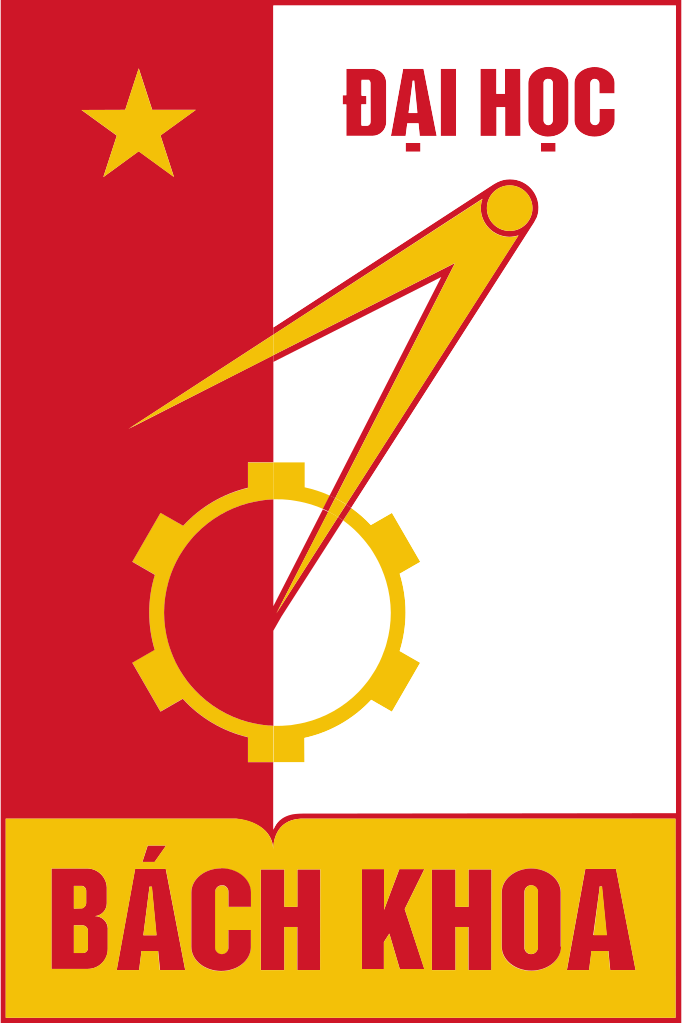
**Hanoi University of Science and Technology**

**School of Information and Communication Technology**

****

**PROJECT**

**HUMAN EMOTION DETECTION THROUGH FACE IMAGE DATA USING CONVOLUTIONAL NEURAL NETWORK**

**Group 15**

**Member Student ID Class**

Nguyễn Tuấn Dũng 20205148 ICT 01-K65

Hoàng Đức Trung 20200640 ICT 01-K65

Đoàn Tuấn Hùng 20205154 ICT 02-K65

**Instructor:** *Prof.* Thân Quang Khoát

**Table of Contents**

1. **The proposed problem**
2. The problem……………………………………………………….2
3. Dataset……………………………………………………………..2
4. **The solution**
5. General idea……………………………………………………….3
6. Details……………………………………………………………..3
7. **Evaluation results and findings**
8. Evaluation results………………………………………………….5
9. Findings……………………………………………………………7
10. **Code overview**……………………………………………….8
11. **Difficulties**…………………………………………………..11
12. **Conclusion**………………………………………………….12
13. **The proposed problem**
14. **The problem**

As humans, we learn to express ourselves and understand others’ feelings. Over time, the process of recognizing each other’s emotions becomes our nature. Yet, having a machine figuring out human sentiments can be challenging: it is virtually impossible to write a program that can tell human emotion apart. There are billions of people on the planet and each of them carries different characteristics and features…

This is where deep learning comes into play.

Deep learning is a subfield of machine learning that utilizes artificial neural networks and representational learning to teach computers to do what comes natural to humans. If computers are provided with sufficient and accurate data and are trained using appropriate algorithms, they can actually perform what we think are only feasible to us.

In this project, our goal is to build a deep learning model that can categorize and detect 7 basic human emotions: happiness, sadness, disgust, fear, surprise, anger and neutrality.

1. **Dataset**

We have chosen the FER-2013 dataset, which was introduced at the International Conference on Machine Learning (ICML) in 2013, to aid the training of our model, it features:

* About 35000 images of human faces.
* Separated into train data and test data
* 7 categories corresponding to the 7 basic human emotions
* 48x48 grayscale format

This is, however, a hard dataset to achieve good results because of the imbalance in the number of images between categories and the mislabeling of some of the images

1. **The solution**
2. **General idea**

We will build a machine learning model using convolutional neural networks, which is mostly used for image recognition and processing. The model will be passed through a number of layers in the network before it produces an output, which is an array of probabilities of every human emotion.

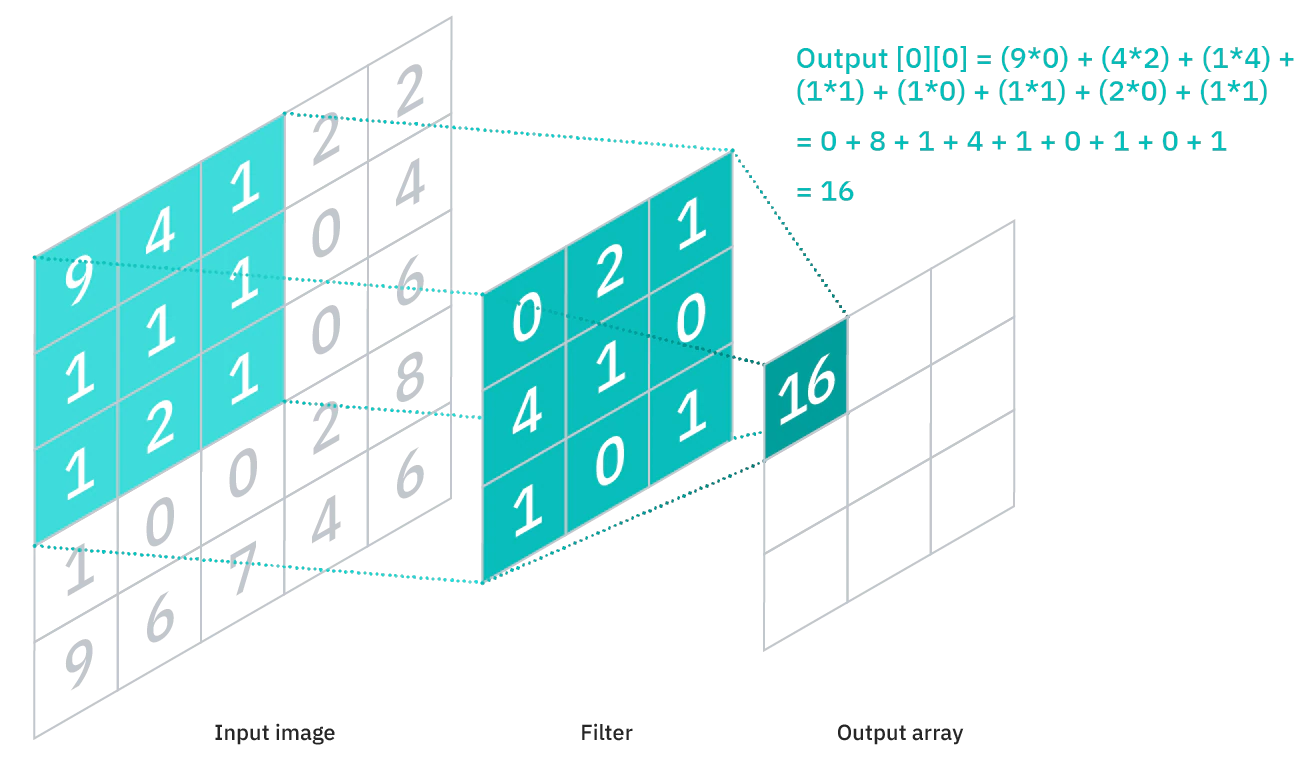
1. **Details**

Since this is an image classification problem, we chose to build and train a model using convolutional neural networks.

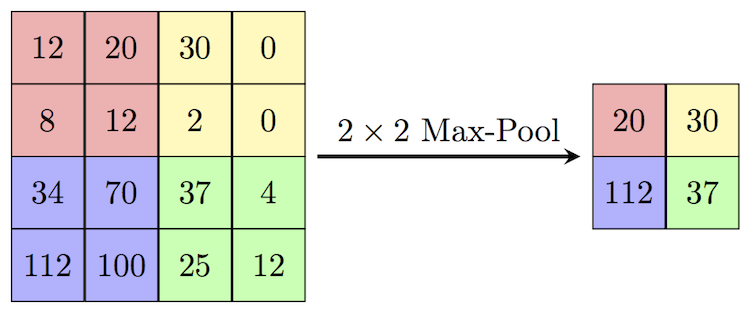
First, the input of the shape (48,48,1) is fed into a conv2D layer. The model then consists of several convolutional layers stacking on top of each other with some max pooling layers and dropout layers in between. These layers will try to detect distinctive features of the inputs to differentiate between them. Finally, the output layer is a dense layer with the output size of 7 corresponding to the number of emotion categories, combined with the softmax activation function to obtain the probabilities of all the emotions.

The role of different layers in the model:

* The Conv2D layer applies a specified number of filters across the image to extract the features from the image and tries to learn the best filter that helps matching the image with the correct label.



* The max pooling layer downsamples the image to produce a downsampled feature map. The downsampled version of the image will help mitigate the computational costs but still contain the sharp and smooth features of it.



* The Dropout layer helps with reducing overfitting and generalization. It works by randomly randomly setting input units to 0 with a specified rate at each step during training time.
* The Flatten layer reduces the multi-dimensional input down to one dimension. This layer is added between the convolutional layer and the fully connected layer
* Lastly, the fully connected layer, or the Dense layer, is a layer where its inside neurons connect to every neuron in the preceding layer. A number of them are usually placed before the output layer and form the last few layers of a CNN Architecture.

Besides, in each of the conv2D layers we specify the activation function to be the relu, or rectified linear unit activation function, which returns the value if it is bigger than zero.

Lastly, when compiling the model, we use the Adam optimizer to adjust the parameters to minimize the losses. And since it is a categorical problem, we set the loss function to be “categorical\_crossentropy”.

1. **Evaluation results and findings**
2. **Model training result**

We tested the model on some test batches and the results are as follow:

Average accuracy: 0.63

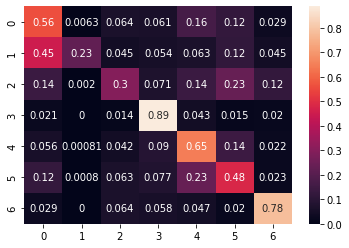


Fig. *Confusion matrix*



1. **Findings**

Here are some of the things we can observe through the evaluation results:

* The model can operate on acceptable accuracy
* The dataset has an imbalance number of images between different categories, which has lead to some bias in the model
* Some images seem to be mislabeled, but the model can still give accurate results
* The model might have a hard time differentiating between some emotions if the intensity of the expressed emotion is low.

1. **Code overview**

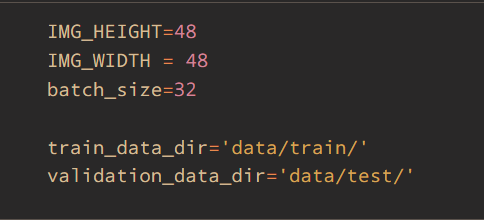
Our project has 2 main files, “Model.ipynb” and “app.py”. The first one is a jupyter notebook file, which is used to train the model and visualize information. The second file is used to take the weights of the trained model to perform a demo on normal, colored images.

1. **The jupyter notebook file**

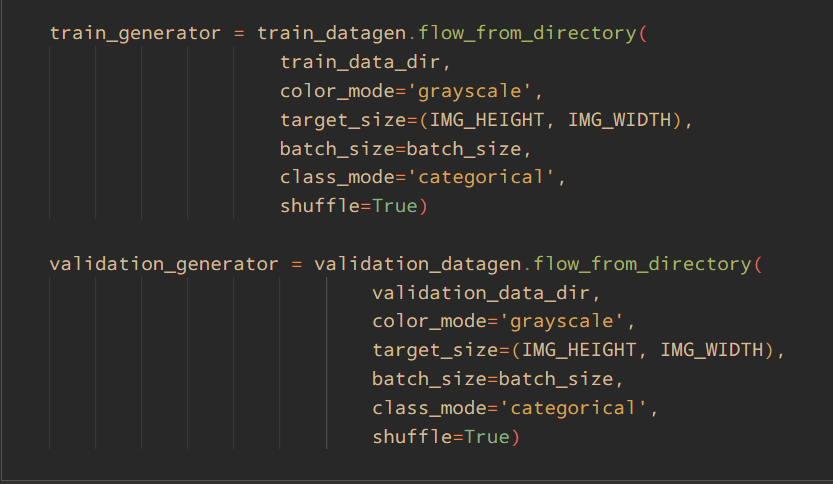
First, we import important libraries:



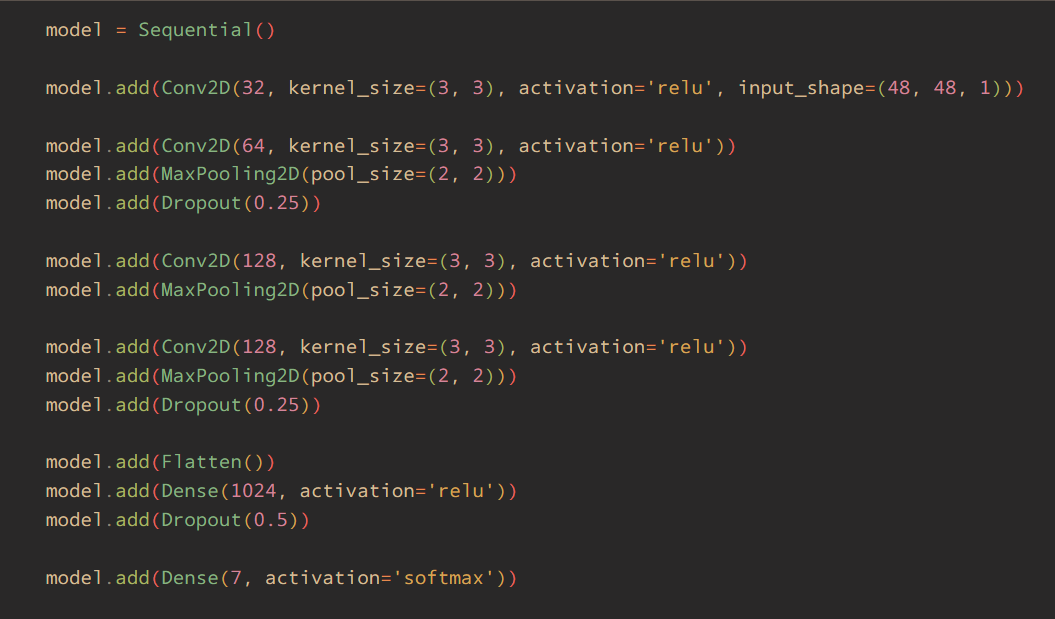
Then, we define some constants:



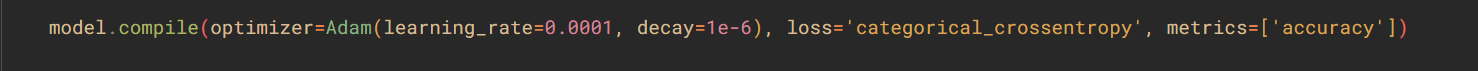
After that, we import training data and validation data with additional preprocessing:



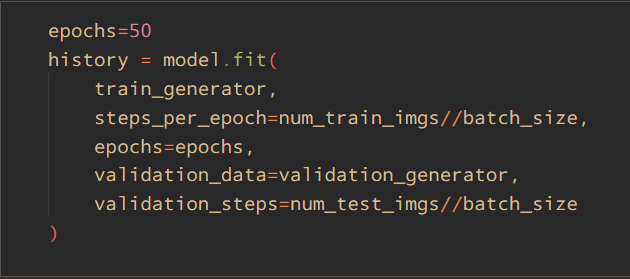
Next, we define the model layers:



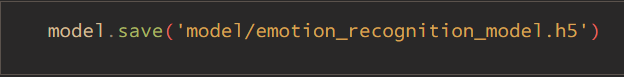
Then we compile the model:



Last but not least, the specified model is trained:

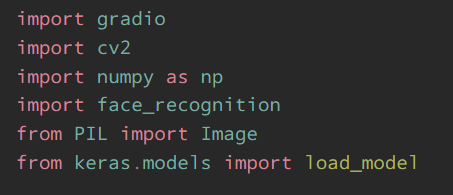


Additionally, we save the model weight into another file:



1. **The python file**

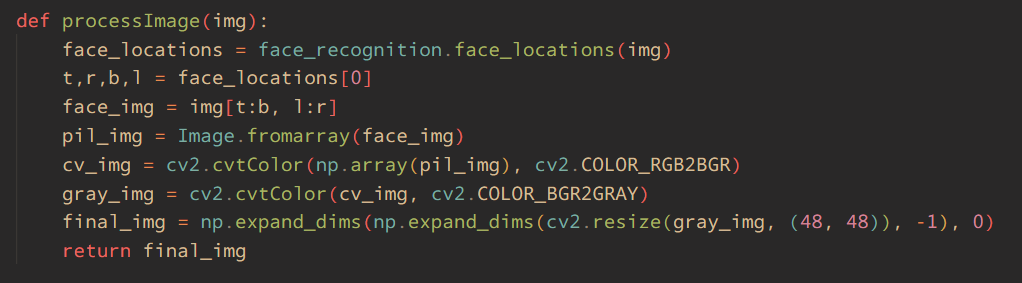
Import required libraries:



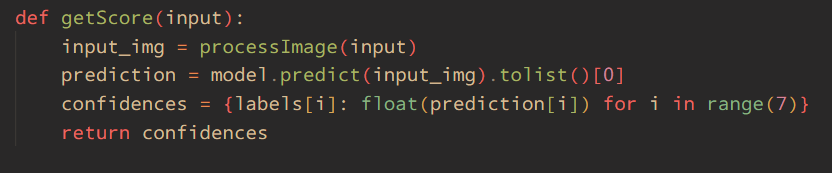
Load the model weights:



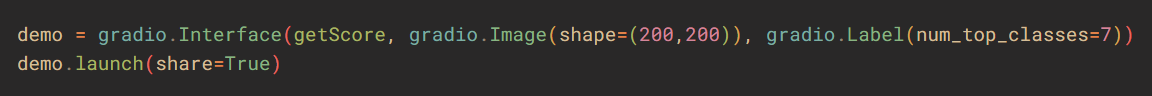
The image preprocessing function, this function first tries to locate human face in the input image, then convert the cropped image into a grayscale image as well as resize it to be of the same dimension as what the model would expect:



The getScore function is used to take the input image, pass it to the image preprocessing function and get the prediction result. Finally it returns a dictionary, in which the key is the emotion label and the value is the score, ranging from 0-1:



Finally, we use the gradio library to perform the demo. This will take an image and output the most probable emotion:



1. **Difficulties**

As mentioned, FER2013 is a difficult dataset:

* The distribution of image data is imbalanced, e.g: There are only 436 training images of the ‘disgust’ category while the ‘happy’ category has 7215.
* Some images are unrelated and mislabeled, which makes training the model harder, some examples:



=> We tried out several model configurations & optimizers to obtain the model that best performed.

Besides, since there are over 28000 training images, the training process took quite a lot of time. We did not have much opportunity to try out other model architecture and techniques, such as transfer learning,...

1. **Conclusion**

Training a good machine learning model is a hard task. During the project, we learned a lot about neural networks and how to build and train a machine learning model.

In the future, we wish to try out different architecture, dataset and various ways to optimize the model so that it can achieve the best accuracy.