CSI4106 Project Report

By

Group 6

Cara Yuejia Li (300049083)
Tong Zhao (300037013)
Daniel Xu (300030558)

CSI 4106
April 19, 2022

Table of Contents

Abstract

Our topic of the project is “Facial Emotion Detection”. Our project is to predict or analyze facial expressions from images using Artificial Intelligence technology (AI).

Artificial intelligence gives computers the ability to analyze, perceive, think and react like humans. At present, artificial intelligence research includes robots, language recognition, image recognition, natural language processing, expert systems, etc. In our proposed project, facial emotion detection is based on human images, so it is also included as artificial intelligence.

Currently, the Facial Emotion Detection technique is strongly needed in the entertainment, business, and even education fields. For example, due to COVID-19, most schools have started offering online courses. During online sessions, teachers might not be able to take care of each student since there could be many students in the same class. Even with students’ cameras on, it is still hard for teachers to pay attention to every student.

Moreover, as per the rules of online exams, many professors require students to turn on their cameras to show their faces while doing exams. However, there are some large classes at the universities that have more than 100 students, and most of the professors are getting trouble proctoring every student regularly. So, with an AI facial emotion detector, teachers/professors can easily and rapidly track every single student’s emotion so that they can either improve their teaching quality or catch students who are against the exam rules.

We used the CNN algorithm and our own model. Also, we tried some other existing models. We built a convolution neural network architecture (CNN) and trained the model on the FER2013 dataset for Emotion recognition from images, including the “train”, “validation”, and “test” data sets. The evaluation metrics included the accuracy rate, precision, recall, and F1 score.

After fine-tuning the hyperparameters and trying many experiments of oversampling and undersampling methods, we finally found the best result with an accuracy of 62% on the FER2013 datasets.

Introduction

Problem to resolve

The problem for our project is to classify images (constructed by pixels) of human emotions into 7 different categories: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. The datasets were provided by Kaggle (Challenges in representation learning: Facial expression recognition challenge 2013).

Most of the previous facial emotion detection research was mainly based on raw images. However, the datasets that we found were only pixels in CSV files. So, in order to create the image data generators, we converted pixels into images, as shown in Fig. 1.

Fig. 1 Pixels into an Image

Reason for selection

Our motivation for selecting this proposed project is to help those people who are afraid of talking with others face-to-face. During this pandemic time, more and more people are used to and prefer to stay at home. In that case, most of them are more comfortable with online communication. So, facial emotion detection features can be used in online chat applications, making the chatting experience more relaxed.

Background information

There are two recent related works from Boston University and the University of Bucharest called “Facial Emotion Recognition: State of the Art Performance on FER2013" (Khaireddin1 and Chen1, 2021) and “Local Learning with Deep and Handcrafted Features for Facial Expression Recognition” (Georgescu1, Ionescu1, Popescu1, 2020), respectively. Both of the researchers used the Convolutional Neural Networks algorithm. One had a top result of 73.28%, and another one had a top result of 75.42%.

In this project, we aim to build facial emotion detection to predict or analyze facial expressions from images using the Convolutional Neural Networks (CNN) algorithm.

In the following sections, we will introduce the algorithms, the steps for preprocessing, the generator, and the model we used and created. In general, we will include background and related work, proposed solution, results, discussion, and conclusion.

The contribution of each group member is shown in Table 1.

Group Member	Task	Time Spent
Cara Yuejia Li	● Created project structure ● Dataset research ● Data preprocessing and classification ● Tested results ● Project report	10 days
Tong Zhao	● Project report ● Dataset research ● Converted pixels to images ● Tested results	10 days
Daniel Xu	● Project report ● Dataset research ● Dataset collection ● Tested results	10 days

Table 1: Group Contribution

Background and Related work

In this project, we have used the Convolutional Neural Network (CNN) algorithm. A Convolutional Neural Network (CNN) is a deep learning algorithm that can take in an input image, assign importance to various aspects or objects in the image and be able to differentiate one from the other.

The Convolutional Neural Network algorithm was first developed and used around the 1980s. At that time, CNN was only used to recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The main drawback for CNNs was that it requires a large amount of data to train and also requires a lot of computing resources. So, at that time, CNNs were only limited to the postal sectors. About 30 years later, in 2012, Alex Krizhevsky, a Ukrainian Canadian computer scientist brought back the branch of deep learning that uses multilayered neural networks. The availability of large sets of data with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

CNN is a class of deep neural networks, most commonly applied to analyze visual imagery. Images are constructed by matrices of pixels. The RGB images are matrices of pixel values having three planes, and the grayscale images are matrices of pixel values having a single plane. For example, a grayscale image is constructed as shown in Fig. 2.

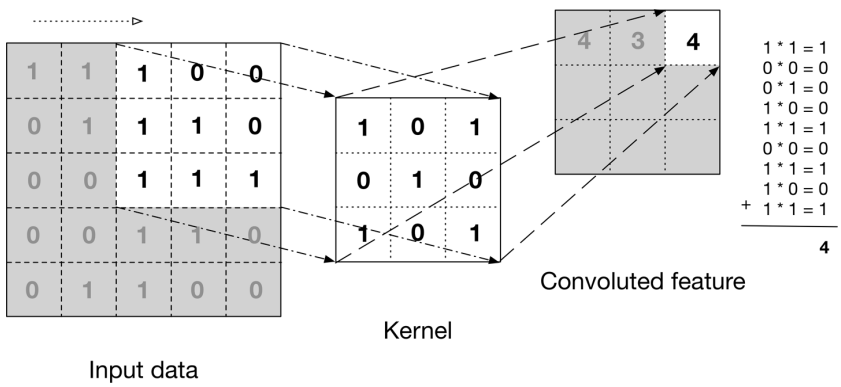

Fig. 2 Grayscale Image

Fig. 2 shows what a convolution is. We take a filter/kernel which is a 3×3 matrix and apply it to the input image to get the convolved feature.

In our project, we have also used accuracy, precision, recall, F1 score, and confusion matrix as the metrics to evaluate our models. The confusion matrix represents the numbers of the 7 emotions prediction results in our project, as shown in Fig. 3. To get the accuracy, precision, recall, and F1 score, we used (TP + TN) / total predictions, TP/ (TP + FP), TP/ (TP + FN), (2 * Precision * Recall) / (Precision + Recall), respectively.

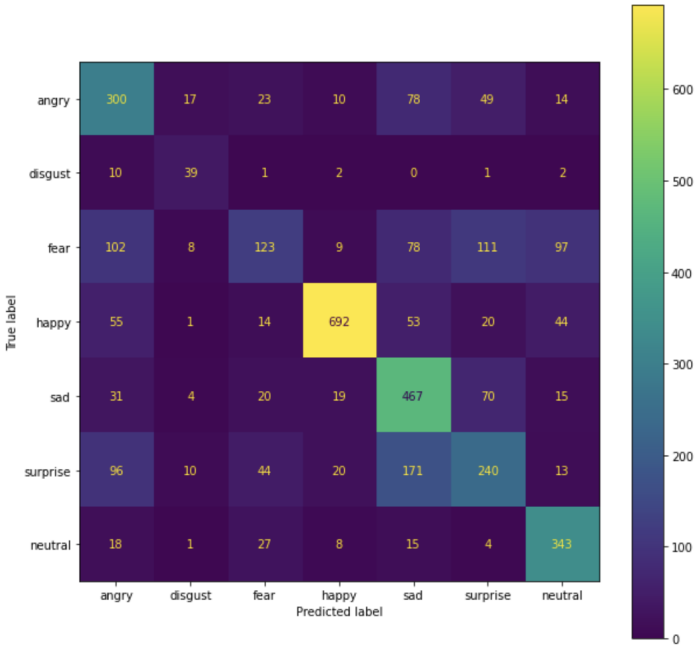

Fig. 3 Confusion Matrix with 7 emotions

There are two recent related works from Boston University and the University of Bucharest called “Facial Emotion Recognition: State of the Art Performance on FER2013” (Khaireddin1 and Chen1, 2021) and “Local Learning with Deep and Handcrafted Features for Facial Expression Recognition” (Georgescu1, Ionescu1, Popescu1, 2020), respectively. Both of the researches used CNN and VGG.

VGG stands for Visual Geometry Group, and it is a standard deep CNN architecture with multiple layers. VGG accepts a 224×224 pixel RGB image as input. In the ImageNet competition, the writers cropped out the middle 224×224 patch of the images. Also, to reduce the number of parameters, the authors propose to use small respective fields to replace the large respective fields. For example, in order to decrease the number of parameters but keep the performance, we can use 2 layers of a 3×3 filter to replace 1 layer of a 5×5 filter, because the number of parameters of 2 layers of a 3×3 filter is 18 (2×3×3=18), while the number of parameters of 1 layer of a 5×5 filter is 25 (1×5×5=25).

The two recent related works mentioned above both used VGG models. However, in our project, we created our own model with 4 blocks of layers, which contains convolutional layers with different batch sizes, batch normalization, max-pooling layers, and flatten layers. For the fully connected layers, we used dense layers with different units and dropout layers to make our model more generalizable. We also tried some pre-trained models called MobileNetV2 and NASNetMobile, but they all have less accuracy than our own model.

Proposed solution

In our project, regarding the problems, we conduct experiments with these steps: loading and splitting datasets, oversampling and undersampling, converting images, creating and training models, and testing and predicting models. We will include the details in the following subsections.

Loading and Splitting Datasets

For this project we used the dataset called “FER2013” to include 7 types of emotions with different images pixels. First, we loaded the dataset which contains all training, validation, and testing data. We converted the 0-6 emotions to their corresponding types of emotions as strings. Then we split the datasets into the training, validation, and testing datasets and after that, we printed some shapes, and column information and created a histogram to show the distribution of the emotions, as shown below in Fig. 4. In addition, we created a new column called “image_name” for those three different datasets and saved the data frames into CSV files. We named each image as “Data_” followed by the ordered number ascendingly starting from 1.

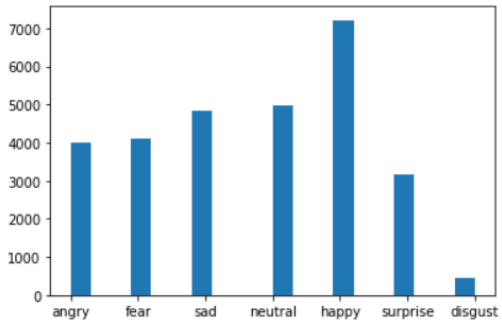

Fig. 4 Histogram of the Emotion Data

Oversampling and Undersampling

The training set consists of 28,709 examples. However, in the histogram shown above in Fig. 4, we can find that the data is imbalanced. Specifically, there are 7215 training data for “happy” emotion, but “disgust” emotion only has 436 data. Therefore, solving the data imbalance problem, we undersampled the largest number of data to 5000 as an average number, using the method “RandomUnderSampler”. Then, we oversampled the rest of the 7 emotions to 5000 for alignment using the method “RandomOverSampler”. So we got a new training data frame called “new_train_df”.

Converting Images

The datasets that we found were only pixels in CSV files (e.g. fer2013.csv), with indications of what each emotion is. So, in order to create the image data generators for the next step, we decided to convert pixels into images. Firstly, we constructed arrays containing the pixels. Then, we used the “NumPy.reshape()” function to build the images. After that, we saved images as JPG files in a specific directory on our local disks, for future purposes. Also, we created an image generator. We used the method “flow_from_datafrade” from the val_img_gen instance to link the test DataFrame and the test data folder. That way, with an image generator, we can ensure that the model receives new variations of the images at each epoch.

Creating and Training Model

For the model, we used the Convolutional Neural Network algorithm with four blocks of layers, which contains two 3D convolutional layers with 32, 64, 128, and 256 batch sizes, batch normalization layers, and a 2D max-pooling layer for each block. Then we flatten the images and added 4 dense layers with Rectified Linear Unit (ReLU) activation to avoid gradient dispersion problems and speed up training. After each dense layer, we added a dropout layer with parameter 0.2 to make our model more generalizable This makes good image manipulation.

We also tried some pre-trained models, for example, NASNetMobile has low accuracy, which is around 0.53 as the model training. Another model called MobileNetV2 did well and accuracy can go around 0.5756 after 7 epochs. We duplicated the input channel from 1 to 3 and it also increases the accuracy from 0.59 to 0.62.

We tried some different hyperparameters to adjust our model. For example, we tried the "RMSprop" optimizer with 1e-3 as the learning rate and the “Adam” optimizer and we found out that the “Adam” optimizer has 0.05 higher accuracy. We also tried 32, 64, 128, and 256 as batch sizes and we concluded that 128 is the best for this model. We also adjusted our models with different numbers of layers and various batch sizes and units.

Testing and Predicting Model

For testing the model, we used a test generator linking to a testing data frame that we split before. As we had created our own model already, we used the “evaluate_generator” method to evaluate the trained model and got the values of loss and accuracy for 7 emotions. Then, we generated predictions using the test generator. After that, we got the classification report in Fig. 5

for the evaluation metrics including accuracy, recall, and F1 score, as well as the confusion matrix in Fig. 6.

	precision	recall	f1-score	support
angry	0.49	0.61	0.54	491
disgust	0.49	0.71	0.58	55
fear	0.49	0.23	0.32	528
happy	0.91	0.79	0.84	879
sad	0.54	0.75	0.63	626
surprise	0.48	0.40	0.44	594
neutral	0.65	0.82	0.73	416
accuracy			0.61	3589
macro avg	0.58	0.62	0.58	3589
weighted avg	0.62	0.61	0.60	3589

Fig. 5 Classification Report

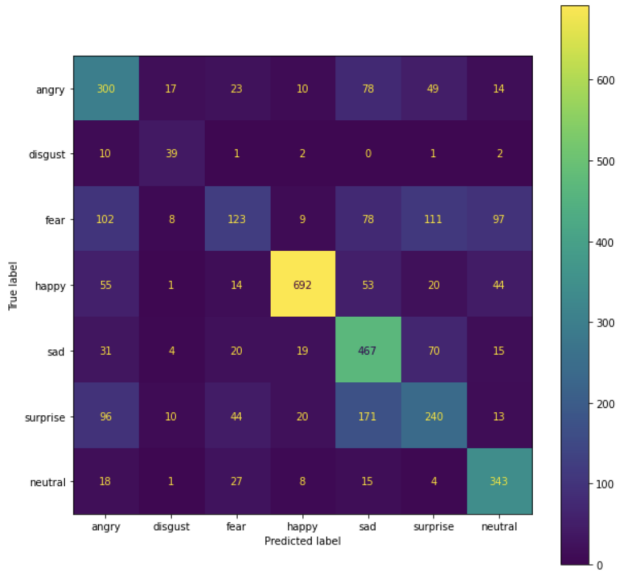

Fig. 6 Confusion Matrix

Results

Models	**Accuracy**
CNN model	0.62
NASNetMobile	0.53
MobileNetV2	0.58

Discussion

After this project, we found out that although we tried some different existing models and these models were described as the model with top accuracy, they may be trained with low accuracy and f1 score. Although the model is good enough, the major problem is the datasets. We read through some documents and found out the highest accuracy obtained from the SOTA algorithm is around 0.70, which means this dataset is not that good. The data is imbalanced with a maximum of around six thousand different numbers of data between some categories. Although we fixed the imbalance issue, it will still have some noisy data as we oversample. Another problem with this dataset is that these 48*48 images are too small and unclear, so it's hard for the machine to distinguish and compare the emotions.

Conclusion

In this project, we built facial emotion detection using our own model with the CNN algorithm. We tested out several other models, but the accuracy is not as good as we expected. Therefore, we decided to use our own model we created, with a top accuracy of 62%.

Overall, this is a very exciting project. In future work, if we have a chance, we plan to have some improvements toward this project, for example first detect the human faces before we detect their emotions, so the top accuracy should be greatly improved. Also, we aim to add a new feature to this project, for example, let it detect human emotions when there are multiple people in the same image.

References

Challenges in representation learning: Facial expression recognition challenge. Kaggle. (2013). Retrieved April 2, 2022, from www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data

Georgescu, M.-I., Ionescu, R. T., & Popescu, M. (2019). Local Learning With Deep and Handcrafted Features for Facial Expression Recognition. *IEEE Access, 7*, 64827–64836. Retrieved April 12, 2022, from https://arxiv.org/pdf/1804.10892.pdf

Khaireddin, Y., & Chen, Z. (2021). *Facial Emotion Recognition: State of the Art Performance on FER2013*. Retrieved April 12, 2022, from https://arxiv.org/ftp/arxiv/papers/2105/2105.03588.pdf

Ma, E. (2019, December 9). What is the VGG neural network? Medium. Retrieved April 13, 2022, from https://becominghuman.ai/what-is-the-vgg-neural-network-a590caa72643

Mandal, M. (2021, May 1). CNN for Deep Learning | Convolutional Neural Networks (CNN). Analytics Vidhya. Retrieved April 13, 2022, from www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

Model Evaluation Metrics in Machine Learning. (n.d.). KDnuggets. Retrieved April 13, 2022, from www.kdnuggets.com/2020/05/model-evaluation-metrics-machine-learning.html

RAO, M. A. (2021, January 28). Realtime Face Emotion Recognition using transfer learning in TensorFlow. Analytics Vidhya. Retrieved April 14, 2022, from https://medium.com/analytics-vidhya/realtime-face-emotion-recognition-using-transfer-learning-in-tensorflow-3add4f4f3ff3

Saha, S. (2018, December 15). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. Towards Data Science. Retrieved April 13, 2022, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53