iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL
FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 Issue: VII Month of publication: July 2021

DOI: https://doi.org/10.22214/ijraset.2021.36459

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VII July 2021- Available at www.ijraset.com

Application of Machine Learning for Emotion Classification

Mr. Shubham Babhulkar[1], Prof. M. S. Chaudhari[2]
[1, 2]*In the Department of computer Science and Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, India*

Abstract: In this paper we propose an implement a general convolutional neural network (CNN) building framework for designing real-time CNNs. We validate our models by creat- ing a real-time vision system which accomplishes the tasks of face detection, gender classification and emotion classification simultaneously in one blended step using our proposed CNN architecture. After presenting the details of the training pro- cedure setup we proceed to evaluate on standard benchmark sets. We report accuracies of 96% in the IMDB gender dataset and 66% in the FER-2013 emotion dataset. Along with this we also introduced the very recent real-time enabled guided back- propagation visualization technique. Guided back-propagation uncovers the dynamics of the weight changes and evaluates the learned features. We argue that the careful implementation of modern CNN architectures, the use of the current regu- larization methods and the visualization of previously hidden features are necessary in order to reduce the gap between slow performances and real-time architectures. Our system has been validated by its deployment on a Care-O-bot 3 robot used during RoboCup@Home competitions. All our code, demos and pre- trained architectures have been released under an open-source license in our public repository.

Keywords: Machine learning, Emotion Classification, CNN.

I. INTRODUCTION

The success of service robotics decisively depends on a smooth robot to user interaction. Thus, a robot should be able to extract information just from the face of its user, e.g. identify the emotional state or deduce gender. Interpret- ing correctly any of these elements using machine learning (ML) techniques has proven to be complicated due the high variability of the samples within each task [4]. This leads to models with millions of parameters trained under thousands of samples [3]. Furthermore, the human accuracy for classifying an image of a face in one of 7 different emotions is 65% ± 5% [4]. One can observe the difficulty of this task by trying to manually classify the FER-2013 dataset images in Figure 1 within the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}. In spite of these difficulties, robot platforms oriented to attend and solve household tasks require facial expressions systems that are robust and computationally efficient. More- over, the state-of-the-art methods in image-related tasks such as image classification [1] and object detection are all based on Convolutional Neural Networks (CNNs). These tasks require CNN architectures with millions of parameters; therefore, their deployment in robot platforms and real-time systems

In this paper we propose an implement a general CNN building framework for designing real-time CNNs. The implementations have been validated in a real-time facial expression system that provides face-detection, gender classification and that achieves human-level performance when classifying emotions. This system has been deployed in a care-O-bot 3 robot, and has been extended for general robot emotion dataset [4]. platforms and the RoboCup@Home competition challenges. Furthermore, CNNs are used as black-boxes and often their learned features remain hidden, making it complicated to establish a balance between their classification accuracy and unnecessary parameters. Therefore, we implemented a real- time visualization of the guided-gradient back-propagation proposed by Springenberg [11] in order to validate the features learned by the CNN.

II. AIM & OBJECTIVE

Commonly used CNNs for feature extraction include a set of fully connected layers at the end. Fully connected layers tend to contain most of the parameters in a CNN. Specifically, VGG16 [10] contains approximately 90% of all its parameters in their last fully connected layers. Recent architectures such as Inception V3 [12], reduced the amount of parameters in their last layers by including a Global Average Pooling operation. Global Average Pooling reduces each feature map into a scalar value by taking the average over all elements in the feature map. The average operation forces the network to extract global features from the input image. Modern CNN architectures such as Xception [1] leverage from the combination of two of the most successful experimental assumptions in CNNs: the use of residual modules [6] and depth-wise separable convolutions [2]. Depth-wise separable convolutions reduce further the amount of parameters by separating the processes of feature extraction and combination within a convolutional layer.

Furthermore, the state-of-the-art model for the FER2-2013 dataset is based on CNN trained with square hinged loss [13]. This model achieved an accuracy of 71% [4] using approximately 5 million parameters. In this architecture 98% of all parameters are located in the last fully connected layers. The second-best methods presented in [4] achieved an accuracy of 66% using an ensemble of CNNs.

III. LITERATURE SURVEY

A. Historical Study

For a recommender system, sentiment analysis has been proven to be a valuable technique. A recommender system aims to predict the preference to an item of a target user. Mainstream recommender systems work on explicit data set. For example, collaborative filtering works on the rating matrix, and content-based filtering works on the meta-data of the items.

1) Data in content-based filtering.
2) The CyberEmotions project, for instance, recently identified the role of negative emotions in driving social networks discussions.
3) Experimental Evidence of Massive-Scale Emotional Contagion through Social Networks.
a) *Affectiva:* Affectiva is an emotion measurement technology company that grew out of MIT's Media Lab which has developed a way for computers to recognize human emotions based on facial cues or physiological responses. Among the commercial applications, this emotion recognition technology is used to help brands improve their advertising and marketing messages. Another major application has been in political polling.[1] In 2011, the company partnered with Millward Brown, which is itself a part of the Kantar Group, the market research, insight and consultancy division of WPP plc, a London-based public company.
b) *Emotient API:* Emotient is an image recognition technology. Emotient allows face emotions, and other features to be recognized and identified in photos and images. The Emotient API allows developers to access and integrate the functionality of Emotient with other applications and to create new applications. Public documentation is not available.
c) *Visge Technologies AB:* Is a private company that produces computer vision software for face tracking (head tracking, face detection, eye tracking, face recognition) and face analysis (age detection, emotion recognition, gender detection), along with a special business unit in automotive industry. The primary product of Visage Technologies is a multiplatform software development kit visageSDK.

IV. PROPOSED SYSTEM

We have proposed a system in which we have three modules as follows:

1) Image analysis
2) Data Set extraction
3) Train the classifier

Use Machine Learning to describe the emotion on a face. We will be using a dataset of more than 33000 images to train our machine to differentiate between different emotions.

In Input processing we will Develop a program that will detect the real time face through webcam on our laptop and it will through cascading will point out the face and eyes and lips.

In image analysis we will analyze the face features and then the output will be the predicted emotion that the face is depicting.

We must train our program to automatically detect the emotions of the face.

For this we will take 80% of the images to train the machine and test our program with 20% images for errors.

As the program keeps on learning the output will be more accurate as it learns.

A. Feature Extraction Function

You need a function that can transform a small patch of image into a vector. If you only reshape the 2D patch into a 1D vector, it is still a function and correct. But in practice it contains several stages. This function is the feature extraction function and should extract features in a wise manner, followed by a normalization.

B. Train the Classifier

Every detection system needs a classifier that looks at your vector and decides if it is the deal or not. In case of face detection, the classifier looks for faces. The main issues are to choose your classifier and set the parameters in a way that you get reasonable results.

V. SYSTEM ARCHITECTURE

A. CNN Architecture

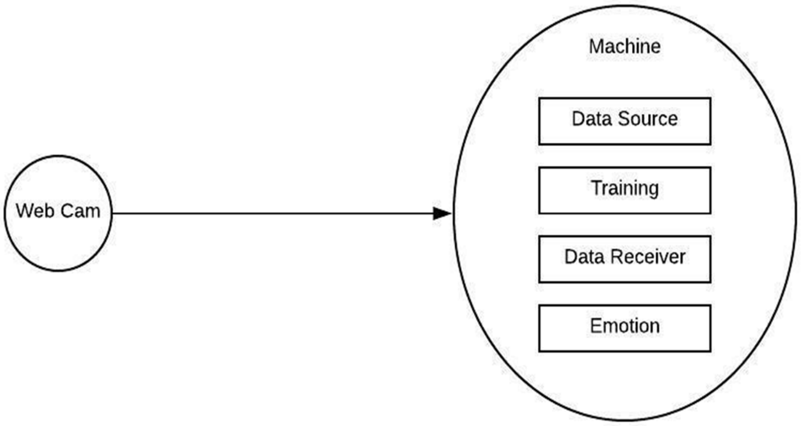


Fig 1

Our Program uses webcam which is a system software and it is used to detect the face.
The CNN we made trains from the dataset from Kaggle and then detects emotion from the face. The machine takes data source and then trains from the dataset provided which contains faces and the emotion attached to it. After the training when a face is given to the program data receiver receives it and compares it with the previous examples.
Comparison gives us the closest emotion that is detected.

B. System Flowchart

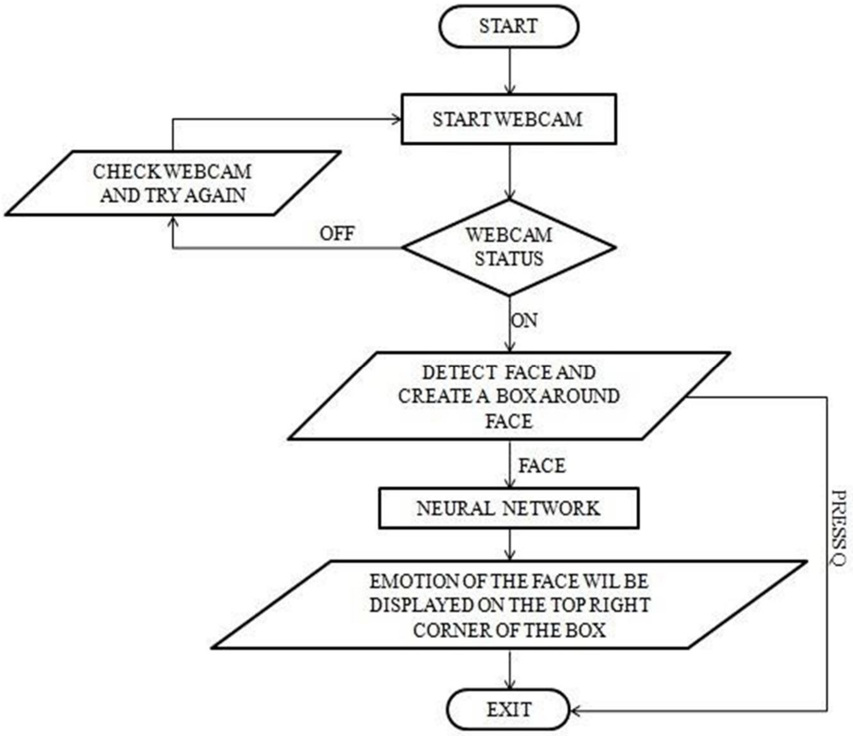
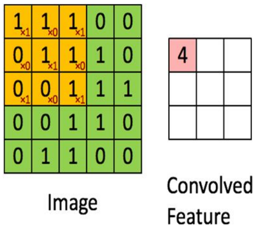

START
START WEBCAM
CHECK WEBCAM AND TRY AGAIN
OFF
WEBCAM STATUS
ON
DETECT FACE AND CREATE A BOX AROUND FACE
FACE
NEURAL NETWORK
PRESS Q
EMOTION OF THE FACE WIL BE DISPLAYED ON THE TOP RIGHT CORNER OF THE BOX
EXIT

Fig 2

1) *Convolution Layer:* Convolution is the mathematical operation that is used in image processing to filter signal, find pattern in signal etc. All neurons in this layer perform convolution on inputs. The most important parameter in a convolutional neuron is the filter size. We shall slide convolution filter over whole input image to calculate this output across the image and here we slide our window by 1 pixel at time this number is called Stride. Typically we use more than 1 filter in one convolution layer



$1_{\times1}$	$1_{\times0}$	$1_{\times1}$	0	0
$0_{\times0}$	$1_{\times1}$	$1_{\times0}$	1	0
$0_{\times1}$	$0_{\times0}$	$1_{\times1}$	1	1
0	0	1	1	0
0	1	1	0	0

4		

2) *Pooling Layer:* Pooling layer is mostly used immediately after the convolutional layer to reduce the spatial size(only width and height, not depth). This reduces the number of parameters, hence computation is reduced. Also, less number of parameters avoid over fitting. The most common form of pooling is Max pooling where we take a filter of size 3X3 and apply the maximum operation over the 3X3 sized part of the image.

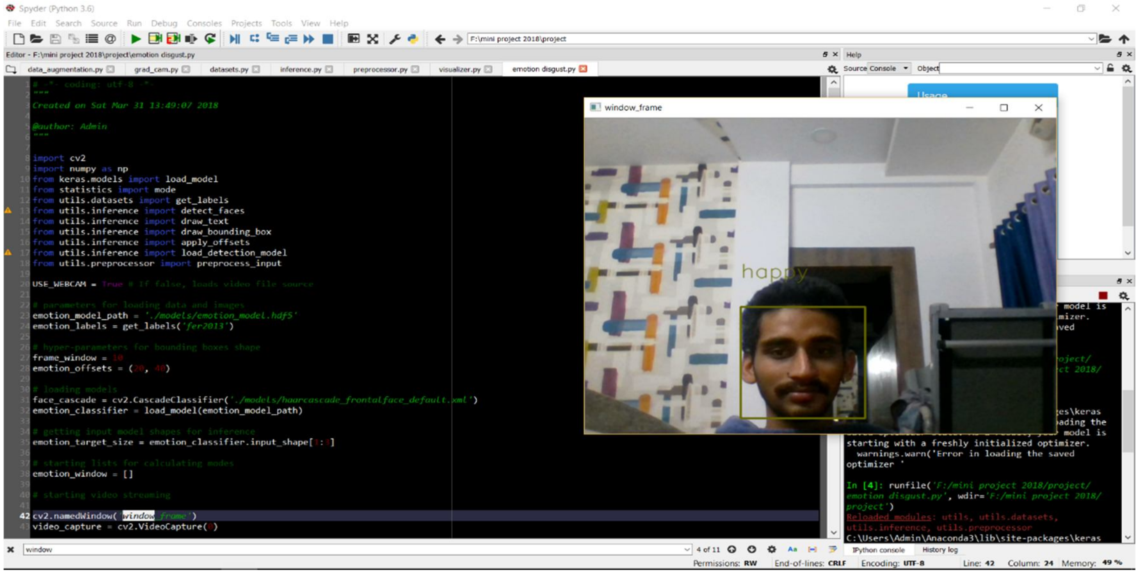
3) *Fully Connected Layer:* If each neuron in a layer receives input from all the neurons in the previous layer, then this layer is called fully connected layer. The output of this layer is computed by matrix multiplication followed by bias offset.

VI. RESULTS

Results of the real-time emotion classification task in un- seen faces can be observed in Figure 5. Our complete real- time pipeline including: face detection, emotion and gender classification have been fully integrated in our Care-O-bot 3 robot.

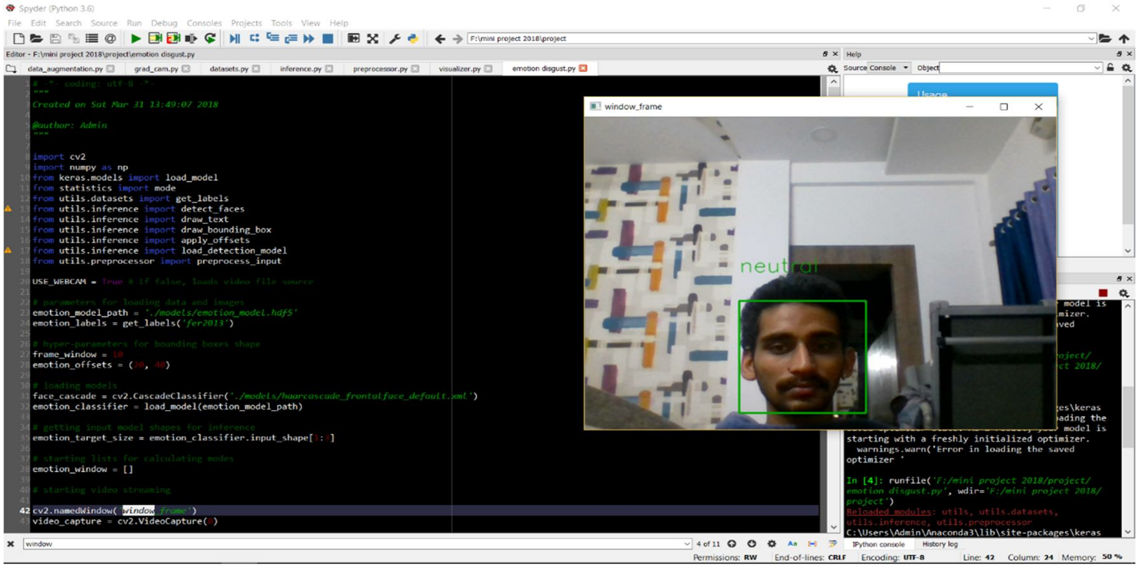
An example of our complete pipeline can be seen in Figure 6 in which we provide emotion and gender classification. In Figure 7 we provide the confusion matrix results of our emotion classification mini-Xception model. We can observe several common misclassifications such as predicting “sad” instead of “fear” and predicting “angry” instead “disgust”.

A comparison of the learned features between several emo- tions and both of our proposed models can be observed in Figure 8. The white areas in figure 8b correspond to the pixel values that activate a selected neuron in our last convolution layer. The selected neuron was always selected in accordance to the highest activation. We can observe that the CNN learned to get activated by considering features such as the frown, the teeth, the eyebrows and the widening of one’s eyes, and that each feature remains constant within the same class. These results reassure that the CNN learned to interpret understand- able human-like features, that provide generalizable elements. These interpretable results have helped us understand several common misclassification such as persons with glasses being classified as “angry”. This happens since the label “angry” is highly activated when it believes a person is frowning and frowning features get confused with darker glass frames**.**



```python
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 31 13:49:07 2018

@author: Admin
"""

import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

USE_WEBCAM = True # If false, loads video file source

# parameters for loading data and images
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
```

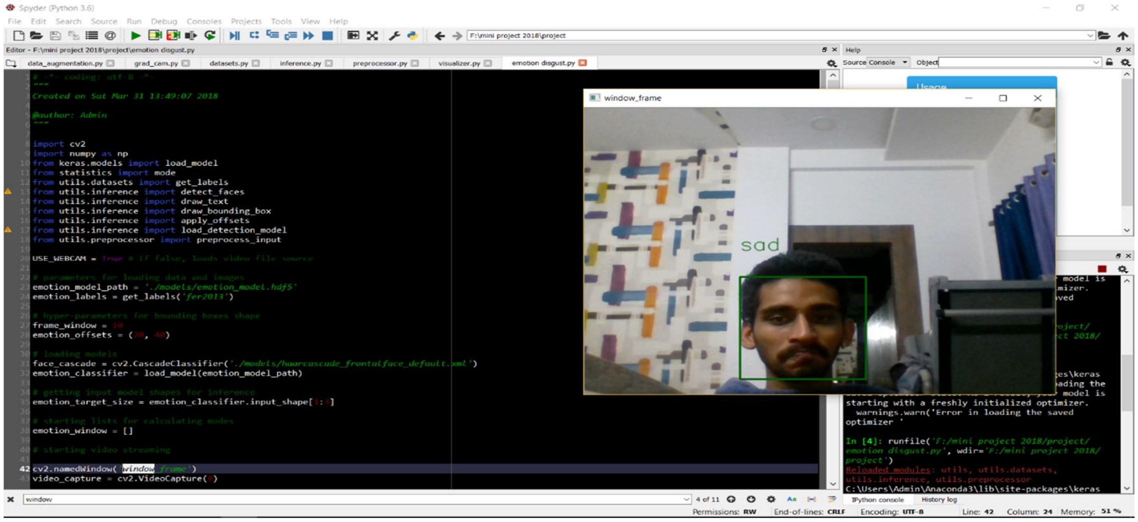
```
model is
mizer.
ved

oject/
ct 2018/

es\keras
ading the
model is
starting with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved
optimizer '

In [4]: runfile('F:/mini project 2018/project/
emotion disgust.py', wdir='F:/mini project 2018/
project')
Reloaded modules: utils, utils.datasets,
utils.inference, utils.preprocessor
C:\Users\Admin\Anaconda3\lib\site-packages\keras
```



```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 31 13:49:07 2018

@author: Admin
"""

import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

USE_WEBCAM = True # If false, loads video file source

# parameters for loading data and images
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
```

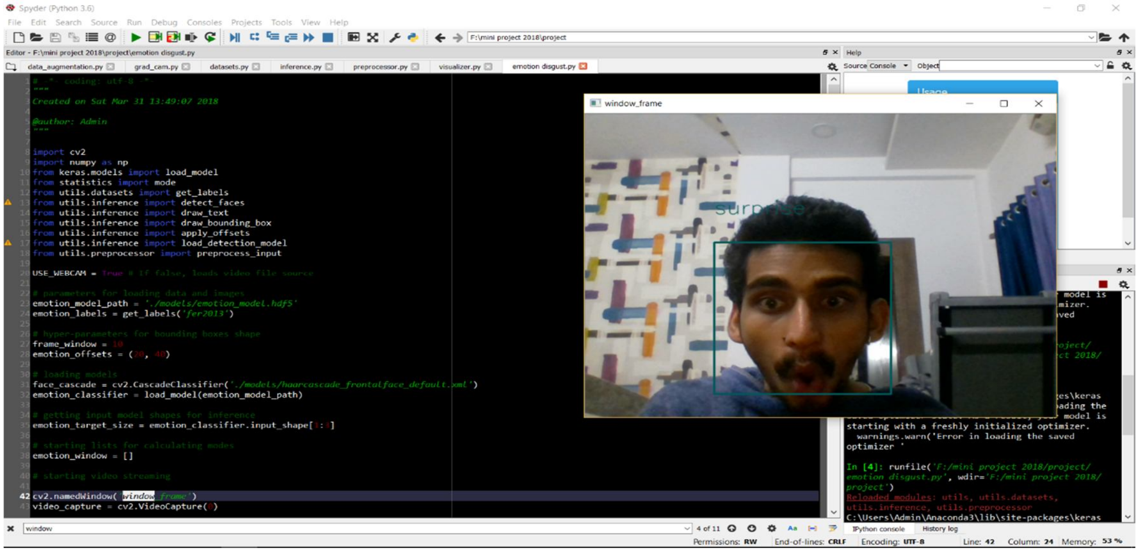
```
starting with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved
optimizer '

In [4]: runfile('F:/mini project 2018/project/
emotion disgust.py', wdir='F:/mini project 2018/
project')
Reloaded modules: utils, utils.datasets,
utils.inference, utils.preprocessor
C:\Users\Admin\Anaconda3\lib\site-packages\keras
```

window



```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 31 13:49:07 2018

@author: Admin
"""

import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

USE_WEBCAM = True # If false, loads video file source

# parameters for loading data and images
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
```

```
starting with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved
optimizer '

In [4]: runfile('F:/mini project 2018/project/
emotion disgust.py', wdir='F:/mini project 2018/
project')
Reloaded modules: utils, utils.datasets,
utils.inference, utils.preprocessor
C:\Users\Admin\Anaconda3\lib\site-packages\keras
```



```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 31 13:49:07 2018

@author: Admin
"""

import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

USE_WEBCAM = True # If false, loads video file source

# parameters for loading data and images
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
```

```
starting with a freshly initialized optimizer.
  warnings.warn('Error in loading the saved
optimizer '

In [4]: runfile('F:/mini project 2018/project/
emotion disgust.py', wdir='F:/mini project 2018/
project')
Reloaded modules: utils, utils.datasets,
utils.inference, utils.preprocessor
C:\Users\Admin\Anaconda3\lib\site-packages\keras
```

REFERENCES

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions. CoRR, abs/1610.02357, 2016.

[2] Andrew G. Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.

[3] Dario Amodei et al. Deep speech 2: End-to-end speech recognition in on three machine learning contests, 2013

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 315–323, 2011.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[6] Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Interna- tional Conference on Machine Learning, pages 448–456, 2015.

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic