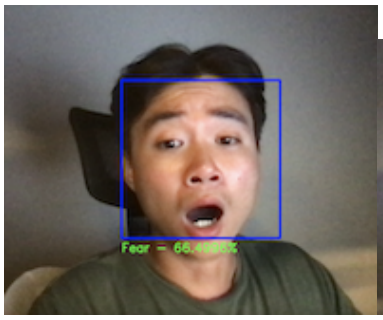
(b) Happy



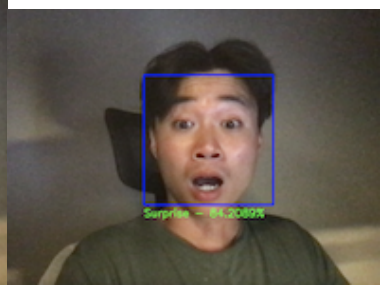
(c) Disgust



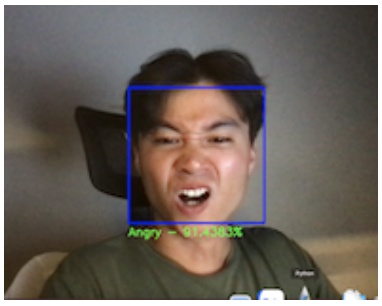
(d) Sad



(e) Fear



(f) Surprise



(g) Angry

Figure 4.4: Classified facial expressions using laptop webcam.

4.2.2 *Simulation 2 - Using video with multiple faces and good lighting*

To expand the scope of our testing and further evaluate the performance of our model, we utilized a video (MDI Management Development International, 2016) provided by MDI Management Development International. This video featured multiple individuals exhibiting various facial expressions, providing a diverse range of data for assessment. Similar to our previous testing conditions, the video maintained optimal conditions with good lighting, subjects facing directly forward, and a clean background.

(a) Neutral (b) Happy

(c) Disgust (d) Not sure

Figure 4.5: Emotions can be recognized in various cases and objects.

During this evaluation, our model exhibited exceptional performance, successfully detecting and classifying all facial expressions without any false-positive cases 4.5. However, we introduced an additional rule to enhance the reliability of our results. Specifically, we implemented a threshold where only emotions with a probability exceeding 50% were considered valid. If none of the

detected emotions met this threshold, the model labeled the frame as "Not sure" and moved on to next frames with more valuable facial emotional information. By implementing this rule, we aimed to ensure that our model prioritizes confident and accurate predictions, filtering out frames that may contain ambiguous or less reliable emotional cues.

4.2.3 Simulation 3

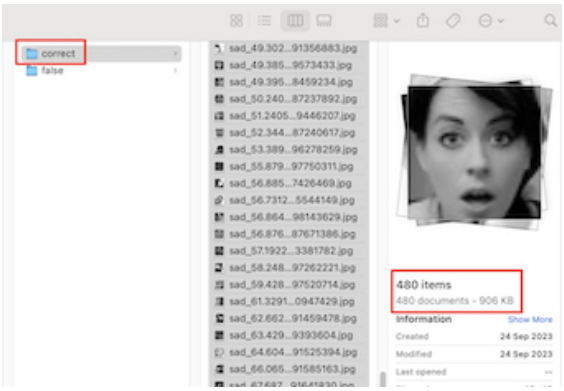


Figure 4.6: Number of correct cases over 700 samples test set

To further assess the capabilities of our model, we conducted an additional test. In this test, we randomly selected 700 images from the Test Set of FER-2013. Using the TensorFlow API, we systematically examined each of these samples. This expanded Test Set now comprises 700 samples, each labeled according to their file names.

We developed a Python script to iterate through the selected samples, leveraging our system to infer the emotional content of each image. Subsequently, we compared the model's predictions with the known labels of the samples to determine correctness. Correctly identified cases were cataloged in a "correct" folder, while instances of misclassification were stored in a "false" folder. This arrangement enables us to conduct further investigations on the misclassified cases, as referenced in the provided listing 4.2.

```
...
def determineResult(predict_label, filename):
    if predict_label in filename:
        return 'correct'
    return 'false'

face_detector1 = cv2.CascadeClassifier('../haarcascade_frontalface_default.
    xml')
predict_labels = ['angry', 'disgust', 'fear',
                  'happy', 'neutral', 'sad', 'surprise']

# Tensor serving endpoint config
```

```
endpoint = 'sagemaker-tensorflow-serving-2023-09-24-06-34-31-904'
config = botocore.config.Config(read_timeout=80)
runtime = boto3.client('runtime.sagemaker', config=config)

# Loop through and segment the faceprint out of the samples
for filename in os.listdir('test_set'):
    frame = cv2.imread(filename=os.path.join('test_set', filename))
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Infer emotion
    face = cv2.resize(gray, (48, 48))
    face = face[..., np.newaxis]
    face = np.expand_dims(face, axis=0)
    data = np.array(face)
    payload = json.dumps({"instances": data.tolist()})

    response = runtime.invoke_endpoint(EndpointName=endpoint,
                                       ContentType='application/json',
                                       Body=payload)
    result = json.loads(response['Body'].read().decode())

    predict_label = predict_labels[(
        np.array(result['predictions'][0]).argmax())]
    predict_certainty = float(
        np.array(result['predictions'][0]).max()) * 100
    result = determineResult(predict_label, filename)

    # Put prediction info overlay
    text = predict_label + '_' + str(predict_certainty) + ' - '

    cv2.imwrite(os.path.join('result', result, text + filename), frame)
...
```

Listing 4.2: Simulation 3 code.

Below is our simulation result:

Test set size	Correct samples	Incorrect samples
700	480	220

We are pleased to report that our model achieved an impressive accuracy rate of approximately 68%. This high accuracy underscores the effectiveness of our system in recognizing and categorizing facial expressions, demonstrating its strong potential for various applications.

4.2.4 Summary

In summary, the results obtained from our proof-of-concept program in this section demonstrate its remarkable capabilities. Under the specified conditions, the program consistently achieved a recognition accuracy of over 90% for human faceprints. Furthermore, it successfully extracted and recognized facial expressions in less than 0.5 seconds, achieving an accuracy of 62.37% across all seven categories of emotions.

In the realm of Restaurant AI, the integration of face recognition and FER opens up innovative avenues for enhancing customer experiences. Imagine a scenario where customer's facial expressions are analyzed as they interact with the dining environment. If a customer leaves with a satisfied - happy expression, the system could prompt a feedback mechanism, enabling them to rate their experience. This real-time feedback loop provides valuable insights for restaurant owners to gauge customer satisfaction. Moreover, face recognition technology can be leveraged to identify and welcome returning customers, offering a personalized touch. By employing FER, the system can further interpret subtle expressions, helping discern not just overall satisfaction but also specific sentiments during the dining experience. This holistic approach to Restaurant AI goes beyond the conventional realm, actively contributing to customer engagement, feedback mechanisms, and the establishment's overall service quality.

CHAPTER 5. CONCLUSION

In conclusion, this thesis has achieved all the objectives set out in section Objectives 1.2.2, involving the development of a simple yet effective CNN architecture for FER. By incorporating various techniques and leveraging the AWS stack, we were able to enhance the model's performance and significantly reduce training time. The results of our experiments clearly demonstrate the model's capability to accurately detect and classify facial emotions, providing an API endpoint that is easily integratable into existing systems.

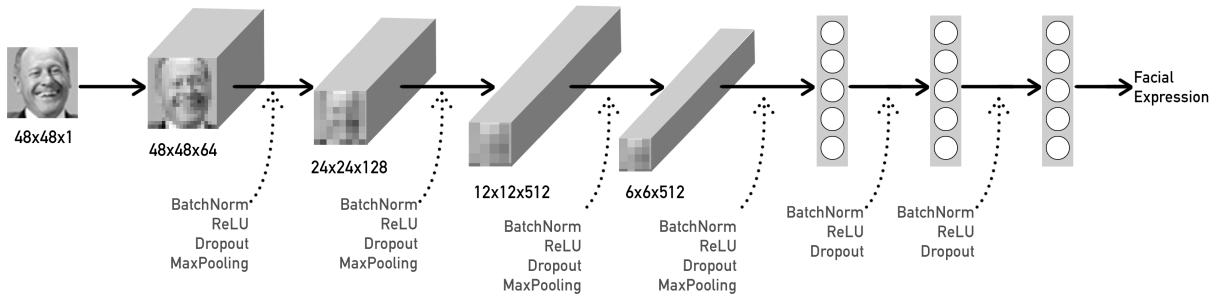
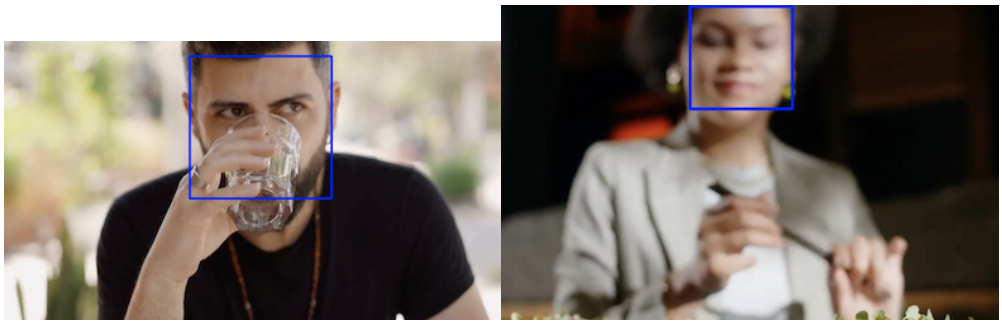


Figure 5.1: The CNN Model Network Architecture Designed

Throughout the project, we focused on improving both the accuracy and efficiency of the FER model. We carefully designed the CNN architecture, considering factors such as model depth, convolutional layers, and pooling operations to ensure optimal performance. Additionally, we implemented techniques like batch normalization, dropout, and max-pooling to enhance the model's generalization and reduce overfitting.



(a) Facial Recognition even if it was covered except the eyes.

(b) Facial Recognition even if blurred.

Figure 5.2: Testing with real and complicated data.

Through extensive testing and validation, we gradually achieved high accuracy rates, showcasing the model's ability to accurately recognize and classify various facial expressions. The exposed API endpoint provided a user-friendly interface that allows easy integration with other systems and applications.

Overall, our work contributes to the growing field of facial expression recognition and highlights the effectiveness of CNN architectures combined with AWS stack techniques. The developed model provides a powerful tool for accurately detecting and classifying facial emotions, with an exposed endpoint that facilitates easy integration into various applications and systems. Specifically, within the FnB industry, our model provides valuable insights into customer emotions, feedback analysis, and market research, contributing to enhanced customer satisfaction, improved product offerings, and overall business growth.

CHAPTER 6. FUTURE DIRECTIONS

In our future work, we have identified several areas that we would like to explore further to enhance our facial expression recognition system:

- Incorporating More Angles Faces Data: To improve the model's ability to detect side faces, we plan to enrich our dataset with face images from many more angles. By including a diverse range of facial orientations, we aim to train the model to accurately recognize and classify facial expressions from different angles.

- Hyperparameter Optimization: We intend to experiment with a variable number of M convolutional layers as a hyperparameter. By leveraging SageMaker's automatic model tuning capabilities, we can explore different configurations and let the system optimize the model's architecture to maximize validation accuracy. This approach will allow us to discover the most suitable number of convolutional layers for our specific FER task.

- GPU Training: We are interested in exploring the performance of GPU training compared to CPU training. GPUs offer parallel processing capabilities, which can significantly accelerate model training and inference times. By utilizing GPU resources, we can assess the impact on training speed and potentially achieve faster convergence and improved overall efficiency.

- Continuous Model Improvement: To continuously enhance our model's performance, we plan to leverage SageMaker techniques for data management and model deployment. We aim to leverage Amazon S3 to feed new data continuously, enabling the model to learn from a larger and more diverse dataset. By training new models on a regular basis, preferably during off-peak hours, and deploying the best-performing model to the API endpoint, we can ensure that our model adapts to new data, mitigates overfitting, and improves its accuracy over time.

- Consistent Lighting and Background: We recognize the importance of standardizing lighting conditions and background cleanliness for robust FER. In our future work, we will further refine our preprocessing techniques to remove variations introduced by different lighting setups and ensure a consistent background. This step will enhance the model's

resilience to environmental factors, leading to more reliable and precise results.

By focusing on these areas in our future work, we anticipate achieving even better accuracy and robustness in recognizing facial expressions. These advancements will further solidify the utility and effectiveness of our system in real-world FnB applications.

REFERENCES

Abdullah, S. M. A., Ameen, S. Y. A., M. Sadeeq, M. A., & Zeebaree, S. (2021). Multimodal emotion recognition using deep learning. *Journal of Applied Science and Technology Trends*, *2*(02), 52–58. https://doi.org/10.38094/jastt20291

Agarwala, A., & Vemulapalli, R. (2018). A compact embedding for facial expression similarity.

Agrawal, A., & Mittal, N. (2019). Using cnn for facial expression recognition: A study of the effects of kernel size and number of filters on accuracy. https://doi.org/10.1007/s00371-019-01630-9

Alizadeh, S., & Fazel, A. (2017). Convolutional neural networks for facial expression recognition. https://doi.org/10.48550/arXiv.1704.06756

Bazzo, J. J., & Lamar, M. V. (2004). Recognizing facial actions using gabor wavelets with neutral face average difference,

Benitez-Quiroz, C. F., Srinivasan, R., & Martinez, A. M. (2016). The emotionet database includes 950,000 images with annotated action units. https://gas.graviti.com/dataset/graviti/EmotioNet

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brownlee, J. (2018). A gentle introduction to dropout for regularizing deep neural networks [A single model can be used to simulate having a large number of different network architectures by randomly dropping out nodes during training. This is called dropout]. https://machinelearningmaster

Brownlee, J. (2019a). A gentle introduction to batch normalization for deep neural networks [Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch]. https://machinelearningmastery.com/batch-normalizatior

Brownlee, J. (2019b). A gentle introduction to pooling layers for convolutional neural networks [Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map]. https://machinelearningmastery.com/pooling-layers

Brownlee, J. (2020). A gentle introduction to the rectified linear unit (relu) [The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero]. https://machinelearningmastery.com/rectified-lin

Chollet, F. (n.d.). Imagedatagenerator. https://keras.io/api/preprocessing/image/

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2019). A survey of deep learning and its applications: A new paradigm to machine learning. *Arch Computat Methods Eng*, 1071–1092. https://doi.org/10.1007/s11831-019-09344-w

de Wijk, R. A., Ushiama, S., Ummels, M., Zimmerman, P., Kaneko, D., & Vingerhoeds, M. H. (2021). Reading food experiences from the face: Effects of familiarity and branding of soy sauce on facial expressions and video-based rppg heart rate. https://doi.org/10.3390/foods10061345

Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011). Static facial expressions in the wild (sfew). https://paperswithcode.com/dataset/sfew

Ekman, P., & Friesen, W. (1978). Facial action coding system: A technique for the measurement of facial movement, consulting psychologists press.

et al, W.-J. Y. (2014). Casme ii: An improved spontaneous micro-expression database and the baseline evaluation. https://doi.org/10.1371/journal.pone.0086041

Fan, H., & Zhou, E. (2016). Approaching human level facial landmark localization by deep learning [300-W, the First Automatic Facial Landmark Detection in-the-Wild Challenge]. *Image and Vision Computing*, *47*, 27–35. https://doi.org/https://doi.org/10.1016/j.imavis.2015.11.004

Fasel, B., & Luettin, J. (2003). Automatic facial expression analysis: A survey.

Fesabas, A. K. R., Asirvatham, D., & Poulain, J. P. (2021). Facial expression recognition in food studies. *Asia-Pacific Journal of Innovation in Hospitality and Tourism (APJIHT)*, *10*, 73–79.

Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., & Lee, D.-H. (2013). Challenges in representation learning: A report on three machine learning contests. in international conference on neural information processing, https://www.kaggle.com/datasets/msambare/fer2013

Gozde, Y., I, O., S, K., C, O., K, P., TE, L., & F, B. (2019). Facial expression recognition for monitoring neurological disorders based on convolutional neural network. https://doi.org/10.1007/s11042-01

Gross, R., Matthews, I., Cohn, J., Kanade, T., & Baker, S. (2009). Multi-pie. http://www.cs.cmu.edu/afs/cs/proj

Haar, A. (1910). Zur Theorie der orthogonalen Funktionensysteme. https://doi.org/10.1007/BF01456326

Handley, L. (2023). A.i. could 'remove all human touchpoints' in supply chains. here's what that means. https://www.cnbc.com/2023/06/19/supply-chains-how-ai-could-remove-all-human-touchpoint

Inc, A. W. S. (2022). Amazon sagemaker automatic model tuning. https://aws.amazon.com/sagemaker/automat

Jun, H., Shuai, L., Jinming, S., Yue, L., Jingwei, W., & Peng, J. (2018). Facial expression recognition based on vggnet convolutional neural network, 4146–4151. https://doi.org/10.1109/CAC.2018.862

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/arXiv.14

Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D. H. J., Hawk, S. T., & van Knippenberg, A. (2010). Presentation and validation of the radboud faces database. https://doi.org/10.1080/0269993090

Li, S., Deng, W., & Du, J. (2017). Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild.

Liang, D., Liang, H., Yu, Z., & et Y. Zhang. (2020). Deep convolutional bilstm fusion network for facial expression recognition. https://doi.org/10.1007/s00371-019-01636-3

Lopes, A. T., de Aguiar, E., Souza, A. F. D., & Oliveira-Santos, T. (2017). Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order. https://doi.org/10.1016/j.patcog.2016.07.026

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. https://doi.org/10.1109/CVPRW.2010.5543262

Lyons, M., Kamachi, M., & Gyoba, J. (1998). The japanese female facial expression (jaffe) database. https://doi.org/10.5281/zenodo.3451524

McKendrick, J. (2021). Ai adoption skyrocketed over the last 18 months. https://hbr.org/2021/09/ai-adoption-sl

MDI Management Development International. (2016). The 7 basic emotions - do you recognise all facial expressions? https://www.youtube.com/watch?v=embYkODkzcs

Mellouka, W., & Handouzi, W. (2020). Facial emotion recognition using deep learning: Review and insights.

Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). Affectnet: A database for facial expression, valence, and arousal computing in the wild. http://mohammadmahoor.com/affectnet/

Pantic, M., Valstar, M., Rademaker, R., & Maat, L. (2005). Web-based database for facial expression analysis. https://doi.org/10.1109/ICME.2005.1521424

PexBell. (2021). Restaurant stock footage - restaurant free stock videos - restaurant no copyright videos. https://www.youtube.com/watch?v=78ixbqwFLbc

Purkayastha, D., & Malathi, D. (2022). Leveraging transfer learning for effective recognition of emotions from images: A review. In D. P. Agrawal, N. Nedjah, B. B. Gupta, & G. Mar-

tinez Perez (Eds.), *Cyber security, privacy and networking* (pp. 13–24). Springer Nature Singapore.

Rashad, F. (2020). Additive margin softmax loss (am-softmax). https://towardsdatascience.com/additive-margi

Rawal, N., Koert, D., Turan, C., Kersting, K., Peters, J., & Stock-Homburg, R. (2022). Exgennet: Learning to generate robotic facial expression using facial expression recognition. https://doi.org/10.3389/frobt.2021.730317

Safaryan, B. (2023). Three reasons generative ai will be the next disruption for e-grocery stores. https://www.forbes.com/sites/forbestechcouncil/2023/05/10/three-reasons-generative-ai-will-be-the-n

Sardar, S., Ray, R., Hasan, M. K., Chitra, S. S., Parvez, A. T. M. S., & Avi, M. A. R. (2022). Assessing the effects of covid-19 on restaurant business from restaurant owners' perspective. https://doi.org/10.3389/fpsyg.2022.849249

Schenkel, T., Ringhage, O., & Branding, N. (2019). A comparative study of facial recognition techniques: With focus on low computational power.

Singapore Computer Society. (2020). Simplifying the difference: Machine learning vs deep learning. https://www.scs.org.sg/articles/machine-learning-vs-deep-learning

Sun, D. W. (2012). Computer vision technology in the food and beverage industries.

SuperDataScience Team. (2018). Convolutional neural networks (cnn): Softmax and cross-entropy [To normalize the output of a network to a probability distribution over predicted output classes]. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-softmax-crosse

Tuoi Tre News. (2021). Fnb businesses struggle to stay afloat during new normal in ho chi minh city. https://tuoitrenews.vn/news/business/20211110/fampb-businesses-struggle-to-stay-afloat-during-

Unzueta, D. (2022). Fully connected layer vs. convolutional layer: Explained [A fully connected layer refers to a neural network in which each input node is connected to each output node. In a convolutional layer, not all nodes are connected]. https://builtin.com/machine-learning/fully-conne

Valstar, M. F., Jiang, B., Mehu, M., Pantic, M., & Scherer, K. (2011). The first facial expression recognition and analysis challenge. https://doi.org/10.1109/FG.2011.5771374

Vincent, M. (2023). The many ways ai & chatgpt will disrupt food tech. https://www.greenqueen.com.hk/ai-cha

Xia, Y., Zheng, W., Wang, Y., Yu, H., Dong, J., & Wang, F.-Y. (2022). Local and global perception generative adversarial network for facial expression synthesis. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(3), 1443–1452. https://doi.org/10.1109/TCSVT.2021.30740

Yin, L., Wei, X., Sun, Y., Wang, J., & Rosato, M. J. (2006). A 3d facial expression database for facial behavior research. https://doi.org/10.1109/FGR.2006.6

Zacks. (2016). Baidu (bidu) and kfc team up for smart restaurant in china. https://www.nasdaq.com/articles/baic

Zhao, G., Huang, X., Taini, M., Li, S. Z., & Pietikäinen, M. (2011). Facial expression recognition from near-infrared videos. https://doi.org/10.1016/j.imavis.2011.07.002

Zhong, Y., Qiu, S., Luo, X., Meng, Z., & Liu, J. (2020). Facial expression recognition based on optimized resnet. *2020 2nd World Symposium on Artificial Intelligence (WSAI)*, 84–91. https://doi.org/10.1109/WSAI49636.2020.9143287

Zou, X., & Zhao, J. (2015). Nondestructive measurement in food and agro-products. https://doi.org/10.1007/97