Abstract:

Facial expressions are the primary means by which humans communicate their feelings. It's easy for humans to identify certain emotions, but computers have a hard time doing so. The expressions on people's faces are unique to each individual. Many strategies have been used, but reaching high levels of accuracy and building robust facial recognition systems is still difficult because of the complexity of human faces. It's impossible to generalize about how each image's brightness, contrast, and resolution varies. This is why it is so difficult to recognize facial expressions. There is a lot of work being done in the field of facial expression recognition. We focused on the recognition of seven fundamental human emotions in this research. These emotions are angry, disgust, fear, happy, sad, surprise and neutral. A system for facial expression recognition (FER) will be implemented in this research using neural networks, specifically Convolutional Neural Networks (CNNs).

Introduction:

An important role is played by human facial expressions when it comes to involvement, engagement or communication. You can tell someone's emotional state just by looking at their face. We are increasingly reliant on machines to solve our problems as technology progresses. Machine learning is an important goal in computer vision. Facial expression detection from a facial image is an intriguing and difficult subject in the realm of computer vision.

Due to the fact that facial expressions are one of the most natural and instinctual ways of expressing emotions, these kinds of expressions are widely acknowledged in the field of facial expression identification. It has a variety of uses, including human-computer interactions, consumer satisfaction, the gaming industry, and the acquisition of real-time human feedback. Due to the nature of the problem, classification approaches like as Support Vector Machines, Random Forest, or Logistics Regression may be used. Along with these more traditional machine learning techniques, neural networks have shown considerable promise in resolving this challenge. Computer vision advancements have consistently improved upon previous techniques and demonstrated tremendous potential.

The main contribution of this research is a simplified training procedure that leads to the lightweight but very accurate CNN for multiple facial analysis tasks. The network is pre-trained on a large facial dataset FER13[1].

We used three convolution layers followed by three max pooling layers in this model. The activation function for convolution is leaky relu, and results of convolutions are normalized using batch normalization. Following flattening, we used four dense layers, each of which has been batch normalized. Additionally, dropout is applied in each dense layer to minimize overfitting.

Methods:

The goal here is to categorize an input image that contains an expression; the image must be classified as one of the seven emotions or expressions. A convolutional neural network has been applied in this case. Here, we begin by creating a model from scratch. The model is composed of three convolutional layers, three max pooling layers, and four dense layers.

Each convolution layer in our convolution section is followed by a max pooling layer. The result of convolution was sent through the leaky relu activation function and batch normalized after max polling.

We flattened the result after the four steps of convolution and pooling and fed it into an ANN. Four dense layers comprise the ANN. Each dense layer has an activation function to account for non - linearity, a dropout mechanism to account for overfitting, and batch normalization to compensate for data normalization.

The output layer is composed of seven nodes representing seven distinct sorts of expressions. The softmax has been used for activating the output.

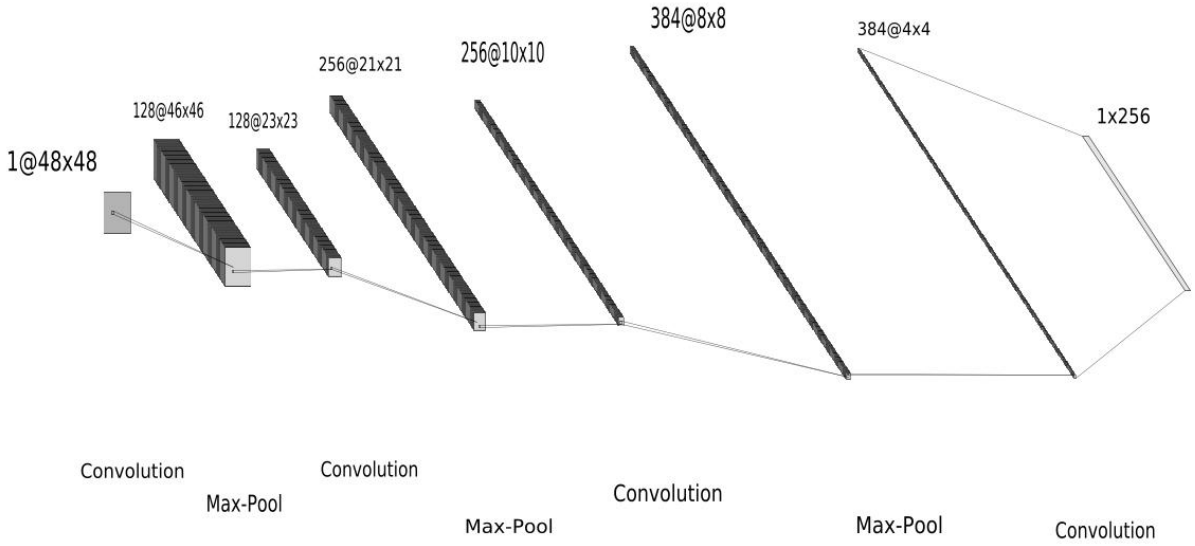


figure:Convolution architecture

Optimizer
Optimizers are techniques or strategies for adjusting the parameters of a neural network, such as weights and learning rate, in order to reduce losses. By minimizing the function, optimization algorithms are used to solve nonlinear equations.

Equation of adam optimizer

I tested several popular optimizers, including SGD, SGD with momentum, Adam, and AdaDelta. While SGD, AdaDelta and SGD with momentum demonstrated excellent performance, they were found to be relatively slow to converge. Adam performed the best; the difference in results was not substantial; I chose Adam due to its popularity.

Activation Function

An activation function is a function that is included in an artificial neural network to assist it in learning complex patterns in the input. When compared to the neuron-based model seen in our brains, the activation function is ultimately responsible for determining what to fire to the next neuron.

```
f(x)=max(0.01*x , x).
```

As, It's a seven class classification problem, so I used a softmax in the last layer that returns one of the most likely classes from the group. I chose the leaky relu activation for hidden layers since it eliminates the problem of gradients disappearing and exploding. I utilized the normal initializer because it has been known to work well with leaky relu activation functions.

Dropout

Dropout is a regularization strategy that prevents complex co-adaptations on training data, decreasing overfitting in neural networks. It's a quick and easy technique to average models with neural networks.
In the first three dense layers, I used a dropout of.25, and in the last dense layer, I used a dropout of .4.

Normalization

Deep neural networks with a large number of layers are difficult to train because they are sensitive to the learning algorithm's initial random weights and architecture. The distribution of inputs to layers deep in the network may fluctuate after each mini-batch when the weights are adjusted, which could be one cause for the difficulties. This could cause the learning process to track a moving target indefinitely.
Batch normalization is a technique for training very deep neural networks in which the inputs to each layer are standardized for each mini-batch. This stabilizes the learning process and significantly reduces the number of training epochs needed to create deep networks.

3.1 Experimental settings:

CNN (convolution neural network) is responsible for the entire feature engineering process, including the extraction of features from images. In a typical CNN design, the image's low-level features are extracted by the starting layers, while its high-level features are extracted by the end layers There are a number of CNN architectures that can be utilized to improve the accuracy of the results. In this experiment I'm using more than 35,000 photos depicting seven different emotions that are included in the FER2013 collection. I've trained the network with batch size of 32 with epoch size of 13. I've used sparse_categorical_crossentropy as a loss function that produces a category index of the *most likely* matching category and also used Confusion matrix to evaluate the model accuracy.

Faces in grayscale 48x48 pixel photos are the data. Faces have been automatically registered such that they are about aligned in each image. The goal is to categorize each face into one of seven categories: angry, disgusted, fearful, happy, sad, surprised, and neutral. "emotion" and "pixels" are columns in train.csv. The "feeling" field contains a numeric code ranging from 0 to 6 for the image's emotion. Each image's "pixels" column has a string enclosed in quotes. This string contains space-separated pixel values in row major order. The single column in test.csv is "pixels," and your objective is to guess the emotion.

3.2 Evaluation criteria:

An accuracy of around 82% percent was achieved with epoch 8. When we increased the epochs to 13, the accuracy increased to 90%. Fig2 describes training with epoch 13. Precision, recall, F1-score, and Support were calculated for each of the seven types of emotions, as well as the average value. These metrics were computed with the help of the sklearn library. The overall Sparse Categorical Accuracy of the trained model is 90% with loss value of 1.738 and Test Data Accuracy is 83% . Overall F1-score is .83 including maximum score of 0.90 for 'Surprise' and minimum score of 0.76 for 'Fear' and weighted average of precision is 0.84

	precision	recall	f1-score	support
Angry	0.85	0.80	0.82	3995
Disgust	0.91	0.81	0.85	436
Fear	0.72	0.81	0.76	4097
Happy	0.95	0.85	0.89	7215
Sad	0.81	0.79	0.80	4830
Surprise	0.90	0.89	0.90	3171
Neutral	0.75	0.86	0.80	4965
accuracy			0.83	28709
macro avg	0.84	0.83	0.83	28709
weighted avg	0.84	0.83	0.83	28709

Fig 2. Classification Report (Precision, Recall, F1-score, Support)

3.3 Results:

Accuracy is the evaluation metric that we will use in this case. On the test data, the final single-network model achieved an accuracy of 83.2%. I use confusion matrix (Fig3) scores to conduct error analysis. The confusion matrix is a table where every column represents the predicted label and the rows represent the true label. In my first designed network, I faced a bias problem after adding the 4th convolution layer. The range of the accuracy for different emotions was 58% to 90% and after removing that layer and changing the activation function to leaky relu the overall accuracy have been improved for different emotions Around 89% of images labeled as Surprise were correctly classified as such, while the accuracy for the remaining emotions was also in the range of 79 to 89 percent. Among the seven classes, Surprise had the highest precision and recall, while Sad had the lowest precision and recall.

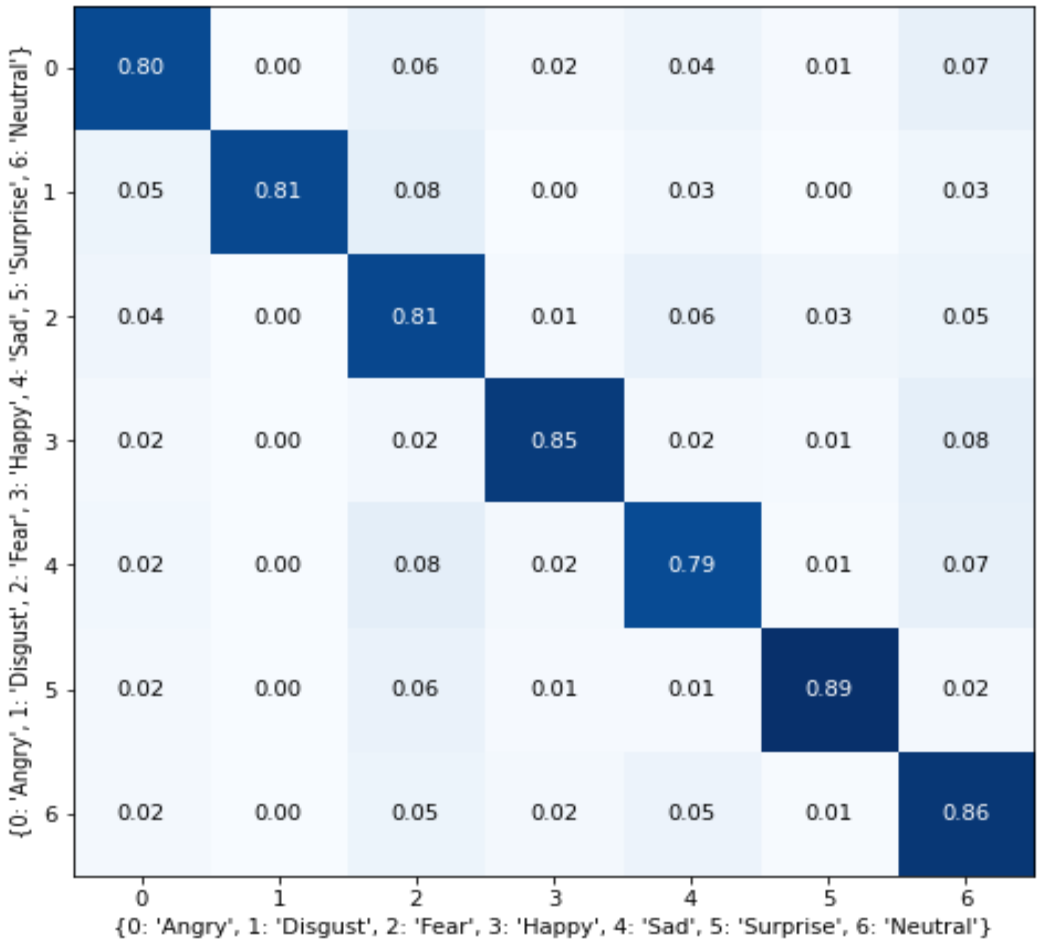


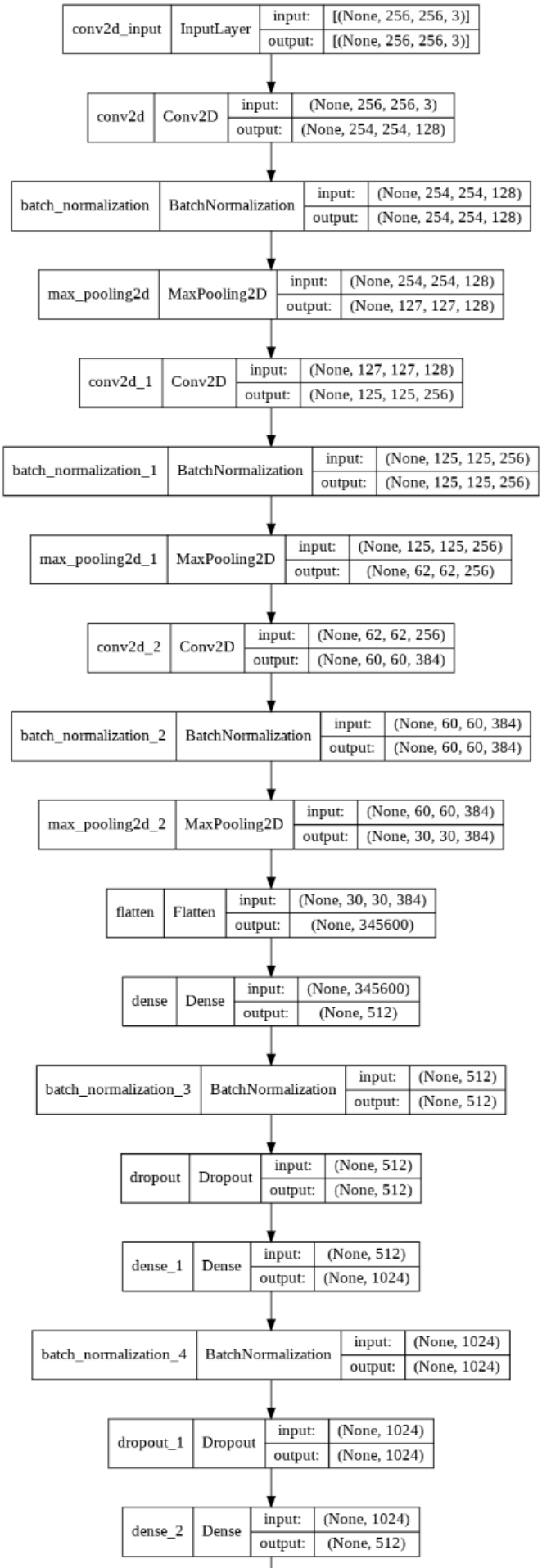
Fig3. Confusion Matrix

Conclusion:

The results demonstrate that the system's overall performance is pretty promising. On the FER2013 dataset, I constructed and trained my own customized CNN architecture, which obtained 83.2 % single-network accuracy on the test data. It involved picture enhancement, then fine tuning the model architecture and hyperparameters. I used dropouts, batch normalization, padding, pooling, strides, kernel size and number of kernels, activation function, weight initialization strategies, several optimizers, padding, and pooling. The proposed model architecture outperforms several modern techniques and is still relatively affordable.

Future study will include exploring other image processing methodologies on the FER2013 dataset, as well as evaluating composites of different deep learning architectures in order to further improve my performance in facial emotion recognition as the network can be further enhanced, and its excellent performance can be studied in greater depth.

1. FER13 dataset


conv2d_input InputLayer input: [(None, 256, 256, 3)] output: [(None, 256, 256, 3)]
conv2d Conv2D input: (None, 256, 256, 3) output: (None, 254, 254, 128)
batch_normalization BatchNormalization input: (None, 254, 254, 128) output: (None, 254, 254, 128)
max_pooling2d MaxPooling2D input: (None, 254, 254, 128) output: (None, 127, 127, 128)
conv2d_1 Conv2D input: (None, 127, 127, 128) output: (None, 125, 125, 256)
batch_normalization_1 BatchNormalization input: (None, 125, 125, 256) output: (None, 125, 125, 256)
max_pooling2d_1 MaxPooling2D input: (None, 125, 125, 256) output: (None, 62, 62, 256)
conv2d_2 Conv2D input: (None, 62, 62, 256) output: (None, 60, 60, 384)
batch_normalization_2 BatchNormalization input: (None, 60, 60, 384) output: (None, 60, 60, 384)
max_pooling2d_2 MaxPooling2D input: (None, 60, 60, 384) output: (None, 30, 30, 384)
flatten Flatten input: (None, 30, 30, 384) output: (None, 345600)
dense Dense input: (None, 345600) output: (None, 512)
batch_normalization_3 BatchNormalization input: (None, 512) output: (None, 512)
dropout Dropout input: (None, 512) output: (None, 512)
dense_1 Dense input: (None, 512) output: (None, 1024)
batch_normalization_4 BatchNormalization input: (None, 1024) output: (None, 1024)
dropout_1 Dropout input: (None, 1024) output: (None, 1024)
dense_2 Dense input: (None, 1024) output: (None, 512)