AI Programming Project: Facial Expression Classification

AIP391: Quach Phu Quy (Team Leader), Nguyen Luong Phuc

March 23, 2022

Abstract: This report is a complication of the results of research and development of Facial Expression Classification models by using CNN architecture and some datasets such as FER2013, CK+, AffectNet Sample. After getting this result, we use it to demonstrate the knowledge of our members through the explanation and statistics of some models and then to express how we can apply it to the real world application. The main objective through this project is to understand how CNN architecture works and how to apply it to detect emotion in real time.

Keywords: face recognition, emotion detection, deep learning, computer vision, CNN.

1 Introduction

Facial expression refers to the movement of facial muscles that carry emotions expressed by a person. It provides information about the mental status of that person. Emotion is a mental condition that the person goes through. Analysis of facial expression has many applications such as in lie detectors, robotics and in art, ...

Improvement in the skill of facial expression recognition is required for an intelligent agent to communicate with human as a part of machine-human collaboration as well as with robots as a part of robot-robot interaction. Fluctuation in recognition rate among the classes is one of the issues for most of the research as they have lower recognition rate to detect emotions like disgust and fear. The purpose of this research is to develop a facial expression recognition system which can classify an image into different classes of emotion. Convolutional neural network in facial expression recognition has been applied in a few research but inconsistency in recognition rate among classes is one of the issues for most of the research as they have lower recognition rate in disgust and fear. We come up with the idea of CNN with data augmentation and combined dataset collected from several datasets which leads this research to higher validation accuracy as well as higher and basically better performance in real time. By using face detection system, we try to improve better detection in real time application. The remaining sections of this article consist of: related work on facial expression recognition, an overview of the methodology of this research, data collection and preprocessing, experiment with data augmentation, how the proposed system has been implemented, result and discussion, and conclusion.

2 Related Works

2.1 CNN - Convolutional Neural Network

2.1.1 Introduction

In the last few years of the IT industry, there has been a huge demand for once particular skill set known as Deep Learning. Deep Learning a subset of Machine Learning which consists of algorithms that are inspired by the functioning of the human brain or the neural networks.

These structures are called as Neural Networks. It teaches the computer to do what is naturally similar to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Autoencoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNets.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term 'Convolution" in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

2.1.2 Basic CNN Architecture

There are two main parts in a CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

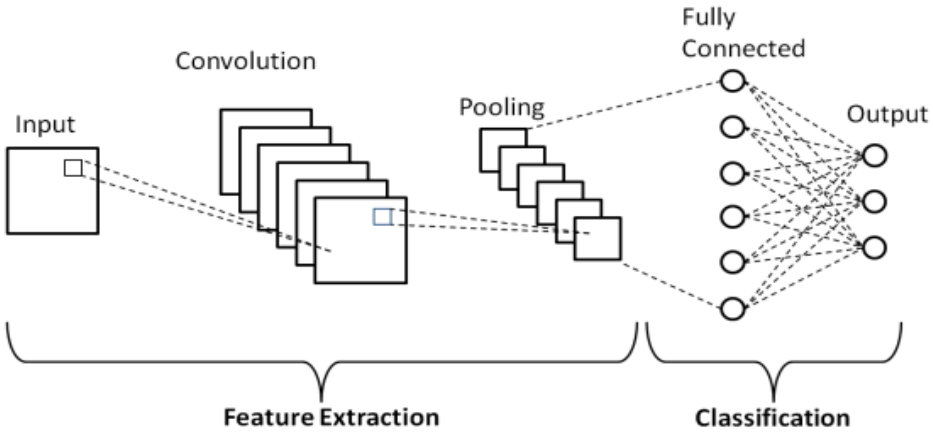

Figure 1: Basic CNN architecture

Convolution Layers

Convolutional Layer : Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a "convolution". In the context of a CNN, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. (Figure 2)

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the

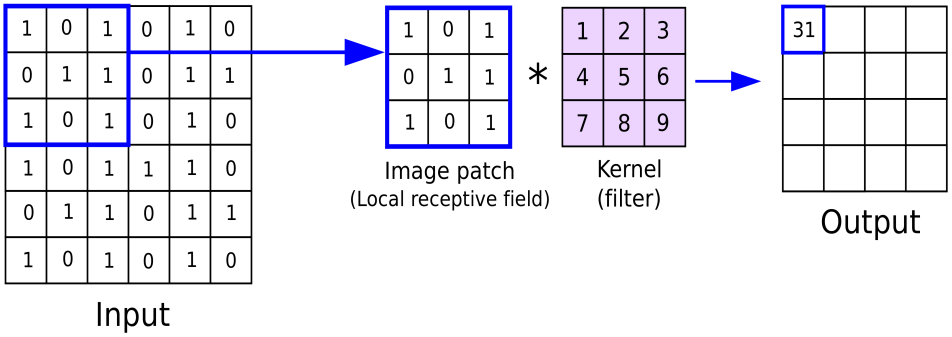

Figure 2: Convolution operation with kernel size 3x3 to get a feature map from input

image. In summary, we have a input, such as an image of pixel values, and we have a filter, which is a matrix of weights, and the filter is systematically applied to the input data to create a feature map.

Pooling Layer :In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. In Max Pooling, the largest element is taken from feature map (Figure 3). Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling (Figure 4). The Pooling Layer usually serves as a bridge between the Convolutional Layer and the Fully Connected Layer.

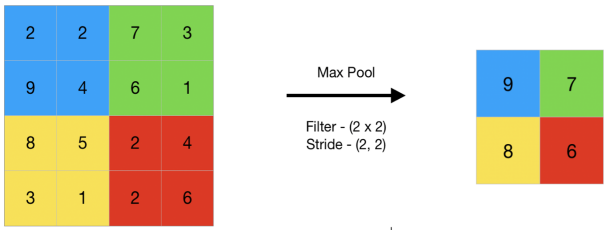

Figure 3: Max Pooling

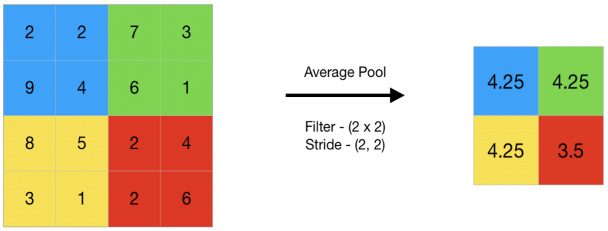

Figure 4: Average Pooling

Fully Connected Layer : The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place. After passing through the fully

connected layers, the final layer uses the Softmax activation function (instead of ReLU) which is used to get probabilities of the input being in a particular class (classification). And so finally, we have the probabilities of the object in the image belonging to the different classes. (Classification part in Figure 1)

2.1.3 Summary

Various techniques have been proposed to even further improve performance. For instance, the Sigmoid activation function has been replaced by Rectified Linear Unit (ReLU) activation to avoid gradient dispersion problems and speed up training. Different pooling methods such as average pooling and max pooling are used to downsample the inputs and aid in generalization. Dropout, regularization, and data augmentation are used to prevent overfitting. Batch normalization has been developed to help prevent gradient vanishing and exploding, which makes the training model runs faster.

A great deal of research has also been done in creating different optimization algorithms used in training. Though there is no systematic theoretical guideline on choosing an optimizer, empirical results show that a suitable optimization algorithm can effectively improve a model's performance. The most commonly used optimizer is Stochastic gradient descent (SGD). It is a simple technique that updates the parameters of a model based on the gradient of a single data point. Numerous variations of this algorithm have been proposed to speed up training. AdaGrad adaptively scales the learning rate for each dimension in the network. RMSProp radically diminishes the learning rate. Adam combines the advantages of AdaGrad and RMSProp by scaling the learning rate and introducing momentum of gradient, etc.

Among many others, one significant factor that could impact performance is the learning rate. A large learning rate could lead to oscillations around the minima or divergence in the loss. A small learning rate would slow down the model's convergence significantly and could trap the model at a non-optimal local minimum. A commonly used technique is to employ a learning rate scheduler that changes the learning rate during training. An adaptive learning rate schedule tries to automatically adjust the learning rate based on the local gradients during training. Cosine annealing resets the learning rate periodically and reuses "good weights" during the training process, etc. With all the developments above and the extensive research in CNNs, they have become an extremely favorable tool when tackling tasks in image processing, pattern recognition, and feature extraction. Once a large facial expression dataset, FER2013, was introduced at ICML in 2013, it became a benchmark in comparing model performance in emotion recognition. Many CNN variants have achieved remarkable results with a classification accuracy between 65% and 78%.

However, in order to improve the ensemble performance even further, we aim to first implement this models again with some fixed datasets and mixing datasets. Other research work has tried to improve their model's performance on FER2013 or AffectNet or CK+ is out of this scope. The purpose of this paper is just to understand how to apply models to the recognize facial expressions in realtime.

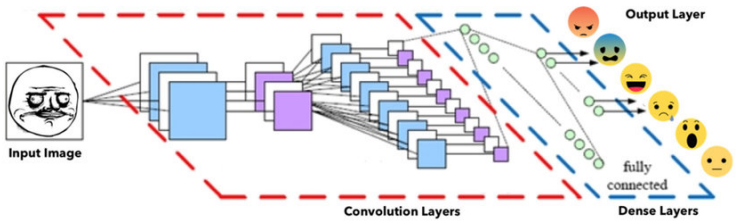

Figure 5: Facial Expression Classification using CNNs

2.2 Some researchs relating to this project

We conducted with some research papers with the similar approach as what we aim to archive in this project:

Facial Expression Recognition using Convolutional Neural Networks: State of the Art - Christopher Pramerdorfer, Martin Kampel - 2016 [11]: The authors review the state of

the art in image-based facial expression recognition using CNNs and highlight algorithmic differences and their performance impact. On this basis, they identify existing bottlenecks and consequently directions for advancing this research topic. Furthermore, they demonstrate that overcoming one of these bottlenecks – the comparatively basic architectures of the CNNs utilized in this field – leads to a substantial performance increase. By forming an ensemble of modern deep CNNs, we obtain a FER2013 test accuracy of 75.2%, outperforming previous works without requiring auxiliary training data or face registration

Convolutional Neural Network Hyperparameters optimization for Facial Emotion Recognition - Adrian Vulpe-Grigoraşi, Ovidiu Grigore - Mar, 2021 [13]: This paper presents a method of optimizing the hyperparameters of a convolutional neural network in order to increase accuracy in the context of facial emotion recognition. The optimal hyperparameters of the network were determined by generating and training models based on Random Search algorithm applied on a search space defined by discrete values of hyperparameters. The best model resulted was trained and evaluated using FER2013 database, obtaining an accuracy of 72.16%.

Facial Emotion Recognition: State of the Art Performance on FER2013 - Yousif Khaireddin, Zhuofa Chen - May 2021 [5]: This paper achieves single-network state-of-theart classification accuracy on FER2013 using a VGGNet. We thoroughly tune all hyperparameters towards an optimized model for facial emotion recognition. Different optimizers and learning rate schedulers are explored and the best initial testing classification accuracy achieved is 73.06 %, surpassing all single-network accuracies previously reported. We also carry out extra tuning on our model using Cosine Annealing and combine the training and validation datasets to further improve the classification accuracy to 73.28 %. For future work, we plan to explore different image processing techniques on FER2013 and investigate ensembles of different deep learning architectures to further improve our performance in facial emotion recognition.

Facial Expression Recognition Using Residual Masking Network - Luan Pham, The Huynh Vu, Tuan Anh Tran - 2021 [10]: This paper forward a system for facial expression recognition, in which the main contribution is a novel Masking Idea that is implemented in the Residual Masking Network. This Residual Masking Network contains several Masking Blocks which are applied across Residual Layers to improve the network's attention ability on important information.

3 Data Preparation

Having sufficient labeled training data that include as many variations of the populations and environments as possible is important for the design of a deep expression recognition system. In this section, we discuss the publicly available databases that contain basic expressions and that are widely used in our reviewed papers for deep learning algorithms evaluation. We also introduce newly released databases that contain a large amount of affective images collected from the real world to benefit the training of deep neural networks. The only problem is that we didn't have access to the dataset AffectNet but at least we just have some samples, i.e nearly 10% of the original samples. The below table provides an overview of these datasets, including the main reference, number of samples, number of classes and the size of each example of corresponding dataset.

FER2013: The FER2013 database was introduced during the ICML 2013 Challenges in Representation Learning. FER2013 is a large-scale and unconstrained database collected automatically by the Google image search API. All images have been registered and resized to 48*48 pixels after rejecting wrongfully labeled frames and adjusting the cropped region. FER2013 contains 28,709 training images, 3,589 validation images and 3,589 test images with seven expression labels (anger, disgust, fear, happiness, sadness, surprise and neutral)

CK+/CK+48: The Extended CohnKanade (CK+) database is the most extensively used laboratory-controlled database for evaluating FER systems. CK+ contains 593 video sequences from 123 subjects. The sequences vary in duration from 10 to 60 frames and show a shift from a neutral facial expression to the peak expression. Among these videos, 327 sequences from 118 subjects are labeled with seven basic expression labels (anger, contempt, disgust, fear, happiness, sadness, and surprise) based on the Facial Action Coding System (FACS). This version of CK+ used for this project, which is CK+48, has 981 samples with 7 labels like above. Because CK+ does not provide specified training, validation and test sets, the algorithms evaluated on this database are not uniform. For static-based methods, the most common data selection method is to extract the last one to three frames with peak formation

and the first frame (neutral face) of each sequence. Then, the subjects are divided into k groups for person-independent k-fold cross-validation experiments, where commonly selected values of k are 5, 8 and 10.

AffectNet Sample: AffectNet contains more than one million images from the Internet that were obtained by querying different search engines using emotion-related tags. It is by far the largest database that provides facial expressions in two different emotion models (categorical model and dimensional model), of which 450,000 images have manually annotated labels for eight basic expressions. But unfortunately, we are not able to use this data, so we use another dataset similarity to AffectNet but with less images than the original.

We also modify the FER2013 and AffectNet samples to only train with 5 classes instead of 7 and a combination of FER2013 & AffectNet samples together. In the AffectNet samples data, we reduce the samples from the labels "fear", "disgust" and "contempt" and keep the same amount of FER2013 samples. This make us have a new auxiliary dataset FER2013 & AffectNet with total of 57,277 samples, of course we resize the original 256x256 size of each AffectNet image to the size of 48x48.(Table 1)

Dataset	N.o. samples	N.o. classes	Size
FER2013	35,887	7	48x48
CK+48	981	7	48x48
AffectNet Sample	41,553	8	256x256
FER + AffectNet	57,277	5	48x48

Table 1: Statistics about the datasets.

Here are some samples of different datasets:

Figure 6: Sample of each emotion in FER2013 Dataset

Figure 7: Sample of each emotion in AffectNet Sample Dataset

After mixing FER2013 with AffectNet Sample, we got a new dataset including train data with 49,176 samples ('Angry': 8,995, 'Happy': 12,215, 'Neutral': 9,965, 'Sad': 9,830, 'Surprise': 8,171) and test data with 8,543 samples('Angry': 1,458, 'Happy': 2,274, 'Neutral': 1,733, 'Sad': 1,747, 'Surprise': 1,331) (Figure 8)

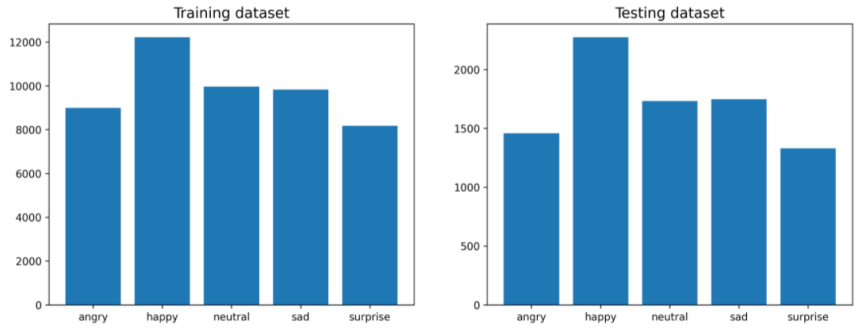

Figure 8: FER2013 combined with AffectNet Sample Data

4 Methods

4.1 Models

4.1.1 Baseline CNN Architecture

A baseline model will establish a minimum model performance to which all of our other models can be compared, as well as a model architecture that we can use as the basis of study and improvement.

In order to better understand the problem, we decided to first try to tackle this problem from scratch, building a vanilla CNN using four Convolution Layers 3x3x32 with no padding, ReLU filters, interleaved with three 2x2 MaxPooling layers, and completed with a Fully connected layer and softmax layer. We also added dropout with rate 20% to aovid overfitting and archive accuracy up to 63%. (Figure 9)

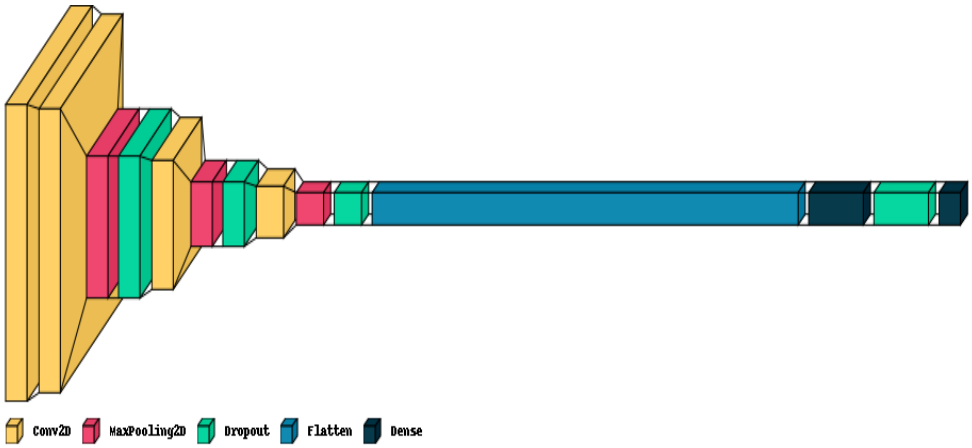

Figure 9: Baseline CNN architecture

4.1.2 VGG - Very Deep Convolutional Networks (VGGNet)

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The “deep” refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers.

The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures. VGGNets are based on the most essential features of convolutional neural networks (CNN). The VGG network is constructed with very small convolutional filters. Here is a brief summary of the architecture of VGG (Figure 10):

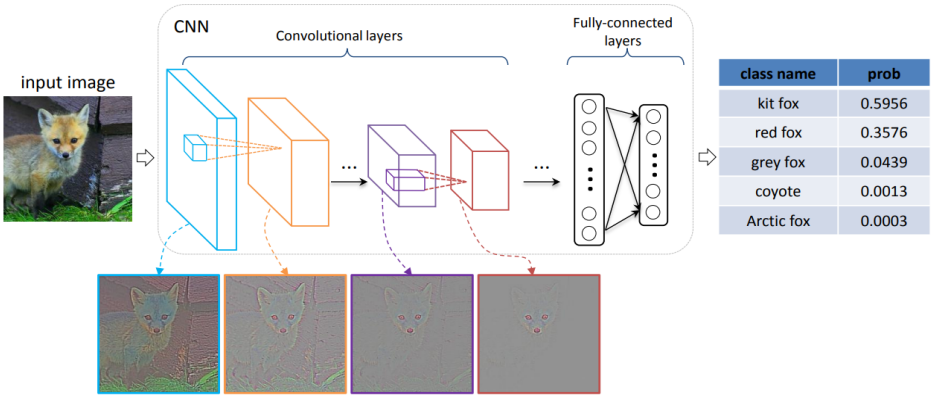

Figure 10: VGG architecture

- **Input**:The VGGNet takes in an image input size of 224×224. For the ImageNet competition, the creators of the model cropped out the center 224×224 patch in each image to keep the input size of the image consistent.
- **Convolutional Layers**:VGG's convolutional layers leverage a minimal receptive field, i.e., 3×3, the smallest possible size that still captures up/down and left/right. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from AlexNet that reduces training time. ReLU stands for rectified linear unit activation function; it is a piecewise linear function that will output the input if positive; otherwise, the output is zero. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution (stride is the number of pixel shifts over the input matrix).
- **Hidden Layers**: All the hidden layers in the VGG network use ReLU. VGG does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time. Moreover, it makes no improvements to overall accuracy.
- **Fully-Connected Layers**:The VGGNet has three fully connected layers. Out of the three layers, the first two have 4096 channels each, and the third has 1000 channels, 1 for each class.

4.1.3 Proposed 5-block VGG-style Architecture

There is a version based on VGG architecture is VGG16. The VGGNet-16 supports 16 layers and can classify images into 1000 object categories, including keyboard, animals, pencil, mouse, etc. Additionally, the model has an image input size of 224-by-224.

Due to limitation of usage and computing power, our member can not use the VGG16 because of increasing of input size, which may lead to long long hours of training with limitation of our computing power. So we come up with the solution is use the CNN model with VGG-Style architecture, concretely is 5-block VGG-Style Model.

Comparing the accuracy of the baseline model with the average results is obviously worse. One of the problems is that the baseline model is quite simple. The convolution layers has small number of kernels so it doesn't extract enough features on the face in the dataset. Also, this dataset, i.e FER2013, is basically an imbalanced dataset, so by the goes on, we need to supply more overfitting-avoiding techniques such as batch normalization and dropouts.

In general, this 5-block VGG-Style architecture with just having more blocks and additionally use batch normalization. The architecture involves stacking convolutional layers with small 3×3 filters followed by a max pooling layer. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network such as 32, 64, 128, 256 for the first four blocks of the model. Padding is used on the convolutional layers to ensure the height and width of the output feature maps matches the inputs. Concretely, there are 5 blocks

where each block contains multiple convolution layers interleaved with one 2x2 MaxPooling. Batch normalization has been used on the feature map of each convolution layer and dropout is used after the 2x2 MaxPooling layer. We also increase the amount of filters/kernels in each block, i.e first block has initial number of kernels is 64 in each convolution layer in the block (instead of 32 like the one in the baseline CNN model), the second block has double amount of the kernels in the previous block, that is 128, and so on to 512 kernels in fifth block but we keep the same amount of kernels in the third and fourth block to 256 one. Finally is the Fully Connected Layer using Softmax activation to output the probability of each labels. (Figure 11)

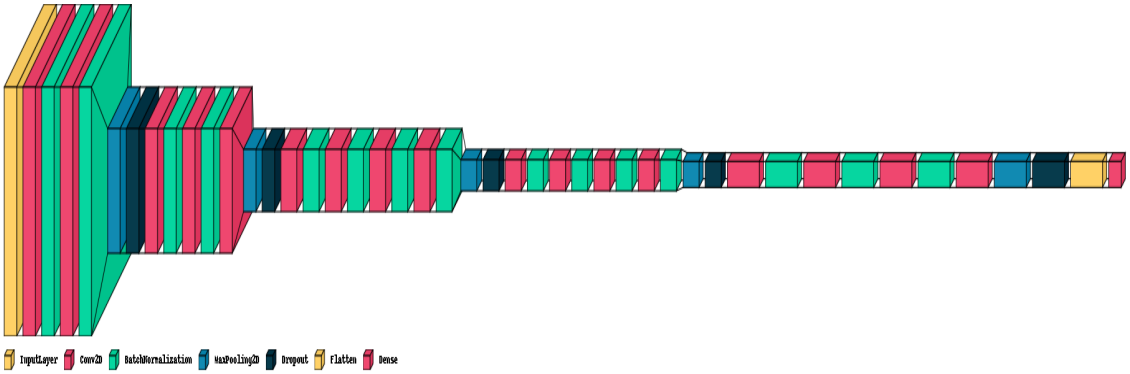

Figure 11: Proposed CNN architecture with 5 blocks

4.2 Techniques

4.2.1 Data Augmentation

Data augmentation is a set of techniques to artificially increase the amount of data by generating new data points from existing data. This includes making small changes to data or using deep learning models to generate new data points. (For example, Figure 12)

We researched and experimented with commonly used techniques in existing relating papers and achieved our best results with horizontal mirroring, ±10 degree rotations, ±10% image zooms, and ±10% horizontal/vertical shifting.

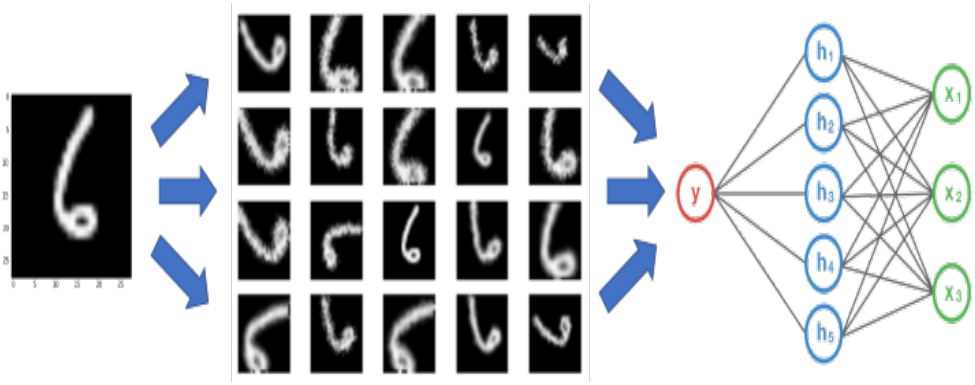

Figure 12: Data Augmentation in play

4.2.2 Batch Normalization

We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, and get 10 times or more improvement in the training speed.

Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). To explain covariance shift, let's have a deep network on cat detection. We train our data on only black cats' images. So, if we now try to apply this network to data with colored cats, it is obvious; we're not going to do well. The training set and the prediction set are both cats' images but they differ a little bit. In other words, if an algorithm learned some X to Y mapping, and if the distribution of X changes, then we might need to retrain the learning algorithm by trying to align the distribution of X with the distribution of Y. (Figure 13)

Batch Norm works in a very similar way in Convolutional Neural Networks. Although we could do it in the same way as before, we have to follow the convolutional property. In convolutions, we have shared filters that go along the feature maps of the input (in images, the feature map is generally the height and width). These filters are the same on every feature map. It is then reasonable to normalize the output, in the same way, sharing it over the feature maps.

In other words, this means that the parameters used to normalize are calculated along with each entire feature map. In a regular Batch Norm, each feature would have a different mean and standard deviation. Here, each feature map will have a single mean and standard deviation, used on all the features it contains.

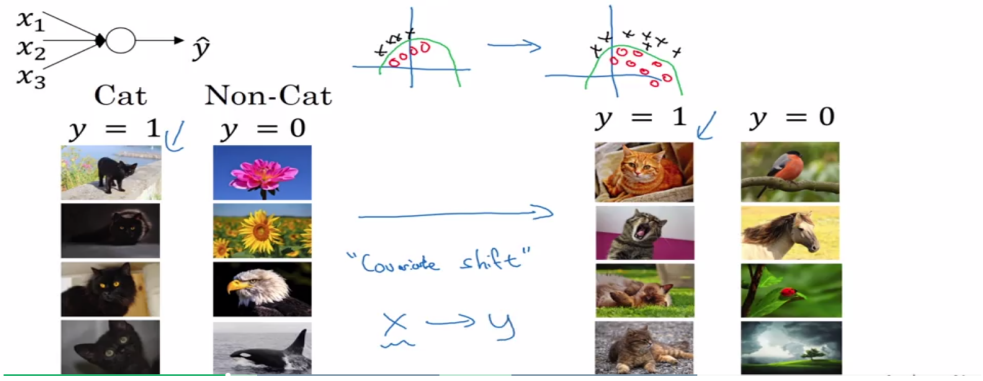

Figure 13: Batch Normalization in Neural Network

4.2.3 Dropouts

When a fully-connected layer has a large number of neurons, co-adaption is more likely to happen. Co-adaptation refers to when multiple neurons in a layer extract the same, or very similar, hidden features from the input data. This can happen when the connection weights for two different neurons are nearly identical.

This poses two different problems to our model:

- Wastage of machine's resources when computing the same output.
- If many neurons are extracting the same features, it adds more significance to those features for our model. This leads to overfitting if the duplicate extracted features are specific to only the training set.

So we need Dropout! Dropouts are the regularization technique that is used to prevent overfitting in the model. Dropouts are added to randomly switching some percentage of neurons of the network. When the neurons are switched off the incoming and outgoing connection to those neurons is also switched off. This is done to enhance the learning of the model. Dropouts are usually advised not to use after the convolution layers, they are mostly used after the dense layers of the network. It is always good to only switch off the neurons to 50%. If we switched off more than 50% then there can be chances when the model leaning would be poor and the predictions will not be good. (For example, Figure 14)

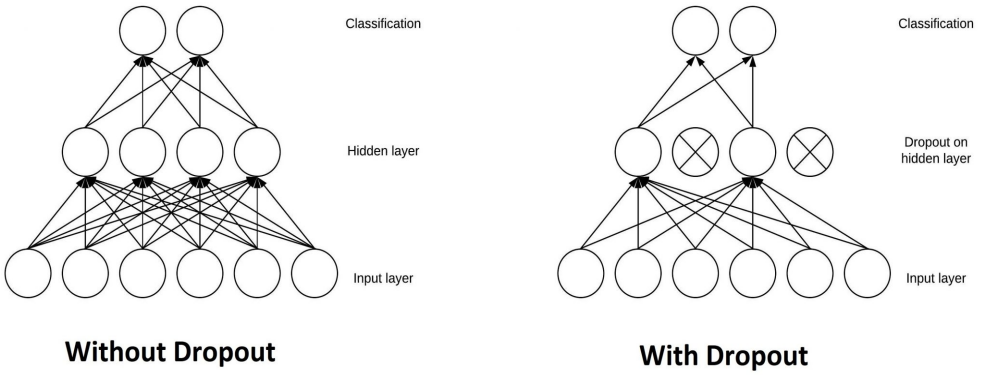

Figure 14: Dropouts being used in CNNs

5 System Design & Implementation

Firstly detect the objective of this project, which is implement a facial expression classification with CNN model then use it to detect emotion of a person in real time and more importantly, understanding how CNNs works and fit to this project. **Secondly**, we collected the datasets for training the model, and then we trained the CNN models with these ones. The output of this process is a collection of ".h5" model files. These files can be used to recognize emotion in realtime application. **Finally**, we run the program, detect face in sequential frames of the video stream using Haar Cascade model, and then use the detected face image to put in the trained model, ultimately the model output the label corresponding to the image, which then shows on the screen of the programm which emotion (label) is showing on the face in the video stream. (For example, Figure 15)

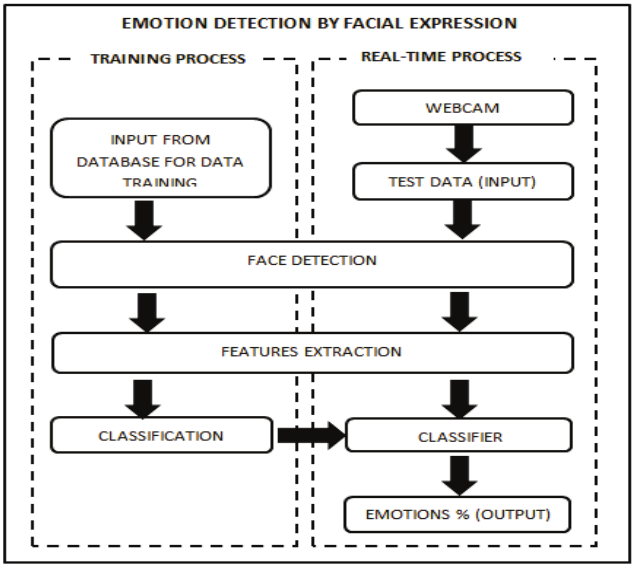

Figure 15: Workflow of real time emotion detection using CNNs model

6 Results & Discussion

6.1 FER2013 and Auxiliary data

By using 5-block VGG-Style architecture, training processes with 100 epochs, we get the following results on the datasets FER2013 and the auxiliary dataset in the table 2:

By reducing the number of class in the mixing data to 5 classes, the imbalanced problem also

Dataset	N.o samples	N.o. classes	N.o. parameters	Accuracy
FER2013	35,887	7	13,111,367	69.3%
FER + AffectNet Samples	57,277	5	13,110,341	71.4%

Table 2: Accuracy on the FER2013 and the auxiliary dataset

reduced a little bit just because the number of samples of class fear and disgust are quite smaller than the other's ones, so we obtain a better accuracy on the mixing dataset. Furthermore, larger amount of samples makes the accuracy of the model training higher.

With the dataset FER2013, the accuracy does not improve since the 60th epoch (Figure 16). With the mixing data, the accuracy does not improve since nearly 50th epoch, although in 85th epoch the model archive over 72% (Figure 17). Here are the training history of the models on FER2013 and the mixing data.

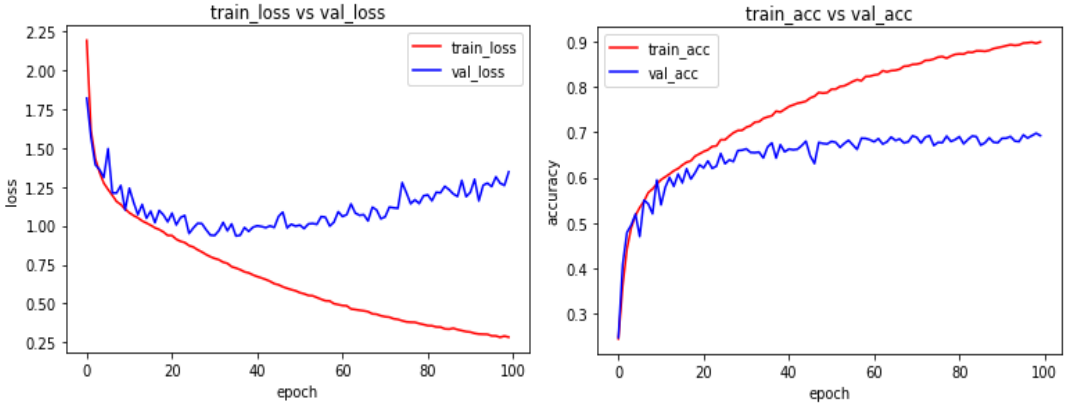

Figure 16: Loss and Accuracy on FER2013

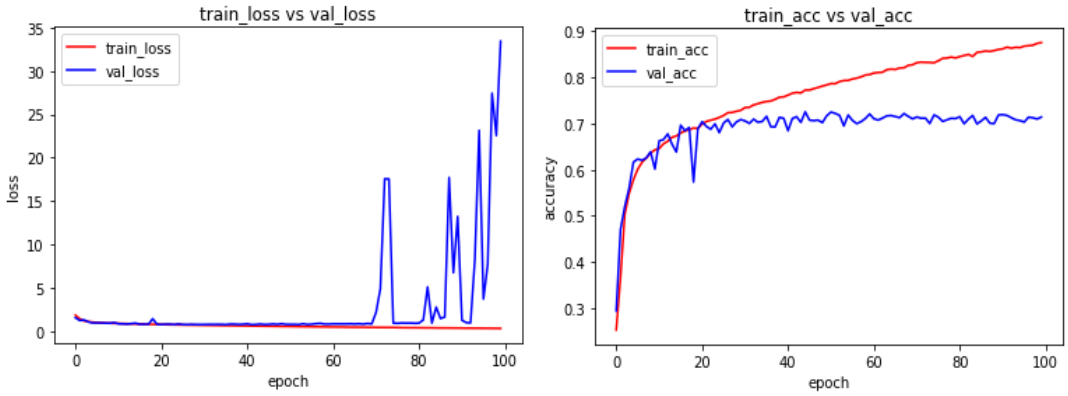

Figure 17: Loss and Accuracy on the mixing data FER + AffectNet samples

6.2 CK+ dataset

This dataset we only trained with 30 epochs due to the small size of the dataset. Because CK+ dataset doesn't have test set so we have to use k-fold cross validation witk k equals to 10 and ultimately the average accuracy we obtained after running 10 folds is 90.1% (See in table 3). The high accuracy 90.1% on CK+ makes sense because all the images in this dataset is balanced. It does not have so many noises in images like the ones in FER2013. All pose of faces in images are fixed, i.e not having any rotation, inclined pose, i.e straight forward face to the camera, ...

The accuracy of model when real time running program working is quite different. Despite the model has only accuracy of 69% and 71% on the test set but in the real time, they perform so much better than the model training with CK+ dataset. The main difference is the CK+ has little of nosie while the training set is so balanced, that leads to problem kind of overfitting of the model with this dataset. The real time frames extracted from webcam having so many noise in the background, that's why the model trained with CK+ detect so bad at test real time.

Dataset	N.o samples	N.o. classes	N.o. parameters	10-fold Cross Validation
CK+48	981	7	13,111,367	90.1%

Table 3: Accuracy on the CK+48 dataset

7 Conclusion

In conclusion, by using 5-block VGG-Style architecture and some regularization methods as well as the mixing data, we obtain the accuracy of our model with the highest one is 71.4%. This result clearly makes sense because we don't use any best standard CNNs models due to limitation of our computer power. In FER2013 dataset, the "digust" label is unbalanced and very smaller than the others's amount of data, which impacts the model's result and might lead to misclassifying emotions. Furthermore, when running the project in real time, the classifier sometimes detects "sad" label as neutral or fear, which is quite realistic in real life. Our team believe, if we have enough strong computer power, we are totally be able to implement the best CNNs architectures such as VGG16 or ResNet50, and then aim to obtain better results in accuracy.

Obviously, this result does not outperform the best benchmarks for facial expression classification problem, but through this project, our members in the team totally understand how to deploy a AI/Computer Vision project like this project. We understand how CNNs works and how to apply CNNs models to classify image in general and emotion in particular, which is, in our objective and opinions, the most important and valuable lessons.

8 Appendix A. Project Plan Management

Project Plan Management	
Task name	Maker
Finding and pre-processing datasets	Quach Phu Quy and Nguyen Luong Phuc
Finding related researches and references	Nguyen Luong Phuc
Researching CNN architectures	Nguyen Luong Phuc
Implementing and Evaluating models	Quach Phu Quy
Making Slide for representation	Quach Phu Quy
Writing report	Quach Phu Quy and Nguyen Luong Phuc
Representing	Quach Phu Quy and Nguyen Luong Phuc

9 Appendix B. Source code & Data

Source code & Data		
Item	Link	Description
Data	$https://tinyurl.com/2s3wj5wu$	FPT Edu Mail Require
Source Code	$https://tinyurl.com/2p9ymtud$	FPT Edu Mail Require

References

[1] Felipe Zago Canal, Tobias Rossi Müller, Jhennifer Cristine Matias, Gustavo Gino Scotton, Antonio Reis de Sa Junior, Eliane Pozzebon, and Antonio Carlos Sobieranski. A survey on facial emotion recognition techniques: A state-of-the-art literature review. *Information Sciences*, 582:593–617, 2022.

[2] Li Cuimei, Qi Zhiliang, Jia Nan, and Wu Jianhua. Human face detection algorithm via haar cascade classifier combined with three additional classifiers. In *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, pages 483–487, 2017.

[3] Huilin Ge, Zhiyu Zhu, Yuewei Dai, Biao Wang, and Xuedong Wu. Facial expression recognition based on deep learning. *Computer Methods and Programs in Biomedicine*, 215:106621, 2022.

[4] Aya Hassouneh, A.M. Mutawa, and Murugappan M. Development of a real-time emotion recognition system using facial expressions and eeg based on machine learning and deep neural network methods. *Informatics in Medicine Unlocked*, 20:100372, 06 2020.

[5] Yousif Khaireddin and Zhuofa Chen. Facial emotion recognition: State of the art performance on fer2013, 05 2021.

[6] Amil Khanzada, Charles Bai, and Ferhat Celepcikay. Facial expression recognition with deep learning, 04 2020.

[7] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, page 1–1, 2020.

[8] Wafa Mellouk and Handouzi Wahida. Facial emotion recognition using deep learning: review and insights. *Procedia Computer Science*, 175:689–694, 01 2020.

[9] Prameela Naga, Swamy Das Marri, and Raiza Borreo. Facial emotion recognition methods, datasets and technologies: A literature survey. *Materials Today: Proceedings*, 2021.

[10] Luan Pham, The Huynh Vu, and Tuan Anh Tran. Facial expression recognition using residual masking network. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4513–4519, 2021.

[11] Christopher Pramerdorfer and Martin Kampel. Facial expression recognition using convolutional neural networks: State of the art. 12 2016.

[12] Ketan Sarvakar, R. Senkamalavalli, S. Raghavendra, Santosh Jankatti, R. Manjunath, and Sushma Jaiswal. Facial emotion recognition using convolutional neural networks. *Materials Today: Proceedings*, 08 2021.

[13] Adrian Vulpe-Grigoraşi and Ovidiu Grigore. Convolutional neural network hyperparameters optimization for facial emotion recognition. In *2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–5, 2021.