Image Classification using CNN model

Priyanka Vyas
Department of Computer Science
Kent State University
Ohio, United States of America
pvyas2@kent.edu

Abstract—This study explains the use of convolutional neural networks (CNNs) for categorization of image dataset and emotion recognition dataset. Such tasks are crucial in contemporary computer vision and have a wide range of applications in industry. This study explores how a CNN model performs when applied to the image classification problem. This report further elaborates the CNN architecture used to optimize the recognition performance and metrics that defines the model performance like overall accuracy, confusion matrix and training/validation curves of accuracy and loss.

Keywords—convolutional neural network, image classification, Emotion Recognition.

I. INTRODUCTION

One of the fundamental challenges in computer vision is to classify an image into one of a number of predetermined categories. The fact that image categorization has so many applications makes it a classic problem. The ability of neural networks to learn and recognize patterns in large, complicated data sets has made them prominent in the fields of facial recognition and motion detection. Processing a lot of visual data in real-time is necessary for motion detection and facial recognition. Neural networks have ability to automatically extract relevant features from the input, and they can then use these features to categorize or detect different emotions or images.

Neural networks have been implemented in emotion recognition to categorize emotions based on physiological information, voice tones, and facial and vocal expressions. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid models are some of the neural network architectures that can be used for emotion recognition, according to a review study by Poria et al. (2017). The researchers discovered that deep learning models with good accuracy rates in emotion recognition tasks include CNNs and RNNs. Neural networks have been used to classify images into different categories, including objects, and animals. In a study by Krizhevsky et al. (2012), it was demonstrated that a deep neural network known as AlexNet outperformed conventional machine learning techniques. Overall, the literature supports the use of neural networks for image classification and emotion recognition because of their capacity for handling huge, complicated datasets and their capacity for learning and recognizing patterns in the data. Although Yann LeCun's LeNet (LeCun et al., 1989) and (LeCun et al., 1998) popularized convolutional neural networks, which have been known since at least the 1990s, it has only lately become practical to train these massive, intricate networks on extremely huge datasets. Modern GPU computing and parallel computing techniques have greatly boosted the capacity to train CNN models since (Steinkrau et al., 2005) demonstrated the value of employing GPUs for machine learning, which has led to their rise in academia and industry.

Further sections in this study with explain the data-preprocessing, model building, experimental evaluation, performance and comparative study performed under the scope of project.

II. DATA PROCESSING

A. Dataset introduction and pre-processing

The dataset used to model and train the network are image Datasets, these images were used as an input parameter and are fed into the CNN model to train and get the output as the label defined for that image. The images used as an input to CNN architecture are Emotion Recognition dataset (grayscale) and Image Classification i.e. collected data (RGB), therefore each channel is individually convoluted and then combined to form a pixel. The details about the two dataset is as below:

1.Emotion Recognition Dataset: This dataset is available on Kaggle.com and contains 7 different categories like "angry", "disgust", "fear", "happy", "neutral", "sad", "surprise". The dataset is available in two folders i.e. train and validation.

2.Image Classification Dataset: This is a collected dataset wherein every individual took pictures from different type of devices. The collected dataset was cleaned and then split-in sets to first create, train and evaluate the CNN model and understand its performance.

Since the dataset is collected from different devices, it offers a lot of variations in terms of image quality, resolution, lighting, color balance and noise. These variations introduce noise in the data and makes it difficult for CNN model to learn the underlying pattern in the image. Hence, to mitigate this problem the data preprocessing technique used in this approach to normalize the data and ensure effective learning.The most commonly adopted techniques are:

1.Image Normalization: This is achieved by scaling the pixel images to a fixed range, usually between 0 and 1 or -1 and 1. This is useful to reduce the effect of variation in image brightness and contrast.

2.Data Augmentation: By performing various image transformations, including rotations, flips, and distortions, on the already-existing images, so that more training data is created. This increases training data diversity and reduces overfitting.

3.Image resizing and Cropping: This involves scaling or cropping the images to a set size so that the CNN model can process them. This is achieved without distorting the information while also maintaining the original images' aspect ratio.

To explore and visualize the data in the dataset, firstly, all the useful libraries like matplotlib,os, numpy, pandas, seaborn are imported and then using the load_img function, images from the train set are loaded for the respective classes like expression "happy" and 'Class_1_Touch' are displayed using the matplotlib library.

To pre-process the data in this project the following two methods are used:

Approach 1: The path for respective training, validation and test folders of the stored data is defined. '**ImageDataGenerator**' class is used to rescale the pixel values to a range between 0 and 1, applying shear range, zoom range and horizontal flip augmentation. The 'rescale' parameter is set to 1/255, which scales the pixel values from the range of 0-255 to the range of 0-1. The 'train_datagen' uses the augmentation techniques like range and flip, 'valid_datagen' and test_datagen' rescale the data, 'target_size' specifies size to which all images are resized to 224x224.

Emotion recognition dataset details:
Train: Found 28821 images belonging to 7 classes.
Test: Found 7066 images belonging to 7 classes.

Image classification dataset details:
Train:Found 12466 images belonging to 3classes
Valid: Found 1874 images belonging to 3 classes.
Test: Found 4105 images belonging to 3 classes.

Approach 2: Imported all necessary libraries, including opencv for image processing, numpy for numerical operations and scikit-learn for label encoding and one-hot encoding. Defined path to folder containing the images. To store the images and their related labels, the code initializes empty lists. Each class label and each image file within the training folder are iterated through by a loop, which loads the image and applies some preprocessing to it before adding it to the list of images and the list of labels. Using label encoding, the labels are transformed from their original string format to integer format.

To construct a binary vector representing each class label, the labels are then one-hot encoded. The information is changed into NumPy arrays and then reshaped into the appropriate input shape for a neural network. The train_test_split function from scikit-learn is then used to divide the data into training and test sets. The preprocessed image arrays are appended to x_train, and the corresponding label is appended to y_train. The labels are converted from strings to integers using LabelEncoder(). The labels are then one-hot encoded using **OneHotEncoder**(). The data is converted to numpy ndarrays and the shapes of the data are printed. The data is split into training and testing sets using train_test_split()

III. Model Training and Evaluation

A. Model Design and Experimental Result

Model 1:

This method uses approach 1 for data pre-processing, the model uses image classification dataset and begins with an input layer that accepts images with a resolution of 224x224 and three channels (RGB). Following a ReLU activation function to provide non-linearity, the first layer is a Convolutional layer with 32 filters (also known as feature maps), each with a 3x3 kernel size. Following this layer is a Max Pooling layer with a pool size of 2x2, which aids in reducing the output's spatial dimensions (width and height) and the model's parameter count. The following layers are comparable to the first, with an increase in the number of filters (64, 128, 256), each followed by a Max Pooling layer and a ReLU activation function. In order to prepare for the Fully Connected layers, the output of the last Max Pooling layer is then flattened to a 1D vector. There are 128 units and a ReLU activation function in the first completely connected layer. Following this, a dropout layer is included to help prevent overfitting by randomly setting 50% of the outputs to zero during training. The chance that each input image belongs to each class is provided by a softmax activation function in the output layer, which is made up of 3 units (for the 3 classes in this case).

Layer (type)	Output Shape	Param #
conv2d_45 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_44 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_46 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_45 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_47 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_46 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_48 (Conv2D)	(None, 24, 24, 256)	295168
max_pooling2d_47 (MaxPooling2D)	(None, 12, 12, 256)	0
flatten_11 (Flatten)	(None, 36864)	0
dense_26 (Dense)	(None, 128)	4718720
dropout_31 (Dropout)	(None, 128)	0
dense_27 (Dense)	(None, 3)	387

Total params: 5,107,523
Trainable params: 5,107,523
Non-trainable params: 0

Fig1: Model summary for image classification dataset

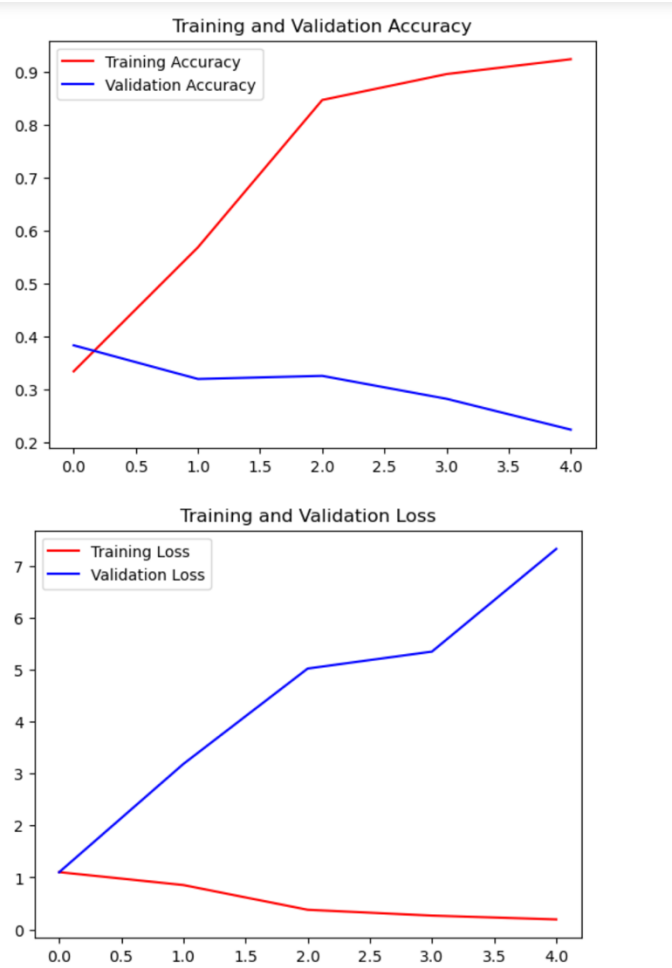

Fig 2: Model accuracy and Model loss curve for epoch 5 and batch size 32 using image classification dataset.

The above fig 2 the CNN model undergoes training and evaluation over 5 epochs. A training set was used to train the model, and a validation set was used to confirm its accuracy. The model was assessed on the training and validation sets throughout each epoch in order to determine its accuracy and loss. The output shows the loss and accuracy numbers for each period. The evaluate () function was used to test the model once it had been trained, and it returns the test loss and accuracy. The result shows the test loss and accuracy values. Due to the low accuracy on the test set and the significant loss, it appears that the model was underperforming. It can be a sign of model overfitting.

```
[[495 315 472]
 [456 318 513]
 [566 378 592]]
```

The above confusion matrix has 3 rows and 3 columns. The (i, j)-th entry of the matrix represents the number of times that the true label was i and the predicted label was j. The output shows that the model made 495 true positive predictions for class 0, 318 true positive predictions for class 1, and 592 true positive predictions for class 2. It also made 315 false negative predictions for class 0, 456 false negative predictions for class 1, and 378 false negative predictions for class 2.

Using emotion recognition dataset, the model starts with a Conv2D layer with 64 filters, followed by a MaxPooling2D layer. The process is repeated with two more Conv2D and MaxPooling2D layers, with 128 and 512 filters respectively. The output is then flattened and passed through two Dense layers with 256 and 512 neurons respectively. Dropout layers are added after each dense layer to prevent overfitting. Finally, the output is passed through a Dense layer with the number of classes and a softmax activation function to get the final class probabilities. The model summary shows the number of parameters in each layer and the total number of trainable parameters.

Layer (type)	Output Shape	Param #
conv2d_32 (Conv2D)	(None, 48, 48, 64)	640
max_pooling2d_32 (MaxPooling2D)	(None, 24, 24, 64)	0
conv2d_33 (Conv2D)	(None, 24, 24, 128)	204928
max_pooling2d_33 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_34 (Conv2D)	(None, 12, 12, 512)	590336
max_pooling2d_34 (MaxPooling2D)	(None, 6, 6, 512)	0
conv2d_35 (Conv2D)	(None, 6, 6, 512)	2359808
max_pooling2d_35 (MaxPooling2D)	(None, 3, 3, 512)	0
flatten_5 (Flatten)	(None, 4608)	0
dense_12 (Dense)	(None, 256)	1179904
dropout_8 (Dropout)	(None, 256)	0
dense_13 (Dense)	(None, 512)	131584
dropout_9 (Dropout)	(None, 512)	0
dense_14 (Dense)	(None, 7)	3591

Total params: 4,470,791
Trainable params: 4,470,791
Non-trainable params: 0

Fig 3: Model summary for emotion recognition dataset

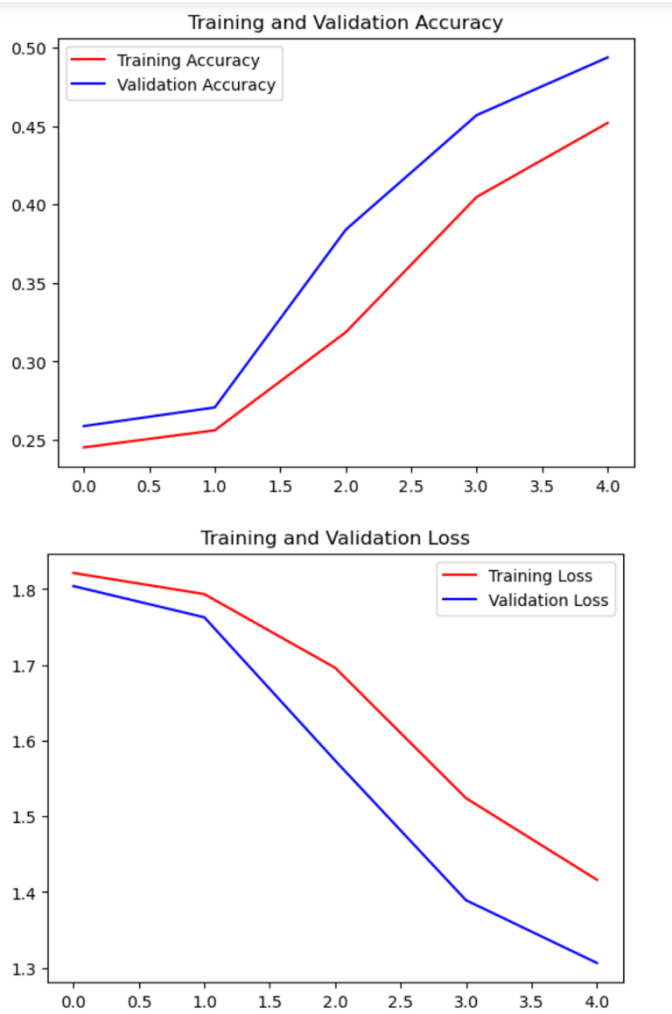

Fig4: Model accuracy and Model loss curve for epoch 5 and batch size 32 using image classification dataset.

From the above curve, it can be interpreted that overall, the model appears to be underfitting, meaning that it is not able to fit the training data well enough to generalize to new data. This could be due to a variety of reasons, such as insufficient training data, overly simplistic model architecture, or insufficient training time.

```
[[129   0  28 237 266 172 128]
 [  6   0   3  36  32  17  17]
 [125   0  23 266 249 201 154]
 [237   0  37 484 518 314 235]
 [132   0  34 327 337 223 163]
 [146   0  29 279 313 222 150]
 [117   0  24 190 215 139 112]]
```

The confusion matrix above is an array of numbers. For example, the number in the first row and first column, 129, represents the number of instances where the true class was 0 and the predicted class was also 0. Similarly, the number in the first row and second column, 0, represents the number of instances where the true class was 0 but the predicted class was 1.

Model 2:

This method uses approach 2 for data pre-processing and the model is designed using Keras sequential API for multiclass classification with 3 output classes. It uses TensorFlow and Keras to create and compile the model. With a kernel size of 3x3, three convolutional layers, each with 32, 64, and 64 filters, and a ReLU activation function. The size and quantity of channels in the input image are represented by the input form of the first layer, which is (28, 28, 1). Two MaxPooling Layers with 2x2 pools in each. To transform the output from the convolutional layers into a 1D array, one flattening layer is used and 64 units in one Dense Layer with ReLU activation. The probability distribution for the three output classes is represented by one output layer with three units and a softmax activation function.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320

max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)        0
2D)

conv2d_4 (Conv2D)            (None, 11, 11, 64)        18496

max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)          0
2D)

conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928

flatten_1 (Flatten)          (None, 576)               0

dense_2 (Dense)              (None, 64)                36928

dense_3 (Dense)              (None, 3)                 195

=================================================================
Total params: 92,867
Trainable params: 92,867
Non-trainable params: 0
_________________________________________________________________
```

Fig5: Model summary for image classification dataset

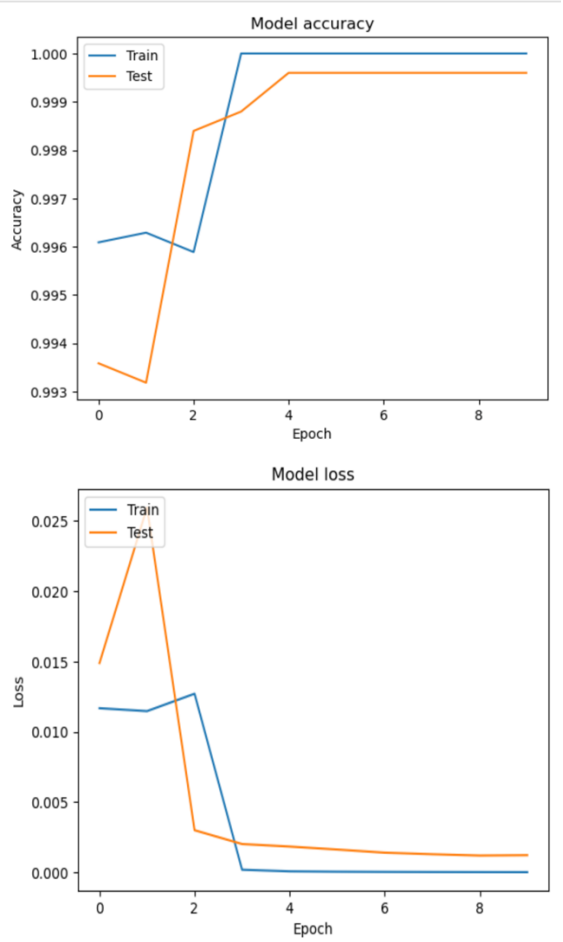

Fig6: Model accuracy and Model loss curve for epoch 10 and batch size 32 using image classification dataset.

As given in figure above the output represents the accuracy and loss of the model after training for 10 epochs, and evaluating on the test dataset. During the training process, the model was trained on batches of data, with 312 batches in total, where each batch contained 312 samples. The loss function used during training was the categorical cross-entropy, and the optimizer used was likely the default one (Adam). The output shows that in the final epoch (epoch 10), the training process achieved a very low loss value of 1.51e-04 and a perfect accuracy of 1.0. This means that the model was able to accurately classify all the training samples in this epoch.

The confusion matrix shows the output as below:

```
[[868   1   0]
 [  0 822   0]
 [  1   0 802]]
```

In the output, it can be observed that the confusion matrix has three rows and three columns, corresponding to the three classes in the dataset. The diagonal elements of the matrix represent the number of correct predictions for each class. For example, the element at position (0,0) represents the number of samples that are actually of class 0 (the first row), and were correctly predicted to be class 0 (the first column). The off-diagonal elements represent misclassifications, where the predicted class does not match the true class, meaning the element at position (0,1) represents the number of samples that are actually of class 0 (the first row), but were incorrectly predicted to be class 1 (the second column).

For emotion recognition dataset, a Conv2D layer with 32 filters, a 3x3 kernel, and the ReLU activation function is the first layer added. The input shape is (48, 48, 1), indicating that the input photos are 48x48 grayscale images. The following layer is a MaxPooling2D layer with a pool size of 2x2 and another Conv2D layer with 32 filters and a 3x3 kernel size. To avoid overfitting, a dropout layer with a 0.25 dropout rate is introduced. Following the addition of another Dropout layer with a dropout rate of 0.25, two additional sets of Conv2D and MaxPooling2D layers with 64 filters and a 3x3 kernel size are added. A further MaxPooling2D layer, a Conv2D layer with 128 filters and a kernel size of 3x3, and a Dropout layer with a dropout rate of 0.25 are added. To create a 1D array from the output, the Flatten() layer is added. Then another Dropout layer with a dropout rate of 0.5 is added, followed by a Dense layer with 256 units and the ReLU activation function. For multi-class classification, the 7-unit output layer with the softmax activation function is added last.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 46, 46, 32)        320

 conv2d_6 (Conv2D)           (None, 44, 44, 32)        9248

 max_pooling2d_3 (MaxPooling  (None, 22, 22, 32)       0
 2D)

 dropout_4 (Dropout)         (None, 22, 22, 32)        0

 conv2d_7 (Conv2D)           (None, 20, 20, 64)        18496

 conv2d_8 (Conv2D)           (None, 18, 18, 64)        36928

 max_pooling2d_4 (MaxPooling  (None, 9, 9, 64)         0
 2D)

 dropout_5 (Dropout)         (None, 9, 9, 64)          0

 conv2d_9 (Conv2D)           (None, 7, 7, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 3, 3, 128)        0
 2D)

 dropout_6 (Dropout)         (None, 3, 3, 128)         0

 flatten_1 (Flatten)         (None, 1152)              0

 dense_2 (Dense)             (None, 256)               295168

 dropout_7 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 7)                 1799

=================================================================
Total params: 435,815
Trainable params: 435,815
Non-trainable params: 0
_________________________________________________________________
```

Fig 7: Model summary for emotion recognition dataset

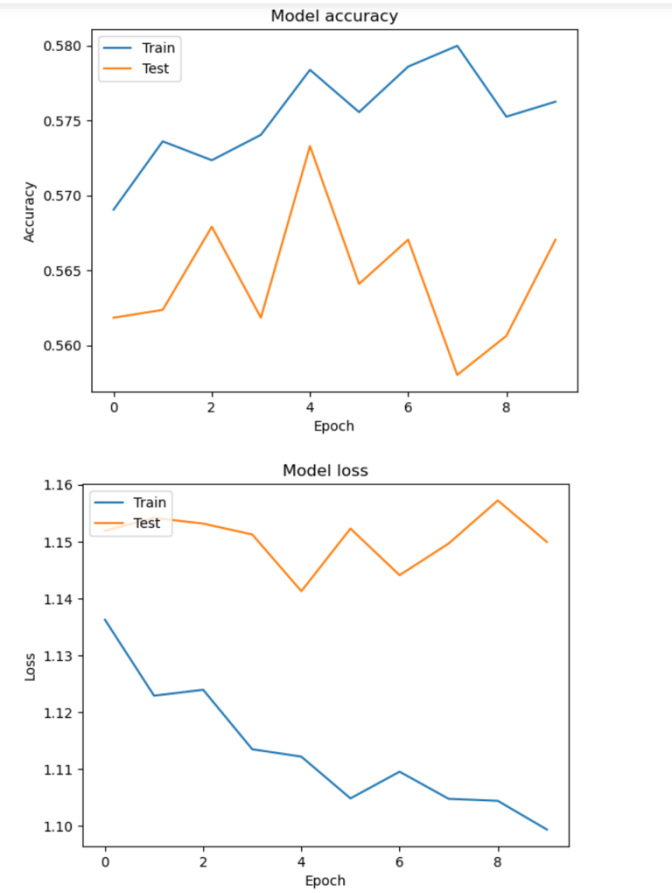

Fig 8: Model accuracy and Model loss curve for epoch 10 and batch size 32 using emotion recognition dataset.

From the above curves, it can be interpreted that the model is not overfitting or underfitting but it is fitting to some extent. The training accuracy of the model is 0.5762 which indicates that the model is able to fit well to the training data. However, the difference between the training accuracy and validation accuracy (0.5762 - 0.5670 = 0.0092) is relatively small which suggests that the model is not overfitting to the training data. The test accuracy of the model is 0.5670, which is slightly lower than the validation accuracy of 0.5670. This suggests that the model is able to generalize well to new data and is not overfitting to the validation set. Overall, the model seems to be fitting reasonably well to the data, with no signs of overfitting or underfitting. However, the accuracy achieved by the model may depend on the particular problem and may need to be improved depending on the specific use case.

```
[[ 375    2   38   76  121  133   24]
 [  31   18    8    9    2   23    2]
 [ 119    1  183   75  119  206  120]
 [  50    1   16 1214  108  100   18]
 [  65    0   36  104  592  180   16]
 [ 109    0   70   86  231  446   13]
 [  21    0   51   56   38   18  441]]
```

The above confusion matrix has 7 rows and 7 columns, indicating that there are 7 classes. The values in the matrix indicate the number of instances that belong to each true label and were predicted as the corresponding predicted label. For example, the value in the first row and first column is 375, which means that 375 instances that belong to class 0 (the first class) were correctly predicted as class 0. The value in the first row and second column is 2, which means that 2 instances that belong to class 0 were predicted as class 1.

B. Model Performance Analysis

Model 1 (Approach 1- imagedatagen)		
Dataset	Image Classification	Emotion Recog
Parameters	5107523	4470791
Epochs	5	5
Batchsize	32	32
Accuracy	Low (31%)	Low (49%)
Performance	Underperforming	Underfitting
Training Time	17 min 25 sec	14 min 30 sec
Model 2 (Approach 2- One hot encode)		
Parameters	92867	435815
Epochs	10	10
Batchsize	32	32
Accuracy	99%	56%
Performance	Good	Good
Training Time	217ms per epoch	2 sec

The above summary in table shows the overall model performance analysis and the findings on the basis of the approach adopted for data pre-processing (once the data was cleaned and regrouped into required classes and to be used as a path folder or input) and CNN models created, trained and compiled for the image classification problem. It is also observed that the accuracy of the Model 1 increases with the increases in the number of epochs, but the computation time is too high and heats the device as well. Hence, Approach 2_One hot coding can be preferred over Approach 1_Imagedatagen as it can be observed One hot encoding is the crucial process of transforming the variables in categorical data that will be fed into machine and deep learning algorithms, resulting into improving predictions and model classification accuracy. With this method of encoding, a new binary feature is created for each potential category, and the feature of each sample that originally belonged to that category is given a value of 1. In training for learning approaches, one hot encoding is a crucial step in the feature engineering process. This method i.e. approach 2 (One hot coding) improves the predictions, computation time, performance as well as classification accuracy of model.

IV. Experimental environment

Python, its libraries and TensorFlow is used to create, compile and train the neural network. To run the code in Jupyter, through anaconda navigator, an environment named 'TensorFlow' with necessary installed packages was first created to create the experimental environment for project. Apple M1 pro with GPU integrated on the same chip as CPU is the machine used to work on the project. Some limitations that were experienced while working on the data was to re-group and cleaning the dataset. It interrupted the model training and also provided false or no output.

Conclusion

Through this study it is learned and observed that in CNN model, the performance and model depth are strongly correlated, and the performance of CNN models can be significantly improved by fine-tuning the hyperparameters. Underfitting or Overfitting affects the mapping of input to output and in turn affects the overall performance of the model leading to high computational speed. Through this study it was aimed to strike an optimum balance while developing the model and this could be achieved by a balance between the bias and invariance to avoid both underfitting and overfitting scenarios.

References

- Poria, S., Cambria, E., Hazarika, D., Majumder, N., & Zadeh, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion, 37, 98-125.
- Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
- LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.
- LeCun, Yann, Bottou, L´eon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
- Steinkrau, Dave, Simard, Patrice Y, and Buck, Ian. Using gpus for machine learning algorithms. In null, pp. 1115–1119. IEEE, 2005.
- Huang, G. et al. (2017) 'Densely Connected Convolutional Networks', 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, CVPR, pp. 2261–2269. doi: 10.1109/CVPR.2017.243