Comparative Analysis Of Deep Learning Models For Analysis Of Neurological Disorders From Facial Features

Project report submitted to Visvesvaraya National Institute of Technology, Nagpur in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology In Computer Science and Engineering

by

Ujjwal Sharma	**BT18CSE021**
Kaustubh Kathare	**BT18CSE024**
Abin Joseph	**BT18CSE025**
Aditya Dhane	**BT18CSE028**

under the guidance of

Dr. S. R. Sathe

Department of Computer Science and Engineering Visvesvaraya National Institute of Technology Nagpur 440 010 (India)

2022

Comparative Analysis Of Deep Learning Models For Analysis Of Neurological Disorders From Facial Features

Project report submitted to Visvesvaraya National Institute of Technology, Nagpur in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology In Computer Science and Engineering

by

Ujjwal Sharma	**BT18CSE021**
Kaustubh Kathare	**BT18CSE024**
Abin Joseph	**BT18CSE025**
Aditya Dhane	**BT18CSE028**

under the guidance of

Dr. S. R. Sathe

Department of Computer Science and Engineering Visvesvaraya National Institute of Technology Nagpur 440 010 (India)

2022

Department of Computer Science and Engineering

Visvesvaraya National Institute of Technology, Nagpur

Declaration

We, hereby declare that this project work titled "**Comparative Analysis Of Deep Learning Models For Analysis Of Neurological Disorders From Facial Features**" is carried out by us in the **Department of Computer Science and Engineering** of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Ujjwal Sharma
(BT18CSE021)

Kaustubh Kathare
(BT18CSE024)

Abin Joseph
(BT18CSE025)

Aditya Dhane
(BT18CSE028)

Date:

Department of Computer Science and Engineering

Visvesvaraya National Institute of Technology, Nagpur

CERTIFICATE

This to certify that the project titled "**Comparative Analysis Of Deep Learning Models For Analysis Of Neurological Disorders From Facial Features**", submitted by **Ujjwal Sharma, Kaustubh Kathare, Abin Joseph, Aditya Dhane** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**, VNIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

(Dr. P. S. Deshpande)
Head,
Department of Computer Science and Engineering VNIT, Nagpur

(Dr. S. R. Sathe)
Project Guide
Professor,
Department of Computer Science and Engineering, VNIT Nagpur.

Date:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, **Dr. S. R. Sathe**, for giving us the opportunity to work on this research project of Comparative analysis, guiding us thoughtfully throughout this project and giving us directions whenever required. We would also like to thank **Sridhar Reddy Gogu** for his guidance and help in the implementation of the project and for providing the necessary resources whenever required by us.

We would like to thank all the teaching and non-teaching faculty of the Computer Science and Engineering Department for supporting us and providing the necessary facilities at all times. We would like to express our gratitude for availing college resources which enabled us to swiftly complete our project.

We would like to acknowledge the contributions of those who have constantly supported and encouraged us which helped in the successful completion of the project.

ABSTRACT

Facial nerve paralysis, such as Bell's palsy is a neuromuscular disorder that results in the loss of muscle control on the affected facial region. An objective and quantitative grading assessment is required for early diagnosis and postoperative recovery. The diagnosis currently relies on the visual inspection by a clinician. The approaches for automatic detection/diagnosis have been emerging in recent years. However, most, if not all, of the approaches use handcrafted features and classifiers, the deep-learning based approaches are yet to be developed. This paper seeks to design a self-sufficient system that isn't reliant on manual feature extraction to provide diagnosis. Designing such a system is not only pertinent to our current problem but would allow us to design an expert system capable of diagnosing all such facial disorders. We look into modeling this self-sufficient system with the help of ML, DNNs and CNNs and study the nuances of their architecture to better tackle our end goal.

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DT	Decision Tree
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
YFP	YouTube Facial Palsy Dataset
DL	Deep Learning

INDEX

CHAPTER 1

INTRODUCTION

Facial paralysis is a medical condition where the patient loses his or her facial movement ability. It is due to neural damage and usually occurs on only one side of the face. The patient can recover from this condition; however, serious sequelae may remain if not treated properly and early.

1.1 Symptoms

Once the facial nerve is damaged, the movement functions will be partially or completely lost, hence causing paralysis to the affected side of the face, which is also known as facial palsy. Typical symptoms of such disorders include inability to frown, reduced elevation of the eyebrow and closure of the eye, loss of blinking and squinting control, droopy lower eyelid, decreased tearing, dropping of the mouth to the affected side, inability to whistle or blow, altered taste, etc. Facial palsy patients may subsequently suffer from various sequelae, including hyperkinesis, synkinesis and atrophy. All of these conditions could result in marked facial disfigurement, interrupt basic human function such as eating, drinking and speaking. The functional disability or impairment may further lead to a wide range of psychosocial problems which are getting attention in recent years.

1.2 Assessment

A variety of facial nerve grading scales have been developed over the years including the most widely used House-Brackmann Scale. These scales classify the degree of facial nerve damage based on a series of rigorously-validated measures, including facial symmetry at rest, differential voluntary facial muscle movement, and secondary features such as synkinesis. To evaluate the degrees of severity of the facial paralysis, a clinician may ask the patient to perform some of these facial expressions. Then, the clinician assigns a score for each expression based on clinical observation. Such an evaluation is subjective. The scores given by the clinicians may differ largely, and this may lead to different treatment decisions. This is highly undesirable from both a medical diagnostics and a treatment consideration, so it is necessary to develop an **objective and quantitative assessment tool** for facial paralysis.

1.3 Approach

Several approaches for automatic detection/diagnosis have been suggested recently. However, most of the approaches use handcrafted features, an accurate and real time deep-learning based approach is yet to be developed. This paper seeks to design a self-sufficient system that isn't reliant on manual feature extraction to provide diagnosis. Designing such a system is not only pertinent to our current problem but would allow us to design an expert system capable of diagnosing all such facial disorders. We look into modeling this self-sufficient system with the help of ML, DNNs and CNNs and study the nuances of their architecture to better understand our end goal.

CHAPTER 2

PROBLEM STATEMENT

2.1 Objective

Our objective is to perform comparative analysis of deep learning methods for the assessment of facial paralysis related diseases. An objective and quantitative assessment tool is necessary for early detection of facial palsy which will help in speedy recovery of the patient. Our main focus is to develop a deep learning model which can accurately classify the disease without the need of providing handcrafted features like distances between facial landmarks or areas of iris etc. and compare our model with existing models. Using this model, we hope to be able to contribute to a larger expert system that is used for automatically diagnosing all patients.

2.2 Challenges

Our first and foremost challenge will be to carefully sanitize the dataset. The dataset needs to be preprocessed to remove all noise and outliers. Secondly, the dataset labels need to be correctly parsed and assigned to the corresponding image. The labeling is important because we need to ensure a symmetric distribution of data to better train our models.

Our next goal is to ensure the models do not underfit, i.e. they are able to understand the classification task correctly. The models need to be able to identify the low-level features that are relevant to the classification task instead of just overfitting to the training data. We need to recognize the correct regularization methods to achieve this objective.

Our main goal is to identify the correct architectures and the relevant hyperparameters that are needed to ensure that the model is performing optimally.

CHAPTER 3

LITERATURE SURVEY

3.1 Modeling and Synthesizing Idiopathic Facial Paralysis [1]

This paper deals with the question of computationally modeling the facial characteristics of BP and synthesizing them. They used a CLM based face tracker to track 68 facial points. These points are preprocessed to remove the effects of translation and rotation by taking the relative positioning with the tip of the nose as the reference point.
In a person without FP, each facial feature of one side of the face has the same distance to the tip of the nose as its corresponding point on the other side of the face. In a person with asymmetric facial expressions, this is not true.
This information is utilized by the authors to design a scaling factor by comparing the spatial distances for the left half and the right half. For a normal patient, this scaling factor comes close to 1, however for affected patients, this score deviates from 1.
This scaling factor was generated for 29 facial points recorded using the tracker. These scaling factors are aggregated to provide an overall score for the patient. It was noted that expressions such as closing of the eyes and smiling had the most dominant scaling factor compared to other expressions.

3.2 Efficient quantitative assessment of facial paralysis using iris segmentation and active contour-based key points detection with hybrid classifier[2]

In this paper they assess facial paralysis by measuring the symmetry between left and right side of the face. For measuring symmetry they created a feature vector using vertical distances between facial landmarks(key points) on each side as well as ratios calculated from iris exposure extraction. They have used Daugman's algorithm and Localized Active Contour Model(LACM) for getting iris exposure and key point extraction respectively.
For both binary and grade classifications they have considered Decision Tree, naïve bayes, Support Vector Machine, and Regularized Logistic Regression as appropriate classification methods. On top of these methods they have also considered a hybrid classifier (rule based + Regularized Logistic Regression).
For grade classification they have used H-B grade scale i.e. 6 class classification. They have considered eye, mouth and forehead regions separately. For these regions they trained 3 hybrid classifiers and one final hybrid classifier combine the results of 3 regional classifiers.
They found that hybrid classifiers provide significant improvement in performance compared to simple classifiers. Forehead and mouth regions give better prediction compared to the eye region. They got 94% accuracy for the H-B grading by using this iris and key points based approach

3.3 Automatic Degree Evaluation of Facial Nerve Paralysis Based on Triple-stream Long Short Term Memory[3]

Here the authors have proposed a triple-stream LSTM which gets the features from the face present in the video for the diagnosis.
The face has been divided into three regions, R5, R6, R7 of which (R5, R6), (R6, R7) overlap. Here the regions R1, R2, R3, R4 are the regions of the forehead, eyes, nose and mouth respectively. They are then merged (so as to have some common features) to form R5, R6, R7.
For the prediction part, they have used three LSTM based models which run in parallel for three of the regions which were specified.
To fuse the vectors which are produced by the LSTMs three more parameters a1, a2, a3 are trained separately by a 1D convolutional layer.
For the diagnosis the patients are asked to perform 5 actions: close eyes, plump cheeks, crew-up nose, open mouth and raise eyebrows.
Using these actions a prediction is made for the grade. They got an average accuracy of around 85%. Note that a different and larger dataset was used for the task.

3.4 A Review on Automated Facial Nerve Function Assessment from Visual Face Capture [4]

In this paper the authors have done a comprehensive review of the studies in automated facial nerve function assessment from visual face capture. The authors systematically reviewed the literature published in the English language from 1977 to 2019. The publications were from the resources of PubMed to Google Scholar. They defined two reviewing criterion, namely Inclusion, i.e facial nerve function assessment from images or videos of the face using computational measures, Exclusion, i.e assessment from non-visual face capture, e.g. electroneurography and electromyography and manual or subjective assessment methods. They extensively reviewed the techniques/studies for automated facial nerve assessment along with two main dimensions , namely computational measurements and assessment outcomes. They divided computational measurements into two different kinds - Computational measures in 2D and Computational measures in 3D. In Computational measures in 2D they specified three categories , The Role Of Facial Landmarks , Static Measures (e.g, distances between intensities of pixels, mediate descriptors of visual texture such as Local Binary Pattern (LBP) histogram features) and Dynamic Measures (e.g , division of face into regions according to facial landmarks or using other segmentation techniques). In Computation measures in 3D they specified two categories , Landmark Based Measures (Angles , distances, and surface between 3D landmarks on the normal side of the face are calculated and compared with the affected (paralysed) side of the face), Surface Based Measurement.
In the assessment of outcomes they divided the outcomes into two main categories - non-semantic numerical values quantifying static, dynamic and synkinetic facial features and semantic grade of facial nerve function designed by the clinician. They further declared that most of the existing solutions fall in the first category.

CHAPTER 4

IMPLEMENTATION RESOURCES

4.1 Python

Python is the gold standard for solving Machine Learning / Deep Learning problems. It offers rich extensibility via the use of open-source libraries. In our work, we utilize **numpy** (for vector computations), **pandas** (for statistical operations), **OpenCV** (for image operations) and **keras** (for high-level ML/DL endpoints).

4.2 Google Drive + Colab

In our project, we used google drive for storage and transfer of data. Google Colab was used for training Deep Learning Models on fast GPUs which massively improved our training time. It also allowed parallel computation on multiple machines reducing our runtime exponentially.

4.3 MTCNN/ Haar Cascades

- MTCNN and Haar Cascades are open-source face extraction libraries available for Python. We utilized them to remove background noise from images to improve the performance of our model.
- Haar Cascades is a fast face extraction. However, it is often inaccurate.
- MTCNN offers pin-point precision and highly accurate face extraction for multiple faces from an image. It also provides flexibility in the shape of the extracted face. However, it has heavy computation and increases computation time.

CHAPTER 5

DATASET

Our goal is to detect Bell's palsy without using manual feature extraction. First Dataset uses handcrafted features given by OpenFace which will be used to train ML and DNN models. This is to compare our final model based on the Second dataset with automated feature extraction which takes videos directly as an input with handcrafted feature model.

5.1 Handcrafted features

5.1.1 OpenFace

OpenFace is an open source toolkit capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code for both running and training the models.[5] It requires a video as an input. OpenFace processes the video frame by frame and gives us basic facial features for each frame separately.

OpenFace facial landmarking provides 68 2D points in terms of x and y coordinates (x0 - x67, y0 - y67) which can be used to calculate distances and areas.

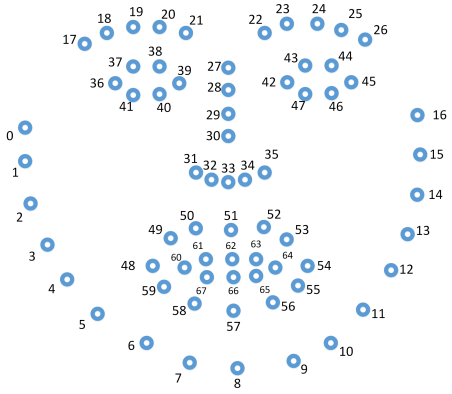

Figure 5.1: OpenFace Landmarking Output [5]

Openface also provides 17 Action units (AU01 - AU45) describing facial movements by their appearance on the face. These action units have intensity parameter (5 point scale) indicating intensity from minimal to maximum.

5.1.2 Features

Using Pandas Python library we calculated 19 euclidean distances between facial landmarks given by OpenFace. For example distance between point 39 and 48 which indicated distance between left eye and left mouth.Similarly we calculated Areas of right eye, left eye, right half mouth, left half mouth using landmarks given by

OpenFace.

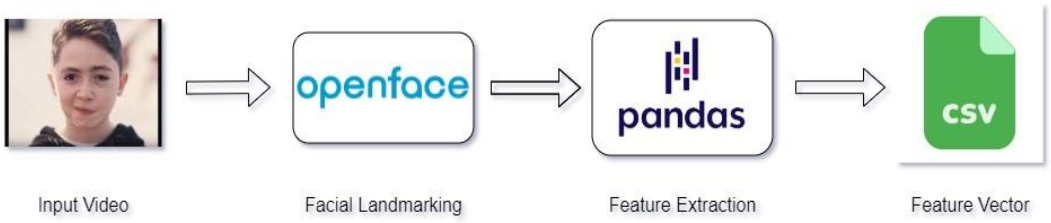

Figure 5.2: Data pipeline for handcrafted dataset

Combining these 19 distances, 4 Areas and 17 Action units we got 40 total features for each frame per video. After that we combined these features using statistical measures as mean, standard deviation to get a single feature vector for the entire video. We also tried using batches of 200 frames which resulted in overfitting and skewed the results.

5.1.3 Final Dataset

We had 41 videos of Bell's palsy patients and 33 videos of normal people. After sending these videos to OpenFace and preprocessing output using pandas we get

Class	**Feature vectors**
Bell's palsy feature vectors	41
Normal feature vectors	33

Table 5.1: Feature vectors for Handcrafted Dataset

5.1.4 Training and Testing Split

We have done 4:1 Training:Testing Split

	Training	**Testing**
Normal	33	8
Bell's Palsy	27	6

Table 5.2: Training and Testing split for Handcrafted Dataset

5.2 CNN Dataset

5.2.1 Background

Main goal of this project is to create a model which can diagnose Bell's Palsy by automatically detecting facial features. In contrast to earlier dataset in which we were using OpenFace and then calculating features like distances and areas to provide input to the model, now we have to directly provide an image as an input to the model.
CNN is the best suited model for this type of requirement. Although we are directly providing images to CNN first we have to do some preprocessing like face extraction to improve our results.

5.2.2 YFP Dataset

Youtube Facial Palsy(YFP) Dataset contains 32 videos of 21 patients from YouTube, and a few patients have multiple videos. Each video is an image sequence with 6 FPS. With videos we have manually labeled local palsy regions (eyes and mouth) labeled by three independent clinicians. [6]

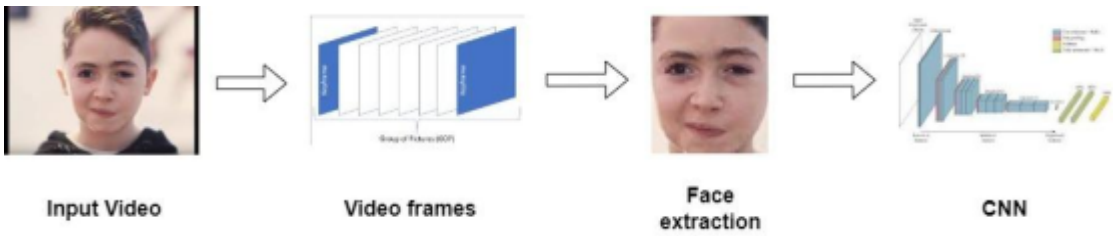

Figure 5.3: Data pipeline for CNN Dataset

5.2.3 Face Extraction

CNN requires an image as an input. Thus we first convert YFP videos into video frames.
These images have a lot of background noise in them. Information is only present on the face, so before feeding these images to CNN, we first have to perform face extraction to remove the background noise.
We explored two methods for face extraction Haar Cascade and MTCNN

5.2.4 Haar vs MTCNN

Haar Cascade is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge and line detection features. Haar cascade is a fast method for face extraction.[7]
MTCNN or Multi-Task Cascaded Convolutional Neural Networks is a neural network which detects faces and facial landmarks on images. MTCNN is one of the most popular and most accurate face detection tools today. It consists of 3 neural networks connected in a cascade.[8]
MTCNN is slower than the Haar cascade method but its advantages include higher Accuracy and rectangular face bounding as Haar cascade can only have square as a bounding box for the face.

	Normal images	Bell's Palsy images
Haar Cascade	2474	2473
MTCNN	3273	2716

Table 5.3: Haar vs MTCNN

The table shows the number of faces extracted by Haar Cascade and MTCNN. As we can see MTCNN has extracted more images in both the classes(Normal and Bell's Palsy). Thus we have chosen MTCNN for face extraction.

5.2.5 Training and Testing Split

We have done 4:1 Training:Testing Split

	Training	Testing
Normal	2673	600
Bell's Palsy	2202	514

Table 5.4: Training and Testing split for CNN Dataset

CHAPTER 6

Machine Learning

6.1 K-Means

6.1.1 Introduction

K-Means is an Unsupervised Learning algorithm that labels unlabelled data into different clusters which is predefined in terms of “k”. It attempts to split the data into “k” groups that are closest to “k” centroids .

6.1.2 Why K-Means

- It is a centroid based algorithm so it is simple to implement w.r.t other machine learning algorithms.
- It generalizes to clusters of different shapes and sizes, such as elliptical clusters.
- It can also be used to detect hidden patterns in the data that are not so intuitive to a human.

6.1.3 Results

Model	**Accuracy**
K-Means (k = 2)	54.28 %

Table 6.1: K-Means Results

6.1.4 Conclusion

As seen in the results the accuracy of the K-Means is not very high rather it performs like a random model . The reasons for such low accuracy might be

- It depends on the initial values (centroids) and can yield different results on different runs
- Outliers might be clustered. Centroids are being dragged by outliers, or outliers might have gotten their own cluster rather than being ignored. Outliers may be present in our dataset because of preprocessing
- The number of dimensions is very high in our case, so the distance-based similarity measure converges to a constant value (Curse Of Dimensionality) .

6.2 Decision Trees

6.2.1 Introduction

Decision Tree is a supervised Learning Algorithm. It is a mathematical method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree [9]. Learned trees can also be re-represented as sets of if-then rules to improve human readability. The main aim is to minimize the entropy of the system so for this we used a measure called the Gini Index. The Gini Index or Gini Impurity which is calculated by subtracting the sum of the squared probabilities of each class from one. It is a greedy algorithm as it goes down the tree, it just picks the decision that reduces entropy the most at that stage.

6.2.2 Why Decision Trees

- The algorithm used (ID3) is simple to understand, interpret and visualize. Output of a Decision Tree can be easily interpreted by humans (in *If-Else* format).
- They can be used for both classification and regression problems and no feature scaling is required .
- Non linear relations does not affect the performance of a Decision Tree like other machine learning algorithms as Decision Tree uses statistical measures (entropy)
- They can also handle missing values as they take account of the whole dataset instead of one data point.

6.2.3 Results

Model	**Accuracy**
Decision Tree	75%

Table 6.2: Decision Tree Result

6.2.4 Visual Representation

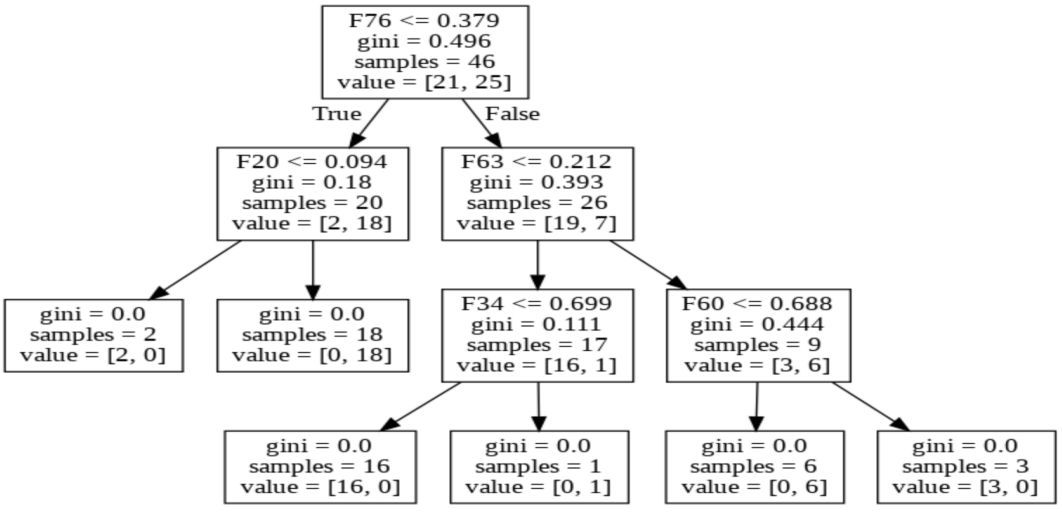

Figure 6.1: Visualization of Decision Tree For Our Dataset

6.2.5 Conclusion

As seen in the results the accuracy of the Decision Trees is higher than K-Means but still it is not very high. The reasons for such accuracy might be

- **Overfitting and High Variance** : This is the main problem of DT. They generally lead to overfitting which makes the predictions incorrect. In order to fit the data (even outliers) they keep generating new nodes of the tree and the Decision Tree becomes too complex to be interpreted by humans. Because of this it loses its generalization capabilities. Also there is a high probability of high variance in the output values which may lead to wrong predictions and hence decrease the accuracy.
- They also become unstable by adding a new data point in the dataset which leads to regeneration of the overall tree.

6.3 Random Decision Forest

6.3.1 Introduction

Random Forest or Random Decision Forests is a Supervised Machine Learning Algorithm**.** They are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time.
It randomly re-samples the input data for each tree (*bootstrap aggregating* or *bagging*). Randomize a subset of the attributes each step is allowed to choose from .For classification tasks, the output of the random forest is the class selected by most trees

6.3.2 Why Random Forest

- Similar advantages as Decision Trees
- Random Forest is based on the bagging algorithm and uses Ensemble Learning technique. In this way it reduces overfitting problems in decision trees and also reduces the variance and therefore improves the accuracy. [11]
- Random Forest is usually robust to outliers and can handle them automatically. [11]
- The Random Forest algorithm is very stable. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees. [11]

6.3.3 Results

Model	Accuracy
Random Forest (500 Voters)	83.34 %

Table 6.3: Random Decision Forest Result

6.3.4 Conclusion

As seen from the results the Random Forest outperforms the Decision Trees as it randomly resamples and trains many different trees .
We tried training several random forest models using different numbers of voters. We observed that if the voters were less than the model was overfitting to the data and was performing poorly in the validation dataset. For large numbers of voters again there was overfitting as the data was less compared to the models to train so the model directly learned the mapping and hence performed poorly during validation .
So we found 500 voters as the optimal value .
Random Forest randomly produces a lot of trees and combines their outputs to finally predict so this algorithm requires much more computational power and resources .

6.4 Artificial Neural Networks

6.4.1 Introduction

Artificial neural networks (ANNs) is a Supervised Learning Algorithm.ANNs is a general and practical method for learning discrete valued, vector valued and real valued functions from examples.There exists algorithms like backpropagation that use gradient descent to tune network parameters to best fit a training set of input-output pairs [9]. ANNs are inspired by biological neural networks that constitute the brain.
ANN consists of a collection of connected units or nodes called (artificial) neurons, which loosely model the (real) neurons in a biological brain. Neurons in the cerebral cortex are connected via axons. A neuron “fires” to the neurons it’s connected to, when enough of its input signals are activated. ANNs are very simple at the individual neuron level but layers of neurons connected to each other in an overlapping manner yields a learning behavior.Neurons have several inputs with a bias. An artificial neuron simply computes a weighted linear sum of the inputs and weights (synapses) with bias and outputs the value after applying an *activation function* (used to introduce non linearity) . In the given figure the neuron is using a sigmoid activation function .

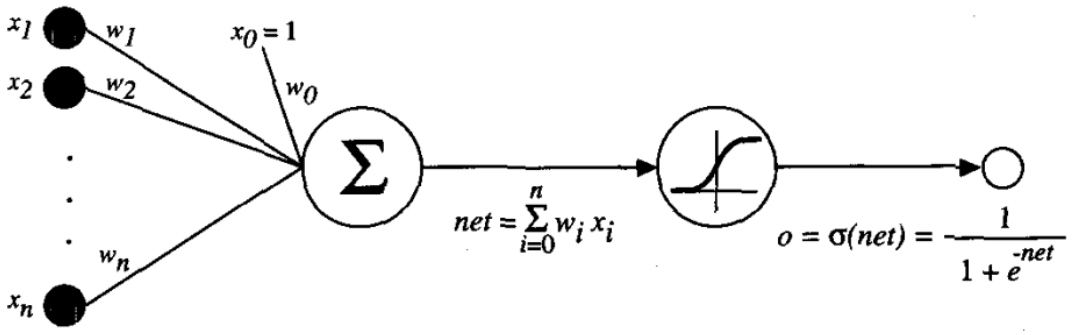

Figure 6.2: Structure Of A Neuron [9]

Learning in ANNs corresponds to learning the weights corresponding to each neuron in the network. Several techniques for training ANNs are developed but as mentioned earlier Backpropagation is by far the most effective technique to train neural networks. Backpropagation uses gradient descent**.** The idea behind Gradient Descent is to compute the rate of change of the loss (gradient) with respect to each parameter, and modify each parameter in the direction of decreasing loss.
The Activation Function is important because in the inner parts of the model, it allows the output function to have different slopes at different values (non linearity) . Using combinations of these different slopes , ANNs can approximate arbitrary functions. There are different types of activation functions as mentioned in the figure.

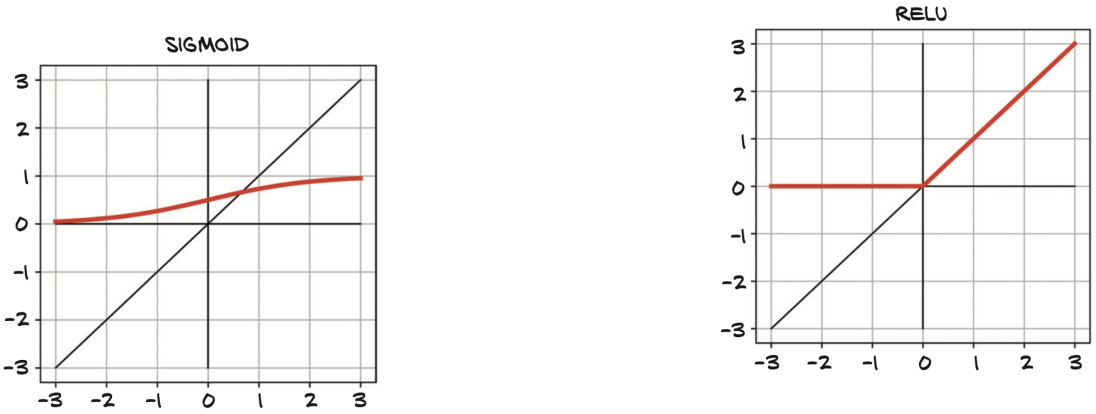

Figure 6.3: Different types of Activation Functions [10]

A loss function (or cost function) is a function that computes a single numerical value that the learning process will attempt to minimize. Conceptually, a loss function is a way of prioritizing which errors to fix from our training samples, so that our parameter updates result in adjustments to the outputs for the highly weighted samples instead of changes to some other samples' output that had a smaller loss [10]. We used "Binary Cross Entropy" Loss which is a loss between predicted labels and true labels specified in the dataset

Training a model involves two steps: optimization and generalization. The main aim of regularization is to ease these two steps. The effective way to stabilize

generalization is to add a regularization term to the loss. This term is used so as to make weights of the model smaller on their own, limiting how much training makes them grow i.e it is a penalty on larger weight values. This makes the loss smoother hence avoids overfitting . Two types of regularization of this kind are L2 regularization, which is the sum of squares of all weights in the model, and L1 regularization, which is the sum of the absolute values of all weights in the model [10] .

Another form of regularization is Batch Normalization .The main idea behind batch normalization is to rescale the inputs to the activations of the network so that inputs have a certain desirable distribution, this helps avoid the inputs to activation functions being too far into the saturated portion of the function which leads in killing the gradients and slowing training [10](vanishing gradient problem).

6.4.2 Why Artificial Neural Networks

- ANNs are robust and can easily approximate complex functions .
- ANNs learning methods are quite robust to noise in the training data. The training examples may contain errors, which does not affect the final output [9].
- ANNs where the fast evaluation of the learned target function is required so this will help predict fast in real time scenarios [9].

6.4.3 Results

Model	**Accuracy**
Deep Neural Network (100→ 50→ 10→ 1)	85.71 %

Table 6.4: Deep Neural Network Result

6.4.4 Conclusion

As seen from the results, the DNN Model outperformed all other models . The main reason for this is the capability of ANNs to approximate complex functions . We have also used all the mentioned regularization techniques (L2, L1, Batch Normalization) and this was the best result we got.

The result is still not satisfactory because due to the preprocessing, a lot of information from the videos (YFP dataset) are lost as 1 video is aggregated into a single feature vector.

6.5 Conclusion

We explored different machine learning algorithms to solve our problem and got results varying from 54.28% to 85.71 % accuracy on the validation dataset with manual handcrafted feature extraction.

Further we explored CNNs , RNNs , LSTMs to deal with this issue of information loss and manual feature extraction.

CHAPTER 7

CNN

7.1 Motivation

Images used in Computer Vision tasks are fairly large in size, if used directly as a feature vector; even a three-layer DNN will require millions of trainable weights. This poses two problems, we require a massively large dataset to effectively train such a large neural net and it exponentially increases the time required as well.

However, images are generally made up of a lot of localized features, meaning we can use a pixel to approximate the behavior of its neighboring pixels. This is the premise of CNNs. They capitalize on this idea to reduce the number of weights, tackling both of the aforementioned problems. Hence, they are used as the gold standard for Image Classification Problems and warrant a discussion in this paper.

7.2 Introduction To CNNs

CNNs are akin to DNNs in their structure, except they introduce the idea of Convolutional Layers and Pooling Layers to curb the problems of DNNs.

7.2.1 Convolutional Layer

Convolutional layer is essentially a kernel that is used to detect local input patterns. Applied on a subset on the input, it can identify low-level features such as edges. This filter is slid across the spatial dimensions of the input, to identify all such local features.

When multiple convolutional layers are utilized, it is shown to extract non-linear combinations of features as well. The filter allows for parameter reuse reducing the number of weights and the sparse connectivity compared to DNNs improves computational complexity.

7.2.2 Pooling Layer

Pooling Layer deals with the approximation of locale to reduce the size of the input. The idea of neighborhood pixels being closely related, allows them to be packed into a lower dimension without a loss of complete information. This helps the model to become invariant to translations while reducing computational complexity.

7.2.3 Hyperparameters

Hyperparameters are used to describe the nuances of the model architecture. They are pivotal in determining the overall performance of the model. They include decisions like activation functions, optimizers and learning rates. However, in CNNs there are a plethora of additional hyperparameters like filter size, output depth, stride and padding that need to be described. This is one of the drawbacks of choosing a CNN model over DNN.

7.3 CNN Models

In this paper, we followed two models for CNNs. First, we designed a CNN model from scratch, then we attempted to design a custom CNN based on pre-trained models using transfer learning.

7.3.1 Basic Model

We designed a simple CNN model from scratch by adding a few convolutional layers and pooling layers to the DNN model we designed in previous experiments. This model was meant to be a precursor to our larger experiment. The performance of this model is to serve as a litmus test for the pre-trained models. Any functional model is supposed to perform at least as good as a model designed from scratch.

7.3.2 Transfer Learning Based Model

Our dataset is not sufficient for training a model from scratch and teaching it to extract low-level features from a human face required for the classification task. The alternative is to take aid from state-of-the-art classifiers and mold them to fit our problem statement. This is known as transfer-learning.

In our paper, we take a look at three important classifiers in VGG16, VGG19 and ResNet as our base model for transfer learning. Each classifier is trained on millions of images to be able to properly extract low-level features and combine them into high level features to be able to properly classify images.

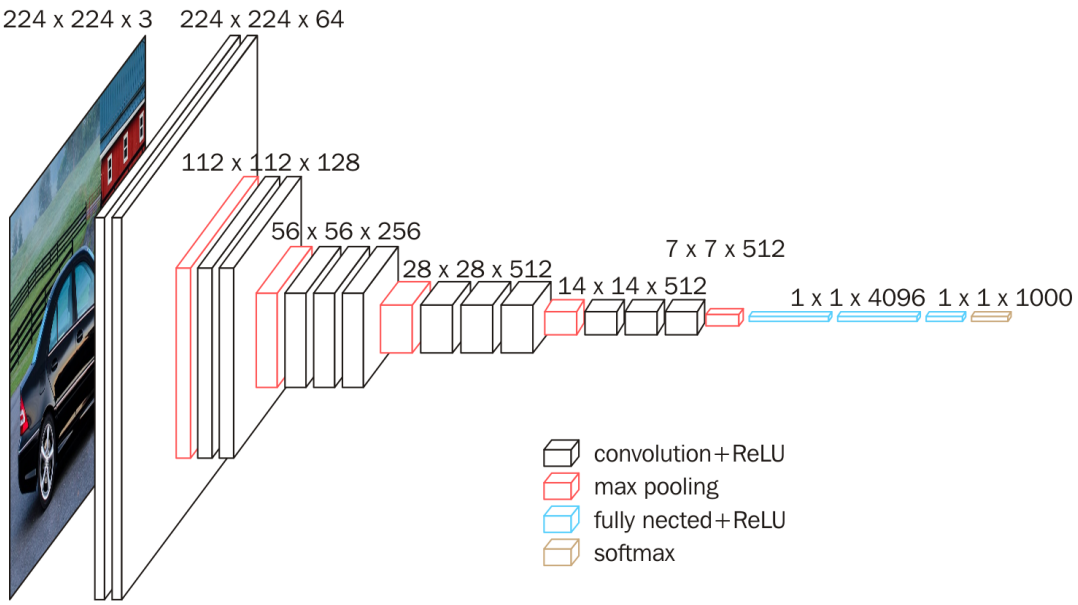

Figure 7.1: VGG16 Model Architecture [12]

Each classifier presents an architecture as shown above albeit with different dimensions. In transfer learning, we remove the output layer, and append custom layers to the pre-final layer. These appended layers are trained to mold the original classifier to accommodate our specific task.

7.4 Experiments

In this paper, we performed extensive research on several variants of the transfer learning models to find the best optimized model for our problem.

Control Parameters - For the CNN model experimentation, we utilized the YFP dataset (which contains ~5000 images). Each model was trained upto 15 epochs and the performance was observed. The performance metrics used in evaluation were Accuracy, Precision and Recall for both training and validation datasets.

Variable Parameters - Each base classifier was experimented with appended nets of different sizes and the performance was observed.

7.4.1 VGG16 Models

Base	Layers	Accuracy	Precision	Recall
VGG16	16→8→4	92.34%	99.07%	85.54%
VGG16	64→32→16	94.76%	98.90%	90.54%
VGG16	128→64→32	93.65%	99.55%	87.80%

Table 7.1: VGG16 Model Results

Observations

- VGG16 has an output layer of size 512. Our observations show that VGG16 worked best with appended layers of smaller size. With an increase in net size, the accuracy further decreased.
- VGG16 models were stable in training and converged well to high performance metrics.

7.4.2 VGG19 Models

Base	Layers	Accuracy	Precision	Recall
VGG19	16→8→4	92.64%	89.44%	96.79%
VGG19	64→32→16	84.10%	86.55%	80.77%
VGG19	128→64→32	68.35%	63.05%	89.58%

Table 7.2: VGG19 Model Results

Observations

- Similar to VGG16, VGG19 performed better with smaller nets.
- VGG19 performed poorly compared to VGG16 and had higher variance during training.

7.4.3 ResNet Models

Base	Layers	Accuracy	Precision	Recall
ResNet	16→8→4	89.82%	88.10%	92.17%
ResNet	64→32→16	84.88%	97.28%	71.89%
ResNet	128→64→32	96.47%	97.53%	95.36%

Table 7.3: ResNet Models Results

Observations

- ResNet has an output size of 2048. Hence, it performed better with increase in the net size.
- During ResNet training, we tried unfreezing the batch normalization layers. Batch normalization layers used pre-trained statistics instead of the new statistics for normalization. This is why it was important to unfreeze these layers.
- ResNet also showed better performance with increase in epochs. While the other models stabilized during the earlier epochs.

7.4.4 Regularization Models

Base	Layers	Accuracy	Precision	Recall
VGG16 + R	64→32→16	94.86%	98.91%	90.78%

Table 7.4: Regularization Model Results

Observations

- We observed that during training of the models, the training accuracy was steeply increasing, indicating overfitting of the data. To curb this issue, we tried dropout regularization to help the model generalize for unseen data.
- We observed best results on the VGG16 model, and it showed very good convergence and stability with high performance as well.
- This model was used as a reference metric for our future models because of its properties.

7.5 Conclusion

CNNs performed very well compared to previous experimentations. They performed well in both training and testing sets and showed good stability properties required for generalization of the model. Our goal for removing the manual intervention by using handcrafted features was successful. CNNs were able to correctly identify the low level features that are required for the classification task by themselves. This makes the goal of designing an expert system that can correctly predict a disorder by identifying the correct features required plausible.

CHAPTER 8

CNN VARIANTS

8.1 Using regularization on the custom layers

8.1.1 Overfitting

While training the previous models we saw that there was slight overfitting happening due to which there was a difference in the training and testing metrics.

In overfitting the model models the training data too well i.e it also learns the noise or the non-relevant features of the training data and as a result it also searches for them in the testing data where if they are absent (due to the testing data belonging to a different distribution) may lead to misclassification.
Thus the model doesn't generalize well.

Some methods to curb overfitting are:

- Training the model on more samples.
- Reducing the complexity of the model.
- Regularization

Out of these methods we'll first look at Regularization.

8.1.2 Dropout Regularization

In dropout regularization, during the model training, we randomly select a node in the layer (after which we add the dropout regularization) and deactivate it for the current epoch.

This is as if we are training a new neural network due to the updated connections with the previous and the next layers.

It also makes the model more robust as the network is forced to learn features that occur in different subsets of nodes.
Note that during testing no nodes are dropped.

During our testing we appended each dense layer with a dropout layer.

8.1.3 Results

Base model	Dropout value	Layers	Accuracy	Precision	Recall
VGG16	0.4	64→32→16	94.76%	97.49%	92.66%
VGG16	0.4	64	92.37%	88.99%	97.95%

Table 8.1: Dropout Regularization Results

Here we have used VGG16 as the base model as ResNet had metrics fluctuating across epochs.
The dropout value of 0.4 gave us the best results.
Layers 64*32*16 denotes that after getting the output of the base model we added a Dense layer with 64 nodes, followed by a Dense layer with 32 nodes and then by 16 nodes and then finally the output is fed to a single node with Sigmoid activation to get a value between 0-1.

8.1.4 Conclusion

Adding the dropout layer showed a small improvement in the recall as compared to the earlier model without dropout.

Also the metrics after adding dropout were not fluctuating when compared to the older model.

8.2 Data augmentation

8.2.1 Data augmentation for reducing overfitting

As said before, training the model on more samples was another way to reduce overfitting.
So by using data augmentation we can increase the dataset size by applying different transformations on the input images to increase their number.
We tried using:

- Horizontal flip
- Vertical flip
- Shear
- Rotation

The idea was to help our model discard spatial relations in the image and to focus on the relevant features at hand.

8.2.2 Results

Base	Layers	Accuracy	Precision	Recall
VGG16	64	94.49%	91.75%	98.63%

Table 8.2: Data Augmentation Results

8.2.3 Conclusion

Slight increase in the metrics was observed.

The reason is that in the data pipeline we are already cropping the face of the person so there is no significant advantage due to the transformations that we are doing.

8.3 Unfreezing layers of the base model

8.3.1 Unfreezing

Here we are using transfer learning so before training the model we freeze the layers of the base model so as to not harm or drastically change the weights of the base model.

The base model (VGG16) was trained on the ImageNet dataset.
So to further fine tune the weights to our specific use-case/dataset we unfroze the last block of VGG16.
The blocks at the start handle low level features so it is not suggested to unfreeze them.

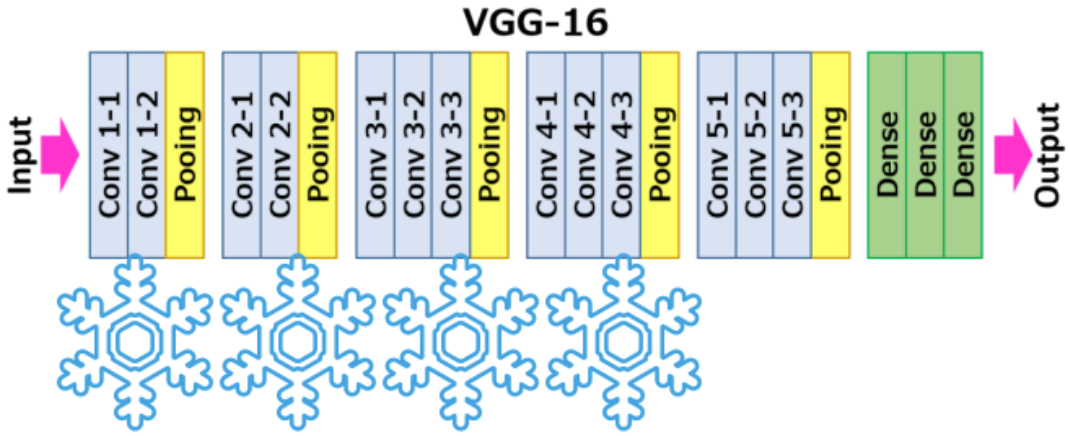

Figure 8.1: Unfreezing last block of VGG16 [13]

8.3.2 Results

Base	Dropout	Layers	Accuracy	Precision	Recall
VGG16	0.4	64	94.46%	93.85%	95.17%

Table 8.3: Unfreezing Layers Results

8.3.3 Conclusion

While training we observed that unfreezing after training and then re-training the model led to overfitting.
So we first unfroze the last block of VGG16 before training which gave these results.
But it seems that it has affected the identification of mid-level features so there is no drastic improvement.

8.4 Removing the terminal layers of the base model

8.4.1 Removing layers of the base model

Here we are removing the last block of VGG16 and sending this output to the newly appended nodes.
The intuition for this is that as an image goes through the convolution layers of the model, the model starts to work with more high level features.
So for example the starting layers could be detecting edges and the later layers would be detecting lips, eyes and so on.

So we thought that if we could work on the lower level features i.e with less summarization, we could get better results as the task requires to detect the asymmetry in the expressions.

So we removed the last block of VGG16 and used the output of Block 4 directly as an input to the newly added layers.

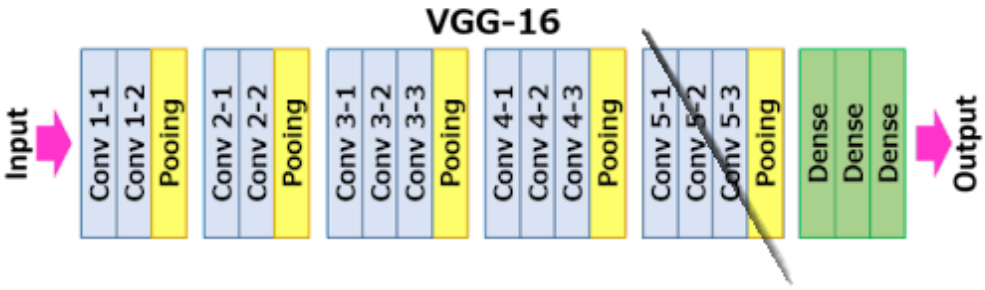

Figure 8.2: Removing the last block of VGG16 [13]

8.4.2 Results

Base	Dropout	Layers	Accuracy	Precision	Recall
VGG16 (Truncate at block 3)	0.4	64	98.59%	97.28%	100%

Table 8.4: Removing Layers Results

8.4.3 Conclusion

This configuration gave us the best results.
Since the recall is 1, no false negatives were produced by this model.

8.5 Conclusion

These were the modifications we tried on the previously discussed CNN based models and they have resulted in an improvement in the metrics.

CHAPTER 9

LSTM

9.1 Background

Muscle movements are significant indicators for the diagnosis of Bell's palsy. For example, patients affected with Bell's palsy cannot shut their eyelid of the affected site. But these movements cannot be captured in a single frame. In a single frame we can only have an open/close eyelid as a source of information; however, a flickering eyelid is a more comprehensive source of information. Thus a single frame cannot accurately represent the overall features required to give a diagnosis. To get these features we have to correlate the information available in multiple consecutive frames, which can be done using LSTMs.

9.2 Introduction

Long short-term memory (LSTM) is an artificial RNN architecture capable of learning long term dependencies in data. It can process images as well as videos by using feedback connections.

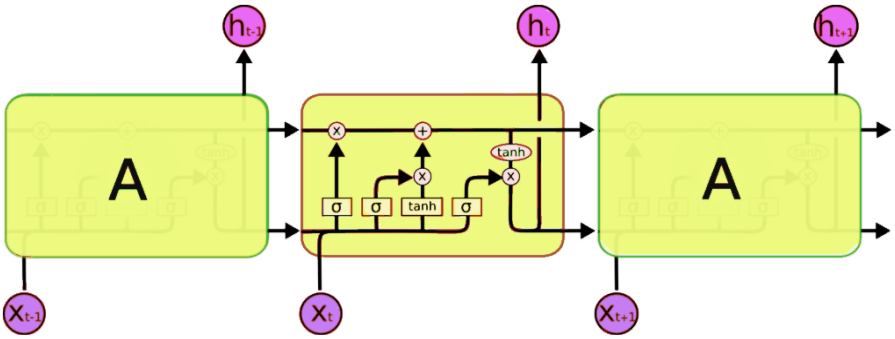

Figure 9.1: LSTM Diagram [14]

LSTM has a recurring module of the model which has a combination of four layers interacting with each other. An LSTM module has a cell state and three gates(intput, output and forget) which helps to selectively learn, unlearn or retain information from each of the units. The cell state has only a few linear connections due to which information can flow through easily. Input gate which uses 'sigmoid' and 'tanh' functions to control the information flow to the current cell state. Forget gate uses a sigmoid function to decide which information from the previous cell state should be forgotten. Finally, the Output gate of the unit decides which information should be passed on to the next hidden state. Although LSTMs provide us with a large range of parameters such as learning rates, and input and output biases: LSTMs take longer to train, require more memory to train and are easy to overfit. [14]

9.3 Results

Training			**Testing**		
Accuracy	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**
0.87	0.84	0.86	0.90	0.85	0.96
Epochs: 10, Units: 32, Batch Size: 4, Dropout: 0.2					

Table 9.1: LSTM Results

9.4 Conclusion

Accuracy of the LSTM models is not satisfactory. During training the training metrics were very fluctuating between epochs which suggests that the model is not stable.

CHAPTER 10

Live/Rolling Prediction

10.1 Background

A single frame cannot accurately represent the overall features required to give a diagnosis.
This is due to the fact that the facial movements of a person while talking might confuse the model into thinking that the person has Bell's Palsy and so may give an inaccurate result.

So we combine the output of our model on multiple frames to further strengthen the prediction.
This method also helps reduce the effect of some outlier frames as their effect gets reduced as we look at some of the previous frames to give a result.

10.2 Methodology

We pass the frames of the video to the model one by one and get the prediction for that frame.
We now use the prediction of the last "**k**" frames to make a prediction for the current frame.

The model which was used for the prediction is the VGG16 based model which was truncated at block 3. (as it had the best metrics)

For combining the output of previous k frames we used exponentially moving weighted average (EWMA).

$$EWMA = \begin{cases} current\_prediction & if\ first\ frame\ of\ video \\ \alpha \cdot current\_prediction\ +\ (1-\alpha) \cdot EWMA & otherwise \end{cases}$$

Here,

- current_prediction is the output of the model for the current frame.
- α is the weightage we give to the current prediction.
- EWMA on the RHS is the previous value of EWMA and the LHS has the calculated value for the current EWMA which will be on the RHS for the next frame and so on.

10.3 Prediction

Although the algorithm for prediction is online, we tried it on a video and added overlays on it for the face bounding box and also showed the output of the model on the top left corner.

The bounding box also changes its color based on the value it predicts.

In our case 0 is Bell's Palsy and 1 is Normal, so if the prediction is near 0, the bounding box will be red and if it is near 1 it will be green and if it is somewhere in between it will be a mixture of green and red with their proportions depending on the value predicted.

For the EWMA we used $\alpha = 0.1$ so that we can give more weightage to the older frames.

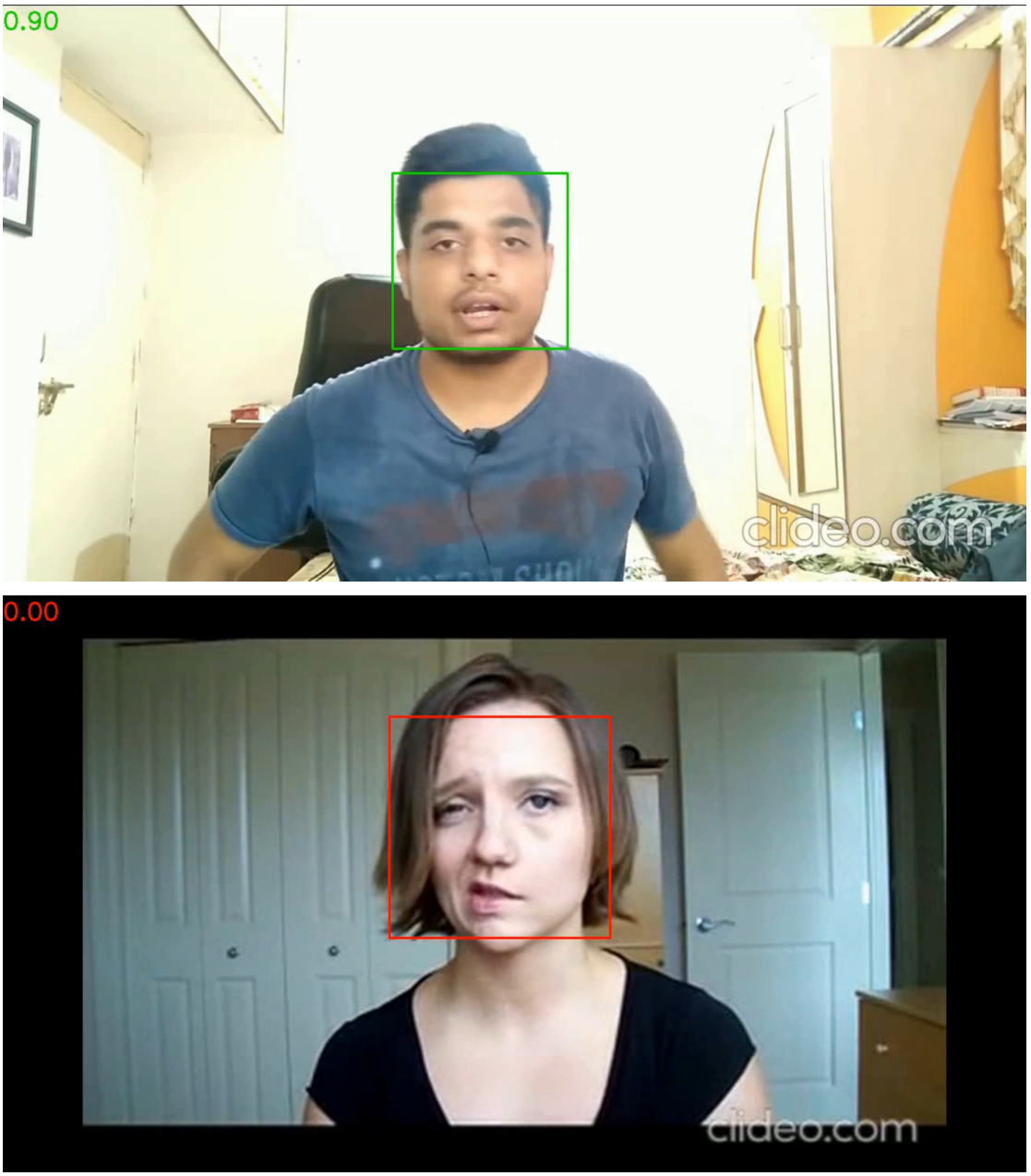

Figure 10.1: Live Prediction example

CHAPTER 11

GRADE PREDICTION

11.1 Introduction

Grade Prediction is the assessment of the degree of a disorder in conjunction to predicting the presence of a disorder. Facial nerve paralysis disorders like stokes or Bell's palsy, result in the loss of muscle control in the affected facial region. An accurate real time grading assessment is required to provide appropriate treatment measures.

11.2 Grading Metrics

11.2.1 HB Scale

The House-Brackmann (HB) Scale is the standard scale used by clinicians to give an assessment for Bell's Palsy.

Grade	Description	Function %
I	Normal	100
II	Slight	76 - 99
III	Moderate	51 - 75
IV	Moderately Severe	26 - 50
V	Severe	1 - 25
VI	Total	0

Table 11.1: HB Scale [15]

11.2.2 Custom Scale

Each image has a predictor for the mouth region and the eyes region. Our custom scale maps these predictors to a 3 point grading system as follows:

Grade 1 → Normal (Assigned to patients with no indication of the disorder)

Grade 2 → Slight (Assigned to patients that are affected in at least one region)

Grade 3 → Severe (Assigned to patients affected severely in both regions)

This grading metric is used in this paper instead of the standard HB scale to accommodate for the faults in the dataset. The insufficient size of the dataset would greatly diminish the size of each class. Also, in order to ensure that each class gets an equi-distribution of images, this particular grading was followed.

11.3 Dataset

The dataset used for Grade Prediction is the YFP dataset. Each data point had a corresponding XML file that assigned predictors for the eye and mouth region. We parsed this XML, converted these predictors to our custom scale and classified the images accordingly. This grading has ensured a near equal distribution of data points for each class.

Grade \ Split	**Training**	**Testing**
Grade I	2073	600
Grade II	1920	561
Grade III	1764	485

Table 11.2: Graded Dataset

11.4 Model

We model the mapping of the predictors to our custom scale as a multi classification problem. To tackle this, we modified our 0/1 classifier to fit this task. We used the best model from the CNN variants (truncated base classifier) as the basis for our new model. We changed the activation function from sigmoid to softmax, to accommodate categorical classification. Then, we removed dropout regularization to allow for faster learning. We finally changed the output layer to a one-hot encoded 3 node layer.

11.5 Experiment

Control Parameters - For the Grade Prediction experimentation, we utilized the YFP dataset (which contains ~5000 images) classified by the grading metrics discussed above. Each model was trained upto 20 epochs and the performance was observed. The performance metrics used in evaluation were Accuracy, Precision and Recall for both training and validation datasets.

Training			**Testing**		
Accuracy	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**
0.70	0.71	0.69	0.72	0.72	0.71
Epochs: 10, Layers: 64 → 32 → 16, Batch Size: 16					

Table 11.3: Grade Classifier Results

Observations

- The Grade Classifier gave an overall accuracy of 72% for a 3 bin classification.
- The Classifier however, showed good stability properties and converged to the final score, instead of fluctuating.

11.6 Classifier Variants

In the YFP dataset, grades have been assigned for both the eyes as well as for the mouth. It will be optimal to train two separate region specific classifiers. Extracting the regional components can be done using Haar cascades. We can then combine the results of the two models(run in parallel) and make a final grade prediction.

This will be an improvement over the current model, because it utilizes the complete information provided from the dataset.

11.7 Conclusion

The performance of the grade classifier was inadequate. The suggested variants curb some of the issues faced by the original classifier. However, it goes against the ethos of the project, which was to design a self-sufficient model that is capable of identifying the necessary features required for the medical assessment. Future work will look into improving and working on these issues.

CHAPTER 12

CONCLUSION

12.1 Summary

To summarize, our paper reviewed different models tasked with the assessment of Facial Paralysis and similar neurological disorders.

We first designed various Machine Learning models and Artificial Neural Networks which worked on handcrafted features to perform the prediction task. We were able to correctly classify patients with an accuracy of 85.71%. These models were meant to be a litmus test for our larger experiment. They were meant to showcase the performance of traditional methods for solving the problem. Any model we would suggest had to at least match the performance of these models.

Our main work consisted of designing CNNs for our classification task. We explored basic and transfer learning models and designed several variants, each outperforming the previous one. Our best consisted model gave us an accuracy of 98.59% for the 0/1 classification task. We attempted other methods like LSTMs, albeit unsuccessfully.

We used non-machine learning techniques of weighted averaging to improve our performance. This technique also enabled our model to account for environmental changes and provide a real-time prediction.

Finally, we worked on the grade classification problem, building on our success from the 0/1 classifier. We were able to get an accuracy of 72% which is close to some of the papers referenced in the literature survey. We also suggested some variants for our model, which could significantly improve the performance.

12.2 Future Work

We scaled a lot of terrain in this paper. However, there is a lot that is left to be desired. In the future, we hope to extend our work, by firstly working the grade classifier variants and bringing it up to speed with the state-of-the-art classifiers.
Next, we hope to design an intuitive UI for a layman to be able to interact with our model. Finally, we hope to explore other new ideas that could bridge the gap in this domain.

REFERENCES

[1] M. Moosaei, M. J. Gonzales, and L. D. Riek, “Modeling and Synthesizing Idiopathic Facial Paralysis” In 14th IEEE International Conference on Automatic Face & Gesture Recognition (2019).

[2] Barbosa, J., Lee, K., Lee, S. et al. “Efficient quantitative assessment of facial paralysis using iris segmentation and active contour-based key points detection with hybrid classifier”. BMC Med Imaging 16, 23 (2016).

[3] Chunlei Yang, Jinlong Kang, Xiaoting Xue, Yang Zhou, Hui Wang, Zhaoxing Wan, Tongsheng Su, Fei Xie and Pengfei Xu “Automatic Degree Evaluation of Facial Nerve Paralysis Based on Triple-stream Long Short Term Memory” In the 3rd International Symposium on Image Computing and Digital Medicine (2015).

[4] Jianwen Lou, Hui Yu, Fei-Yue Wang “A Review on Automated Facial Nerve Function Assessment From Visual Face Capture” In IEEE Transactions on Neural Systems and Rehabilitation Engineering (2020).

[5] “OpenFace 2.2.0: a facial behavior analysis toolkit” (Oct 2019), ·github.com/TadasBaltrusaitis/OpenFace/wiki/Output-Format

[6] “YFP dataset” (Dec. 2018), sites.google.com/view/yfp-database

[7] “Face detection with Haar Cascade " (Dec 2020), towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08

[8] “Face Recognition with Facenet and MTCNN” (March 2020), arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

[9] Tom Mitchell, “Machine Learning, McGraw Hill” (1997).

[10] Eli Stevens, Luca Antiga, and Thomas Viehmann, “Deep Learning with PyTorch :Manning” (2020).

[11] “Advantages And Disadvantages Of Random Decision Forest” (Feb. 2019), http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html

[12] “CNN Models” (June 2020), datawow.io/blogs/cnn-models

[13] “Neural Style Transfer”, renom.jp/notebooks/tutorial/image_processing/neural-style-transfer/notebook.html

[14] “Time Series - LSTM Model”, tutorialspoint.com/time_series/time_series_lstm_model.htm

[15] JW House, DE Brackmann. "Facial nerve grading system" In Otolaryngol Head Neck Surg (1983).