Journal of Advanced Research in Instrumentation and Control Engineering
Volume 8, Issue 1&2 - 2021, Pg. No. 12-17
Peer Reviewed Journal

Article

Face Recognition Thourgh Simulation

K Thamizhmaran

Assistant Professor, Department of Electronics Communication Engineering , Government College of Engineering, Bodinayakanur, Tamil Nadu, India.

I N F O

E-mail Id:
tamil10_happy@rediff.com
How to cite this article:
Thamizhmaran K. Face Recognition Thourgh Simulation. *J Adv Res Instru Control Engg* 2021; 8(1&2): 12-17.

Date of Submission: 2021-04-03
Date of Acceptance: 2021-04-23

A B S T R A C T

The use of machines in society has increased widely in the last decades. Nowadays, machines are used in many different industries. As their exposure with humans increase, the interaction also must become smoother and more natural. To achieve this, machines must be provided with a capability that let them understand the surrounding environment. When machines are referred, this term comprises to computers and robots. A distinction between both is that robots involve interaction abilities into a more advanced extent since their design involves some degree of autonomy. When machines can appreciate their surroundings, some sort of machine perception has been developed. Humans use their senses to gain insights about their environment. Therefore, machine perception aims to mimic human senses to interact with their environment. Nowadays, machines have several ways to capture their environment state trough cameras and sensors. Hence, using this information with suitable algorithms allow to generate machine perception. In the last years, the use of Deep Learning algorithms has been proven to be phenomenally successful in this regard. For instance, Jeremy Howard showed on his Brussels 2014 TEDx's talk how computers trained using deep learning techniques were able to achieve some amazing tasks. These tasks include the ability to learn Chinese language, to recognize objects in images and to help on medical diagnosis. Affective computing claims that emotion detection is necessary for machines to better serve their purpose. For example, the uses of robots in areas such as elderly care or as porters in hospitals demand a deep understanding of the environment. Facial emotions deliver information about the subject's inner state. If a machine can obtain a sequence of facial images, then the use of deep learning techniques would help machines to be aware of their interlocutor's mood. In this context, deep learning has the potential to become a key factor to build better interaction between humans and machines, while providing machines with self-awareness about its human peers, and how to improve its communication with natural intelligence.

Keywords: Image, Network Loss, Face Recognition, Python

Introduction

Also, a focus on some parameters and its effect on the model's accuracy prediction was performed. These parameters were chosen because their influence over the network's behaviour:

- Network los

- Learning rate
- Dropout
- Optimizers

As described by Rosalind Picard, "... affective computing is the kind of computing that relates to, arises from, or influences emotions or other affective phenomena". Affective computing aims to include emotions on the design of technologies since they are an essential part of tasks that define the human experience: communication, learning, and decision-making.

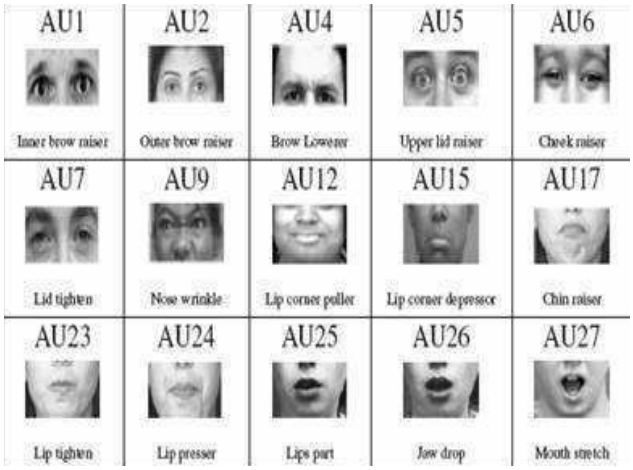

Figure 1.Facial action units

One of the main foundations behind affective computing is that without emotions, humans would not properly function as rational decision-making beings. Some research show that there is no such a thing as "pure reason". Emotions are involved in decision-making since a fully scientifically approach would turn into an extreme time-consuming process, not suitable for daily tasks. Research around this topic have shown that the brain does not test each probable option, but it is biased by emotion to quickly decide. An emotion is defined as a class of qualities that is intrinsically connected to the motor system. When a particular emotional state is triggered, the motor system will provide the corresponding set of instructions to reproduce the modulations connected to that class. So far, emotions' importance has been addressed without taking human interaction into consideration. Empathy is a human capacity that makes us aware and provides us with understanding about what other beings might be experiencing from their current's position. Moreover, empathy allows us to build close relationships and strong communities. Therefore, it is fundamental towards a pro-social behaviour, which includes social interaction and perception. Thus, it is especially important for affective computing to develop ways to accurately measure these modulations since they can lead to a better understanding of a subject's emotional state. The two main ways to do so is by detecting facial and vocal emotions.

Facial Emotion Recognition

The work by psychologist Paul Ekman has become fundamental to the development of this area. Nowadays, most of the face emotion recognition studies are based on Ekman's Facial Action Coding System. This system provides a mapping between facial muscles and an emotion space. The main purpose behind this system is to classify human facial movements based on their facial appearance. This classification was first developed by Carl-Herman Hjorth, who is a Swedish anatomist. However, this mapping might face some challenges. For instance, gestures involved on facial emotions can be faked by actors. The absence of a real motivation behind the emotion does not prevent humans to fake it. For instance, an experiment describes when a patient, who is half paralyzed is asked to smile. When it is asked, only a side of the mouth rises. However, when the patient is exposed to a joke, both sides of the mouth raise. Hence, different paths to transmit an emotion depend on the origin and nature of a particular emotion. With respect to computers, many possibilities arise to provide them with capabilities to express and recognize emotions. Nowadays, it is possible to mimic Ekman's facial units. This will provide computer with graphical faces that provide a more natural interaction. When it comes to recognition, computers have been able to recognize some facial categories: happiness, surprise, anger and disgust.

Convolution Neural Networks

The study of the visual cortex is closely related to the development of the convolution neural networks. Back in 1968, Hubel and Wiesel presented a study focused on the receptive fields of the monkey's visual cortex. This study was relevant because of the striate cortex (primary visual cortex) architecture description and the way that cells are arranged on it.

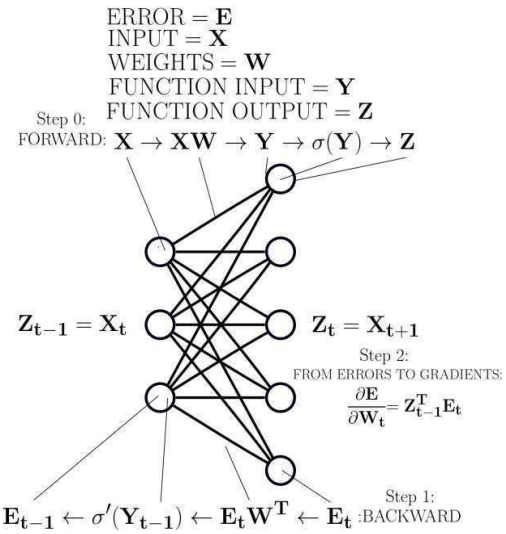

Figure 2.Convolutions Neural Network

N mathematics, a convolution operation is defined to mix two functions. An analogy commonly used is that this operation works as a filter. A kernel filters everything that is not important for the feature map, only focusing on some specific information.

To Execute this Operation, Two Elements are Needed:

- The input data
- The convolution filter (kernel)

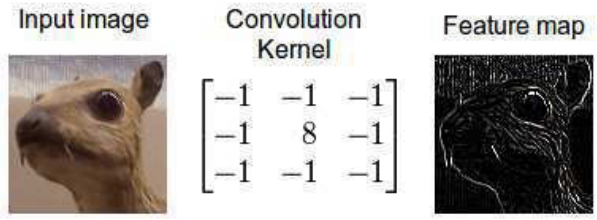

Figure 4.Convolution operation

The result of this operation is a feature map. Figure 3 provides a graphical explanation about the mechanics on the convolutional operation. The number of feature maps (output channels) provides the neural network with a capacity to learn features. Each channel is independent since they aim to learn each a new feature from the image that is being convoluted. Finally, the type of padding defines the algorithm to be used when performing the convolution. There is a special case on the input's edges. One type of padding will discard input's border, since there is no more input next to it that can be scanned. On the other hand, the other padding will complete the input with a value of 0. It is a matter of reducing parameters while convoluting.

2 Spatial sub-sampling

Spatial sub-sampling is an operation also known as pooling. The operation consists of reducing the values of a given area to one. So, it reduces the influence of the feature position on the 4. Figure -Image sequence for subject S130 from CK+. Subject displays the surprise emotion.

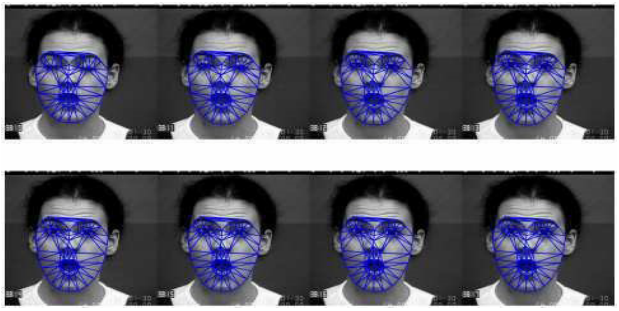

Figure 4.Image sequence for subject S130 from CK+

Feature map by diminishing its spatial resolution. This is done by choosing the most responsive pixel after a convolution operation. There are two types of polling: average and maximum. The average one computes the mean on the defined area, while the maximum only selects the highest value on the area. The area size can lead to reduction on the prediction performance, if the value is too large. It proceeds in a similar fashion.

Implementation Framework

Nowadays, many frameworks have been developed for deep learning. Some of the most popular ones include libraries such as: Caffe, Theano and Tensor Flow. Also, implementing a framework from scratch using a programming language was never considered. It would have been out of scope since it requires a big amount of effort, and the duration of such a project usually takes years. The use of Python as the front-end API on all these frameworks shows that it is the preferred language for machine learning. Usually, Python is combined with a programming language that provides support for low level operations such as: C or C++, to act on the back end.

Layer's Specification

The convolutional layer contains the following hyper parameter set; there is no rule of thumb to determine values for these parameters. Most of the time they are tuned by trial and error:

- Number of output channels
- Size of the kernel's receptive field
- Shape of the kernel's stride
- Type of padding operation

Program

```
import glob

from shutil import copyfile

emotions = ["neutral","angry","contempt","disgust","
fear","happy","surprise","sadness"]

participants=glob.glob("source emotion*")

for x in participants:

part="%s" %x[-4:]

for sessions in glob.glob("%s\\*"%x):

for files in glob.glob("%s\\*"%sessions):

current_session=files[20:-30]

file=open(files,'r')

emotion=int(float(file.readline ()))

sourcefile_emotion=glob.glob("source_im-
age\\%s\\%s\\*"%(part,current_session))[-1]

sourcefile_neutral=glob.glob("source_im-
age\\%s\\%s\\*"%(part,current_sesssion))[0]

dest_neut="sorted_set\\neutral\\%s"%sourcefile_neu-
tral[25:]

dest_emot="sorted_set\\%s\\5s"%(emotion[emotion],-
```

```
sourcefile_emotion[25:])
copyfile(sourcefile_neutral,dest_neut)
copyfile(sourcefile_emotion,dest_emot)
import cv2
import glob
faceDet = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
faceDet_two = cv2.CascadeClassifier("haarcascade_frontalface_alt2.xml")
faceDet_three = cv2.CascadeClassifier("haarcascade_frontalface_alt.xml")
faceDet_four = cv2.CascadeClassifier("haarcascade_frontalface_alt_tree.xml")
emotions = ["neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"]
def detect_faces(emotion):
files = glob.glob("sorted_set\\%s\\*" %emotion) #Get list of all images with emotion
filenumber = 0
for f in files:
frame = cv2.imread(f) #Open image
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #Convert image to grayscale
#Detect face using 4 different classifiers
face = faceDet.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
face_two = faceDet_two.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
face_three = faceDet_three.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
face_four = faceDet_four.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
#Go over detected faces, stop at first detected face, return empty if no face.
if len(face) == 1:
facefeatures = face
elif len(face_two) == 1:
facefeatures = face_two
elif len(face_three) == 1:
facefeatures = face_three
elif len(face_four) == 1:
facefeatures = face_four
else:
facefeatures = ""
#Cut and save face
for (x, y, w, h) in facefeatures: #get coordinates and size of rectangle containing face
print ("face found in file: %s" %f)
gray = gray[y:y+h, x:x+w] #Cut the frame to size
try:
out = cv2.resize(gray, (350, 350)) #Resize face so all images have same size
cv2.imwrite ("C:\\Users\\admin\\Desktop\\face.jpg" %(emotion, filenumber), out) #Write image
except:
pass #If error, pass file
filenumber += 1 #Increment image number
for emotion in emotions:
detect_faces(emotion) #Call functiona
import cv2
import glob
import random
import numpy as np
emotions = ["neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"] #Emotion list
fishface = cv2.createFisherFaceRecognizer() #Initialize fisher face classifier
data = {}
def get_files(emotion): #Define function to get file list, randomly shuffle it and split 80/20
files = glob.glob("C:\\Users\\admin\\Desktop\\face.jpg" %emotion)
random.shuffle(files)
training = files[:int(len(files)*0.8)] #get first 80% of file list
prediction = files[-int(len(files)*0.2):] #get last 20% of file list
return training, prediction
def make_sets():
training_data = []
training_labels = []
```

```
prediction_data = []
prediction_labels = []
for emotion in emotions:
training, prediction = get_files(emotion)
#Append data to training and prediction list, and generate labels 0-7
for item in training:
image = cv2.imread(item) #open image
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #convert to grayscale
training_data.append(gray) #append image array to training data list
training_labels.append(emotions.index(emotion))
for item in prediction: #repeat above process for prediction set
image = cv2.imread(item)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
prediction_data.append(gray)
prediction_labels.append(emotions.index(emotion))
return training_data, training_labels, prediction_data, prediction_labels
def run_recognizer():
training_data, training_labels, prediction_data, prediction_labels = make_sets()
print ("training fisher face classifier")
print ("size of training set is:", len(training_labels), "images")
fishface.train(training_data, np.asarray(training_labels))
print ("predicting classification set")
cnt = 0
correct = 0
incorrect = 0
for image in prediction_data:
pred, conf = fishface.predict(image)
if pred == prediction_labels[cnt]:
correct += 1
cnt += 1
else:
incorrect += 1
cnt += 1
return ((100*correct)/(correct + incorrect))
#Now run it
metascore = []
for i in range(0,10):
correct = run_recognizer()
print ("got", correct, "percent correct!")
metascore.append(correct)
print ("\n\nend score:", np.mean(metascore), "percent correct!")
#Change from:
emotions = ["neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"]
#To:
emotions = ["neutral", "anger", "disgust", "happy", "surprise"]
def run_recognizer():
training_data, training_labels, prediction_data, prediction_labels = make_sets()
print ("training fisher face classifier")
print ("size of training set is:", len(training_labels), "images")
fishface.train(training_data, np.asarray(training_labels))
print ("predicting classification set")
 cnt = 0
correct = 0
incorrect = 0
for image in prediction_data:
pred, conf = fishface.predict(image)
if pred == prediction_labels[cnt]:
correct += 1
cnt += 1
else:
cv2.imwrite("C:\\Users\\admin\\Desktop\\face.jpg"%(emotions[prediction_labels[cnt]], emotions[pred], cnt), image) #<-- this one is new
 incorrect += 1
cnt += 1
return ((100*correct)/(correct + incorrect))
```

Conclusion

In this project, a research to classify facial emotions over static facial images using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While

good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While the results achieved were not state-of-the-art, they were slightly better than other techniques including feature engineering. It means that eventually DL techniques will be able to solve this problem given an enough amount of labeled examples. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, facial emotion detection software includes the use of feature engineering. A solution totally based on feature learning does not seem close yet because of a major limitation: the lack of an extensive dataset of emotions. For instance, ImageNet contest uses a dataset containing 14 197 122 images. By having a larger dataset, networks with a larger capability to learn features could be implemented. Further improvement on the network's accuracy and generalization can be achieved through the following practices. The first one is to use the whole dataset during the optimization. Using batch optimization is more suitable for larger datasets. Another technique is to evaluate emotions one by one. This can lead to detect which emotions are more difficult to classify. Finally, using a larger dataset for training seems beneficial. Nowadays, optimizers work on a stochastic basis because of large datasets (millions of examples). However, this was true for our project. Given a limited dataset, trying on the whole dataset could have led to a better feature learning. Also, the use of some optimizers reported on this research would have had a different behaviour. This behaviour can be displayed on the loss curve having a smoother shape or by avoiding an early convergence. Second, due to time constraints, it was not possible to evaluate each emotion. On this way, it would have been possible to detect which emotions are easier to classify, as well as which ones are more difficult. Moreover, pre-training on each emotion could lead to a better feature learning. After that, the network could have received this learning (transfer learning). This could have improved on reducing the training time; as well as, minimizing to a higher degree the cost function. Also, using a larger dataset can lead to higher scale training. Training into a larger input space and for more time improves the network accuracy. A larger training scale allows the network to learn more relevant features. If this is not achieved, feature engineering is still required for this task. However, such a dataset might not exist nowadays. Using several datasets might be a solution, but a careful procedure to normalize them is required.

References

1. Berretti S et al. Superfaces: A super resolution model for 3d faces. In ECCV Workshops. 2012; 73-82.
2. Bettadapura V. Face expression recognition and analysis: the state of the art. arXiv preprint arXi. 2012.
3. Cui X et al. Data augmentation for deep neural network acoustic modelling. IEEE/ACM Trans. Audio, Speech and Lang. 2012; 23(9): 1469-1477.
4. Dhall A et al. Emotion recognition in the wild challenge. In Proceedings of the 16th International Conference on Multimodal Interaction. 2014; 461-466.
5. Hinton G et al. Deep neural networks for acoustic modelling in speech recognition", The shared views of four research groups. *Signal Processing Magazine, IEEE* 2012; 29(6): 82-97.
6. Gorbenko A, Popov V. On face detection from compressed video streams. *Applied Mathematical Sciences* 2012; 6(96): 4763-4766.
7. Hinton GE et al. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint. 2012.
8. Hoai M. Regularized max pooling for image categorization. *In BMVC* 2014; 2: 6.
9. Howard AG. Some improvements on deep convolutional neural network-based image classification. arXiv. 2013.
10. Sutskever I et al. Sequence to sequence learning with neural networks. *In Advances in Neural Information Processing Systems* 2014; 3104-3112.
11. Jia Y et al. Caffe: Convolution architecture for fast feature embedding. arXiv preprint arXiv. 2014.
12. Kim Y et al. Deep learning for robust feature generation in audio-visual emotion recognition. IEEE International Conference on Acoustics, Speech and Signal Processing. 2013; 3687-3691.