5-2019

Real-Time Facial Emotion Recognition Using Fast R-CNN

Salem Bin Saqer AlMarri
sxa4391@rit.edu

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact rit.scholarworks@rit.edu.
Real-Time Facial Emotion Recognition Using Fast R-CNN

by

Salem Bin Saqer AlMarri

A Thesis Submitted in Partial Fulfilment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Department of Electrical Engineering and Computing Sciences
Kate Gleason College of Engineering
Rochester Institute of Technology
Dubai, United Arab Emirates
May 2019
Real-Time Facial Emotion Recognition Using Fast R-CNN

by

Salem Bin Saqer AlMarri

A Thesis Submitted in Partial Fulfilment of the Requirements
for the Degree of Master of Science in Electrical Engineering
Department of Electrical Engineering and Computing Sciences

Approved by:

Dr. Abdulla Ismail
Thesis Advisor – Professor of Electrical Engineering
Department of Electrical Engineering and Computing Science
RIT Dubai

Dr. Boutheina Tili
Thesis Committee Member – Associate Professor of Electrical Engineering
Department of Electrical Engineering and Computing Science
RIT Dubai

Dr. Jinane Mounsef
Thesis Committee Member – Assistant Professor of Electrical Engineering
Department of Electrical Engineering and Computing Science
RIT Dubai
Table of Contents

Chapter 1: Introduction to Neural Networks ................................................................. 1
  1.1 Models of a Neuron .............................................................................................. 2
  1.2 Rosenblatt’s Perceptron .................................................................................... 10
  1.3 Neural Networks Applications .......................................................................... 13
Chapter 2: Convolutional Neural Networks ................................................................. 16
  2.1 What is a convolutional neural network? ............................................................ 16
  2.2 Convolutional Neural Network Learning ......................................................... 18
  2.3 ConvNet Classifiers ......................................................................................... 19
Chapter 3: Facially Expressed Emotions .................................................................... 22
  3.1 Facial Action Coding System ........................................................................... 23
  3.2 Emotion Classification ..................................................................................... 25
Chapter 4: Principle Previous Work in Object Detection and Face Emotion Recognition .................................................................................................................. 28
Chapter 5: Face Emotion Recognition Using Fast R-CNN ........................................... 30
  5.1 Deep Learning .................................................................................................. 30
  5.2 Fast R-CNN – Object Detector ......................................................................... 34
  5.3 Difference between CNN and Fast R-CNN ..................................................... 35
Chapter 6: Experimentation and Analysis .................................................................. 36
  6.1 JAFFE Database Experiment ........................................................................... 39
  6.2 FERC-2013 Database Experiment ..................................................................... 39
  6.3 RAVDESS Database Experiment ...................................................................... 39
  6.4 Live Deployment Experiment .......................................................................... 47
Chapter 7: Conclusion and Future Work ..................................................................... 52
  7.1 Conclusion ......................................................................................................... 52
  7.2 Future Work ...................................................................................................... 53
References .................................................................................................................. 54
Appendix ..................................................................................................................... 58
Deep Learning Toolbox – Functions ......................................................................... 58
MATLAB Codes ......................................................................................................... 62
Abstract

In computer vision and image processing, object detection algorithms are used to detect semantic objects of certain classes of images and videos. Object detector algorithms use deep learning networks to classify detected regions. Unprecedented advancements in Convolutional Neural Networks (CNN) have led to new possibilities and implementations for object detectors. An object detector which uses a deep learning algorithm detect objects through proposed regions, and then classifies the region using a CNN. Object detectors are computationally efficient unlike a typical CNN which is computationally complex and expensive. Object detectors are widely used for face detection, recognition, and object tracking. In this thesis, deep learning based object detection algorithms are implemented to classify facially expressed emotions in real-time captured through a webcam. A typical CNN would classify images without specifying regions within an image, which could be considered as a limitation towards better understanding the network performance which depend on different training options. It would also be more difficult to verify whether a network have converged and is able to generalize, which is the ability to classify unseen data, data which was not part of the training set. Fast Region-based Convolutional Neural Network, an object detection algorithm; used to detect facially expressed emotion in real-time by classifying proposed regions. The Fast R-CNN is trained using a high-quality video database, consisting of 24 actors, facially expressing eight different emotions, obtained from images which were processed from 60 videos per actor. An object detector's performance is measured using various metrics. Regardless of how an object detector performed with respect to average precision or miss rate, doing well on such metrics would not necessarily mean that the network is correctly classifying regions. This may result from the fact that the network model has been over-trained. In our work we showed that object detector algorithm such as Fast R-CNN performed surprisingly well in classifying facially expressed emotions in real-time, performing better than CNN.

Keywords: Artificial Intelligence, Convolutional Neural Network, Facial Emotion Recognition.
List of Tables

Table 1.1: Notations used to describe a nonlinear model of a neuron .............................................. 3
Table 1.2: Notation description of signal-flow graph of a perceptron .................................................. 11
Table 3.1: Single action units of facial action coding system ................................................................. 24
Table 6.1: Database properties of altered JAFFE Database ..................................................................... 36
Table 6.2: Fast R-CNN training options using JAFFE Database ............................................................... 36
Table 6.3 JAFFE trained Fast R-CNN benchmarked against other published work ............................... 38
Table 6.4: Database properties of facial expression recognition challenge 2013 ................................. 39
Table 6.5: Fast R-CNN training options using FER-2013 database ....................................................... 39
Table 6.6: Database properties of altered FER-2013 Database .............................................................. 41
Table 6.7: Fast R-CNN training options using FER-2013 database ....................................................... 41
Table 6.8: FER-2013 trained Fast R-CNN benchmarked against other published work ....................... 42
Table 6.9: RAVDESS database processing configuration ......................................................................... 43
Table 6.10: Fast R-CNN training options using RAVDESS database .................................................... 43
Table 6.11: Fast R-CNN training options using RAVDESS database for three different network models ......................................................................................................................................................... 44
Table 6.12: RAVDESS trained Fast R-CNN benchmarked against other published work ................. 45
List of Figures

Figure 1.1: Non-linear model of a neuron .............................................................. 2
Figure 1.2: Result of affine transformation of external bias input on neuron k on output $u_k$ .... 44
Figure 1.3: Threshold function and a Sigmoid function for varying slope parameter ($\alpha$) ........ 4
Figure 1.4: Feedforward network with a single layer of neurons .................................... 5
Figure 1.5: Fully connected feedforward network with one hidden layer and one output layer.... ................................................................. 6
Figure 1.6: Time response of a system to a feedforward weight $\omega$ ................................. 8
Figure 1.7: Data Pyramid ...................................................................................... 9
Figure 1.8: Transition pathway of Data Pyramid .......................................................... 9
Figure 1.9: Rosenblatt’s Perceptron ......................................................................... 10
Figure 1.10: Signal-flow graph of a perceptron .......................................................... 11
Figure 1.11: Hyperplane presentation of decision boundary for a two-class pattern classification ........................................................................ 12
Figure 2.1: Convolutional operation .......................................................................... 17
Figure 2.2: Pooling operation ................................................................................... 17
Figure 2.3: Fully connected layer ............................................................................. 18
Figure 2.4: Representation of sentiment analysis of deep convolutional neural network .... 21
Figure 3.1: Representation of facial action coding system ............................................ 23
Figure 3.2: Plutchik’s wheel of emotions ..................................................................... 25
Figure 3.3: Lovheim cube of emotion ....................................................................... 26
Figure 5.1: Deep learning, a subset of machine learning which is a subset of artificial intelligence ........................................................................ 30
Figure 5.2: Extracted features of 8 channels (labels) of layers 3, layer 8 and layer 14 as visualized by Deep Dream feature visualization technique ........................................ 33
Figure 5.3: Fast R-CNN Architecture ....................................................................... 35
Figure 6.1: Image samples obtained from JAFFE database [1] ....................................... 37
Figure 6.2: Detection precision analysis of Fast R-CCN (VGG-16) trained on JAFFE .......... 38
Figure 6.3: Image samples obtained from FER 2013 [2].................................................. 40
Figure 6.4: Detection precision analysis of Fast R-CCN (VGG-16) trained on FER-2013 ........ 41
Figure 6.5: Samples of processed images from RAVDESS database [3]................................. 43
Figure 6.6: Average precision performance analysis on three network models......................... 45
Figure 6.7: Average miss rate analysis on three different network models............................... 46
Figure 6.8: An undertrained network model, trained with max epochs of 3 (left) and an overtrained network model trained with max epochs of 30. (right)......................................................... 46
Figure 6.9: Detected and classified facial emotion using trained Fast R-CNN object detector (right), in comparison with a facial expression obtained from the training dataset (left)........ 48
Figure 6.10: Percentage of detected emotions expressed by 100 different participants............ 48
Figure 6.11: Trained Fast R-CNN classifying 5 expressed emotions........................................ 50
Figure 6.12: Rapidly protoyped system platform for Fast R-CNN deployment.......................... 51
Chapter 1: Introduction to Neural Networks

A neural network is a complex, nonlinear, parallel system consisting of smaller, simpler interconnected processing units that can perform computations [4]. Neural networks can be modified by changing the strength of each connection between processing units to achieve a desired output through a process called learning.

Neural networks automatically extract features from training sets during the learning process. With the ability to generalize, the trained neural network could classify new input data into the trained output classes [5]. There are several properties of neural networks. Neural networks can be linear or nonlinear. The importance of nonlinearity is when the network is expected to compute a desired output from a nonlinear input signal such as a speech signal. A neural network generally maps a set of inputs to a set of outputs (input-output mapping), where a set of training examples consists of specified unique input signals and corresponding desired output response. At random, the training examples are chosen from a set and fed to the network. The strength of connections between nodes of the network are then modified to minimize the difference between the desired response and the actual response produced by the network. This process is repeated until the difference between desired response and actual response is zero or minimal, thus achieving a steady-state for the network. Neural networks are adaptive, as they are able to adapt their synaptic weights (strength of interconnections between nodes) to surrounding changes.
1.1 Models of a Neuron

The simple processing unit interconnected with other units of the neural network is called a neuron. The neuron consists of three basic elements. The first element is a set of synapses or connecting links, each of which is characterized by a weight or strength of its own. The input signal to one set of the connecting links is multiplied by the strength of that connection. In the case of an artificial neuron, the range of values may include negative or positive values. The second element is adder, a linear combiner that sums each respective input signal multiplied by the strength of the connection. The third element is activation function, used to limit the amplitude of a neuron’s output to intervals of [0,1] or [-1,1], depending on the activation function.

1.1.1 Nonlinear Model of a Neuron

Fig 1.1 depicts a schematic of model of a single neuron.

![Figure 1.1: Non-linear model of a neuron.](image)

The mathematical relationships describing the neuron model are given in equations 1.1 – 1.5. Equation 1.1 and 1.2, a pair of equations that describe the general form of neuron k. In
equation 1.3, bias $b_k$ results in affine transformation of output $u_k$ thus forming $v_k$. Equation 1.4 and 1.5 are formulated by combining equations 1.1, 1.2, and 1.3. The notations used in the equation are given in Table 1.1. A graph of affine transformation resulted by input bias $b_k$ of output $u_k$, which is the linear transformation, is shown in Fig 1.2.

Table 1.1: Notations used to describe a nonlinear model of a neuron.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>Neuron.</td>
</tr>
<tr>
<td>$b_k$</td>
<td>externally applied bias, which increases or decreases net input of activation function. (+ve/-ve)</td>
</tr>
<tr>
<td>$x_m$</td>
<td>input signals to neurons.</td>
</tr>
<tr>
<td>$w_{km}$</td>
<td>synaptic weights of neuron.</td>
</tr>
<tr>
<td>$u_k$</td>
<td>linear combiner output of input signals.</td>
</tr>
<tr>
<td>$\varphi(\cdot)$</td>
<td>activation function.</td>
</tr>
<tr>
<td>$y_k$</td>
<td>output signal of neuron.</td>
</tr>
<tr>
<td>$v_k$</td>
<td>induced local field or activation potential of neuron.</td>
</tr>
</tbody>
</table>

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$ (1.1)  

$$y_k = \varphi(u_k + b_k)$$ (1.2)  

$$v_k = u_k + b_k$$ (1.3)  

$$v_k = \sum_{j=0}^{m} w_{kj} x_j$$ (1.4)  

$$y_k = \varphi(v_k)$$ (1.5)
1.1.2 Types of Activation Function

There are two basic types of activation functions denoted by $\varphi(v)$, which define the output of a neuron in terms of an induced local field $v$ as illustrated in Fig 1.3 [6]. Threshold activation function (equation 1.6), is similar to a step function, if an input value is above or below a certain threshold the neuron is activated. Sigmoid activation function (equation 1.7), outputs a more smooth, continuous range of values between -1 and 1 to predict outputs probability in classifying multiple classes.

1. **Threshold Function** [0,1]

   \[
   \varphi(v) = \begin{cases} 
   1 & \text{if } v \geq 0 \\
   0 & \text{if } v < 0 
   \end{cases}
   \]  \hspace{1cm} (1.6)

2. **Sigmoid Function** [-1,1]

   \[
   \varphi(v) = \begin{cases} 
   1 & \text{if } v > 0 \\
   0 & \text{if } v = 0 \\
   -1 & \text{if } v < 0 
   \end{cases}
   \]  \hspace{1cm} (1.7)

*Figure 1.2: Result of affine transformation of external bias input on neuron $k$ on output $u_k$.*

*Figure 1.3: Threshold function and a Sigmoid function for varying slope parameter ($a$).*
1.1.3 Network Architecture

Neurons of neural networks can be structured differently into two different classes of networks, depending on the desired learning algorithm that would be used to train the network.

1. Single-Layer Feedforward Networks

A single-layer network (shown in Fig 1.4), is strictly a feedforward type of networks, which projects inputs from layer of source nodes on output layer of neurons (computation nodes). The “single-layer” is a computation performed by a single layer, which is the output layer. No computation is made by the input layer of source nodes.

![Feedforward network with a single layer of neurons](image)
2. Multilayer Feedforward Networks

A multilayer network is a feedforward network with a layer of source nodes, one or more layer of hidden neurons, and the output layer of neurons (shown in Fig 1.5). Both the hidden layer and the output layer consist of computational nodes. The term “hidden” layer refers to the fact that such a layer is not directly visible from either the output or the input of the network, and they receive inputs from source nodes, or nodes of previous hidden layers and project outputs to upcoming nodes of either hidden layers or the output layer of the network.

![Multilayer Feedforward Network Diagram]

*Figure 1.5: Fully connected feedforward multi-layer network with one hidden layer and one output layer.*
1.1.4 Feedback Network

A dynamic system with feedback loop is a system whose input is influenced by some or all elements of the output from that same system. The feedback loop is characterized by operators of A and B. The behavior of a feedback system is controlled by the weight of \( \omega \). Equation 1.8 is referred to as a closed-loop operator, whose A has been made equivalent to the weight of \( \omega \) and B has been made equivalent to the unit-delay operator of \( z^{-1} \). The network output is given by equation 1.9.

\[
\frac{A}{1 - AB} = \frac{w}{1 - wz^{-1}} \tag{1.8}
\]

\[
y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n - l) \tag{1.9}
\]
Depending on the value of the feedforward weight $\omega$, which can be less than one, equal to one, or greater than one, the system’s time response would represent the system as being stable, linear divergent, or exponential divergent. This is illustrated in Fig 1.6.

woord less than one, $|\omega| < 1$, system is stable where output signal is exponentially convergent, i.e., the output is exponentially decreasing towards a lower value.

woord equal to one, $|\omega| = 1$, system is unstable where output signal is linearly divergent, i.e., the output is linearly increasing.

woord greater than one, $|\omega| > 1$, system is unstable where output signal is exponentially divergent, i.e., the output is exponentially increasing.

Figure 1.6: Time response of a system to a feedforward weight $\omega$
1.1.5 Knowledge Representation

Knowledge representation refers to how information must be represented for systems to utilize to carry out complex tasks. Data, a value measurement by itself, has no meaning without a clear context. For instance, speed is a value measurement, and a car traveling at a speed of \( x \) is the context, which then becomes a piece of information. When information is processed and analyzed, knowledge is formed. When knowledge and experience are combined, the result is wisdom. Data, information, knowledge and wisdom share a pyramid relationship shown in Fig 1.7 and Fig 1.8 [7].

![Figure 1.7: Data pyramid](image)

![Figure 1.8: Transition pathway of data pyramid](image)
There are two primary characteristics of knowledge representation. The first one is the information or set of information that has been fully developed or formulated. The second one is how the information will be converted into a set of instructions to be used subsequently. To achieve correct knowledge representation, information must be structured in a way that gives it purpose within the neural network. Knowledge representation is important for a neural network in such a way that is fundamental in how a model of neural network is supposed to learn and afterwards start generalizing. The learning process requires a set of input-output pairs, input signal and corresponding desired response, which is referred to as the training data.

1.2 Rosenblatt’s Perceptron

Rosenblatt’s perceptron is the simplest form of a neural network consisting of a single neuron with adjustable synaptic weights and bias. The single neuron is strictly used as a classifier for two classes of linearly separable patterns, which lie on opposite sides of a hyperplane. In Fig 1.9, a linearly separable pattern of two classes are classified using a Rosenblatt’ perceptron.

![Pair of linearly separable patterns](image)
The perceptron, as shown in Fig 1.10, with notations described in Table 1.2, is based on McCulloch-Pitts model of a neuron where summing nodes make up a linear combination of inputs $x_m$ from the synapses of neurons $w_m$, along with an external bias $b$, which results in an induced local field or activation potential $v$ (equation 1.10), which is then applied to a hard limiter $\phi(\cdot)$.

$$v = \sum_{i=1}^{m} w_i x_i + b$$  \hfill (1.10)

**Figure 1.10: Signal-flow graph of a perceptron**

**Table 1.2: Notation description of signal-flow graph of a perceptron**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_m$</td>
<td>input signals to neurons.</td>
</tr>
<tr>
<td>$w_m$</td>
<td>synaptic weights of neurons.</td>
</tr>
<tr>
<td>$b$</td>
<td>externally applied bias.</td>
</tr>
<tr>
<td>$v$</td>
<td>induced local field or activation potential of neurons.</td>
</tr>
<tr>
<td>$\phi(\cdot)$</td>
<td>activation function.</td>
</tr>
<tr>
<td>$y$</td>
<td>output signal of neurons.</td>
</tr>
</tbody>
</table>
The way a perceptron is used for a two-class pattern classification problem is by taking different input signals $x_m$ and categorizing them into one of two classes, $C_1$ or $C_2$, depending on the neuron’s output $y$ value, whether it is +1 or -1. Results of the pattern classification can be plotted on a two-dimensional plane to better understand the behavior of the perceptron as shown in Fig 1.1. The two classes on the two-dimensional plane are separated by a decision boundary, which is a hyperspace; the line that separates the two classes from each other.

![Figure 1.11: Hyperplane presentation of decision boundary for a two-class pattern classification](image)

**1.2.1 The Perception Convergence Theorem**

The perceptron is a linear classifier. If the training set is linearly separable, the perceptron is guaranteed to converge, and the input vectors will be classified correctly, and positive examples will be separated from the negative examples by a hyperplane. If the training set is not linearly separable, then it will be impossible to separate positive examples from negative ones by the hyperplane without the perception convergence theorem. The theorem, fixed-increment convergence, lets the subsets of training vectors be linearly separable and to converge by adapting each synaptic weight on an iteration-by-iteration basis.
1.3 Neural Networks Applications

Neural network classification, or forecasting applications, were widely implemented in industry, business and science fields with many different applications due to neural network’s properties of learning and generalization, regardless of the algorithm, whether it is backpropagation, radial basis functions, genetic, or Kohonen’s learning vector quantization. Neural networks have satisfied a diverse assortment of needs in three categories: pattern classification, prediction and optimization [8].

1.3.1 Conventional Applications

1- Classification:

Artificial neural networks can process many input signals and infer the non-linear relationships, allowing them to be used for image processing and character recognition. Image processing applications cover a wide range of industries, image recognition for smartphones, or to fight crimes by detecting wanted criminals from public crowds, as well as processing satellite imagery for agricultural purposes to distinguish different properties of different crops on a huge farming field. Character recognition applications are more relevant to banking and security industries where handwriting on cheques can be recognized and processed to digitized numbers. Legitimate handwritten signatures can also be distinguished from forged ones [8].

2- Forecasting:

With the ability to model unforeseen non-linear relationships from different input data, neural networks are capable at forecasting future data trends. There are many use cases for artificial neural networks forecasting in fields of sales, financial allocation between products, capacity utilization, and stock market. In marketing analysis, neural networks have been used to cut costs of marketing campaigns by forecasting unlikely customers from potential customers based on the pattern of their collected data sets. In the airline industry, neural networks have been used to forecast future pattern of the most demanded seats by customers based on previous data sets [8].
3- **Optimization:**

Optimization methods are used to maximize or minimize certain objective functions of nonlinear systems. Artificial neural networks help approximate objective functions, thus allowing the use of suitable techniques to develop polynomial equations to calculate the solution required to achieve optimization for the system [9].

### 1.3.2 Modern Applications

There are many new neural network applications that have been developed, and many more are yet to come. Three modern applications of neural networks are summarized. The first application is an emotion recognition system using a deep learning approach from audio–visual emotional big data [10]. Hence, the system incorporates the use of a deep learning approach, along with emotional big data, which is comprised of extremely large data sets of speech and video. This system consists of two convolutional neural networks. The first CNN takes a processed speech signal as an image of a Mel-spectrogram, and the second CNN takes the frames of a video signal as a picture. The output of both CNNs is then fused using two consecutive extreme learning machines. Next, the fused output is processed by a support vector machine for the final classification of emotion. This application of automatic emotion recognition has a great potential in various industries and can be employed to support the decision making of bigger AI systems embedded within smart cities in terms of instructiveness, health care, or even security.

The second application is for mapping oil spills from dual-polarized SAR images. This application uses neural network algorithm that forecasts the probability maps of oil spills. The signals are obtained by synthetic aperture radar, a form of radar used to create two or three-dimensional reconstruction of objects, in this case the sea, where it is being used to detect oil spill areas by using backscattering technique between oil and pixels of the sea. The issue is that noise can interfere with the signals obtained by SAR and thus decrease the accuracy. With this application, ANN removes the factor that decreases the accuracy of the collected signal and incorporates its algorithm to generate probability maps of oil spills. Such an application
increases the rate at which oil spills are detected, and the forecasting aids in preventing further damage to marine wildlife caused by oil spills [11].

The third application is deep convolutional neural networks for breast cancer screening [12]. This application is a computer-aided diagnosis system that is based on deep convolutional neural network. Such systems help radiologists to classify mammography mass lesions that may be cancerous. With transfer learning and fine-tuning strategies to train the CNN model, the overall system achieved a 98.23% accuracy when classifying new images. This kind of application has a great potential in saving lives of millions of women around the globe by detecting breast cancer, which is one of the most common invasive disease among women worldwide.
Chapter 2: Convolutional Neural Networks

2.1 What is a convolutional neural network?

A convolutional neuronal network is a feed-forward multilayer perceptron that is used for pattern classification and is inspired by natural visual perception mechanism of living creatures. Such networks usually consist of an input, an output, and multiple hidden layers. Hidden layers typically consist of convolutional, pooling, fully connected and normalization layers. Convolutional neural networks can be used for various applications, including image classification, object detection, object tracking, text detection and recognition, speech and natural language processing [13].

2.1.1 Convolutional Layer

One or more of these layers can be part of a convolutional neural network as a hidden layer, which applies a convolution operation to input before passing it to the next layer. What is meant by convolution operation is different from what is known in mathematics or engineering. The definition is strictly taken from image processing [14]. The input to the convolution layer is convolved with what is called a kernel, convolution matrix, or a filter.

As an illustration, consider the example shown in Fig 2.1. The input to the layer is 5x5 matrix, and the convolution matrix is 3x3. The 5x5 matrix includes 25 elements, and the 3x3 includes 9 elements. The first 9 elements of 5x5 make up a 3x3 matrix. This 3x3 matrix has its first $A_{11}$ multiplied by $B_{11}$ of the convolution matrix(kernel/filter). The process is repeated for every matrix, and the result is summed into $C_{11}$ of the convoluted matrix. Then the second set of 9 elements of input data 5x5 are taken to make up another 3x3 matrix, and the process is repeated until the convoluted matrix of 3x3 is filled with 9 elements. It is important to mention that the distinction between correlation and convolution is that the matrix filter used in convolution is flipped vertically.
2.1.2 Pooling Layer

Like the convolution layer, the input data is convoluted with a filter to form a convolution matrix. In a pooling layer, for example, a cluster of 3x3 is taken from the input data 5x5. This is illustrated in Fig 2.2, there are two pooling functions, average or max, depending on the function. The average value of all elements is then placed in $C_{11}$, or the max value of the cluster is placed into $C_{11}$. The process is repeated, depending on the function, from element $C_{11}$ until element $C_{33}$ for a 3x3 pooled region. It is important to note that the size of the pooled region and the stride must be specified. What is meant by stride is the cluster taken from input data, for example, 3x3 cluster taken from input data 5x5. For each pooled element, the cluster is shifted by one element along the row of the input data matrix. Once the end of the row is reached, the cluster goes back from where it started, and the cluster is now shifted by one element along the column until the whole matrix is pooled.

**Figure 2.2: Pooling operation**
2.1.3 Fully Connected Layer

After multiple layers of convolution and pooling, the output is received to a fully connected layer as input as shown in Fig 2.3. The fully connected layer is usually placed at the end of the CNN architecture, or somewhere intermediate.

![Fully Connected Layer Diagram](image)

*Figure 2.3: Fully connected layer*

This kind of layer is very similar to the feedforward multilayer perceptron. In this layer all the filtered, down sampled information is gathered for the CNN to start learning. Most of the weights are also located in this part of the network.

2.2 Convolutional Neural Network Learning

For any given task required to be carried out by CNN, certain parameters must be tuned respectively for the given task. Various mechanisms and techniques such as weight initialization and network regularization are implemented to optimize the neural network.
2.2.1 Weight Initialization

To train the CNN in a stable manner, a correct weight initialization is required; otherwise, this would lead to various difficulties in the learning process, depending on the learning method used. For example, in the case where a back-propagation method is used, incorrect weight initialization would result in either vanishing or exploding gradient problem. There are various techniques that can be used for weight initialization. These techniques are Gaussian Random Initialization and Uniform Random Initialization.

1. **Gaussian Random Initialization**
   
   This initialization technique is used for convolutional and fully connected layers of the CNN with random matrices of sampled elements from Gaussian distribution with zero mean and small standard deviation.

2. **Uniform Random Initialization**
   
   This initialization technique is used for convolutional and fully connected layers of the CNN with random matrices of sampled elements from uniform distribution with zero mean and small standard deviation.

2.2.2 Network Regularization

Convolutional neural networks are expected to learn and generalize; however, in some cases, networks fail to learn and generalize because they tend to over-fit on the training data during the learning process. The network is expected to learn rather than memorize to be able to generalize and better adapt to unseen data. Regularization is the set of approaches or techniques used to prevent over-fitting. Data augmentation, dropout, batch normalization, ensemble model averaging, and early stopping are examples of regularization techniques.

2.3 ConvNet Classifiers

ConvNet (convolutional neural network) can classify random data into categories. Data are categorized based on features, characteristics and qualities shared with other sets of data in respective categories. When compared with other classification algorithms, a ConvNet is easier to train because it requires fewer connections and parameters, as proven by results obtained
from ImageNet Large Scale Visual Recognition Competition (ILSVRC) [15]. In this competition, deep ConvNet achieved top-1 in terms of performance, performing considerably better than previously used algorithms. The deep ConvNet classified 1.2 million high-resolution images in the competition into 1000 different classes. ConvNets are used in various applications, classifying various ranges of data from images to texts etc. Several modern ConvNet classifiers will be discussed in this section.

1. **Face recognition based on convolution neural network**

   In [16], a face recognition method based on ConvNet is proposed, consisting of nine layers, three of which are convolution layers, two pooling layers, two fully connected layers and a SoftMax regression layer. The network was tested on Caffe, a deep learning framework capable of processing over 60 million images per day. Caffe framework trained and tested two different face databases. ORL face database consisted of a total of 400 pictures for 40 people. AR face data base contained a total of 2600 pictures for 100 people. After approximately 200 iterations, the network reached an accuracy rate of 90%. After 2000 iterations, it reached 97%. Finally, after 10,000 iterations the recognition rate for all face data sets of AR and ORL reached an accuracy of 99.82% and 99.78%, respectively. When compared with other methods used for AR database and ORL database, the ConvNet used topped the comparison due to excellent convergence and strong robustness.

2. **Hand gesture recognition based convolutional neural network**

   In [16], a hand gesture recognition algorithm based on ConvNet is proposed to recognize 10 gestures. The network was trained using 6000 hand gesture images and 1100 testing samples. The network achieved a 98% recognition rate, topping other algorithms frequently used in recognition algorithms.
3. Deep Convolutional Neural Networks for Twitter Sentiment Analysis

In [17], a deep learning system for sentiment analysis of tweets based on convolutional neural network is proposed. The system is used to classify a tweeter’s attitude towards something as negative, neutral or positive. This is illustrated in Fig 2.4.

![Diagram of sentiment analysis of deep convolutional neural network](image)

*Figure 2.4: Representation of sentiment analysis of deep convolutional neural network*

The architecture of the network includes a sentence matrix, single convolutional layer followed by a non-linearity layer, max pooling layer and SoftMax classification layer.
Chapter 3: Facially Expressed Emotions

The first scientific study on facial expression of emotion was published by Charles Darwin in 1872 [18]. His publication provided evidence that some emotions have universal facial expressions and explained why particular expressions occur with specific emotions, although some emotions can be expressed while concealing the true emotion. It was suggested that true feelings can be displayed despite efforts to conceal them. In addition, facial emotions are expressed with different groups of facial muscles, both voluntary and involuntary. Depending on the state of mind, involuntary muscles contract regardless of one’s will, thus revealing true emotion. An example described by the writer was that when a man is moderately angry or even enraged, he will be able to command the movements of his body, but muscles of the face are the least obedient to his will, therefore expressing his true emotions. This hypothesis was developed by Darwin, who further claimed that if one cannot deliberately contract a facial muscle, then he will not be able to prevent the involuntary muscle contraction. In [19], Dr. Paul Ekman mentions that he proved Darwin’s hypothesis to be correct by using the Facial Action Coding System, were less than 25% of his subjects could not deliberately control an involuntarily activated muscle contraction.
3.1 Facial Action Coding System

This system, as shown in Fig 3.1, is used to recognize and name human facial movements and to identify which group of muscle movements contributes to each emotion expressed by the face. Any move by any muscle is named for every different direction. This system was first developed by Carl-Herman Hjortsjo, a Swedish anatomist [20]. He paired 24 emotional conditions with sets of facial muscle movements. Some examples of the emotional conditions are: mournful, tormented, perplexed, anguished, scornful, and nauseated. Refering to Table 3.1, a smile could require up to 8 facial action units. An example is the combination of #16, the lower lip depressor, or #12, the greater zygomatic muscle. This system was later adopted by a group of researchers, including Dr. Paul Ekman, to become the standard way to systematically categorize physical expressions of emotions. With 46 different action units, the system’s usefulness was proven to not only psychologists and animators but also to experts from the commercial industry willing to make use of it for advertising, or even for enhancing shopping experiences.

Figure 3.1: Representation of facial action coding system
### Table 3.1: Single action units of facial action coding system

<table>
<thead>
<tr>
<th>AU No.</th>
<th>FACS Name</th>
<th>Muscular Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inner Brow Raiser</td>
<td>Frontalis, Pars Medialis</td>
</tr>
<tr>
<td>2</td>
<td>Outer Brow Raiser</td>
<td>Frontalis, Pars Lateralis</td>
</tr>
<tr>
<td>4</td>
<td>Brow Lowerer</td>
<td>Depressor Glabellae; Depressor Supercilli; Corrugator</td>
</tr>
<tr>
<td>5</td>
<td>Upper Lid Raiser</td>
<td>Levator Palpebrae Superioris</td>
</tr>
<tr>
<td>6</td>
<td>Cheek Raiser</td>
<td>Orbicularis Oculi, Pars Orbitalis</td>
</tr>
<tr>
<td>7</td>
<td>Lid Tightener</td>
<td>Orbicularis Oculi, Pars Palpebralis</td>
</tr>
<tr>
<td>8</td>
<td>Lips Toward Each Other</td>
<td>Orbicularis Oris</td>
</tr>
<tr>
<td>9</td>
<td>Nose Wrinkler</td>
<td>Levator Labii Superioris, Alaeque Nasi</td>
</tr>
<tr>
<td>10</td>
<td>Upper Lip Raiser</td>
<td>Levator Labii Superioris, Caput Infraorbitalis</td>
</tr>
<tr>
<td>11</td>
<td>Nasolabial Furrow</td>
<td>Zygomatic Minor Deepener</td>
</tr>
<tr>
<td>12</td>
<td>Lip Corner Puller</td>
<td>Zygomatic Major</td>
</tr>
<tr>
<td>13</td>
<td>Cheek puffer</td>
<td>Caninus</td>
</tr>
<tr>
<td>14</td>
<td>Dimpler</td>
<td>Buccinator</td>
</tr>
<tr>
<td>15</td>
<td>Lip Corner Depressor</td>
<td>Triangularis</td>
</tr>
<tr>
<td>16</td>
<td>Lower Lip Depressor</td>
<td>Depressor Labii</td>
</tr>
<tr>
<td>17</td>
<td>Chin Raiser</td>
<td>Mentalis</td>
</tr>
<tr>
<td>18</td>
<td>Lip Puckerer</td>
<td>Incisivii Labii Superioris; Incisivii Labii Inferioris</td>
</tr>
<tr>
<td>20</td>
<td>Lip Stretcher</td>
<td>Risorius</td>
</tr>
<tr>
<td>22</td>
<td>Lip Funneler</td>
<td>Orbicularis Oris</td>
</tr>
<tr>
<td>23</td>
<td>Lip Tightener</td>
<td>Orbicularis Oris</td>
</tr>
<tr>
<td>24</td>
<td>Lip Pressor</td>
<td>Orbicularis Oris</td>
</tr>
<tr>
<td>25</td>
<td>Lips Part</td>
<td>Depressor Labii, or Relaxation of Mentalis or Orbicularis Oris</td>
</tr>
<tr>
<td>26</td>
<td>Jaw Drop</td>
<td>Masseter; Temporal and Internal Pterygoid</td>
</tr>
<tr>
<td>27</td>
<td>Mouth Stretch</td>
<td>Ptergooids; Digastric</td>
</tr>
<tr>
<td>28</td>
<td>Lip suck</td>
<td>Orbicularis Oris</td>
</tr>
<tr>
<td>38</td>
<td>Nostril Dilator</td>
<td>Nasalis, Pars Alaris</td>
</tr>
<tr>
<td>39</td>
<td>Nostril Compressor</td>
<td>Nasalis, Pars Transversa and Depressor Septi Nasi</td>
</tr>
<tr>
<td>41</td>
<td>Lid Droop</td>
<td>Relaxation of Palpebrae Superioris</td>
</tr>
<tr>
<td>42</td>
<td>Slit</td>
<td>Orbicularis Oculi</td>
</tr>
<tr>
<td>43</td>
<td>Eyes Closed</td>
<td>Relaxation of Palpebrae Superioris</td>
</tr>
<tr>
<td>44</td>
<td>Squint</td>
<td>Orbicularis Oculi, Pars Palpebralis</td>
</tr>
<tr>
<td>45</td>
<td>Blink</td>
<td>Relaxation of Palpebrae and Contraction o Orbicularis oculi,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

24
3.2 Emotion Classification

3.2.1 The Six Basic Emotions:

Paul Ekman mentioned the six basic emotions: surprise, fear, anger, disgust, sadness, and happiness [21]. Each expressed emotion is described by three parts of the face, which are brows-forehead, eyes-lids, and lower face.

3.2.2 Plutchik’s Wheel of Emotions:

The Wheel of Emotions (shown in Fig 3.2) is an emotion classification system developed by Robert Plutchik in 1980 that visualizes eight basic emotions to help explain the Psychoevolutionary Theory of Emotion [22]. The theory considers emotion as not only a feeling state, but a complex chain of events that once simulated would evoke feelings resulting in actions.
For example, the emotion of happiness would evoke the person into smiling, or even laughing. The eight basic emotions consist of two-four sets, an emotion and its pair opposite: joy – sadness; trust – disgust; fear – anger; and anticipation – surprise.

Any two basic emotions next to each other may be combined to produce an emotion. The opposite emotion of the produced emotion is also found on the other side of the 2D emotion wheel and is also made up of two basic emotions that are next to each other. There are eight primary colors for eight primary emotions, so the color would vary depending on the emotion’s intensity.

3.2.3 Lovheim Cube of Emotion:

This is a three-dimensional model (shown in Fig 3.3) with signal substances that form three axis of the system and eight basic emotions taking place at eight different corners of the cube [23]. The three signal substances are dopamine, noradrenaline and serotonin.

![Figure 3.3: Lovheim cube of emotion](image)
3.3 Facial Expression Data Base

This is a database consisting of images or video clips of facially expressed emotions. Such data is essential for training, testing and validating algorithms used to develop expression recognition systems. Most databases are based on the eight basic emotions. Examples of facial expression databases are:

1. Extended Cohn-Kanade Dataset (CK+):
   
   This is a database made up of 123 subjects, 593 images, and 327 emotion labelled sequences. The images are mostly gray with a resolution of 640*490 [24].

2. Japanese Female Facial Expressions (JAFFE):
   
   This is a database made up of 10 subjects, 213 gray images of 256*256 resolution [1].

3. MMI Facial Expression Database:
   
   Is a database made up of 75 subjects, 2900 videos of 720 * 576 resolution [25].
Chapter 4: Principle Previous Work in Object Detection and Face Emotion Recognition

Object detection, like face detection is one of several sub-domains of computer vision. Most algorithms of face detection are based on deformable part models (DPM), suitable for identifying difficult objects, but considered to be too slow to meet real-time requirements. A photo of 640x480 resolution is processed in one second without multi-threading support [26]. Recently, deep convolutional neural networks (CNN) has demonstrated an outstanding performance in a variety of vision tasks. CNNs learn by automatic feature extraction from datasets used for training. To achieve better accuracy, for face emotion recognition, a larger dataset is required for training the deep convolutional neural network. A clean, consistent data is recommended for training the network, composed mainly of object of interest in different variations. To speed up the training duration, the use of a graphical processing unit is recommended. When compared to shallow neural networks, deep CNNs have a better performance and accuracy. Where a typical neural network may one or two hidden layers, a deep neural network has more than two hidden layers.

In [27], Kumar, Kant and Sanyal proposed a better approach to predict human emotions (frame by frame) using deep convolutional neural network and how emotion intensity expressed by a face changes from low level to high level emotion. In their proposed approach, a 9-layer convolutional neural network model is trained to classify 7 facially expressed emotions. The network model was trained using the FER-2013 database [2]. For emotion analysis of micro expressions, classified emotions are measured in percentages as an approach to assess the overall network performance.

In [28], Sang, Cuong and Ha, proposed a discriminative deep feature learning approach with dense convolutional networks (DenseNet) for facial emotion recognition. An auxiliary loss is employed in this proposal, to regulate the training process of neural networks. Thus, reducing the intra-class variation of deep features and enhancing the discriminative power of the learned networks. The experimental results show that their proposed approach achieves superior
performance in comparison with other recent state-of-the-art methods implemented using the FERC-2013 dataset.

In [29], Wang, Dong and Hu, proposed a deep cascade convolutional network that uses Fast Region-based Convolutional Neural Network (Fast R-CNN) for face detection. With this proposed method, the highest recall rate of true positive against false positive have been achieved on the challenging FDDB benchmark, thus, outperforming the current state-of-the-art methods [30]. The proposed method first utilizes the cascade CNN structure to reject background regions quickly and then uses Fast R-CNN object detector. The Fast R-CNN produces a bounding-box and classified each object being the human face.

In [31], Li Shang, Qin and Chen, proposed a technique to detect and recognize fish species under water using Fast R-CNN object detector. The Fast R-CNN improves the mean average precision (mAP) by 11.2% when compared to deformable parts model (DPM), achieving a mAP of 81.4%, and performing faster than the R-CNN object detector by 80 times. The proposed model is promising for automatic fish identification systems to help marine biologists estimate fish existence and quantity, which would effectively help understand oceanic geographical and biological environments.

In [32], Li, Zhang, Zhang, and their colleagues proposed a Faster R-CNN (Faster region-based Convolutional Neural Network) based method for facial expression recognition. In their proposal, initially the facial expression image was normalized to extract implicit features using the trainable convolutional kernel. A maximum poling technique was also used to reduce dimensions of extracted features, along with a SoftMax classifier and a regression layer which is used to classify the facial expression. Experimental results show the performance and the generalization ability of the Faster R-CNN for facial emotion recognition, resulting in a measure average precision (mAP) of 82%.
Chapter 5: Face Emotion Recognition Using Fast R-CNN

5.1 Deep Learning

Artificial Intelligence is divided into three main branches (shown in Fig 5.1); namely supervised learning, unsupervised learning, and reinforcement learning. Deep learning is a special approach in machine learning as it covers all three branches of artificial intelligence. Deep learning addresses problems such as knowledge representation, reasoning and planning [33].

![Figure 5.1: Deep learning, machine learning, and artificial intelligence](image)

A feature is a characteristic that might be used in solving a problem and is what makes a machine learning algorithm work. Although a feature may be extracted manually, there are no definite algorithms that can be used to extract specific features from data such as images or video. Deep learning is based on feature learning, which is a set of techniques that enables systems to automatically discover the representations needed for feature classification from raw data. In deep learning, tasks are divided and distributed into machine learning algorithms,
which are organized in consecutive layers that build up a network capable of carrying out
natural decision-making processes like the human brain [34]. This makes deep learning one of
the most powerful ways to recognize features automatically.

5.1.1 Deep Learning Toolbox for MATLAB

Formerly known as “Neural Network Toolbox”, this is a software framework for
designing, testing and implementing various types of neural networks to perform classification
and regression on various types of data such as images, time-series, and text data [35]. The
framework allows integration with other applications found on MATLAB’s add-on explorer.
Plotting tools provide visualization of network training, network features learned, and
activations of each layer, which can help to understand what the network learned during the
training process.

5.1.2 Preparing Data for Network Training

Neural networks are trained through data. Training data must be arranged according to
its respective label and prepared in a way that makes the data consistent. This consistency
ensures that the network would only be trained on relevant extracted features while
disregarding things like background, background objects, etc.

The data used to train the convolutional neural network in this thesis was obtained from
the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [3]. The database
consists of 24 actors expressing 8 different emotions, which are: neutral, calm, happy, sad,
angry fearful, disgust, and surprise. The data was prepared in the following steps:

1. Download the datasets of each actor from the RAVDESS website.
2. Rename each video file according to the actor’s number and emotion.
3. Create a folder for each emotion to contain every actor’s expression of that emotion.
4. Use MATLAB code to crop several pictures from each video. Then save the cropped
   pictures in the same emotion labelled folder.
5. Use another MATLAB code to detect each actor’s face. Then crop the face and resize
   the facial image to the desired size.
5.1.3 Training the Convolutional Neural Network – Transfer Learning – VGG-16

All images prepared should now become an image data store within MATLAB’s IDE for the training to take place successfully. Each image must be in its relevant emotion label file. The whole file would then be used for the training as an image datastore object. The image datastore object can then be split into two image datastores. One can be used for training and the other for validation. The split can be of any ratio depending on the training strategy.

VGG-16 is a pretrained ConvNet that consists of a 16-depth weight layer with very small (3 x 3) convolution filters used for object recognition [36]. The ConvNet has achieved state-of-the-art results on ImageNet Challenge 2014. A transfer learning technique is used on VGG-16, thus replacing its last three layers while maintaining the models’ original weights and layers. This way, the network would be able to converge and generalize by training with relatively little labelled data. In addition, training time is considered very little when compared to the time taken for training a network from scratch. Network’s learned features could be visualized using Deep Dream Technique through MATLAB as shown in Fig 5.2.
Figure 5.2: Extracted features of 8 channels (labels) of layers 3, layer 8 and layer 14 as visualized by Deep Dream feature visualization technique
5.2 Fast R-CNN – Object Detector

Fast region-based convolutional neural network is an object detector that classifies objects in an image using a deep learning approach. It uses an algorithm called Edge Boxes, which increases the computational efficiency of an object detector by generating efficient region proposals. An edge box is a simple box objectness score that measures the number of edges existing in a box minus the edges that are part of contours overlapping the box’s boundary [37]. The fast R-CNN consists of two-state detection algorithms. The first stage identifies a subset of regions in an image (using edge boxes), which might contain an object. The second stage classifies the object in each region [38]. In the first stage (feature extraction), the entire image is processed by convolutional layers to extract features and is processed by edge boxes to generate region proposals. A region of interest pool then combines convoluted features with region proposals as an input for the classification layers. The classification layers are part of the second stage (object classification). Using Fast R-CNN object detector for facial emotion recognition results in real-time classification of emotions.

5.2.1 Training Fast R-CNN – Transfer Learning – VGG-16

Training an object detector such as Fast R-CNN requires the use of an image labeler. An image labeler is an application through which a collection of images can have defined rectangular regions of interest (ROI) labels. Once each image within a collection has its ROI defined with a label, the labels can be exported to a MATLAB workspace. Once exported as a ground truth to workspace, a command such as `trainingData` would return the ground truth object as a table of training data.
The Fast R-CNN object detector trained for this thesis is also based on VGG-16 ConvNet. The transfer learning technique is also applied to the VGG-16 ConvNet. The training data used for this object detector is the same as the one used in the previous network in Chapter 4. The image datastore of the previous network was imported to image labeler, and a label was created for each emotion. The rectangular region of interest (ROI) label for each image must be specified. Since each image within this image datastore is a cropped image of a face expressing an emotion, the region was set to [1,1,224,224] for each image in its respective emotion label column. Fast R-CNN architecture is shown in Fig 5.3.

![Fast R-CNN Architecture](image.png)

*Figure 5.3: Fast R-CNN Architecture*

### 5.3 Difference between CNN and Fast R-CNN

The way Fast R-CNN differs from a typical CNN is that regions within an image are classified efficiently in real-time from input data such as image, video or live webcam. Whereas, CNN does not run in real-time, even with a high-end hardware resource a noticeable delay is still presented. In CNN, the whole image is processed for classification resulting in an output for object class only. There is no output on where the classified object was located on the processed image. Adding a bounding box regression for a typical CNN is possible. However, only one object at any given time.

---

1 Fast R-CNN architecture obtained from [https://www.mathworks.com/help/vision/ug/faster-r-cnn-basics.html](https://www.mathworks.com/help/vision/ug/faster-r-cnn-basics.html)
Chapter 6: Experimentation and Analysis

A series of experimentations were performed to enhance the classification of facially expressed emotions using convolutional neural network. Furthermore, the convergence of the deep network trained using Fast R-CNN is verified and validated for the required purpose. With Fast R-CNN, regions within an image will efficiently be classified in real-time.

6.1 JAFFE Database Experiment

The Japanese Female Facial Expression Database is a free of charge non-commercial database published in 1998, JAFFE database is an early standard database used by researchers to benchmark algorithms [1]. The database consists of 10 Japanese female models expressing 7 facial expressions in 213 images with a grayscale resolution of 256x256. Samples from the database are in Fig 6.1. The database resolution was altered from [256, 256, 1] to an RGB resolution of [224, 224, 3], thus suiting the input image layer of the pretrained network model, VGG-16 as specified in Table 6.1. The network model was transformed into a Fast R-CNN object detection network by adding a region-of-interest pooling layer and a bounding box regression layer. The network was then trained with training options of Table 6.2

<table>
<thead>
<tr>
<th>Data Properties</th>
<th>Training Set</th>
<th>Test Set</th>
<th>Emotions</th>
</tr>
</thead>
<tbody>
<tr>
<td>224x224pixels</td>
<td>80%</td>
<td>20%</td>
<td>Angry, Disgust, Fear, Happy,</td>
</tr>
<tr>
<td>RGB</td>
<td>170 images</td>
<td>42 images</td>
<td>Sad, Surprise, and Neutral</td>
</tr>
</tbody>
</table>

| Table 6.2: Fast R-CNN training options using JAFFE Database |
|-----------------|---------------|-----------|-----------------------------|
| Solver          | MiniBatchSize | InitialLearnRate | MaxEpochs |
| SGDM            | 1             | 1e-4       | 20                        |
| PositiveOverlapRange | [0.15 1]       |             |                           |
| NegativeOverlapRange | [0.1 0.15]     |             |                           |
Evaluating detection precision of the above trained network through MATLAB returns three arrays; average precision, recall, and precision as shown in Fig 6.2. For comparative analysis, a mean average precision (mAP) can be estimated from the average precision vector of the multiclass object detector, representing network’s recognition rate. Experimental results of Fast R-CNN object detector, a transformed VGG-16 network model trained on JAFFE database is benchmarked against experimental results obtained from [39]. Fast R-CNN object detector topped the performance as depicted in Table 6.3.

Figure 6.1: Image samples obtained from JAFFE database [1]
Detection Precision Analysis of Fast R-CNN (VGG-16) - JAFFE

Figure 6.2: Detection precision analysis of Fast R-CCN (VGG-16) trained on JAFFE

Table 6.3: JAFFE trained Fast R-CNN benchmarked against other published work

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST R-CNN</td>
<td>97.6</td>
</tr>
<tr>
<td>SBN-CNN [40]</td>
<td>95.24</td>
</tr>
<tr>
<td>RETRaIN [39]</td>
<td>88.09</td>
</tr>
<tr>
<td>CB-CNN [41]</td>
<td>86.74</td>
</tr>
<tr>
<td>CS-LBP [42]</td>
<td>86.68</td>
</tr>
<tr>
<td>LDP [43]</td>
<td>86.19</td>
</tr>
<tr>
<td>LDTP [44]</td>
<td>85.71</td>
</tr>
<tr>
<td>LBP [45]</td>
<td>85.23</td>
</tr>
<tr>
<td>FL-CNN [46]</td>
<td>84.00</td>
</tr>
<tr>
<td>LDN [47]</td>
<td>81.42</td>
</tr>
<tr>
<td>CN-CNN [41]</td>
<td>53.57</td>
</tr>
</tbody>
</table>
6.2 FERC-2013 Database Experiment

To contribute over existing efforts, the deep network was first trained using a database made available for Facial Expression Recognition Challenge [2]. FER-2013 database specifications are in Table 6.4, and image samples are in Fig 6.3. Alterations were made to both the VGG-16 model and the database to make them compatible for training. The image input layer of VGG-16 model was altered from [224, 224, 3] to [48, 48, 3] along with the last three layers of the model. The image dataset was also altered from grayscale [48, 48, 1] to [48, 48, 3]. In addition, the pretrained network model has been transformed into a Fast R-CNN object detection network by adding a region-of-interest pooling layer and a bounding box regression layer. The network was then trained with training options of Table 6.5.

Table 6.4 Database properties of facial expression recognition challenge 2013

<table>
<thead>
<tr>
<th>Data Properties</th>
<th>Training Set</th>
<th>Emotions</th>
</tr>
</thead>
<tbody>
<tr>
<td>48x48 pixels</td>
<td>28,709 examples</td>
<td>Angry, Disgust, Fear, Happy,</td>
</tr>
<tr>
<td>Grayscale, 1 bit</td>
<td></td>
<td>Sad, Surprise, and Neutral</td>
</tr>
</tbody>
</table>

Table 6.5: Fast R-CNN training options using FER-2013 database

<table>
<thead>
<tr>
<th>Solver</th>
<th>MiniBatchSize</th>
<th>InitialLearnRate</th>
<th>MaxEpochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGDM</td>
<td>1</td>
<td>1e-4</td>
<td>1</td>
</tr>
<tr>
<td>PositiveOverlapRange</td>
<td>[0.2 1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NegativeOverlapRange</td>
<td>[0 0.2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmallestImageDimension</td>
<td>400 ~ 600</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unfortunately, after several training attempts with different training options, training duration per session required more than 50 hours, the Fast R-CNN object detector did not converge on the given training dataset. A validation test was made for the trained network on the same trained dataset which resulted in a zero precision. The reason for such poor performance could be that the network may have been altered incorrectly. Another experiment was attempted, this time without altering the VGG-16 network model using the altered database properties in Table 6.6, and training options in Table 6.7. With total training duration of more than 170 hours, were the network was trained seven times with different training options. All seven trained object detectors were tested on 20% of the dataset which were not part of the training set. For comparative analysis, the average recognition rate of the network is benchmarked against other results obtained from [48], and [49] in Table 6.8. Fast R-CNN object detector (VGG-16) is shown to be the least performing method when implemented on the FER-2013 database. AlexNet based CNN is shown to surpass other methods in the table as CNN networks classify whole images, unlike object detectors which first identify regions before classifying them into different classes.

Assuming Fast R-CNN’s architecture was changed from VGG-16 to VGG-19 or AlexNet, the object detector would still not surpass the methods in terms of accuracy, especially when compared to the AlexNet based CNN. A results table was generated when the trained Fast R-CNN was used to classify the test set. The table includes information such as bounding box

Figure 6.3: Image samples obtained from FER 2013 [2]
region, emotion label, and score for every image in the test set. Fast R-CNN performed poorly on FER-2013 because the trained object detector was able to identify regions for classification in 586 of 5,741 images, thus the object detector classified only 10.2% of the test set and the performance analysis is depicted in Fig 6.4.

Table 6.6: Database properties of altered FER-2013 Database

<table>
<thead>
<tr>
<th>Data Properties</th>
<th>Training Set</th>
<th>Test Set</th>
<th>Emotions</th>
</tr>
</thead>
<tbody>
<tr>
<td>224x224 pixels</td>
<td>80%</td>
<td>20%</td>
<td>Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral</td>
</tr>
<tr>
<td>RGB</td>
<td>22,968 images</td>
<td>5,741 images</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7: Fast R-CNN training options using FER-2013 Database

<table>
<thead>
<tr>
<th>Solver</th>
<th>MiniBatchSize</th>
<th>InitialLearnRate</th>
<th>MaxEpochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGDM</td>
<td>1</td>
<td>1e-4</td>
<td>20</td>
</tr>
<tr>
<td><strong>PositiveOverlapRange</strong></td>
<td>[0.15 1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NegativeOverlapRange</strong></td>
<td>[0.1 0.15]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.4: Detection precision analysis of Fast R-CNN (VGG-16) trained on FER-2013

Figure 6.4: Detection precision analysis of Fast R-CNN (VGG-16) trained on FER-2013
6.3 RAVDESS Database Experiment

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), consists of 7356 files, videos of 24 professional actors expressing 8 different emotions in 3 seconds short videos [3]. After experimenting with various facial emotion libraries, the RAVDESS was one of the most recently published databases, consisted of high quality RGB videos, real-time emotion expression with a white background. Training a neural network requires a clean consistent data to avoid extracting unnecessary features during the learning process.

Processing raw data in video format to images of cropped faces expressing emotions resulted in 4,254 unique facial expression for 8 different emotions using processing configuration of Table 6.9, resulted in the following processed images shown in Fig 6.5. The cropped images were resized to a dimension of [224, 224, 3], matching the image input layer of the existing pretrained network model. Transfer learning technique was performed on the pretrained network model by replacing the last three layers, and then the network was transformed into a Fast R-CNN object detection network. VGG-16 network model was trained more than ten times on different training options. VGG-19 and ResNet-50 network models were trained on the same training options. Table 6.10 includes training options of the three network models.

Table 6.8: FER-2013 trained Fast R-CNN benchmarked against other published work

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN (AlexNet) [49]</td>
<td>61.1</td>
</tr>
<tr>
<td>Net B [48]</td>
<td>60.91</td>
</tr>
<tr>
<td>Net B_DAL [48]</td>
<td>58.33</td>
</tr>
<tr>
<td>Net B_DAL_MSE [48]</td>
<td>58.15</td>
</tr>
<tr>
<td>Fast R-CNN (VGG-16)</td>
<td>30.19</td>
</tr>
</tbody>
</table>
Table 6.9: RAVDESS database processing configuration

<table>
<thead>
<tr>
<th></th>
<th>Raw Data</th>
<th>Processed Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension</strong></td>
<td>1280x720 pixels</td>
<td>224x224 pixels</td>
</tr>
<tr>
<td><strong>Data Type</strong></td>
<td>1 video</td>
<td>6 pictures</td>
</tr>
<tr>
<td><strong>Data Quantity</strong></td>
<td>1,440 videos</td>
<td>8,680 pictures</td>
</tr>
<tr>
<td><strong>Data Composition</strong></td>
<td>Full Composition</td>
<td>Cropped Face Composition</td>
</tr>
</tbody>
</table>

![Figure 6.5: Samples of processed images from RAVDESS database [3].](image)

Table 6.10: Fast R-CNN training options using RAVDESS database

<table>
<thead>
<tr>
<th></th>
<th>Solver</th>
<th>Mini-Batch Size</th>
<th>Initial Learn Rate</th>
<th>Max Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Positive Overlap Range</strong></td>
<td>SGDM</td>
<td>1</td>
<td>1e-4</td>
<td>1</td>
</tr>
<tr>
<td><strong>Negative Overlap Range</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VGG-16 was trained more than ten times on varying initial learn rates and max epochs. When VGG-16 was not able to detect any facial expression, the initial learn rate value was decreased to a smaller number and max epochs was increased. If the trained network was classifying multiple object classes on overlapping regions, the max epochs value was decreased. This process was repeated more than twenty times until the desired result was achieved using
the training option found in Table 6.11 to determine the initial-learn rate and max epochs for each network.

Table 6.11: Fast R-CNN training options using RAVDESS database for three different network models.

<table>
<thead>
<tr>
<th>Network Model</th>
<th>Initial Learn Rate</th>
<th>Max Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>0.5e-5</td>
<td>15</td>
</tr>
<tr>
<td>VGG-19</td>
<td>1e-4</td>
<td>30</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>1e-4</td>
<td>30</td>
</tr>
</tbody>
</table>

Once the network model has been trained on the 50% of the processed data from Table 6.9 with training options of Table 6.10 and Table 6.11, a validation test is made to measure the mean average precision of each trained network model. The validation test was made on the other 50% of the processed data on three different trained network architecture (VGG-16, VGG-19 and ResNet-50) which were transformed into a Fast R-CNN object detector. In Table 6.12, recognition rates of VGG-16, VGG-19, and ResNet-50 are benchmarked against result obtained from [50] of a Support Vector Machine trained on the same database.

With 96.3% recognition rate, the SVM method topped the comparison table, as the best scoring Fast R-CNN based method had a recognition rate of 79.18%. SVM method scored a higher recognition rate than Fast R-CNN based methods as it was evaluated using a single parameter, predicted label against the predefined label. The Fast R-CNN object detector was evaluated using three parameters, boxes, scores and labels. Boxes refer to the predicted object region, scores refer to the confidence level of the emotion out of 100%, and labels refer to the class of the label. Moreover, the SVM in [50] is incapable of placing rectangle region of interest over classified objects nor output a probability on predicted class label. SVMs are linear classifiers, they require the use of feature extractors to produce sparse vectors to be used for training. Unlike CNNs, which automatically extract features for learning. With such limitation, SVMs are found at a disadvantage when compared with object detectors such as Fast R-CNN.

Average precision is evaluated using precision and recall, results are shown in Fig 6.6. Precision measures how accurate a prediction is made by the trained network in a score format. Recall measures trained network ability to detect all positive regions, which is the
overlapping region the predicted boundary against the ground truth. Average precision computes average precision for recall value on a scale of 0 to 1. Average miss rate computes miss rate against false positives per-image. The results are shown in Fig 6.7.

Table 6.12: RAVDESS trained Fast R-CNN benchmarked against other published work

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM [50]</td>
<td>96.3</td>
</tr>
<tr>
<td>VGG-19</td>
<td>79.18</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>74.6</td>
</tr>
<tr>
<td>VGG-16</td>
<td>60.9</td>
</tr>
</tbody>
</table>

VGG-16 network model is shown to be performing poorly when compared to VGG-19 and ResNet-50. This is due to different training options, especially the initial-learn rate and max epoch values. All network models were initially tested by classifying real facially expressed emotions which were not part of the training set to evaluate network’s generalization. Unexpectedly, VGG-16 performed far better due to training it just enough and stopping the training process earlier than VGG-19 and ResNet-50. VGG-19 and ResNet-50 have been over-
trained due to high max epochs value. An over-trained network would detect multiple overlapping regions and classify them into multiple object classes with different confident scores (shown in Fig 6.8).

**Figure 6.7:** Average miss rate analysis on three different network models

**Figure 6.8:** An undertrained network model, trained with max epochs of 3 (left) and an overtrained network model trained with max epochs of 30. (right)
6.4 Live Deployment Experiment

The trained Fast R-CNN object detector (VGG-16) for classifying facially expressed emotions has been tested on 100 different people to validate network’s generalization.

**Validation Conditions:**

1. Only detect facially expressed emotions.
2. Input from a live webcam running in real-time.
3. Expressed facial emotion must truly match network’s classified emotion.

6.4.1 Live Verification

The trained Fast R-CNN object detector must only detect facially expressed emotions. This is verified by the pooled region of interest placed over the detected object. If the rectangle region of interest positioned on regions containing faces, then the network is verified to be able to detect faces, and therefore detecting facially expressed emotions.

When the webcam’s resolution was set to low, 320x240, and at a distance less than 30 cm from the face, the trained object detector started detecting faces, and classifying facially expressed emotions. However, when the webcam’s resolution was set to high, or at a distance greater than 30 cm from the face, the trained object detector did not place any region of interest over any face visible, therefore, no facially expressed emotions were detected.

Best results were obtained with a webcam resolution of 320x240 because the network input layer is 224x224. Also, the distance of 30 cm or less between the webcam and the face yields a similar face size of that used during the training of the object detector (shown in Fig 6.9).

**Recommended Deployment Configuration:**

- Moderate Lighting.
- Webcam resolution of 320x240.
- Distance between webcam and face is 20 – 30 cm.
6.4.2 Live Validation Test

The network model was tested on 100 different people who tried to express all eight facial emotions and are shown in Fig 6.10. The network performed impressively in recognizing happy and surprised faces, as they are the easiest to express intentionally. Some participants found it difficult to express negative emotions willingly, also the network had issues recognizing neutral expressions. On average, the network model was able to recognize four different facially expressed emotion per participant out of eight emotions. In Fig 6.11, the Fast R-CNN was able to recognize 5 facially expressed emotions. When the participant attempted expressing a fearful face, the Fast R-CNN detected two object classes, fear and surprise. The detector was more confident that it was a fearful expression rather than a surprised one.
Figure 6.11: Trained Fast R-CNN classifying 5 expressed emotions correctly.
6.4.3 Rapidly Prototyped Deployment System

Through Fab Academy program, an automated system has been designed and fabricated, where the trained object detector would be deployed. The program is a fast-paced, hands-on learning experience program focused on qualifying students to self-fabricate and rapidly prototype projects and machines. Fab Labs such as Fab Lab UAE, is a workshop where the program is offered which includes 3D printing machines, CNC machines, laser cutters, soldering stations and all the required machines used in self-fabrication. The program is led by Professor Neil Gershenfeld, director of Massachusetts Institute of Technology’s Center of Bits and Atoms.

The components making up the proposed system mainly consist of an ATMEGA328P controller, stepper motor and driver, time-of-flight sensor, webcam and laptop. The system also combines wooden structure with 3D printed material Fig 6.12. The system’s algorithm is designed to enter operational mode once a user stands 50 cm away from the system. Once activated, a platform consisting of a webcam and time-of-flight sensor is moved linearly on a vertical axis until it is positioned just above the user’s head, where the distance detected by the time-of-flight sensor is more than 50 cm. The platform is then moved downwards until it is positioned just in front of the user’s face. This way, the system can automatically adjust the webcam platform for users of different heights. Once in front of the user’s face, a serial command is sent from the controller to MATLAB to start classifying user’s facial expression in real-time, results can be shown on the laptop. The system then goes back to its initial position once the user steps away, entering a sleep mode until another user stands in front of the system.
Figure 6.12: Rapidly prototyped system platform for Fast R-CNN deployment
Chapter 7: Conclusion and Future Work

7.1 Conclusion

In this thesis, the use of convolutional neural networks for facial emotion recognition is considered. CNNs are faster in classifying images when compared to deformable parts model (DPM), but do not classify in real-time like the fast region-based convolutional neural network object detector (Fast R-CNN). Several experimental attempts were made to verify and validate both networks convergence and generalization. Best results were obtained using network models trained on RAVDESS library, the network performed impressively by classifying facially expressed emotions in real-time through a webcam device for people of different ages and ethnicities.

When a CNN model is transformed into an object detector like Fast R-CNN, measures typically used to evaluate an object detector’s performance, such as average precision or average miss rate, are concluded to be insufficient. Although a Fast R-CNN model achieves high precision and low miss-rate, the network may have been over trained. During the training process, the model at some point stops generalizing and starts learning statistical noise. When tested, the model would detect multiple overlaying regions and classify them into multiple classes with varying scores. The proposed solution is to train several network models with different training options. The models are then tested on similar data obtained from different libraries, or on data obtained from the real-world. Models which performed poorly were discarded, and training options were iterated until the desired performance is achieved.

In conclusion, object detection algorithms which use deep learning and convolutional neural network such as Fast R-CNN performed impressively when trained to classify facially expressed emotions. The network model able to classify positive emotions effectively such as happy and surprised, achieving a 100% and 94% respectively on participants who were not part of the training set.
7.2 Future Work

Further experimentations could be made by processing more than 300,000 images rather than 4,340 from RAVDESS library videos. In addition, emotion labels could include emotional intensities. The network model would then be able to differ between a happy and a very happy facial expression. The network model was trained using a dataset which is invariant in terms of size, viewpoint, and illumination. As a result, when a user’s face resolution is smaller than that of the training set, due to standing far away from the webcam, the users facial emotion expression is not detected by the network. Also, incorrect classification occurred when the user is not facing the webcam straight forward, or his face was not illuminated by ambient light correctly. Issues like these could be resolved by training the network with images of varying sizes, viewpoints and illumination. Such experimentations could result in a better performing network model.

Future work could involve using Mask Region-based Convolutional Neural Network (Mask R-CNN), which uses bounding box detection to predicts object mask, segmentation of pixels into an object. Such object detectors are currently used to detect objects such as pedestrian, street lanes, light poles, etc. Training Mask R-CNN to also detect facially expressed emotions might be useful for future robots which could be deployed in various sectors such as customer service and hospitality.
References


Appendix

Deep Learning Toolbox – Functions

There are 139 functions within this toolbox making up 6 different categories. Only functions and objects related to the scope of this thesis will be highlighted from this toolbox, and other features of MATLAB.

- **Character Array:**

  The first step in writing a code for deep learning in MATLAB for any purpose is identifying the folder which contains the data that would be used for training or validation, classification or regression by using a character array. The character array would contain the path to folder’s location.

- **Cell Array:**

  The second step is using a cell array to index the classes that would be used in the code. For example, a network is to be trained on the eight basic emotions. The cell array of the classes would be as follows, `categories = {'surprise', 'fear', 'anger', 'sadness', 'sadness', 'happiness'}.`

- **imageDatastore:**

  Creating an `imageDatastore` object is an important step when it comes to train a new network or validate an existing one. This function creates an object within MATLAB’s workspace of processed and imported images according to prespecified properties.

- **splitEachLabel:**

  Is a function used to split `imageDatastore` object labels by proportions. Usually for deep learning, the datastore into two sets of files, one would be used for training and the other for validation when training the network.
SeriesNetwork:

A series network is a neural network for deep learning with layers arranged consecutively starting with an input layer and an output layer. Such network maybe loaded as pretrained network, or imported from other deep learning frameworks, libraries, or ecosystems like Caffe, Keras or Open Neural Network Exchange.

trainingOptions

This function specifies options for training deep learning neural network.

solverName: determines the solver to be used for network training. There are three solvers which are as follows. Stochastic Gradient Descent with Momentum (SGDM) Optimizer, RMSProp Optimizer and Adaptive Moment Estimation (ADAM) Optimizer.

Stochastic Gradient Descent:

This optimizing algorithm updates the network’s weights and biases to minimize loss function with small steps towards the negative gradient of the loss. The small steps refer to the mini-batch, which is a subset of the training set. Once each step is evaluated, it is considered an iteration towards minimizing the loss function. An epoch is when the training algorithm fully passes over all mini-batch (training subset).

Stochastic Gradient Descent with Momentum:

Is the same optimizing algorithm but with momentum. The momentum reduces any oscillation occurring to the algorithm along its steepest decent towards optimum.

Root q Square Propagation:

RMSPROP uses different learning rates for different parameters thus adapting to the loss function being optimized during the network training. The previous optimizing algorithm uses only a single learning rate for all parameters.
Adam:
Is similar to RMSProp but with momentum which reduces oscillation during the network training process.

- **MaxEpoch**: Maximum number of times training data will be fully passed by the solver during network training.

- **MiniBatchSize**: Size of mini-batch used for each training iteration. Mini-batch is a subset used to evaluate the gradient of loss function and update network’s weights.

- **ValidationFrequency**: Frequency of network validation per iteration.

- **ExecutionEnvironment**: hardware resource used to train the network, it can be computer’s Central Processing Unit (CPU), Graphical Processing Unit (GPU), multiple-GPU, or parallel which is a local pool or compute cluster.

- **Layer**
Network layer defines the architecture of neural network for deep learning which can include different deep learning layers such as the following:

1. **Input Layers**
2. **Convolution and Fully Connected Layers**
3. **Sequence Layers**
4. **Activation Layers**
5. **Normalization, Dropout, and Cropping Layers**
6. **Pooling and Unpooling Layers**
7. **Combination Layers**
8. **Object Detection Layers**
9. **Output Layers**
For transfer learning, an existing network must have last layers of its network replaced with same or similar layers, this allows to use a network which can classify 1000 different categories to classify x categories (in our case 6 or 8 categories).

- **trainNetwork**

  Is the command used to train a network for both deep learning classification and regression problem. The image datastore, layers and training options are included during the training process. The training process time may vary depending on the number of layers used, amount of data used to train the network with, along with other options determined under trainingOptions command. The time may take seconds, to minutes, hours or even weeks depending on the hardware used to train the network.

- **Classify**

  After training process is done, the trained network can be saved to be used for classification whenever needed. The network would be used with the command classify to predict class label for the input data.
MATLAB Codes

Video Cropper Tool

The following MATLAB code processes video data into several image data. Were each actor’s video of each expressed emotion is captured into 3 images. The usefulness of this code lies within its capability to process all video files within a single folder.

```matlab
workingDir = tempname;
mkdir(workingDir);
mkdir(workingDir,'images');
dinfo = dir('*.mp4'); % list folder content (video)

for K = 1 : length(dinfo) % repeat for all video in folder
    thisvideo = dinfo(K).name;

    targetVideo = VideoReader(thisvideo,'CurrentTime',1); % read video at 1s
    a = 1;
    img = readFrame(targetVideo); % read video frame
    filename = [dinfo(K).name sprintf('%d.jpg',a)];
    fullname = fullfile(workingDir,'images',filename)
    imwrite(img,fullname) % save video frame at 1s

    targetVideo = VideoReader(thisvideo,'CurrentTime',2); % read video at 2s
    a = 2;
    img = readFrame(targetVideo); % read video frame
    filename = [dinfo(K).name sprintf('%d.jpg',a)];
    fullname = fullfile(workingDir,'images',filename)
    imwrite(img,fullname) % save video frame at 2s

    targetVideo = VideoReader(thisvideo,'CurrentTime',3); % read video at 3s
    a = 3;
    img = readFrame(targetVideo); % read video frame
    filename = [dinfo(K).name sprintf('%d.jpg',a)];
    fullname = fullfile(workingDir,'images',filename)
    imwrite(img,fullname) % save video frame at 2s

end
```
**Crop and Resize Tool**

This MATLAB code processes saved images resulted from the previous code in a way each detected face within each image is recognized, cropped and resized to the desired pixel as per the neural network’s specification. This code is also able to process all images within a folder.

```matlab
OutputFolder = 'C:\Users\almarri\Desktop\resultTEST';
dinfo = dir('*.*'); % list folder content (images)

% Cascade object detector to detect people’s faces
faceDetector = vision.CascadeObjectDetector;

% Specifications to draw bounding box over object of interest
shapeInserter = vision.ShapeInserter('BorderColor','Custom','CustomBorderColor',[0 255 255]);

for K = 1 : length(dinfo)
    thisimage = dinfo(K).name;
    Img = imread(thisimage);

    % Detect faces in saved image
    bbox = step(faceDetector, Img);

    % Draw boxes around detected faces and display results
    I_faces = step(shapeInserter, Img, int32(bbox));

    % Saved Cropped Face
    croppedFace = imcrop(I_faces, [bbox])

    % Resize image to 224x224 pixels
    resizedImage = imresize(croppedFace, [224, 224]);
    imwrite(resizedImage, fullfile(OutputFolder, thisimage));
end
```
Train Convolutional Neural Network

clear all;
close all;
clc;

matlabroot='C:\Users\a\Desktop\megaLibrary\224\New'

rootFolder = 'C:\Users\a\Desktop\megaLibrary\224\New' %database of training images
categories = {'ANG', 'DIS', 'HAP', 'NEU', 'SAD', 'SUR', 'FEA', 'calm'}

imds = imageDatastore(fullfile(rootFolder, categories), 'LabelSource', 'foldernames');
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7, 'randomized');

net = vgg16;

layersTransfer = net.Layers(1:end-3);
% numClasses = numel(categories(imdsTrain.Labels));
layers = [
    layersTransfer
    fullyConnectedLayer(8, 'WeightLearnRateFactor', 10, 'BiasLearnRateFactor', 10)
    softmaxLayer
    classificationLayer];

options = trainingOptions('sgdm', ...
    'MiniBatchSize', 16, ...
    'MaxEpochs', 30, ...
    'InitialLearnRate', 1e-6, ...
    'ValidationData', imdsValidation, ...
    'ValidationFrequency', 30, ...
    'Verbose', false, ...
    'Plots', 'training-progress')

sa3emNet = trainNetwork(imdsTrain, layers, options);

YPred = classify(sa3emNet, imdsValidation);
accuracy = mean(YPred == imdsValidation.Labels)

save sa2emNet
Network Validation Tool

Testing the trained network to validate network’s generalization and accuracy. The code first loads the network to MATLAB’s IDE workspace, then uses human face detector through webcam to detect any face. Once the object detector recognizes a human face, the image is then cropped and resized to 224 x 224 pixels. The cropped face image is then classified according to the trained emotion labels.

clear all;
clc;
% Load trained network
load sa3emNet
% Load and configure webcam
cam = 0
cam = webcam
cam.Resolution = '1280x720'
preview(cam)
cont = 1
% Capture from webcam
while cont == 1
outputFolder='C:\Users\a\Desktop\aNet\test'
img = snapshot(cam);
imshow(img)
% Load cascade object detector for human faces
faceDetector = vision.CascadeObjectDetector;
shapeInserter = vision.ShapeInserter('BorderColor','Custom',
'CustomBorderColor',[0 255 255]);
I = img
imshow(I);shg;
bbox = step(faceDetector, I);
% Draw boxes around detected faces and display results
I_faces = step(shapeInserter, I, int32(bbox));
imshow(I_faces), title('Detected faces');
% Saved Cropped Face
test = imcrop(I_faces,[bbox])
imwrite(test, fullfile(outputFolder, 'test.jpg'))
resizedImg=imresize(test, [224, 224]);
imshow(resizedImg)
imwrite(resizedImg, fullfile(outputFolder, 'test.jpg'));
resizedImg = imageDatastore(fullfile(outputFolder))
% Classify cropped face image according to trained network
[ZPred,scores] = classify(sa3emNet,resizedImg,'ExecutionEnvironment','gpu')
disp('ANG DIS HAP NEU SAD SUR FEA calm')

closePreview(cam)
end
Auto Deep Dream Image Feature Extractor Tool

The following MATLAB code allows network feature visualization using deep dream. Such tool is useful understanding and diagnosing network behavior. The code uses Deep Dream feature visualization technique which synthesizes images that strongly activate network layers [51]. This way image features learned by the network can be visualized and understood.

```matlab
% Load trained network
load sa2emNet

% All network layers
netLayer = 
{'conv1_1','relu1_1','conv1_2','relu1_2','pool1','conv2_1','relu2_1','conv2_2','relu2_2','pool2','conv3_1','relu3_1','conv3_2','relu3_2','conv3_3','relu3_3','pool3','conv4_1','relu4_1','conv4_2','relu4_2','conv4_3','relu4_3','pool4','conv5_1','relu5_1','conv5_2','relu5_2','conv5_3','relu5_3','pool5','fc6','relu6','drop6','fc7','relu7','drop7','fc','softmax'}

% Number of labels (8 emotions)
channels = 1:8;

for j = 1:39

% Visualize network features using deep dream
I = deepDreamImage(sa2emNet,char(netLayer(j)),channels, ... 
    'PyramidLevels',1, ... 
    'Verbose',1, ... 
    'ExecutionEnvironment','gpu')

h=figure

    for i = 1:8
        subplot(3,2,i)
        imshow(I(:,:,,:,i))
    end
```

66