Emotion Recognition from Facial Expressions

Arvinder Singh
Northeastern University
Boston, USA
singh.arv@northeastern.edu

Rahul Kumar
Northeastern University
Boston, USA
kumar.rahul4@northeastern.edu

Uday Raghuvanshi
Northeastern University
Boston, USA
raghuvanshi.u@northeastern.edu

***Abstract*—Facial expressions are one of the easiest and widely used methods of communication between human beings. Facial recognition has great applications in the field of human-computer interaction and other areas. In this project, we aim to implement multiple machine learning models to recognize emotions through facial expressions. We trained our data using the techniques of Support Vector Machines (SVM), Logistic Regression and Deep Neural Networks. For SVM, we extracted the image features, performed clustering to obtain a bag of visual words and obtained histograms of descriptors in the images before feeding them into the SVM classifier. To classify the data using Logistic regression, we normalized the pixel values, performed k-fold cross-validation to obtain the best hyper parameters for Logistic Regression and used those parameters to train the Logistic Regression model. CNN is performed using two models which are pre-trained VGG16 model and our proposed model. The Convolutional layer extracts the spatial features which are then fed to fully connected layers for classification. We aimed at maximizing the accuracy of the facial recognition algorithm so that this can be used in real-world applications. Upon testing on several SVM models, we obtained a test accuracy of 25.69%. For Logistic Regression, we improved the test accuracy up to 38.35%. Our proposed CNN model has exceeded the accuracy of both methods and achieved 66.9% test accuracy.**

***Index Terms*—CNN, Logistic Regression, SVM, VGG, FER2013**

I. Introduction

Using facial expressions while communicating is the most common and efficient form of communication that we humans use in our daily lives. However, computers mostly rely on text, touch and voice inputs to understand us. Using Facial Expression Recognition (FER) techniques helps computers better interpret a human user. Gestures including facial expressions convey nonverbal communication cues that play an important role in building interpersonal relations. Thus, recognizing facial expressions makes Human-Computer interaction easy and efficient. Our project could have numerous real-world applications, for example, companies could use the customers' responses to their products by tracking the facial expressions of customers when they view a product at supermarkets. It could also be deployed as a security system to look for potential miscreants in places such as banks, malls and public parks. Even humanoid robots can interact with humans more efficiently by interpreting human facial expressions.The aim of this paper is to perform FER on the FER2013 dataset using different machine learning techniques and comparing their results.

II. Related works

The early works on emotion recognition from facial expressions are done using Classical machine learning methods. The features are handcrafted[5] and then it is fed to the classical ML networks such as SVM[4]. SVM was proved to be very effective supervised learning and eliminated the limitations of some of tradional learning such as decision trees. The SVM approach involves finding a hyperplane in a high-dimensional space that maximizes the margin between the two classes of data points. This approach minimizes the classification error and maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class. Abdulrahman et al.[6] extracted facial features from images using Local Binary Patterns (LBP) which was then taken as input to SVM classifier. The SVM classifier achieved best accuracy with the radial basis function kernel.

While classical methods like these have been commonly used, the rise of deep learning has led to significant advancements in computer vision tasks, such as image classification. Among the deep CNNs that have gained recognition in this field is AlexNet [7], which won the ImageNet ILSVRC challenge in 2012. The AlexNex is followed by VGGNet[8], which became runner-up in ILSVRC 2014. VGGNet used convolution with small filters (3*3) and made the neural network deeper instead of using larger filter size for convolution as used in earlier CNNs. Even VGGNet was able to achieve depth of 19 layers in VGG19 and this limitation was tackled by ResNet[9] which was able to achieve a depth of 152 layers and won the ILSVRC 2015. The vanishing gradient problem was addressed by ResNet with introducing residual learning framework, where the output of a layer is the sum of the input and the output of a residual function applied to the input. This allowed the network to learn the residual mapping instead of the complete mapping, which made it easier to optimize. Even having larger depth, ResNet avoided overfitting with the use of residual connections and Batch Normalization.

Sang et al.[1] used the CNN for the facial expression recognition task. They proposed a CNN architecture that consists of multiple convolutional layers, pooling layers, and fully connected layers that can learn features from raw pixel data and capture both local and global features. Georgescu et al.[2] combined handcrafted features extracted using Local binary pattern and features from CNN and classified using SVM classifier for facial expression recognition. Singh et al.[3] used CNN to extract features and conducted sensitivity analysis

to evaluate the impact of different hyperparameters on the performance of the CNN model.

III. Dataset

For this project we used the FER2013 Dataset from Kaggle which is a well-studied dataset and has been used in several research papers. The dataset contains 35,887 images that are normalised to 48x48 pixels in grayscale.The dataset has both training and testing data. The images are of 7 facial expressions including Angry, Fear, Happy, Sad, Surprise and Neutral. The image distributions for the training data are: Angry (3,994), Disgust (434), Fear (4096), Happy (7214), Sad (4829), Surprise (3,170), and Neutral (4,964). Thus, we can see that the data is unbalanced.

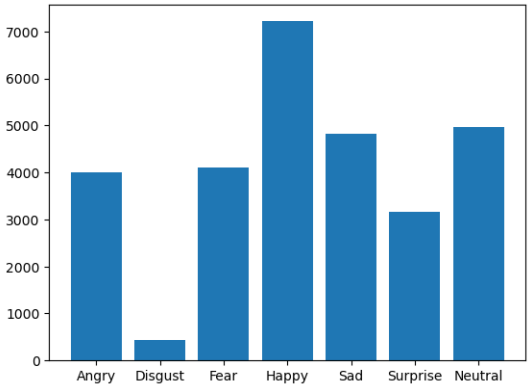


Fig. 1: *Distribution of training examples*

Fig. 2: *Sample Images from the FER2013 dataset*

Another thing to note is that the FER2013 dataset is challenging. The images are not aligned and some of them are labeled incorrectly. We can see from the following images.

Moreover, there are many samples that do not contain faces. Some are shown below:

These challenges make it harder to classify data because the model will have to generalize to an incorrect data.

Fig. 3: *Images classified under 'fear' class in training data*

Fig. 4: *Images with no faces*

IV. Implementation

In order to tackle the FER problem, we have implemented 3 classification based approaches, namely Support Vector Machines (SVMs), Logistic Regression and Convolutional Neural Networks (CNN). The input to all three is in the from of grayscale images. However, the way each method processes images before training the model varies. This has been discussed in each of the implementation below:

A. Support Vector Machines

Support Vector Machine (SVM) is one of the supervised machine learning algorithms that aims at finding the decision boundary/hyperplane that would separate the different classes of images in the feature space. The hyperplane obtained is such that it maximizes the margin among all the classes. The first step is to extract features from images and use them to form a Bag of Visual Words (BoVW) which is the vocabulary. Then each image in training and testing data is represented based on the vocabulary and we obtain histograms of training and testing images along with the labels. Finally, these histograms are fed in the SVM classifier, comparing different SVM models, to obtain the accuracies.

We started by forming an array of training and test images along with the associated labels. These are then fed into a 'getvocab' function which returns a vocabulary of visual words. It loops through all the training images, extracts key-points and descriptors using SIFT extractor and stores them. In the data, there are many images which either don't give any descriptors or the length of its descriptors is not 128 and hence we discarded them. Finally, we performed Kmeans clustering to form the vocabulary of size 20. Using this vocabulary, we iterated again through all the training and test images, extracted descriptors and looked for the distribution of vocabulary in each image. After obtaining the training and test histograms, we feed them into the SVM classifier.

For our classification problem, we tested different models by varying the 'regularizing parameter', 'kernel type' and 'degree'

of polynomial kernel. We performed k-fold cross validation to prevent overfitting. For k-folds, the training data was divided into 4 sets, model was trained on 3 of those and validated on the 4^{th}. The 4 accuracies were then averaged to obtain the training accuracy for the particular set of parameters. We tested for varying regularizing coefficient (0.1,1,10), different kernels (linear, rbf, sigmoid, polynomial) and different degrees for polynomial kernel (2,3,4). combination of parameters that gave us the largest accuracy was selected as the best model. Thus, the best model was obtained after hyperparameter tuning and k-fold cross validation. The results are presented below:

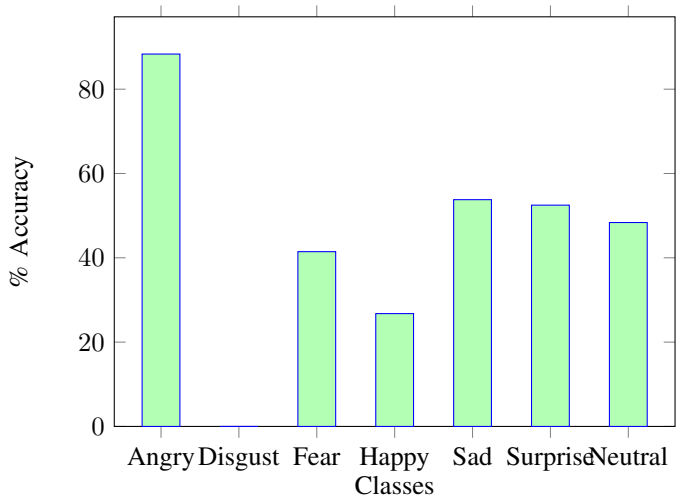
TABLE I: Hyperparameter Tuning on SVM

Hyperparameters			**Accuracy**
Regularizing Coeff (λ)	***Kernel***	***Deg of poly kernel***	***(%)***
0.1	linear	—	25.114
0.1	rbf	—	25.1446
0.1	sigmoid	—	23.2067
0.1	polynomial	2	25.1168
0.1	polynomial	3	25.3224
1	polynomial	4	25.4758
1	linear	—	25.1446
1	rbf	—	25.7755
1	sigmoid	—	16.6155
1	polynomial	2	25.0680
1	polynomial	3	25.2562
1	polynomial	4	24.9948
10	linear	—	25.1446
10	rbf	—	25.1516
10	sigmoid	—	16.2530
10	polynomial	2	25.0889
10	polynomial	3	23.7539
10	polynomial	4	24.0606

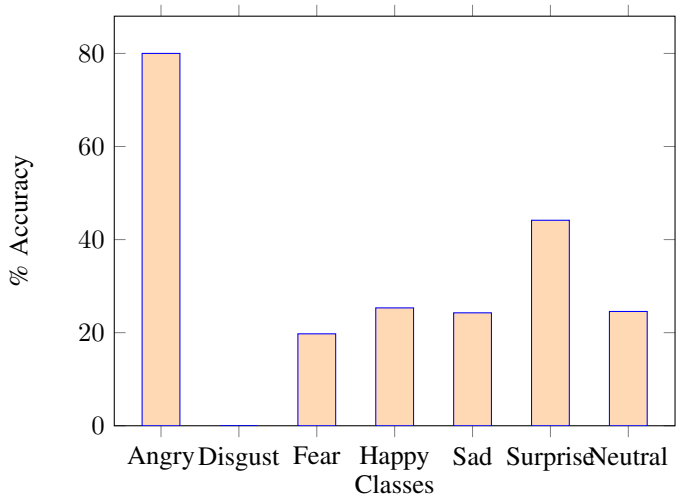
After performing cross validation on all possible set of parameters, the model that gave us the best accuracy (25.7755%) was using 'rbf' kernel with a regularizing coefficient of 1.

- The linear kernel produced same accuracy for all cases because it only forms a linear decision boundary between the classes and the varying regularization coefficient only affects the size of coefficients and not the boundary shape
- The rbf kernel performs the best because it is a non linear kernel that can easily capture complex non-linear decision boundaries, which is true for our case.
- The sigmoid kernel produced significantly lower accuracies than other kernels because it is not able to accurately capture the non-linear relationship between the input features and target variables.
- The polynomial kernel showed varying performance based on degree and coefficient. Degree higher than 4 were also tested but proved to be highly computationally expensive and gave significantly lower performance on test data. It is because further increasing the degree of polynomial kernel led to overfitting.

Once we had the best model, we used it to train the whole data again and observed the total and per class training accuracy. The training accuracy was 28.9195%. The per class accuracies are shown in the bar graph:



The model was then run on the test data and produced a test accuracy of 25.6950%. The per class accuracies for test data are shown in the bar graph:



The result showed 0% accuracy for the 'Disgust' class in both training and test data. The possible reason for this could be the less amount of training data available for 'Disgust' class. The training data only had 434 samples in this class (out of the total 28709 images).

The state of the art for SVMs is able to achieve 50% accuracy. In this model they use different, more complex computer vision methods of extracting image features like combining HOG (Histogram of Gradients) with Face Landmarks. If they didn't use Face landmarks, they achieved 29% accuracy. Thus, including Facial landmarks to improve accuracy is essentially improving on feature extraction techniques using computer vision and not on the machine learning technique. So, considering the use of SVM, we did an excellent job to get an accuracy of 26

B. Logistic Regression

Facial emotion recognition (FER) using Logistic regression is not a very popular approach for analyzing facial expressions and recognizing emotions from images. Logistic regression models can be trained to classify input images into different

emotion categories based on the extracted features. In Facial Expression Recognition, feature extraction is a crucial step that involves extracting relevant information from the face images to accurately classify emotions. Some of the commonly used feature extraction techniques for FER include LBP, HOG, and Deep Learning.

Logistic regression models are simple and interpretable, which makes them not very ideal for FER tasks. They can be trained quickly and are less prone to over-fitting. Furthermore, Logistic regression models can handle noisy and incomplete data, making them useful for real-world applications. However, Logistic regression models have some limitations in FER tasks. They may struggle to capture the complex relationships between facial features and emotions, and may not perform well when the emotion categories are closely related. Moreover, they may not generalize well to new data sets, especially when the images have significant variations in pose, lighting, and occlusion.

Despite these limitations, we wanted to see if we can get a good enough accuracy using pure Machine Learning techniques and limiting the use of Computer Vision to extract features or using Deep learning to extract features and then feeding them to a Logistic regression model.

We started by reading the image data into a numpy array after reshaping every image to be a row vector and then vertically stacking all the pictures in the dataset while also maintaining another numpy array which contains the labels associated with the pictures. After obtaining a numpy array with all the pictures we normalized the pixel coordinates to be in the range of [-1, 1]. Since the labels were strings like - angry, disgust etc. We transformed these labels to one-hot encodings and then we took the index value of the maximum argument for every label. After pre-processing the image and label data into numpy arrays, we then wanted to choose the best hyperparameters to maximize the accuracy on the Logistic Regression model.

We start by hyper-parameter tuning using GridSearchCV, which is a widely used technique for selecting the best hyperparameters of a machine learning model. The approach involves defining a grid of hyperparameter values and training and evaluating the model for each combination of hyperparameters in the grid. The optimal hyperparameters are then selected based on the best evaluation metric achieved during the grid search. We defined our hyper paramater grid as follows:-

TABLE II: Hyperparameter tuning for Logistic Regression

Parameter	**Value**		
Penalty	*None*	*L1 Norm*	*L2 Norm*
C (Regularization Coeff.)	*0.01*	*0.1*	*1*
Solver	*lbfgs*	*newton-cg*	*saga*
Multi-Class	*One-vs-Rest*	*Multinomial*	-
Maximum iterations	*1000*	*10000*	-
Cross-Validation	*5 folds*	-	-

After running GridSearchCV to select the best hyperparameters, we found that newton's method was unable to converge even for maximum iterations = 10,000. The best performing model had parameters-

C: **0.01**,
Multi-class: **Multinomial**,
Penalty: **L2 Norm**,
Solver: **lbfgs**
and the best accuracy = **44.18**%.

Then we trained our model using these parameters, and observed that the accuracy of the model on the training dataset came out to be 38.35% which is a significant improvement over SVM, considering we are using traditional ML; with absolutely no use of Computer Vision or Deep Learning to extract features from the images. The state of the art for Logistic Regression is 54%, in this model they train on deep features extracted from a pre-trained convolutional neural network. So considering the use of pure machine learning techniques, we did an excellent job to get an accuracy of 40%. Also, as discussed above, there was a bias in terms of the number of photos in each class. The disgust class had the least amount of training data available hence, the model performed poorly on the disgust class. It is worth noting that,

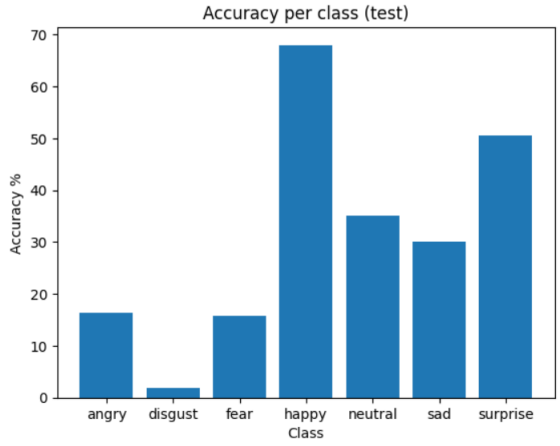


Fig. 5: *Per class accuracy for test data (Logistic Regression)*

even with the use of Computer Vision to extract features such as SIFT or ORB and then training those features using Logistic Regression gave us a lower accuracy than the use of pure Machine Learning. This can be explained because of how the features extracted from images, Sift or ORB features are based on low-level visual features such as edges and corners. These features are not specifically related to the emotions expressed by individuals, but rather the overall visual structure of the image. Therefore, when these features are used for training a logistic regression model, the resulting model may not be able to capture the subtle nuances of facial expressions that are indicative of particular emotions. On the other hand, pure machine learning models that use pixel data as input may be able to capture more detailed and nuanced information about facial expressions. By training the model directly on the raw

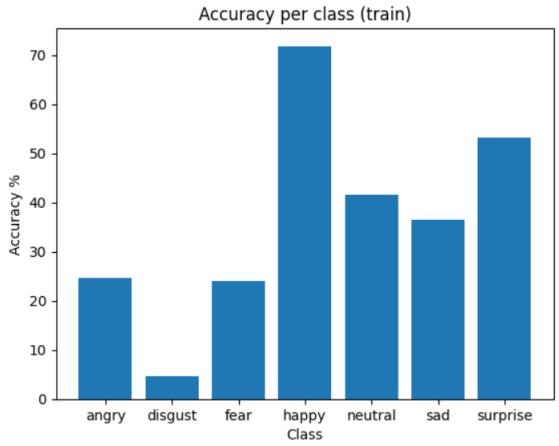


Fig. 6: *Per class accuracy for training data (Logistic Regression)*

image data, the model can learn the subtle visual cues that are indicative of particular emotions. This may result in a more accurate model compared to models that rely on pre-extracted features. Additionally, using pre-extracted features may also result in loss of information, as some relevant features may be discarded during the extraction process. This may further reduce the accuracy of the resulting model. The best way to improve the accuracy, would be to first use Deep Learning to extract deep features that can capture subtle nuances of facial expressions and then use those extracted features to train a logistic regression model. This can be very expensive in terms of the computational power required but is guaranteed to give us a better model.

C. Convolutional Neural Network

In the past decade, Convolutional Neural Network is able to bring significant improvement in the field of image classification and object detection. The ability of Convolutional layers to extract and process spatial features from images sets it apart from traditional learning methods, which requires hand selected feature extractor. CNNs use a series of convolutional layers to learn hierarchical representations of image features, such as edges, textures, shapes, and patterns. Each convolutional layer consists of a set of filters that slide over the input image and perform a convolution operation, resulting in a feature map that highlights the presence of certain visual patterns. The output of each convolutional layer is then passed through a non-linear activation function, such as ReLU, to introduce non-linearity into the model. This allows CNNs to learn complex, non-linear relationships between the input image and its corresponding output label. Before training the model, Data preprocessing and Data augmentation is carried out on test examples to make the model robust, which is explained below:

1)Data preprocessing: In preprocessing, the pixels are normalized over all the training examples. The mean and standard deviation of pixel is calculated for training dataset and then each image is normalized by subtracting the mean pixel value and dividing by standard deviation. This helped in improving the performance and lead to faster convergence.

2)Data augmentation: In augmentation of data, image are horizontally flipped by a probability of 0.5 and images are rotated by 30deg. This helped in generalizing the model and prevented overfitting.

1) Pre-trained model-VGG16: Firstly, we used Transfer learning to build the model which provides initial result in a less time by using the Pre-trained model of VGG16. Since this model are trained on ImageNet which are colored images of size 224x224. But the images in FER2013 dataset has greyscale images of size 48x48. So the image is resized to 224x224 before feeding to the network. Also first convolution layer is modified to input greyscale images. Finally model is fine tuned by training the first convolution layer, and all classification layers of VGG16. Fig. 7 and Fig. 8 shows

TABLE III: Parameter used for Pre-trained VGG16

Parameter	Value
Learning Rate	0.0003
Batch Size	64
Epoch	50
Optimizer	Adamax

the classification loss for testing and validation images and confusion matrix for test images for pre-trained model. It can be seen in fig. 7 that the loss decreases rapidly till 20 epoch and then started to increase which shows the need of decreasing the learning rate after 20 epoch.

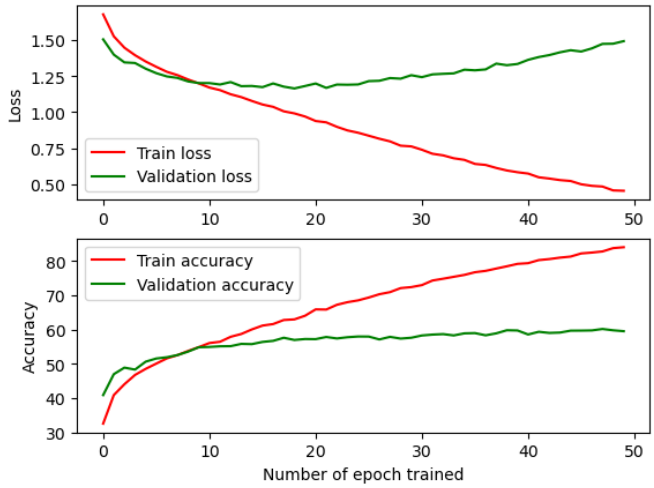


Fig. 7: *Classification loss and accuracy plot for training and validation images for pre-trained network*

2) Proposed model: Our proposed model is inspired from VGG16 due to its simple but effective architecture. Each convolution layer has kernel size 3x3 with stride 1. The original VGG16 network does not contain batch normalization and dropout layer but to make the network robust, we have added both. Also the architecture is designed to take grey

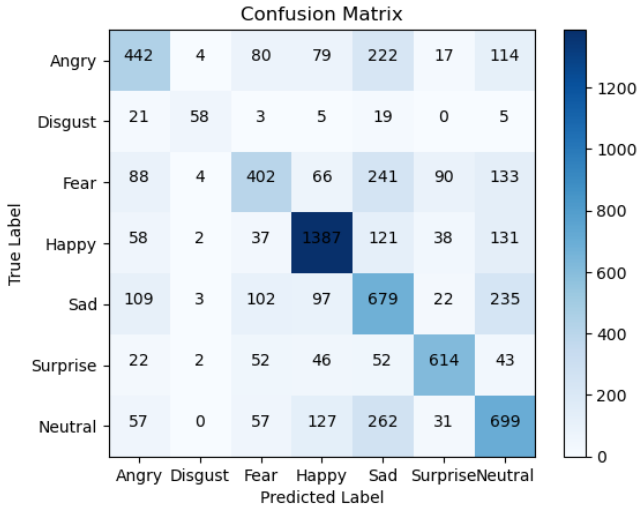


Fig. 8: *Confusion matrix for test images for pre-trained network*

scale images as input of size 48x48. Below is our proposed architecture:

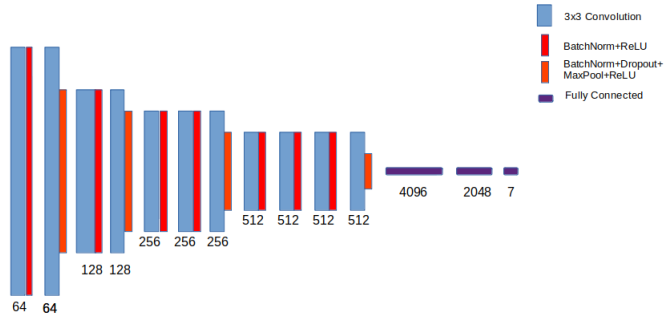


Fig. 9: *Proposed network architecture*

The network consists of total 14 layers in which 11 are convolutional layers and 3 are fully connected layers. The image size is reduced using MaxPool layer of 2x2. The channels are increased from 64 to 512 and features from last convolutional layer is fed to fully connected layers. The output of network gives the probability of 7 classes.

TABLE IV: Parameter used for proposed model

Parameter	Value
Learning Rate	0.001
Batch Size	64
Epoch	100
Optimizer	Adamax

Fig. 10 and Fig. 11 shows the classification loss for testing and validation images and confusion matrix for test images. It can be seen in fig. 10 that the loss decreases rapidly till 50 epoch and after that the rate slowed down.

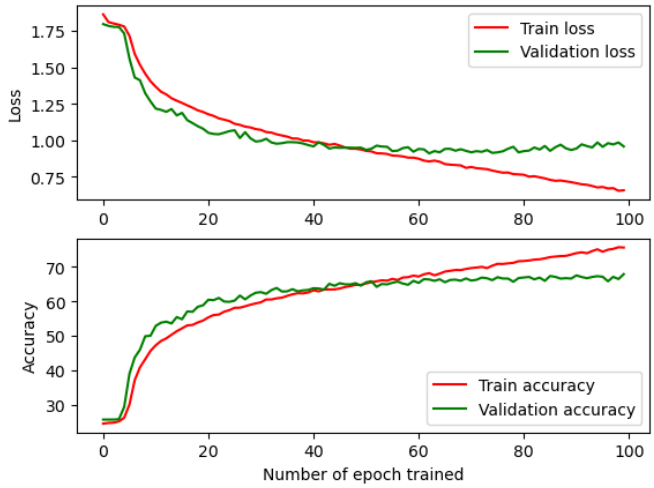


Fig. 10: *Classification loss and accuracy plot for training and validation images for proposed network*

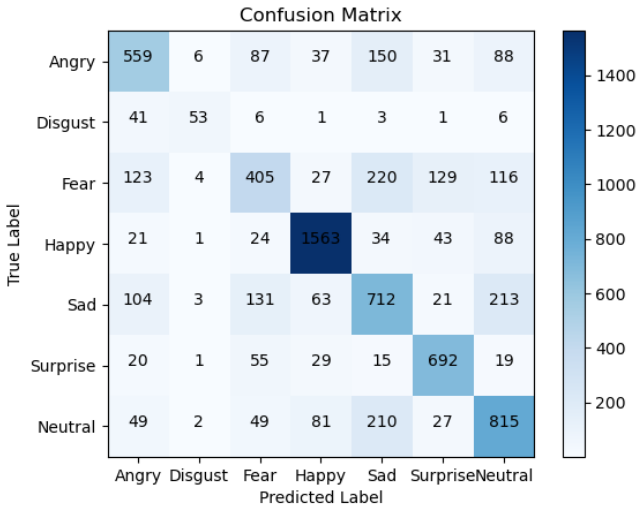


Fig. 11: *Confusion matrix for test images for proposed network*

The proposed network performed well on test set and able to out-perform the pre-trained model. Table V shows the comparison of testing accuracy of CNN models. The reasons for better performance of proposed model is mentioned below:
1) Use of Batch normalization and dropout layers in convolutional layer made the network robust to overfitting and able to generalize
2) Network is designed to take input of size 48x48 which eliminated the requirement of resizing and thus no artifacts are introduced in the image for our model but artifacts introduced in pre-trained model.
3) Due to less number of parameters in our network, all parameters are trained efficiently with reduced training time but for pre-trained model, only fully connected layers are trained.
4) Due to robust network, proposed model is trained with high learning rate.

TABLE V: Test accuracy for pre-trained and proposed model

Model	Test accuracy
Pre-trained model	59.5%
Proposed model	66.9%

Both CNN models are also compared on the accuracy measures such as Precision, Recall and F1score which are defined as below:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

where TP is true positive, FP is false positive and FN is false negative. Precision represents the ability of network to predict image belonging to a particular class accurately. Recall represents the network ability to identify the fraction of images belonging to a class. F1score combines the precision and recall and provides a robust parameter of correctness. Fig. 12 shows the comparison of pre-trained model and custom model on the basis of accuracy measures.

Class	Accuracy Metric					
	Precision		Recall		F1score	
	Pre-trained	Proposed	Pre-trained	Proposed	Pre-trained	Proposed
Angry	0.55	0.61	0.46	0.58	0.5	0.6
Disgust	0.79	0.76	0.52	0.48	0.63	0.59
Fear	0.55	0.54	0.39	0.4	0.46	0.45
Happy	0.77	0.87	0.78	0.88	0.77	0.87
Sad	0.43	0.53	0.54	0.57	0.48	0.55
Surprise	0.76	0.73	0.74	0.83	0.75	0.78
Neutral	0.51	0.61	0.57	0.66	0.54	0.63

Fig. 12: *Accuracy metric for pre-trained and proposed model*

V. Results and Conclusion

In the paper we have discussed the importance of Facial Emotion Recognition, the fer2013 dataset and its shortcomings and finally implemented 3 different approaches to tackle the FER problem. We highlighted the key differences in accuracy's among them and also compared each with their respective state of the art models. Furthermore, we also briefly discussed about how the techniques of feature extraction using computer vision methods impact the performance of our algorithms.

TABLE VI: Classification Results

Machine learning Algorithm	Accuracy (%)
SVM	25.69
Logistic Regression	38.35
CNN	66.9

Based on the analysis and experiments conducted, we can conclude that for facial expression recognition, SVM or logistic regression alone are not a viable approach. The use of computer vision techniques such as SIFT and ORB for feature extraction did not provide better results compared to pure machine learning techniques.

Using the FER2013 dataset and deep learning with the preprocessed grayscale images, we were able to achieve an accuracy of 66.9% on the test set. Further, we used cross validation to tune the hyperparameters for different models. We also compared the performance of our logistic regression and SVM model with that of other state-of-the-art models for facial expression recognition such as Convolutional Neural Networks (CNNs) and found that SVM or Logistic regression model was not as accurate. However, these models had the advantage of being faster and requiring less computational resources compared to CNNs. Overall, If we use CNNs for facial expression recognition, we can achieve higher accuracy. Further research could focus on exploring other feature extraction techniques and experimenting with more complex machine learning models.

References

[1] D. V. Sang, N. Van Dat and D. P. Thuan, "Facial expression recognition using deep convolutional neural networks," 2017 9th International Conference on Knowledge and Systems Engineering (KSE), Hue, Vietnam, 2017, pp. 130-135, doi: 10.1109/KSE.2017.8119447.

[2] M. -I. Georgescu, R. T. Ionescu and M. Popescu, "Local Learning With Deep and Handcrafted Features for Facial Expression Recognition," in IEEE Access, vol. 7, pp. 64827-64836, 2019, doi: 10.1109/ACCESS.2019.2917266.

[3] S. Singh and F. Nasoz, "Facial Expression Recognition with Convolutional Neural Networks," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0324-0328, doi: 10.1109/CCWC47524.2020.9031283.

[4] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995.

[5] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2, Jun. 2006, pp. 2169–2178.

[6] M. Abdulrahman and A. Eleyan, "Facial expression recognition using Support Vector Machines," 2015 23nd Signal Processing and Communications Applications Conference (SIU), Malatya, Turkey, 2015, pp. 276-279, doi: 10.1109/SIU.2015.7129813.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[10] Amir Mohammad Hemmatiyan-Larki; Fatemeh Rafiee-Karkevandi; Mahdi Yazdian-Dehkordi "Facial Expression Recognition: a Comparison with Different Classical and Deep Learning Methods"

VI. Appendix

Link for dataset:

- *https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data*
- *https://www.kaggle.com/datasets/msambare/fer2013*