Facial Expression Recognition on Video Sequences and 3-D Point Clouds using Spatiotemporal Convolutional Neural Networks

Aneri Patel
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat
Email: 201601445@daiict.ac.in,

***Abstract*—Facial expressions are the facial changes in response to a person's internal emotional state, intention. It is also a way of non-verbal communication. Automatic facial expression recognition (FER) is a challenging problem that impacts essential applications in areas like human–computer interaction (HCI) and data driven animation. FER is solved using computer vision and artificial intelligence (AI). 2-D based solutions for FER present difficulties due to self-occlusion and pose variation. Thus, they are not satisfactory for real-world applications. 3-D data based solutions solve the problems that arise with 2-D data. In recent times, 3-D facial data, both 3-D point clouds and video sequences have become increasingly available. This report is concerned with facial expression recognition for two types of data: 1) Videos 2) 3-D point clouds, using 3-D convolutional neural networks (3-D CNN).**

***Index Terms*—Facial expression recognition, 3-D convolutional neural networks, Spatio-temporal convolution, Point Clouds**

I. Introduction

Automatic facial expression recognition (FER) has a vast potential in areas like surveillance and communication. Thus, there is a considerable demand for improving its performance [1]. It is more challenging to perform FER in an uncontrolled environment than in a controlled environment. This is because, in an uncontrolled environment, changes in occlusion, illumination, and noise make it harder to detect expressions [2]. This report focuses on FER in a controlled environment.

Traditional approaches follow a 3-step process: 1) Face detection 2) Facial feature extraction 3) Expression recognition [3]. This report focuses on the implementation of more recent convolutional neural network (CNN) based algorithms. CNN's perform "end-to-end" learning [4], thus significantly reducing the need for preprocessing the input [5].

The majority of FER research is based on 2-D images [6]. In recent times, 3-D facial expression data sets are more accessible. Thus, 3-D data based approaches are now being increasingly used. 3-D data is preferred over 2-D data for FER as it captures more information. At the cost of assimilating more information, 3-D methods require a significantly higher computation power and time than 2-D methods.

II. Related Work

Computer Vision focuses on achieving a high-level understanding of images and video sequences. A notable amount of research in computer vision has been conducted on problems like anomaly detection [7] and event detection [8] in videos. FER is yet another problem in computer vision that has been approached for decades. Currently proposed FER systems are differentiated based on multiple factors. They are either feature-based, model-based, deep learning based [9], convolutional-neural network (CNN) [10], or Recurrent Neural Network - Long Short Term Memory (RNN-LSTM) [11] based [1]. Feature based FER algorithms focus on extracting geometric information of the face surface like spatial relations between pairs of landmarks (Euclidean [12] [13] and Geodesic distances [14]), gradients, and local shapes, directly from the input data [15]. On the other hand, model-based algorithms use a generic model [1]. FER systems are also differentiated on the basis of the type of solution they use. e.g. deep learning, CNN, and RNN-LSTM based algorithms. In this project, a CNN-based approach is used for both implementations. FER systems are also differentiated based on the type of data they use; whether it is Uni-modal or Multi-modal [1]. Uni-modal approaches use a single information source to classify expression/emotions while multi-modal algorithms use multiple sources (audio, video, thermal imaging, etc.) of information for classification. In this project, uni-modal approaches were implemented. FER algorithms are further classified based on the dimensions of data they use; whether it is 2-D or 3-D. 3-D images can be of two types: a) Multi-view images b) Point-clouds [1]. FER is performed on two types of data: a) Spontaneous expressions b) Acted expressions [16]. In this project, partIV-B works on spontaneous expressions while part IV-C works with acted expressions.

III. 3-D Convolutional Neural Networks

The equation for the value (v_{ij}^{xy}) at position (x, y) in the j^{th} feature map in the i^{th} layer of the 3-D CNN is given by the following equation [17]:

$$(v_{ij}^{xy}) = act\_fn(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}) \quad (1)$$

where act_fn is the activation function for the i^{th} layer, m is the index for the set of feature maps in the $(i-1)^{th}$ layer connected to the current feature map,P_i and Q_i are the height and width of the convolution kernel, respectively. b_{ij} is the bias for that specific feature map, and w_{ijk}^{pq} is the value of the weight of the kernel at the position (p, q), connected to the k^{th} feature map [17].

A. 3-D Convolution

3-D convolution is performed with a 3-D kernel and 3-D data.

In general, the equation for a 3-D convolution on an $x \times y \times z$ matrix h with an $l \times m \times n$ matrix x is [18]:

$$\sum_{i=1}^{l}\sum_{j=1}^{m}\sum_{k=1}^{n} h(i,j,k)x(l-i,m-j,n-k) \tag{2}$$

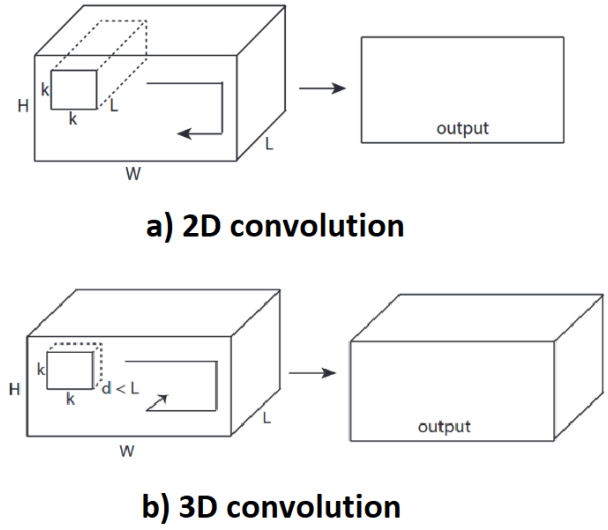

Fig. 1. Visual representation of 2-D & 3-D convolution [19]

B. 3-D pooling

3-D pooling is performed in the n^{th} layer of a CNN to build resilience against distortions in the input from the $(n-1)^{th}$ layer [17].
During 3-D pooling, a 3-D kernel shifts across the data, processing the overlapped data. There are multiple types of 3-D pooling, some of them are mentioned below:

1) Max-Pooling: This type of pooling forwards the maximum value from the overlapped area [20].
2) Weighted-Average Pooling: This type of pooling forwards a weighted average of all pixels in the overlapped area.
3) Average Pooling: The filter forwards the average of all pixels in the overlapped area. (This is a special case of weighted-average pooling where the value of all the weights in the kernel are 1.

Max-pooling is most commonly used because it provides the network with the ability to detect features regardless of rotation or tilting. This is called spatial invariance [20]. Max-pooling is implemented in all the pooling layers used for this project.

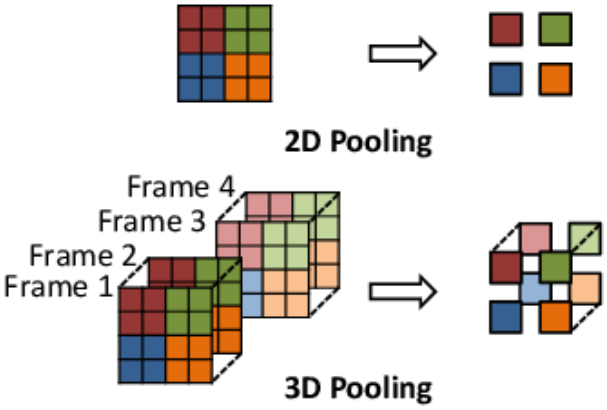

Fig. 2. Visual representation of 2-D & 3-D pooling [21]

C. Fully - Connected (FC) Layer

The Fully Connected layer, is placed after the convolution layers [22]. It gets a flattened input and produces a flattened output. It consists of interconnected nodes. The convolution and pooling layers are independent of the FC layer [22]. The convolution and pooling layers capture the features present in the input data [22]. This is then fed to the FC layer. This is done to reduce computational power required for the last few steps layers [22].

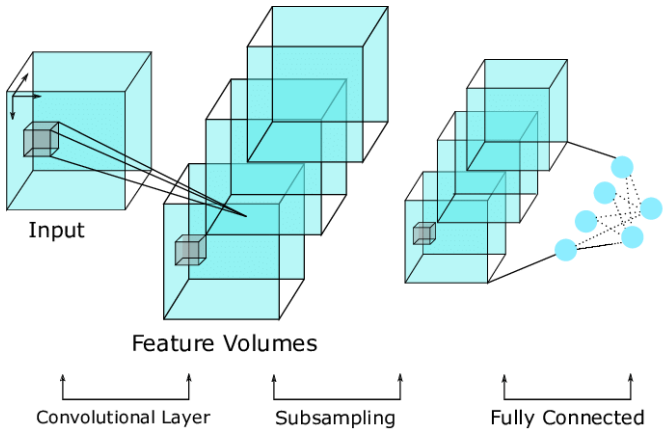

Fig. 3. Basic 3-D CNN architecture [22]

IV. Methodology

A. Experiment

All codes for this project were run on a 25GB NVIDIA Tesla K80 GPU with an Intel Xeon 2.20 GHz processor. All parts of the project were implemented in Python [23].

This report focuses on 2 implementations:

1) 3-D CNN for FER in videos
2) 3-D CNN for FER from 3-D point cloud dataset

B. 3-D CNN for FER in videos

1) Dataset

The BAUM-1 dataset [16] is used for this project. It is an audio-visual affective face database of affective and mental states. The audio is in Turkish. For the scope of this project, the audio aspect of the database is not used.

There are 2 parts of this database: 1. Acted 2. Spontaneous. Both parts are recorded in a studio. This project only uses the spontaneous part of the data set, called BAUM-1s.

Acted datasets are developed under constrained conditions, resulting in recording exaggerated, unnatural expressions. In acted videos, the frames are found to follow a trend where the first frame of each video-sequence contains a neutral expression and the last frame of the sequence reflects the acted expression at its peak [16]. There are 1184 spontaneous video clips and 273 acted video clips in the database. Thus, the 273 acted video clips were discarded for this project.

The dataset contains expressions of 31 subjects, 13 of which are female and 18 male. From the 31 subjects, subject 5 did not provide permission to publish their videos. Thus, only the videos of 30 subjects are used in this project.

Fig. 4. Pictures of the 31 subjects who volunteered for BAUM-1 database. [16]

The target emotions, include the 6 basic ones: Happiness, anger, sadness, disgust, fear, surprise. Many videos in the database target mental states: boredom, contempt, unsure, confused, undecided, thinking, concentrating, interested, curious,and bothered. In this project, only the six basic emotions were targeted. Thus, only 483 videos were used for this project.

2) Face Detection

The video frames have many pixels that contribute to a green background. Thus, face detection is performed on each of the frames in the videos using pre-trained Haar cascade face classifiers. This classifier uses the Viola-Jones algorithm [24]. Once the face is detected, it is cropped from the frame and resized to a size of 64×48 pixels. This significantly reduces the computational power required as the size of input is reduced drastically from 854×480 pixels to 64×48 pixels, by almost 10 times.
A pre-trained face-region Haar Cascade classifier [25] provided by the OpenCV library is used in this project.

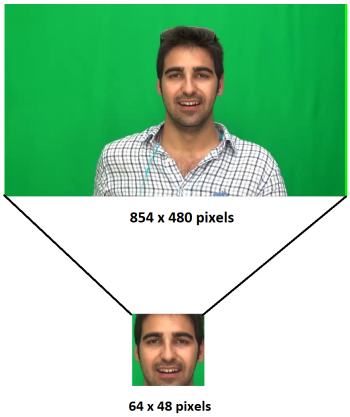

Fig. 5. 1^{st} frame of 10^{th} response video provided by subject 1 before & after face detection [26]

3) Data Structure

The videos in the data set provided are of unequal lengths ranging from less than 7 to more than 1000 frames. 16 frames were sampled from each of the videos at uniform intervals (decided depending on the size of the video) to form a regular input matrix for the CNN. Thus, videos with less than 16 frames were discarded for implementation purposes.
Intuitively, it might seem that on increasing the number of frames sampled and thus, the temporal-information will cause an increase in accuracy, but an increase in accuracy was observed on increasing the number of frames sampled only until 16, and not beyond.
Thus, the project used 483 videos of unequal length, focused on the six basic emotions/expressions of 30 subjects. These 483 videos were split into train and test data sets of 386 and 97 videos each.
The input to the 3-D CNN for training is of dimensions: `(n_train,n_frames,h,w,c)` where:

- `n_train` : number of videos used for training.
- `n_frames`: number of frames sampled from each video.
- `h` : height of each frame (after face detection and resizing).
- `w` : width of each frame (after face detection and resizing).
- `c` : number of channels (3, as images are in RGB format).

Thus, each video is converted into an input sample of dimensions `(n_frames,h,w,c)` For the implementation, the parameters were set to: `n_train : 843, n_frames: 16, h: 64 , w: 48 , c: 3`

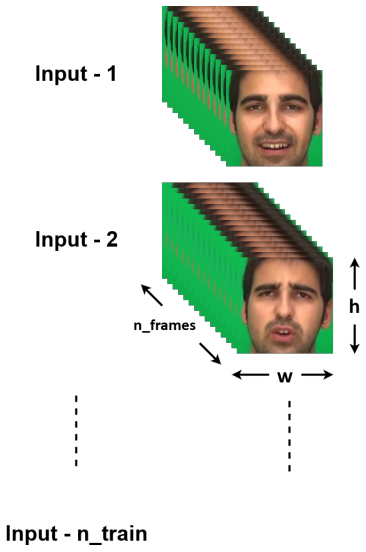

Fig. 6. Visualization of input structure

4) Train-Test Split

If the test-train split is performed without shuffling the videos,the model will train itself to fit only the expressions of the first few subjects. The test-train split is performed after shuffling the data set. It is important to ensure that the samples from each class are not imbalanced in the train and test sets. Thus, the train-test split is performed in a stratified manner to ensure that the percentage of samples from each class is the same in the test and train data sets.
The train-test-split utility [27] offered by scikit-learn in it's module 'models' is used to perform this function with parameters: `random_state=123`, `shuffle=True`, `stratify=labels`.

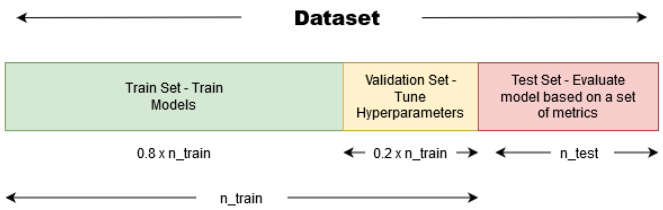

Fig. 7. Visual representation of cross-validation

5) Architecture

The architecture of the implemented 3-D CNN is explained with the following diagram:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_6 (Conv3D)            (None, 16, 64, 48, 64)    1600
_________________________________________________________________
max_pooling3d_5 (MaxPooling3 (None, 8, 32, 24, 64)     0
_________________________________________________________________
conv3d_7 (Conv3D)            (None, 8, 32, 24, 32)     16416
_________________________________________________________________
max_pooling3d_6 (MaxPooling3 (None, 4, 16, 12, 32)     0
_________________________________________________________________
conv3d_8 (Conv3D)            (None, 4, 16, 12, 16)     4112
_________________________________________________________________
max_pooling3d_7 (MaxPooling3 (None, 2, 8, 6, 16)       0
_________________________________________________________________
conv3d_9 (Conv3D)            (None, 2, 8, 6, 8)        1032
_________________________________________________________________
max_pooling3d_8 (MaxPooling3 (None, 1, 4, 3, 8)        0
_________________________________________________________________
conv3d_10 (Conv3D)           (None, 1, 4, 3, 4)        260
_________________________________________________________________
flatten_2 (Flatten)          (None, 48)                0
_________________________________________________________________
dense_4 (Dense)              (None, 48)                2352
_________________________________________________________________
dropout_3 (Dropout)          (None, 48)                0
_________________________________________________________________
dense_5 (Dense)              (None, 24)                1176
_________________________________________________________________
dropout_4 (Dropout)          (None, 24)                0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 150
=================================================================
Total params: 27,098
Trainable params: 27,098
Non-trainable params: 0
```

Fig. 9. Architecture of 3-D CNN implemented on BAUM1-s data set

The activation functions, number of kernels, and dimensions of data at each level of the 3-D CNN are mentioned in figure 8.

All frames of the same colour are grouped together as a single input in 8(i.e. they were sampled from the same video). In the diagram, the number of frames in a single input has been reduced (without scaling) for simplicity purposes (In the diagram, 16 frames were grouped together instead of 64 to represent each input).

The architecture of a 3-D CNN includes an input layer followed by multiple alternating 3-D convolution and 3-D pooling layers. After the convolution and pooling layers, there is a fully-connected (dense) layer.After the FC layer, is the classification layer. [17].
The values of the hyperparameters used for the experiment are as follows: `batch_size = 5`, `no_epochs=90`, `learning_rate=0.001`, `no_classes = 6`, `validation_split = 0.2`, `n_train = 386`, `n_test = 97`

Activation functions [28]: A ReLu [29] activation layer is used for all of the convolution layers. A softmax [30] activation layer is used for the final layer, for multi-class classification.

Error function [31]: Categorical cross-entropy error [32] function is implemented in the classification layer, for better multi-class classification.

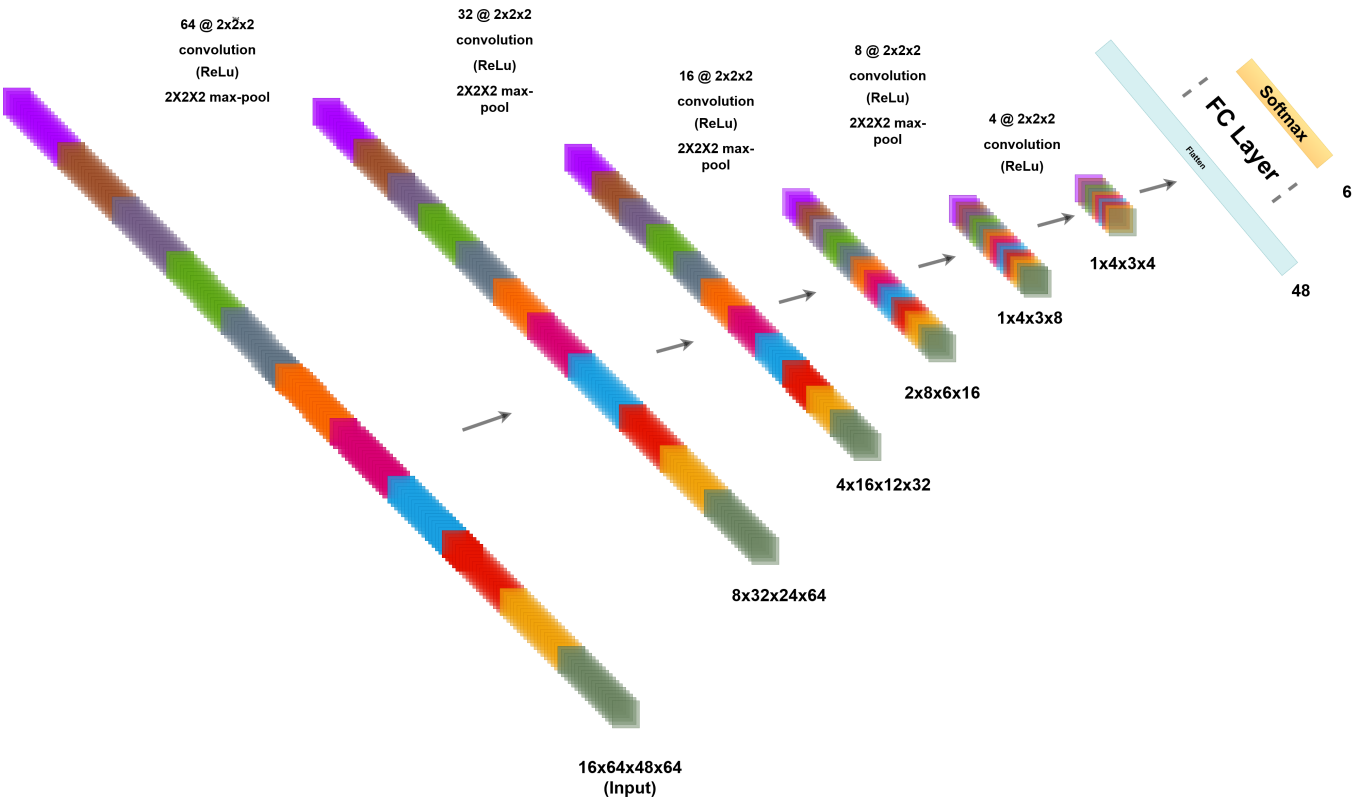

Fig. 8. Structure of 3-D CNN implemented on BAUM1-s dataset

Node dropout [33]: A dropout rate of 0.5 is applied to each fully-connected layer to prevent over-fitting.

Optimizer [34]: Adams optimizer [34] is implemented.

3-D data provides more information but also imposes the cost of a higher dimensionality than 2-D faces. The drawbacks significantly increase for video sequences with higher resolution and frame rates. This raises the total memory required for storage. It also significantly increases computational costs and run-time of algorithms. In this project, the videos used were compressed in mp4 format. On extracting frames from these videos, the MP4 [35] compression was undone, thus counter-intuitively, the memory required to store the frames sampled from parent videos was more than that of the videos themselves. These frames were 2-D images further compressed in JPEG [36] format. Thus, on converting the data to numpy arrays [37], it was counter-intuitively observed that the memory space required to store the numpy arrays was exponentially greater than the memory required to store the videos themselves.

C. 3-D CNN for FER from 3-D point cloud datasets

1) Dataset

3D point cloud data has recently become easily accessible with the development of new technology devices. Even the IPhoneX allows us to capture 3-D images of faces. [26] 3-D images are of two types [38]:

1) Multiviews: such data contains images of the same object from different angles [38].
2) Point clouds: Such data is just a set of points in 3-D space. It can be captured using IR based cameras like Kinect & IphoneX camera [38].

For the scope of this project, a 3-D CNN is used for FER ON 3-D point cloud images. The dataset used for this project is Balcilar's IphoneX dataset [26]. It contains 7 expressions of 50 subjects each. The target emotions are: anger, disgust, fear, happiness, neutral, sadness, and surprise. There is a point cloud corresponding to each expression for each subject in the database. Thus, there are $7 \times 50 = 350$ point clouds in the database.
Each point-cloud consists of 1220 3-D co-ordinates (x, y, z). All the expressions of the subjects were captured in the same orientation. The data set provides normalized point clouds.

2) Voxel Matrices

The term 'voxel' represents the 3-D equivalent of a 2-D pixel. A voxel is a point in a regular 3-D matrix [39]. Neural networks require a regular data structure as input for convolution and pooling [38]. While point clouds are irregular data, voxels are regular data forms. Thus, researchers convert point clouds to voxel matrices, although this conversion causes a loss of structural and textural information [38]. The same conversion method has been approached for this project.

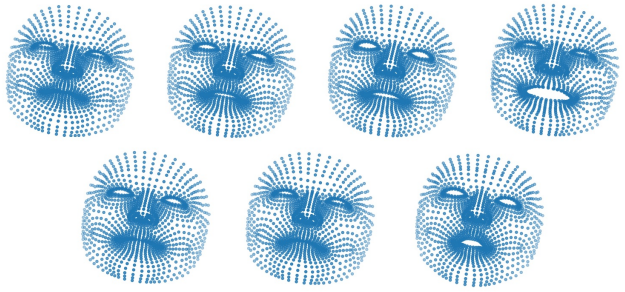

Fig. 10. 3-D point clouds of 7 expressions expressed of the 1^{st} subject in Balcilar's IPhoneX dataset

Algorithm used to convert 3-D point-cloud to 3-D voxel matrix:

Transformation step: Instead of using $1220 \times 3 = 3660$ values directly, the 1220 points are transformed into a regular 3-D $24 \times 24 \times 24$ voxel matrix. As we wish to transform each point cloud to a $24 \times 24 \times 24$ regular matrix, the co-ordinates(x,y,z) are normalized to the $[0 - 23]$ range and rounded off to an integer (ix, iy, iz). This integer represents the index(ix, iy, iz) in the voxel matrix corresponding to the co-ordinate (x, y, z).
Incrementation step: For every combination of (ix, iy, iz) that comes up in the above step, the value of the voxel matrix at the corresponding index in the voxel matrix is increased by 1. Values of the vortex matrix at indices that are combinations of (ix, iy, iz) that don't come up in the transformation step remain 0.
Normalization step: If two or more co-ordinates (x, y, z) return the same transformed value (ix, iy, iz), the corresponding value of the volumetric voxel matrix would become more than 1 at that index (ix, iy, iz). Thus, after the incrementing step, all elements in the volumetric matrix are divided by the maximum value in the voxel matrix to bring all the values in the matrix in the range of $0 - 1$. As a result, the final volumetric matrix representation of the 3-D point cloud is obtained.

The resulting $24 \times 24 \times 24$ matrix has values in the [0-1] range to input to the 3-D CNN. Thus, the dimensions of the training data set are: $(280 \times 24 \times 24 \times 24)$. The dimensions for the test data are: $(70 \times 24 \times 24 \times 24)$.
The converted regular voxel matrix is then used as input to the 3-D CNN. Convolution is performed on this matrix as explained in section III-A.

3) Architecture

The values of the hyperparameters used for the experiment are as follows: `batch_size = 16`, `no_epochs = 50`, `learning_rate = 0.001`, `no_classes = 7`, `validation_split = 0.2`, `n_train = 280`, `n_test = 70`.

Activation, and loss functions, node dropout, and optimizer are the same as that used for the previous implementation.

The architecture of the implemented 3-D CNN is explained with the following diagram:

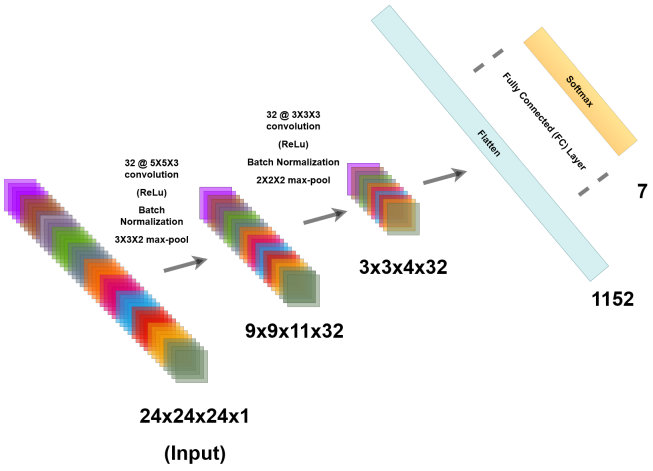

Fig. 11. structure of 3-D CNN implemented on Balcilar's IphoneX dataset

All frames sampled from the same video are grouped by the same colour in figure 11 (i.e. they are from the same video). In the diagram, the number of frames corresponding to each video has been reduced (without scaling) for simplicity purposes. For eg. in the diagram, 3 frames were grouped together instead of 16 to represent each input).

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 20, 20, 22, 32)    2432
_________________________________________________________________
batch_normalization_1 (Batch (None, 20, 20, 22, 32)    128
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 9, 9, 11, 32)      0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 7, 7, 9, 32)       27680
_________________________________________________________________
batch_normalization_2 (Batch (None, 7, 7, 9, 32)       128
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 3, 3, 4, 32)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 178,855
Trainable params: 178,727
Non-trainable params: 128
```

Fig. 12. Architecture of 3-D CNN implemented on point-cloud data set

V. Experimental Results

A. Over-fitting and Under-fitting

To optimise the accuracy of the model, it is important to observe whether it has over-fit or under-fit the data.
When a model does not generalize well from the observed data to unseen data, it is said to be overfit [40]. It occurs because of noise, limited size of training data, and the complexity of classifiers [40].

Overfitting can be prevented by early-stopping [40]. Early-stopping is a method that uses cross-validation error. The accuracy of algorithms stops improving or even get worse after a certain number of epochs, as the model starts learning noise. It is optimal to stop iterations once the cross-validation error starts increasing at $n\_optimal$ epochs. Once the cross-validation error starts increasing, it means that the model has started losing its ability to generalise [40]. If the model is trained for $n\_epochs < n\_optimal$, it is said to be under-fit. If it is trained for $n\_epochs > n\_optimal$, it is said to be over-fit [40].
Early-stopping has been applied in both implementations.

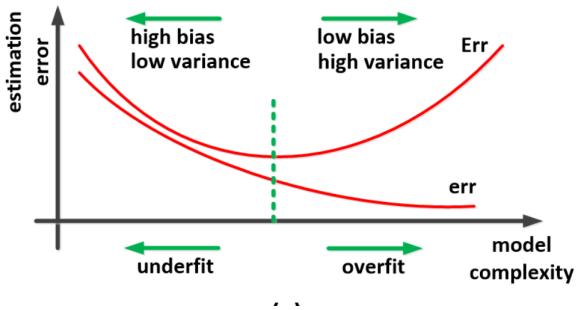

Fig. 13. Visualization of of under & over fitting. The dotted line signifies $n\_optimal$ epochs [41].

B. Metrics

The metrics used to measure performance of both models are:

1) Confusion matrix - It is a 2-D matrix, indexed in one dimension by the true class of an object and in the other by the class of the object predicted by a classifier [42]. In a normalized confusion matrix, the value at the (i, j) index represents the number of samples in the test data that were of class i and classified as class j by the classifier [43]. Thus, the diagonal elements (i, i) show the accuracy of classifying each class of objects present in the data.
2) Accuracy - The accuracy of a classifier is defined as the fraction of samples in the test data that are correctly classified by the classifier. This is equivalent to the sum of values in the first diagonal, in the confusion matrix [43].

C. 3-D CNN for FER in video sequences

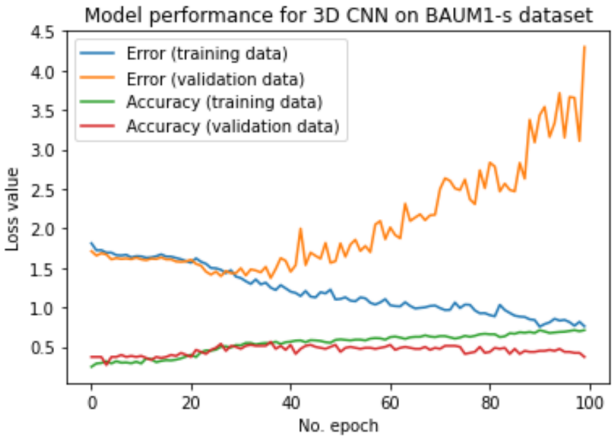

Fig. 14. Plot of accuracy, validation accuracy, error and cross validation error while training on the BAUM1-s dataset.

Accuracy achieved on the test set is: 44.28%. The state of the art accuracy on the BAUM1-s dataset using only the visual modality is 50.11% [44]. The state of the art accuracy on the BAUM1-s dataset using multi-modal methods i.e. by using both sound and audio, is 54.57% [44].

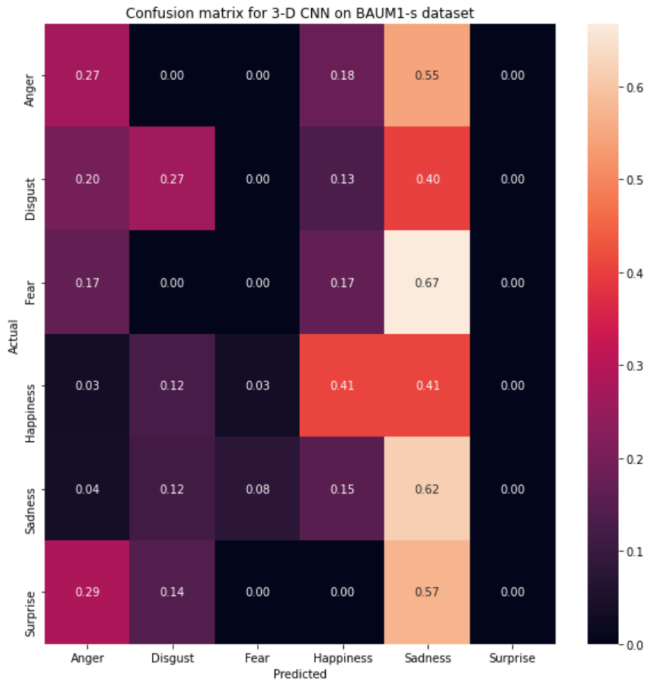

Fig. 15. Confusion matrix for accuracy results on BAUM1-s dataset with aforementioned 3-D CNN

From the confusion matrix, it is notable that the 3-D CNN is not able to learn features relevant to the 'Surprise' class. A possible reason for this is that spontaneous reactions of surprise look similar to those of anger, disgust, and sadness with expressions like raised eyebrows.

D. 3-D CNN for FER from 3-D point cloud

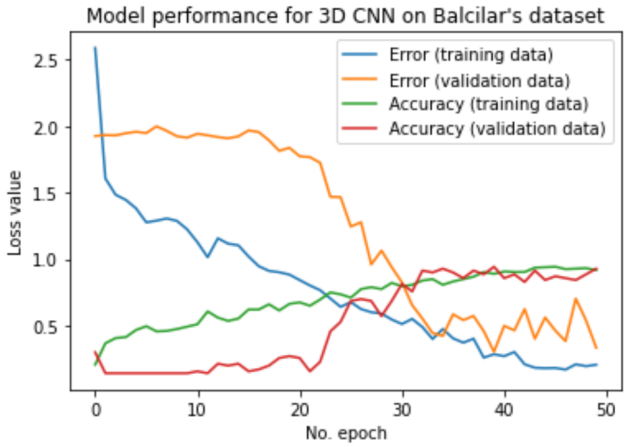

Fig. 16. Plot of accuracy, validation accuracy, error and cross validation error while training on the BAUM1-s dataset with aforementioned 3-D CNN

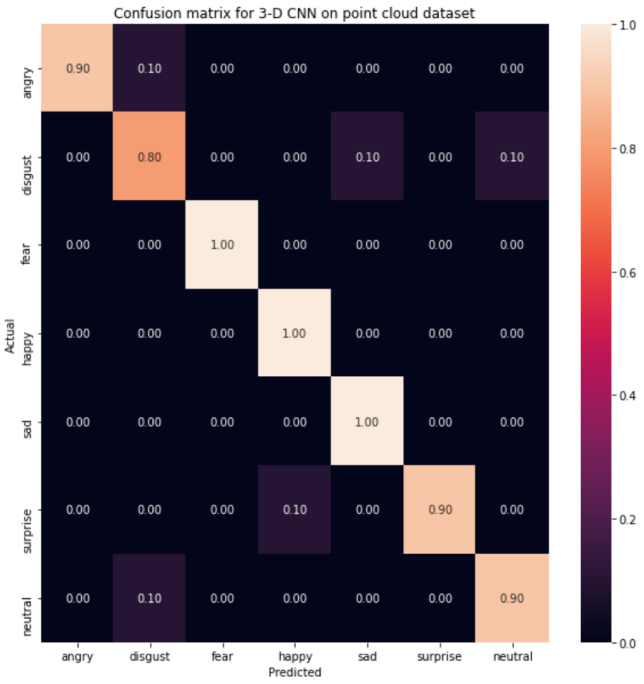

Fig. 17. Confusion matrix for accuracy results on BAUM1-s dataset with aforementioned 3-D CNN

Accuracy achieved on the test set is: 91.28%.

VI. Conclusion & Future Work

Experimental results on the BAUM-1s database show that FER under unconstrained situations is quite challenging.

To reduce the computation and memory costs of using 3-D data, 3-D FER can be performed by mapping the 3-D facial surface onto a 2-D plane. The depth image from the point cloud is computed. A depth image is one where the value of each pixel in the $x-y$ plane is set to the corresponding z co-ordinate of the 3-D co-ordinate in the point cloud [1]. Existing 3-D CNN architectures and methods are unable to completely exploit and assimilate the information in 3-D data [45]. Despite the advance in 3-D algorithms, uni-modal solutions still have drawbacks that make pure 3-D approaches satisfactory only for some applications. These limitations can be overcome by implementing multi-modal 2D+3D analysis [44] [1]. Most FER research has been conducted on the six basic facial expressions: anger, disgust, fear, happiness, sadness, and surprise [1]. Other emotions also need to be considered to develop algorithms that can generalize better.

References

[1] F. Nonis, N. Dagnes, F. Marcolin, and E. Vezzetti, "3d approaches and challenges in facial expression recognition algorithms—a literature review," *Applied Sciences*, vol. 9, no. 18, p. 3904, 2019.

[2] A. Mahmood, S. Hussain, K. Iqbal, and W. S. Elkilani, "Recognition of facial expressions under varying conditions using dual-feature fusion," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[3] M.-C. Su, Y. Hsieh, and D.-Y. Huang, "A simple approach to facial expression recognition," *Proceeding WSEAS*, vol. 2007, 2007.

[4] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[5] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[6] H. Li, J. Sun, Z. Xu, and L. Chen, "Multimodal 2d+ 3d facial expression recognition with deep fusion convolutional neural network," *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2816–2831, 2017.

[7] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*. Springer, 2017, pp. 189–196.

[8] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *arXiv preprint arXiv:1901.06032*, 2019.

[11] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *arXiv preprint arXiv:1808.03314*, 2018.

[12] L. Wang, Y. Zhang, and J. Feng, "On the euclidean distance of images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1334–1339, 2005.

[13] S. Lele and J. T. Richtsmeier, "Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data," *American journal of physical anthropology*, vol. 86, no. 3, pp. 415–427, 1991.

[14] R. Ahdid, K. TAIFI, S. SAFI, and B. MANAUT, "Euclidean & geodesic distance between a facial feature points in two-dimensional face recognition system," *human-computer interaction*, vol. 1, p. 5, 2017.

[15] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan, "Facial landmark detection with tweaked convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 3067–3074, 2017.

[16] S. Zhalehpour, O. Onder, Z. Akhtar, and C. E. Erdem, "Baum-1: A spontaneous audio-visual face database of affective and mental states," *IEEE Transactions on Affective Computing*, vol. 8, no. 3, pp. 300–313, 2016.

[17] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[18] M. Naghizadeh and M. D. Sacchi, "Multidimensional convolution via a 1d convolution algorithm," *The Leading Edge*, vol. 28, no. 11, pp. 1336–1337, 2009.

[19] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[20] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[21] H. Fan, H.-C. Ng, S. Liu, Z. Que, X. Niu, and W. Luk, "Reconfigurable acceleration of 3d-cnns for human action recognition with block floating-point representation," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 287–2877.

[22] A. Roy and D. Mishra, "Ecnn: Activity recognition using ensembled convolutional neural networks," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019, pp. 757–760.

[23] G. Van Rossum *et al.*, "Python programming language." in *USENIX annual technical conference*, vol. 41, 2007, p. 36.

[24] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[25] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[26] M. Balcilar, "Iphonex dataset," https://github.com/balcilar/3D-CNN-Emotion-Recognition/blob/master/faces.py, 2019.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[29] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[30] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *International Workshop on Multiple Classifier Systems*. Springer, 2003, pp. 125–134.

[31] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[32] D. M. Kline and V. L. Berardi, "Revisiting squared-error and cross-entropy functions for training neural network classifiers," *Neural Computing & Applications*, vol. 14, no. 4, pp. 310–318, 2005.

[33] P. Baldi and P. Sadowski, "The dropout learning algorithm," *Artificial intelligence*, vol. 210, pp. 78–122, 2014.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] J. Ozer, *Video compression for multimedia*. AP Professional, 1995.

[36] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[37] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[38] Q. Lu, C. Chen, W. Xie, and Y. Luo, "Pointngcnn: Deep convolutional networks on 3d point clouds with neighborhood graph filters," *Computers & Graphics*, vol. 86, pp. 42–51, 2020.

[39] H. Thaler, P. Ferber, and D. Rottenberg, "A statistical method for determining the proportions of gray matter, white matter, and csf using computed tomography," *Neuroradiology*, vol. 16, no. 1, pp. 133–135, 1978.

[40] X. Ying, "An overview of overfitting and its solutions," in *Journal of Physics: Conference Series*, vol. 1168, no. 2. IOP Publishing, 2019, p. 022022.

[41] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[42] C. Sammut and G. I. Webb, *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated, 2017.

[43] P. Diez, *Smart Wheelchairs and Brain-computer Interfaces: Mobile Assistive Technologies*. Academic Press, 2018.

[44] S. Zhang, S. Zhang, T. Huang, W. Gao, and Q. Tian, "Learning affective features with a hybrid deep model for audio–visual emotion recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3030–3043, 2017.

[45] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5648–5656.