RESEARCH ARTICLE OPEN ACCESS

Emotion Recognition Through Facial Expressions Using CNN

Ankur Jyoti Sarmah[1] Nitin Bhattacharyya[2] Ritik Kumar Jain[3] Priyankar Maroti[4]
[1] *Assistant Professor, Assam Engineering College, Guwahati, Assam, India*
[2] *Final Year B.Tech, Assam Engineering College, Guwahati, Assam, India*
[3] *Final Year B.Tech, Assam Engineering College, Guwahati, Assam, India*
[4] *Final Year B.Tech, Assam Engineering College, Guwahati, Assam, India*

ABSTRACT:
Emotion recognition is a very fundamental part of human interaction. This paper sought out to develop a machine learning model based on the FER2013 dataset through the use of traditional CNN architecture that recogniz-es the seven basic emotions with the help of facial expressions. To increase the accuracy of the model various procedures were carried out such as hyperparam-eter optimization using Random Search from the Keras tuner library, testing out different number of convolution layers and different optimizers. The best model gave 69.55% test accuracy, which contained 8 convolution layers and used Ad-am optimizer.

Keywords: CNN,KerasTuner,RandomSearch,AdamOptimizer,FER2013

Date of Submission: 06-02-2023 Date of acceptance: 18-02-2023

I. INTRODUCTION

The most common way to recognize emotions is through changes in expressions on a face. Humans can easily and instantly classify simple and complex emotions from a facial expression but for machines it is indeed a hard task to accomplish regardless of the complicated nature of the task

The building blocks to make a machine recognize emotions is to first make it possible for the machine to recognize the seven basic emotions. The seven basic emotions are Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. Convolutional neural networks have proved to be useful in this regard.

The FER2013 is a very popular dataset used to classify the seven basic emotions. In this paper a basic CNN machine learning model was first created from scratch and used to classify the seven basic emotions. This model gave a test accuracy of 65%. The further goal then became to optimize this model to achieve a better accuracy just though traditional CNN without using additional data or other methods such as appli-cation of pretrained models or ensemble model. This paper takes a dive into the steps taken to improve the model accuracy and the conclusions made while engaging in these steps

II. Previous work

The Convolutional Neural Network (CNN) has shown considerable promise in Image Processing. Convolutional layers, maximum pooling layers, activation layers, batch normalization layers, drop out layers, and fully linked layers make up a CNN. All of these layers help to speed up the training process. To down sample the inputs and aid in generalization, various pooling approaches like as average pooling and max pool-ing are used. To avoid overfitting, dropout, regularization, and data augmentation are used. To prevent gradient vanishing and bursting, batch normalization was created. There has also been a lot of work put into developing different optimization algo-rithms that are employed in training. Despite the lack of a systematic theoretical guideline for selecting an optimizer, practical data show that using the right optimiza-tion technique can significantly increase the performance of a model.

Stochastic gradient descent is the most widely used optimizer (SGD). It's a simple method for updating a model's parameters based on the gradient of a single data point. By adjusting the learning rate and introducing gradient momentum, Adam combines the benefits of AdaGrad and RMSProp.

In the training of a CNN, the learning rate is a significant element. Oscillations around the minima or loss divergence could result from a high learning rate. A low learning rate would greatly slow the model's convergence and potentially trap it in a non-optimal local minimum. A learning rate scheduler that varies the learning rate during training time-based decay reduces the learning rate either linearly or exponen-tially as the iteration number grows is a regularly used approach. After a fixed number of epochs, step decay reduces the learning rate by a factor. During training, an adap-tive learning rate schedule attempts to automatically alter the learning rate based on local gradients.

CNNs have evolved into a useful tool for image-related tasks as a result of numerous advancements. Ian Good fellow et al. created the FER2013 database, which contains photos with facial emotions, for a Kaggle competition on facial emotion detection in 2013. It contains 35887 photos with facial emotions in an 8-bit grayscale format of 48x48 pixels, separated into three categories: 28709 training data, 3589 test data, and 3589 validation data. All of the images in the database have been classified to fit into one of the seven primary categories of face emotions: anger, disgust, fear, happiness, sad, surprised and neutral. The FER2013 dataset has been used to compare model performance in emotion recognition.

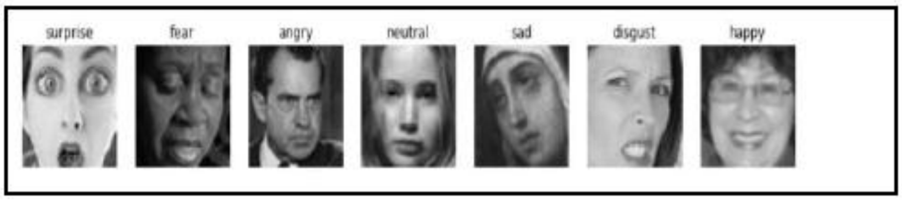

Fig. 1. Seven Basic Emotions

Many CNN variants have achieved remarkable results, with classification accuracy ranging from 70% to 76%. For instance [Ian Goodfellow et.al.][6] used Ensemble ResMaskingNet of 6 CNNs achieved an accuracy of 76.82%, [Khanzada Amil et al.][1] used different CNNs and Ensemble them to improve the performance to achieve an accuracy of 75.8%. [Georgescu et al.][2] used Fusion CNN + Bag of visual words (BOVW) and achieved 75.42% accuracy. [Pramerdorfer et al.][3] used another ensemble of CNNs to achieve an accuracy of 75.2. Different Pre-Trained models were also used by the researchers to achieve high accuracy for the dataset.

Ensembling has demonstrated outstanding performance in terms of performance en-hancement. However, in order to improve ensemble performance even further, we intend to first optimize the ensemble's building blocks, a single network. Other re-search has attempted to improve their model's performance on FER2013 by incorpo-rating auxiliary training data; however, this is beyond the scope of this paper.

Table 1. Results on Fer2013 Dataset.

Model	Network Type	Accuracy %
Ian et. al. [6]	Ensemble ResMaskingNet with 6 other CNNs	76.82
Khanzada Amil et al. [1]	Ensemble CNN	75.8
Georgescu et al.[2]	Fusion CNN + BOVW	75.42
Pramerdorfer et al. [3]	Ensemble CNN	75.2
Kim et al. [4]	Ensemble CNN	73.73
Connie et al. [5]	Hybrid CNN-SIFT	73.4
VGG [2]	CNN	72.7
ResNet [2]	Residual	72.4
Grigoraşi et. al. [7]	CNN hyperparameters optimizations	72.16
Hua et al. [8]	Ensemble CNN	71.91

Inception [2]	Deep CNN	71.6
Tang et. al. [9]	CNN + SVM	71.16
Kaiming et. al. [10]	Residual	69.7
VGG-16 [2]	CNN	68.2
Ionescu et al. [11]	BOVW	67.48

III. Model Creation And Optimization

A basic CNN model was build up from the scratch which used 4 convolution layers, two dense layers, Adam optimizer and batch size 64. This model gave an initial test accuracy of 65%. In the model loss plot, it is observed that the train loss decreases smoothly over the total length of the epochs while the test loss becomes constant with many fluctuations which tells that the model is not able to generalize well from the train data when predicting the test data. Over fitting occurs when the model attempts to learn too many details in the training data while also accounting for noise in the training data. As a result, model performance on unseen or test datasets is very poor. As a result, the network is unable to generalize the features or patterns found in the training dataset. Due to over fitting, the accuracy of our model on the test set goes up to 65% and then continues to fluctuate at that point whereas on the train dataset the accuracy goes upto 92% smoothly over the total length of the epochs.

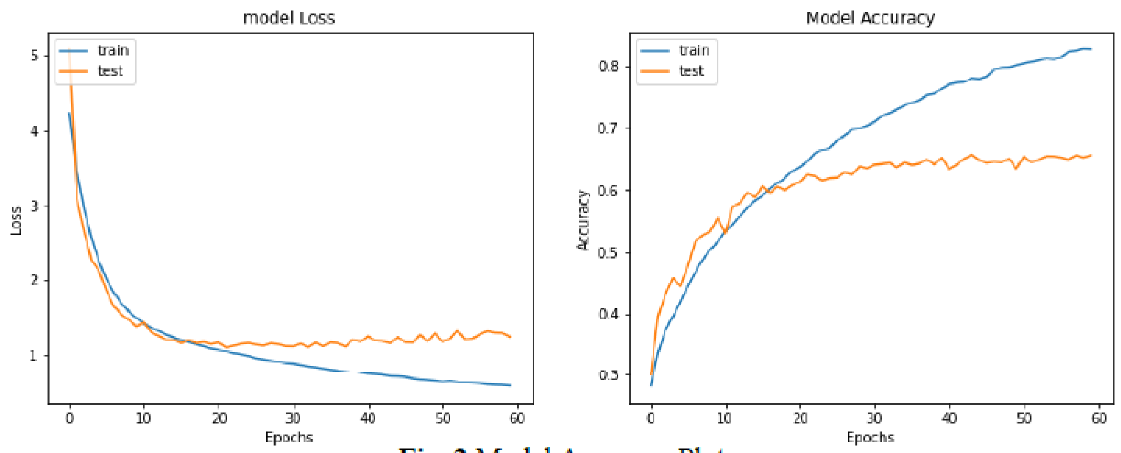

Fig. 2 Model Accuracy Plot

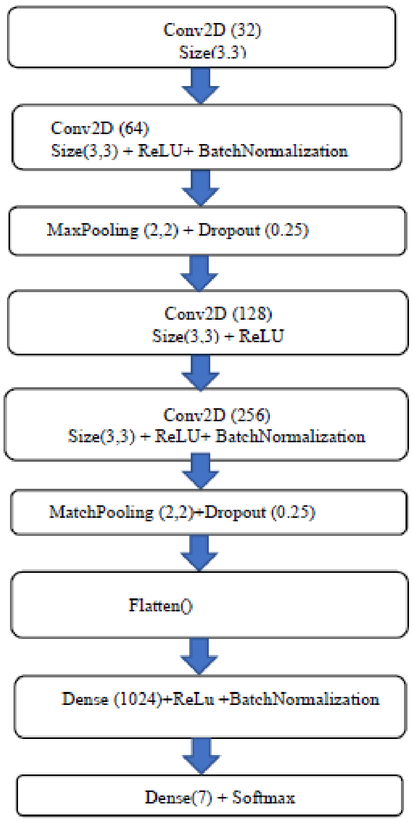
Conv2D (32)
Size(3,3)
Conv2D (64)
Size(3,3) + ReLU+ BatchNormalization
MaxPooling (2,2) + Dropout (0.25)
Conv2D (128)
Size(3,3) + ReLU
Conv2D (256)
Size(3,3) + ReLU+ BatchNormalization
MatchPooling (2,2)+Dropout (0.25)
Flatten()
Dense (1024)+ReLu +BatchNormalization
Dense(7) + Softmax

Fig. 3 Model Architecture

In order to increase the accuracy through hyperparameter optimization various steps were taken. First, from previous works it was seen that best accuracy came about with 5 convolution layers or 8 convolution layers. So hyperparameter optimization were done for both for both 5 convolution layer model and 8 convolution layer model. Sec-ond, through experimenting with various batch sizes, notably 32,64 and 128,it was found that 32 batch size yielded the best results. So the final batch size was settled to be 32. Third, trying out different optimizers. The two optimizers taken up for the pur-poses of this paper were SGD with nesterov and Adam.

3.1 Hyperparameter Optimization

For the purposes of hyperparameter optimization, Keras Tuner library was used. This library contained Random Search which can be used to make combinations of parameters from a given search space to reach the best combination for the best performance.

The search space defined was as follows:
1. No. of kernels in the first convolutional layer minimum 32 and maximum 256 with step size of 32
2. Maximum no. of kernels in the convolutional layer other than the first layer minimum 64 and maximum 512 with step size 64
3. Dropout in the convolutional layer's minimum 0 and maximum 0.5 with step size 0.05
4. Dropout in the dense layer's minimum 0 and maximum 0.5 with step size 0.05

A total of 100 trials were carried out in search of the best combination through random search. Each trial ran for 20epochs and its validation accuracy noted. It should be noted that the best model with 5 convolution layers had a validation accura-cy of 62% after 20epochs and the best model with 8 convolution layers had a valida-tion accuracy of 61.9% after 20 epochs. This was done with the Adam optimizer.

3.2 Mathematical Equations

Activation Functions:

The output of a neural network can be determined using activation functions. These functions are associated with each neuron in the neural network and determine wheth-er it should be active or not based on whether the input of each neuron is important to the model's prediction. The activation function also allows to normalise each neuron's output to a range of 1 to 0 or -1 to 1. It is known neural networks are occasionally trained on millions of data points. As a result, the activation function must be efficient enough to reduce computation time and increase performance. Activation functions used in this paper are

ReLu Function:
ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$.
ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values
SoftMax Activation Function:
SoftMax function is a type of Activation function, it is particularly beneficial when dealing with classification problems. When dealing with many classes, this function is commonly used. It would divide by the sum of the outputs and squeeze the outputs for each class between 0 and 1. The SoftMax function is best employed in the classifier model's output layer, where we're attempting to define the class of each input using probabilities. 14 Note that for Binary classification, we can utilise both the sigmoid and SoftMax activation functions, which are both as approachable. When dealing with multi-class classification problems, however, we frequently combine SoftMax and cross-entropy.

$$\frac{e^{y_i}}{\sum_{i=1}^{k} e^{y_i}}$$

Optimizers:

A neural network optimizer is a function or algorithm that changes the weights and learning rate of the network. As a result, it helps to reduce total loss while also im-proving precision. The optimizers applied in this paper are

Adam [Kingma et. al.][12]:

Adaptive Moment Estimation (Adam) is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponential-ly decaying average of past gradients similar to momentum: There were a lot of terms in the original paper [Kingma et. al.][12]. So below is the simplistic way:

$$\theta_{t+1} = \theta_t - \left(\frac{n\,.\,\hat{m}}{\sqrt{\hat{v}_t}\,.\,\varepsilon}\right)$$

Where,

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = (1-\beta_1)g_t + \beta_1 m_{t-1}$$

$$v_t = (1-\beta_2)g_t^2 + \beta_2 v_{t-1}$$

- Epsilon ϵ, which is just a small term preventing division by zero. This term is usually 10−810−8.
- Learning rate η (although it's αα) in the paper.
- The gradient gg, which is still the same thing as before:

$$g = \nabla J(\theta_{t,i})$$

SGD with Nestrov [Ruder, Sebastian][13]:

Nesterov accelerated gradient (NAG) is an optimization technique that is used during the training of neural networks.

Nesterov's Accelerated Gradient Descent performs a simple step of gradient descent to go from to, and then it 'slides' a little bit further than in the direction given by the previous point just like a ball rolling down a hill analogy.

$$V_t = \gamma\, V_{t-1} + \eta \nabla_\theta J(\theta - \gamma V_{t-1} - 1)$$

$$\theta = \theta - V_t$$

IV. Result Analysis

In order to test the performance of our proposed model on FER2013 dataset, the model was trained on 150epochs with Adam as the optimizer with constant learning rate of 0.0001. The dataset was augmented in form of zoom range of 0.1, rotation range of ±10° rotations, width shift range and height range of 0.1 with horizontal flip set to True. The batch size for the dataset was set to 32.

After the best models for 5 convolution layers and 8 convolution layers, the 5 convolution layer model was trained from the start for 300 epochs and it gave the max test accuracy of 70.58%.

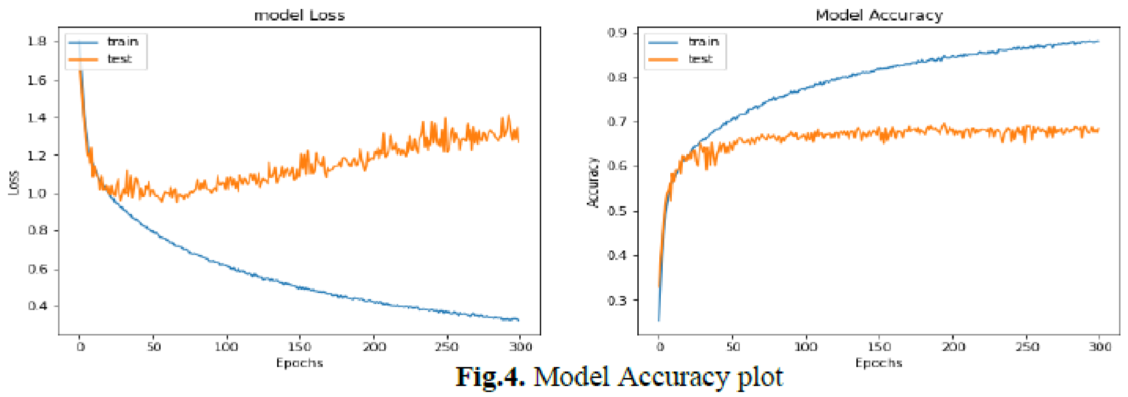

Fig.4. Model Accuracy plot

Although this model had a high accuracy it is observed that it has massive overfitting and was not able to generalize the features properly.

Similarly, the 8-convolution layer model was also trained from scratch for 150 epochs using both Adam and SGD nesterov. It was observed that the Adam model gave the better accuracy of 69.55% compared to the 68.21% of the SGD model. However, the SGD model had better generalization. Since the Adam model had satis-factory tradeoff between good generalization of features and test accuracy, it was selected as the final model.

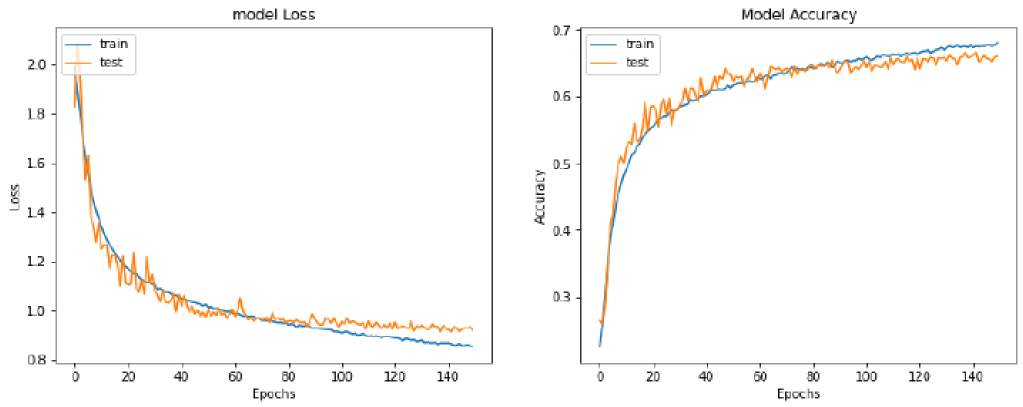

Fig.5. Proposed model accuracy Plot using SGD

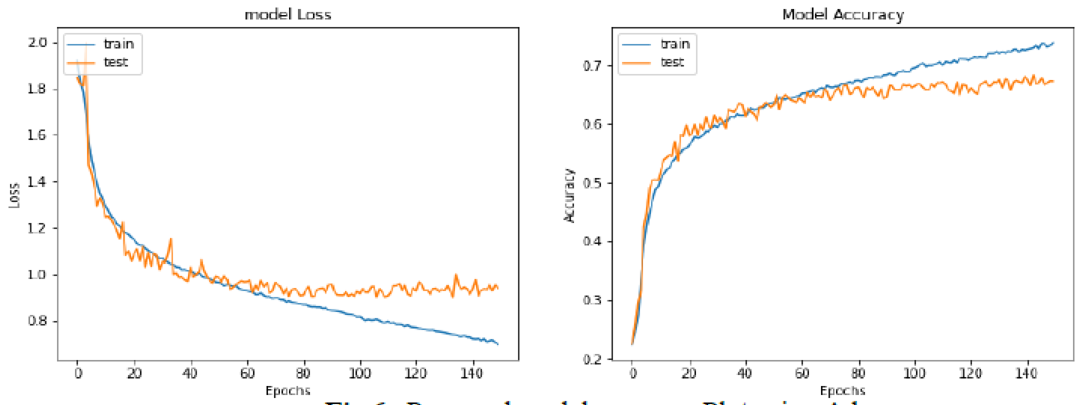

Fig.6. Proposed model accuracy Plot using Adam

Based on the results of our proposed model on the FER2013 dataset and the performance of all other models on the FER2013 dataset as described in the Previous Work Section, a table of results is presented below.

Table 2. Performance Comparision

Model	**Network Type**	**Accuracy %**
Ian et. al. [6]	Ensemble ResMaskingNet with 6 other CNNs	76.82
Khanzada Amil et al. [1]	Ensemble CNN	75.8
Georgescu et al.[2]	Fusion CNN + BOVW	75.42
Pramerdorfer et al. [3]	Ensemble CNN	75.2
Kim et al. [4]	Ensemble CNN	73.73
Connie et al. [5]	Hybrid CNN-SIFT	73.4
VGG [2]	CNN	72.7
ResNet [2]	Residual	72.4
Grigorași et. al. [7]	CNN hyperparameters optimizations	72.16
Hua et al. [8]	Ensemble CNN	71.91
Inception [2]	Deep CNN	71.6
Tang et. al. [9]	CNN + SVM	71.16
Kaiming et. al. [10]	Residual	69.7
Proposed Model	CNN	**69.55**
VGG-16 [2]	CNN	68.2
Ionescu et al. [11]	BOVW	67.48

According to the table, the proposed model and the VGG 16 Model differ by 1.35 percent, with the VGG16 Model having 130-144 million parameters compared to the proposed model's 5,228,487. The difference between the paper proposed by [Kaiming et al.][10] proposing a Residual Network and the proposed model is only 0.2 percent.

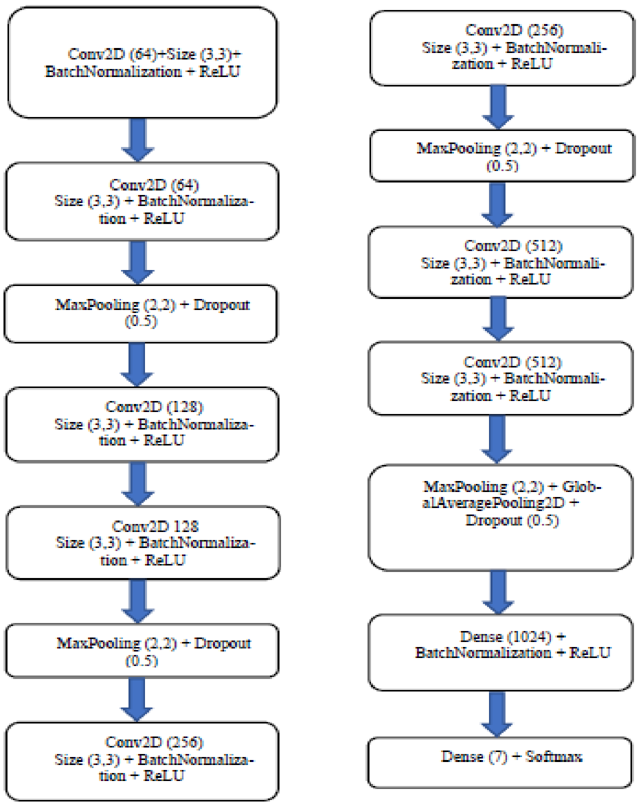
Conv2D (64)+Size (3,3)+ BatchNormalization + ReLU
Conv2D (64) Size (3,3) + BatchNormalization + ReLU
MaxPooling (2,2) + Dropout (0.5)
Conv2D (128) Size (3,3) + BatchNormalization + ReLU
Conv2D 128 Size (3,3) + BatchNormalization + ReLU
MaxPooling (2,2) + Dropout (0.5)
Conv2D (256) Size (3,3) + BatchNormalization + ReLU
Conv2D (256) Size (3,3) + BatchNormalization + ReLU
MaxPooling (2,2) + Dropout (0.5)
Conv2D (512) Size (3,3) + BatchNormalization + ReLU
Conv2D (512) Size (3,3) + BatchNormalization + ReLU
MaxPooling (2,2) + GlobalAveragePooling2D + Dropout (0.5)
Dense (1024) + BatchNormalization + ReLU
Dense (7) + Softmax

Fig.7. Proposed model architecture

V. Conclusion

This paper sought out to develop a machine learning model capable of recognizing emotions from facial expressions using CNN. Further it tried to increase the accuracy of the model by hyperparameter optimization through the use of Random Search algo-rithm to determine the best combination of parameters from a discrete space of possi-ble solutions. The best model obtained from this method had a test accuracy of 69.55% and used a traditional CNN architecture.

The results turned out satisfactory as the model had a good tradeoff between generali-zation of features and test accuracy. In future, the accuracy can be attempted to be increased further by increasing the number of Random Search trials as well as trying out pretrained models such as VGG-16, ResNet etc. and then Ensembling them.

References

[1]. Khanzada, Amil, Charles Bai, and Ferhat Turker Celepcikay. "Facial expression recognition with deep learning." arXiv preprint arXiv:2004.11823 (2020).

[2]. Georgescu, Mariana-Iuliana, Radu Tudor Ionescu, and Marius Popescu. "Local learn-ing with deep and handcrafted features for facial expression recognition." IEEE Ac-cess 7 (2019): 64827-64836.

[3]. Pramerdorfer, Christopher, and Martin Kampel. "Facial expression recognition using convolutional neural networks: state of the art." arXiv preprint arXiv:1612.02903 (2016).

[4]. Kim, Bo-Kyeong, et al. "Fusing aligned and non-aligned face information for auto-matic affect recognition in the wild: a deep learning approach." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016.

[5]. Connie, Tee, et al. "Facial expression recognition using a hybrid CNN–SIFT aggrega-tor." International Workshop on Multi-disciplinary Trends in Artificial Intelligence. Springer, Cham, 2017.

[6]. Goodfellow, Ian J., et al. "Challenges in representation learning: A report on three machine learning contests." International conference on neural information pro-cessing. Springer, Berlin, Heidelberg, 2013.

[7]. Vulpe-Grigoraşi, Adrian, and Ovidiu Grigore. "Convolutional Neural Network Hy-perparameters optimization for Facial Emotion Recognition." 2021 12th International Symposium on Advanced

Topics in Electrical Engineering (ATEE). IEEE, 2021.

[8]. Hua, Wentao, et al. "HERO: Human emotions recognition for realizing intelligent In-ternet of Things." IEEE Access 7 (2019): 24321-24332.

[9]. Tang, Yichuan. "Deep learning using linear support vector machines." arXiv preprint arXiv:1306.0239 (2013).

[10]. Vorontsov, A. O., and A. N. Averkin. "Comparison of different convolution neural network architectures for the solution of the problem of emotion recognition by facial expression." Proceedings of the VIII International Conference "Distributed Compu- 13 ting and Grid-technologies in Science and Education"(GRID 2018), Dubna, Moscow region, Russia. 2018.

[11]. Ionescu, Radu Tudor, Marius Popescu, and Cristian Grozea. "Local learning to im-prove bag of visual words model for facial expression recognition." Workshop on challenges in representation learning, ICML. 2013.

[12]. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[13]. Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).