**MIDTERM PROJECT TITLE: EMOTION RECOGNITION**

\_\_\_(Group 26)\_\_\_

**Subject:** Machine learning and data mining 1

**Subject code:** ICT3.002

**Lecturer:** Dr. Lê Hữu Tôn

**Semester:** 2023-2024

**Names of the group members:**

Nguyễn Trường Sơn

Dương Nghĩa Quyết

**ID:**

BI12-388

BI11-234

# Abstract:

**Introduction**

The project aims to develop an emotion recognition model using dataset of facial images to enhances human-computer interaction and sentiment analysis. The dataset is structured into training, validation, testing and predicting subsets, ensuring a comprehensive approach to model evaluation.

**AI-ML Model Construction**

The model deploys a Convolutional Neural Network (CNN) architecture, built by using TensorFlow and Keras libraries, both are well-known for their efficacy in image classification tasks. The architecture includes convolution layers for feature extraction, max-pooling layers for downsampling, dropout layers to prevent overfitting, flatten layers to vectorize outputs, and dense layers for classification. The codebase comprises of three main scripts: **‘main.py’** for model creating, **‘model\_visualiztion.py’** for viewing the performance, **‘predict.py’** for predicting emotions in new images.

**Evaluation**

The model’s performance is evaluated using accuracy, precision, and recall metrics. The training history, saved using pickle, is visualized in **‘model\_visualizataion.py’**, showed a decrease loss and an increase accuracy, indicating effective model training.

**Predictions**

Testing on a separate set of images in the **‘data\_testing’** folder, the model achieved a 6 out of 7 corrected guesses. While this signifies promising outcomes, the limited dataset size and diversity, along with the complexity of images with multiple entities and varied art styles, suggesting rooms for betterment. Augmenting the dataset and adding training epochs could significantly improve the model’s accuracy.

**Conclusion**

The project showcases the successful development of an emotion recognition model using CNN architecture, emphasizing its potential applications in multiple fields. The model’s performance highlights the need for more diverse and extensive training data to fully realize its potential.

Table of content:

[**Abstract: 2**](#_Toc23154)

[**I. INTRODUCTION 4**](#_Toc27490)

[**1.1 Purpose 4**](#_Toc21561)

[**1.2 Dataset Information 4**](#_Toc4850)

[**II. AI-ML MODEL CONSTRUCTION 5**](#_Toc18039)

[**2.1 Architecture 5**](#_Toc2697)

[**2.2 Building 6**](#_Toc22879)

[**III. EVALUATION 7**](#_Toc14055)

[**3.1 Test data 7**](#_Toc11942)

[**3.2 Model plotting 7**](#_Toc2840)

[**3.3 Save the model 8**](#_Toc3235)

[**IV. PREDICTIONS & CONCLUSION 9**](#_Toc23950)

[**3.1 Prediction 9**](#_Toc23444)

[**3.2 Conclusion 9**](#_Toc31455)

[**\*\*Ref: 10**](#_Toc19935)

# I. INTRODUCTION

## 1.1 Purpose

- The purpose of this project is to develop an emotion recognition model using a dataset of facial images. This is a crucial aspect of human-computer interaction and sentiment analysis. This model aims to interpret facial expressions from various real life images and artworks.

## 1.2 Dataset Information

- The dataset contains images of individuals and drawings expressing two emotions: **Happiness and Sadness**.

- The dataset is sourced from existing data on the Internet. See: Dataset

- The dataset has a total of 165 images

- The dataset is split into 2 types of data in different folders: **‘data\_training’** and **‘data\_testing’**

+ **‘data\_training’** includes:

* Training data : 70%
* Validation data : 20%
* Testing data : 10%

Total size: 165 images (92 happy images and 73 sad images)

+ **‘data\_testing’** consists of new images that have never been in the training before which are for predicting the images

# II. AI-ML MODEL CONSTRUCTION

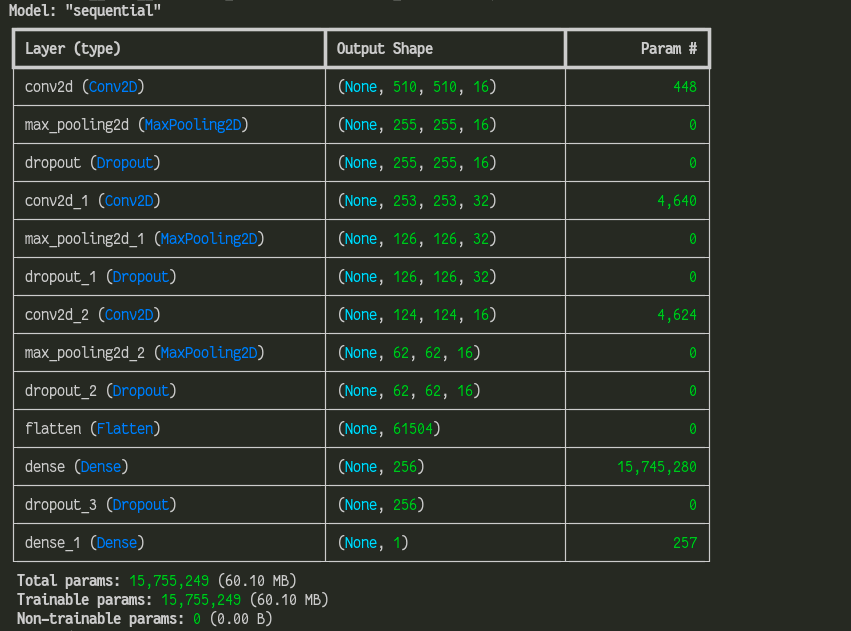
## 2.1 Architecture

- The model is built by using the TensorFlow and Keras libraries. This model’s architecture for this project is a Convolutional Neural Network (CNN), it is a widely-used architecture for image classification tasks.

- The CNN architecture has several layers:

* **Convolution layers**: Extract the features from input images
* **MaxPooling layers**: These layers downsample the feature maps, reducing computational power required
* **Dropout layers**: These layers prevent overfitting problem
* **Flatten layers**: These layers convert the outputs from previous layers into a single vector
* **Dense layers**: These layers connected layers that performs classification based on the extracted values

You can see the information of my model’s architecture here:



## 2.2 Building

- The code that I’ve written is referenced. See: CNN Tutorial

- main.py : has everything for creating and training and validating and testing the model

- model\_visualization.py : plot the model loss and accuracy

- predict.py : use the model to predict if the image is a happy image or a sad image (The data used for this lies in **‘data\_predicting’**)

**For more information about the model’s building and training one can refer to the codebase**

# III. EVALUATION

## 3.1 Test data

- Now we can test our model by using the test\_data generated in main.py

- Evaluation metrics:

* Accuracy: The overall correctness of the model’s predictions.
* Precision: The proportion of true positives among all positive predictions
* Recall: The proportion of actual positives that were correctly identified by the model

Performance after training the dataset for 30 epochs:

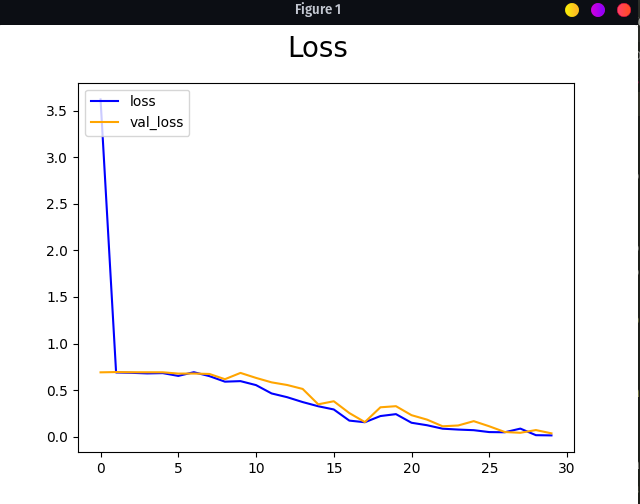


## 3.2 Model plotting

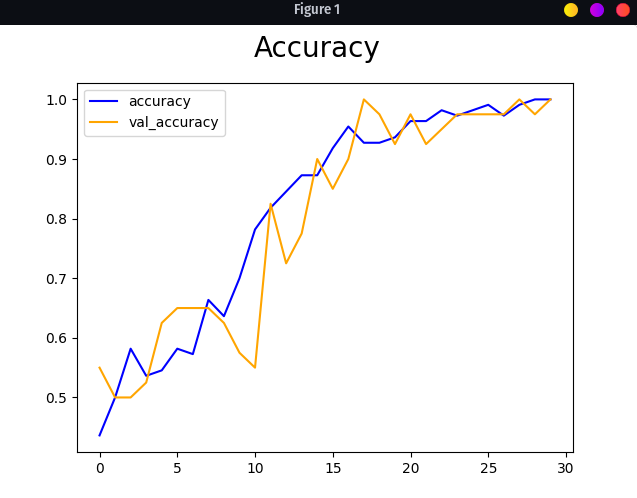
- In main.py the training information history is saved by using pickle and then will be loaded in model\_visualization.py

- In model\_visualization.py there are 2 plotting functions: Plotting the loss and Plotting the accuracy

Here are the results:

+ Loss:

+Accuracy:



- We can see that out model is working correctly as the Loss keep decreasing and the Accuracy is increasing

## 3.3 Save the model

- Last but not least, we can save the model for later, by using **keras.saving\_api** we can save multiple models in the **models** folder

- In the predict.py we can use any kind of model that we have saved by changing the path to the model that we want to use

# IV. PREDICTIONS & CONCLUSION

## 3.1 Prediction

- In **‘data\_testing’** folder I have prepared some images that I’ve got from Internet (not the same as from training)

- I’ve tested the images and out of 7 images, the model predicted the image correctly 6 times at its best, showing that the model I’ve built is working quite well, on the note that we are only feeding the model 165 images which is not many and the images are not diverse enough, also does not work well if multiple entities were in the image combine with the art styles factor of different drawings. If there were many more diverse images and we increase the epoch then the results would be much better.

## 3.2 Conclusion

- As the final words, this project demonstrates the development of an emotion recognition model using a CNN architecture. By training on a dataset of labeled images and using it for predicting images on a separated test set, the model’s ability to classify emotions can be assessed. We aim to enhance the accuracy and the robustness of the model, for the potential applications of this model can be use in various domains such as:

1. Human-Computer Interaction
2. Mental Health Assessment and Therapy
3. Market Research and Customer Feedback Analysis

# \*\*Ref:

1. CNN Tutorial: [Build a Deep CNN Image Classifier with ANY Images](https://www.youtube.com/watch?v=jztwpsIzEGc&t)

2. Dataset: [happy\_or\_sad\_Binary\_Image\_Classification](https://www.kaggle.com/datasets/eneskosar19/happy-or-sad-binary-image-classification?resource=download)

3. CNN Documentation: [Convolutional Neural Network and Regularization Techniques with TensorFlow and Keras](https://medium.com/intelligentmachines/convolutional-neural-network-and-regularization-techniques-with-tensorflow-and-keras-5a09e6e65dc7)