Faculty of Computers and Artificial intelligence Department of Computing and Bioinformatics

جامعة القاهرة
كلية الحاسبات والذكاء الاصطناعي
قسم المعلوماتية الحيوية

Classification and Detection of Fetal Brain Abnormalities using Deep Learning Techniques

Supervised by

Dr. Sabah Sayed

Dr. Zeinab Abd el Haliem

TA. Nora Abd el Hameed

20198110	Aya Khaled Abd El Alim
20198117	Nourhan Fahmy Tamam
20198115	Rahma Rashedy Ramadan
20198010	Eslam hossny mohamed

Graduation Project

Academic Year 2022-2023

Final Documentation

1. Abstract:

Magnetic resonance imaging (MRI) Significantly improved imaging of the fetal brain and uterus, so for medical investigation of the brain it we may consider it as a crucial tool. Approximately between each 1000 there are 3 pregnancies have fetuses with brain abnormality, that's why scanning fetal brain has become fated. Meanwhile, many series of neuro pathological variations occur, some related with serious clinical illnesses. In recent times, fetal MRI is a non-invasive, modern tool to monitor the development of the fetal brain. MRI provides high contrast resolution for the brain texture. ML, DL techniques are usually used to classify and detect tumors and brain abnormalities for fetal MRI photos with none surgical interventions at early stage. As they make use of computer ability of computers to learn and train without being explicitly programmed. Early detection will indicate many things as possible treatments that can be taken, how the mother will manage the pregnancy, and help the parents to understand and prepare themselves for handling the situation. Moreover, early discovery can improve quality of diagnosis and the follow up plans. In this paper we are Counting on deep learning techniques to help us detecting and classifying fetal brain abnormalities using a collection of preprocessing tools and like CNN, Alexnet, Inception3V and CNNSVM to generate wither it's normal or not and try to generate the probability of which type of abnormality the baby could be having.

List of Figures:

List of Tables:

List of abbreviation

DL: Deep Learning

ML: Machine Learning

CNN: Convolutional Neural Network

SVM: Support Vector Machine

MRI: Magnetic resonance imaging

KNN: k-nearest neighbors' algorithm

DT: Decision Tree

NB: Naïve Base

Exp: Experiment

Chapter 1: Introduction

1.1Introduction to the main area:

In today's world, we see that many cases born with disabilities, one of these is brain disabilities. Because of delay of discovery of that appropriate actions are not being taken in the right time so it might lead to permanent disability or even worse consequences like losing the baby. Between each thousand pregnant women there are 3 of them have baby suffer from brain abnormality. In addition to an MRI Scan study in university of California the scanned brains between 9-10 years of 11676 children, they found 1 per 500 has brain abnormalities. Therefore, early detection and classification are important. If somehow, we were able to detect this abnormality in the baby when it is in the fetal stage and operate and medicate, or even know the dimensions the case and whether there is a chance to treat it and the possibilities of its survival. Hence, the role and techniques of machine and deep learning comes to show the usability of it. Machine and Deep learning techniques have the potential to aid this primary detection and classification of fetal brain health, which correspondingly lead to enhancement the diagnosis procedure and follow up plans. In our method of detection, we first capture the brain image of the fetus using MRI (Magnetic Resonance Imaging) technique. Then we have various steps of preprocessing to take in consideration if we want to produce efficient model, like extracting the ROI (Region of Interest) and applying some feature enhancement and reduction techniques (to brief that, we apply all suitable kind of preprocessing available in our hand as much as we can) to obtain more developed and detailed image for the fetus, we do each step and take notice of how each step help in improving the accuracy of the model and the classification of the data. Eventually we apply the model and its various models for classification from both machine and deep techniques.

1.2- Motivation:

Due to delay of discovery of these brain abnormalities, appropriate actions are not being taken in the right time so it might lead to permanent disability or even worse consequences like losing the baby. We aim to Help doctors to try to detect brain abnormalities during pregnancy and even try to classify it before the babies' born in a try of curing the disability avoiding that delay. Usually in some dangerous cases of abnormalities the fetus dies in the mother womb, hours after birth or even days. In other less dangerous cases the fetus could develop to a baby and grow up as a brain disabled person. And there are situations when the mother loses the fetus during the first 3 months and that's usually the period when most mothers know they are pregnant. One of the

reasons they might lose that fetus is because the fetus itself was developing a brain abnormality so then the mother should take an action in treatment. Detection and classification are needed to know all that mentioned previously and more importantly it led to the ability of knowing if the case is treatable.

For example, in some countries if the case is having brain abnormality that is considered dangerous to the extent that it is incompatible with life and no cure for it, the mother here has the choice to terminate it. Briefly, it helps the family and makes them aware and ready in terms of managing the imbalance, this will improve the quality of health management and diagnosis.

Note: Taking actions toward a baby with an abnormality might not take place while the baby is still in the womb. In many cases the abnormality might not affect the baby to born as someone with only a disability but the baby might just be as normal as any baby but with some dangerous symptoms that should be cured or it will affect the baby's life. Also, treatment in many cases doesn't take place while the baby in the womb but the detection itself helps in future planes in treatment after the baby is born.

1.3-Problem definition:

There is always a big gap, which is that the radiologist is not required of him, and it is not his specialty to diagnose radiology, and he is not required to know the diseases related to radiology or to study them, only doing radiology and writing the report, not diagnosing the disease, and his diagnosis cannot be taken or relied upon.

It is the role of the gynecologist and obstetrician (branches that focuses on managing health concerns of the **mother** and **fetus** prior to, during, and shortly after **pregnancy**), which is to diagnose the disease according to what he saw in the rays or MRI's, but here there is a gap, which is that it is a great burden for the gynecologist to study the part devoted to MRI because this subject is far from their specialization, and if we assume that he is able to study it, he cannot memorize it continuously, it comes with practice in the work.

And here comes the role of our model, which makes it easier for gynecologists to know and classify the brain of the fetus, whether it is normal or not, as it makes it easier for the doctor if he is new to practicing the profession or is unable to remember what he studied.

Our application here does not cause the doctor to be dispensed with, but it helps the doctor and removes a great burden from the doctor in memorizing information that is considered a great theoretical burden.

We leave space for them to think, understand and study in his field, which increases his skills in information that the doctor cannot do without or the device plays its role in it, such as other diagnoses and treatment, for example the calculator, it helps in part of solving the problem and not the whole problem, because you solve a large part From the problem, we leave part of the

calculator because it is faster and saves a lot of time instead of recalling the information and helped him give time and dedication to solve the part that the calculator cannot solve, unlike what was in the past.

Thus, a person always strives to make devices that make it easier for him to work and take part of the required steps in the task, so that he devotes himself to the most important and major steps that the devices cannot do.

Here, instead of the doctor remembering or studying and memorizing the basic information, the role of the device comes to give this information to him and apply it to the MRI images. The doctor starts the most important part, which is the treatment based on this diagnosis.

1.4-Project objects:

According to what was previously mentioned in his paper, our motivation is that are trying to help in detection and classification of the fetal brain to determine if there is a fetal deformity and its type before the babies' born in a try of curing the disability and avoid delay which may cause a lot risks and circumstances. This will happen by processing the MRI images and applying machine learning and deep learning techniques.

1.5-Gantt Chart:

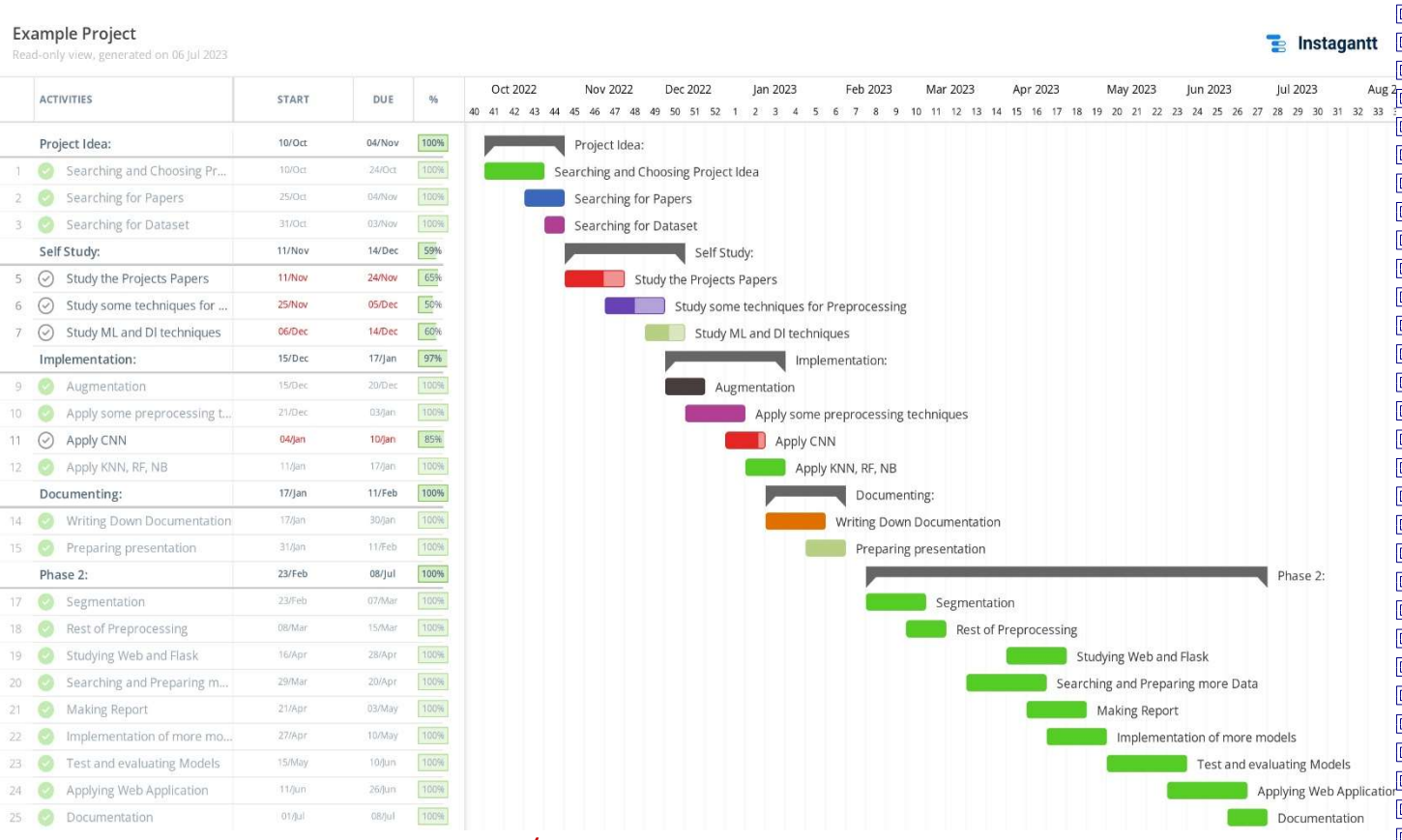


Fig 1.1 (Gantt chart of project time plan)

1.6- Project development methodology:

Our project development methodology follows an iterative testing and continuous integration approach. This means that we regularly test and evaluate our work in progress, and integrate it into the overall project as we go along. By doing so, we can identify and address issues early on, and ensure that our final product meets all the necessary requirements and standards.

First Method (Machine learning)

- Extracted fetal brain MRI dataset from atlas.
- Encountered difficulties due to the small size of the dataset and low image quality.
- Applied a lot of preprocessing techniques to the data, few of them worked but the most didn't help our case.
- Used four machine learning classifiers (KNN, DT, SVM, NB).
- Reported accuracy of each classifier in a table.
- Evaluated the results of the classifier.

As it's obvious from the figures, accuracy of the ml techniques are low and it take long time for each technique to calculate results.

KNN accuracy: 0.7300150829562594
Classification Report:

	precision	recall	f1-score	support
0	0.87	0.59	0.70	357
1	0.65	0.90	0.75	306
accuracy			0.73	663
macro avg	0.76	0.74	0.73	663
weighted avg	0.77	0.73	0.73	663

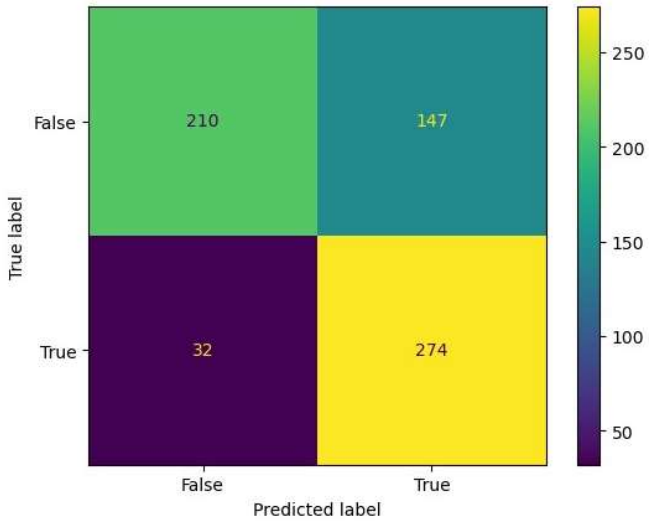


Fig 1.2 Accuracy matrix for KNN

Decision Tree accuracy: 0.6018099547511312
Classification Report:

	precision	recall	f1-score	support
0	0.64	0.61	0.62	357
1	0.56	0.60	0.58	306
accuracy			0.60	663
macro avg	0.60	0.60	0.60	663
weighted avg	0.60	0.60	0.60	663

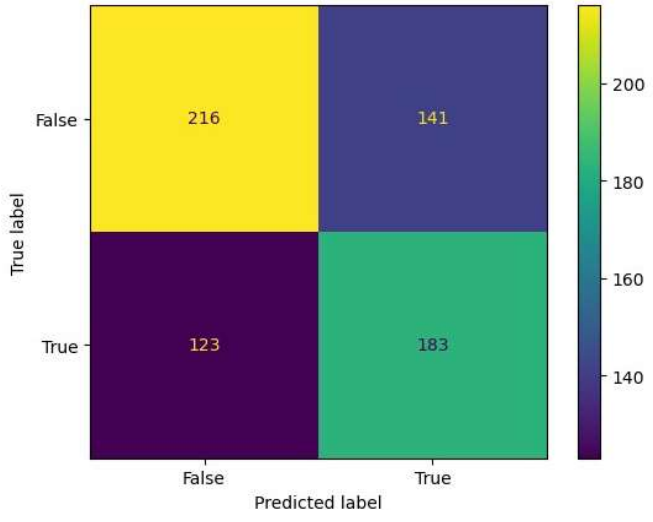


Fig 1.3 Accuracy Matrix For DT

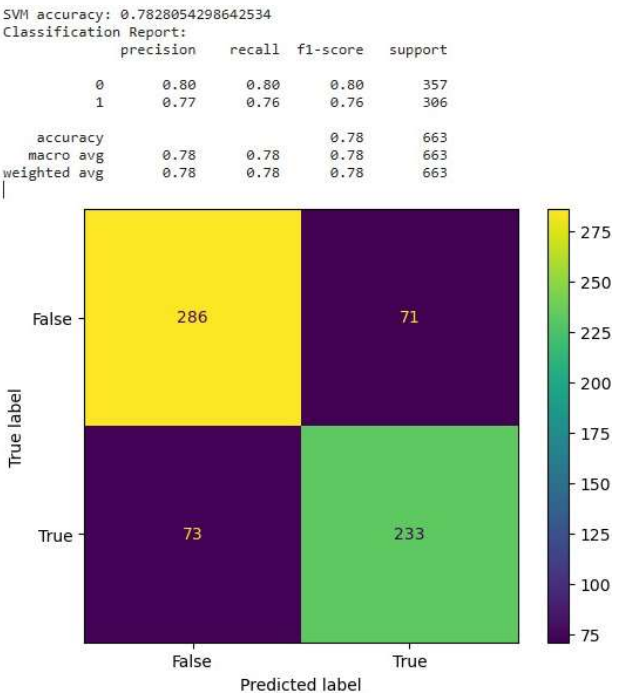


Fig 1.4 Accuracy Matrix for DT

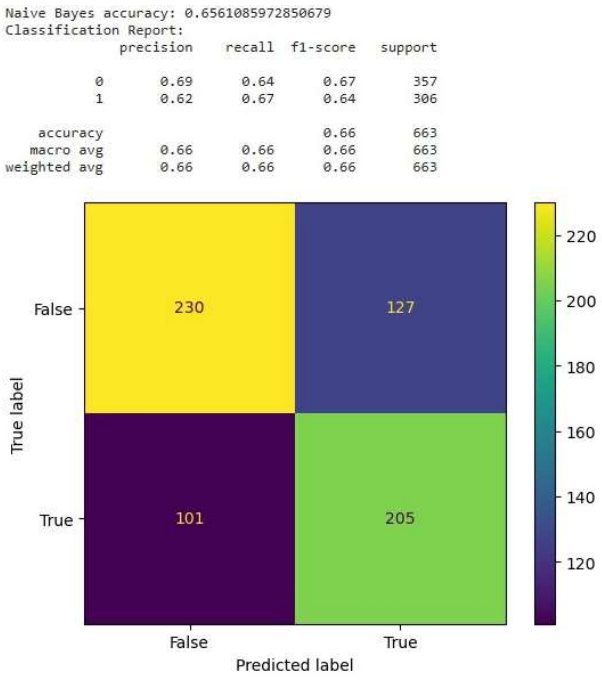


Fig 1.5 Accuracy Matrix For NB

Machine learning techniques takes a really long time to evaluate accuracy and produce lower accuracy in comparison to deep learning techniques with the same preprocessing so we counted on Deep Learning.

Second Method (Deep learning)

5.1- Stakeholders: radiologist,

gynecologist and obstetrician.

5.2- system architecture:

- preprocessing:

Augmentation: 2 types of augmentation has been done, augmentation on all data with batch size = 3, augmentation on only training data with batch size = 7

Resize: big image size produces good accuracy in classification but it requires more time, we used size 150 * 150 not big or small and it's decent for the models.

Enhancing Mask: Unsharp Mask to sharpen the edges enhancing image details and making them appear clearer and more defined.

Normalization: Z-normalization used to standardize image pixel values by subtracting the mean pixel value and dividing by the standard deviation. This helps to normalize the pixel values and make them more consistent across different images.

Dataset: The source of MRI fetal brain images used in this project is from 3 Sources:

1 - Fetal Brain Atlas: It includes 462 images with GA (Gestational Age), around 299 images of healthy brain and around 163 images of abnormal brain) ranging from 16 weeks to 39 weeks.[1]

2- Fetal Brain MRI from Stanford Lucile Packard Children's Hospital which is only belong to normal class can be found in Stanford Digital Repository website. Data Description: folders 1 to 741, subdirectories for each fetal brain MRI in .jpg format, consisting of a sequence for each of the 3 planes (axial, sagittal, coronal), we add some of them to our normal class.[2]

3-Radiopaedia which abnormal chases were downloaded from. [3]

Total data set for both classes reached 1105 image, Normal class = 557, Abnormal = 548.

Plus, we later Add Unknown class having around 500 images from different data sets belong to different categories to handle if the user entered random image.

classification: enter images to DL to classifiers to obtain the result.

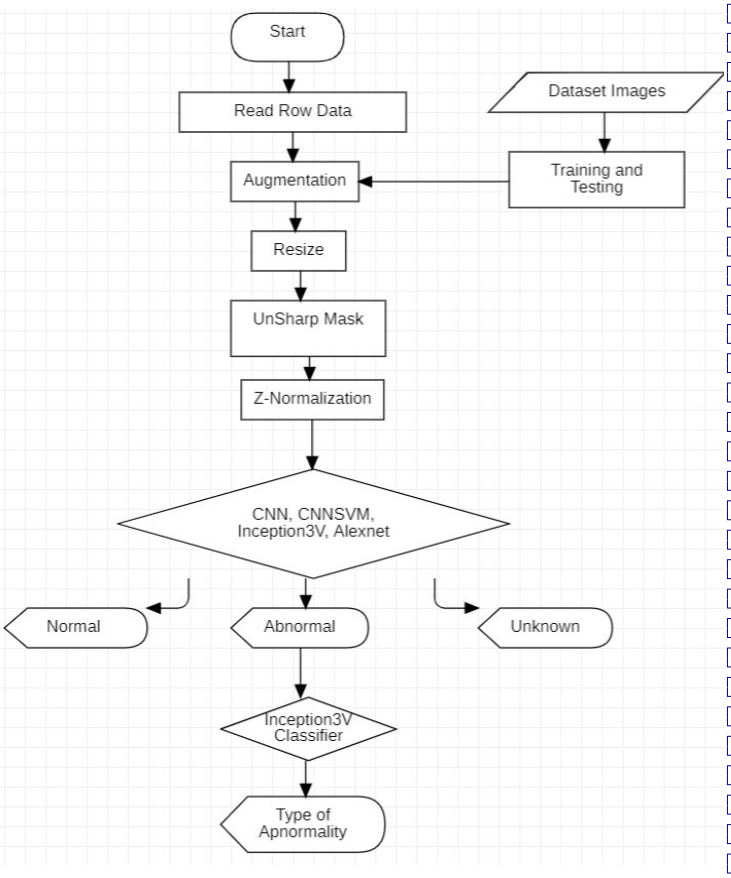
Models: We made 6 models 2 of the are tuned differently (CNN, Inception), with 4 unique model CNN, CNNSVM, Inseption3V, Alexnet.

We Tried different models, including EfficientNetB7, ResNet50 Evaluated results and found CNN, CNN-SVM, Alex Net, and InceptionV3 performed best.

Fine-tuned hyperparameters, starting with batch size and epochs and ending with model architecture.

We found that the best batch size is 64 and 30 epochs for augmented data and batch size 32

for non-augmented.



Flow chart of the Proposed system architecture.

If the image classification is abnormal it will enter another classifier (Inception3) that will classify its category.

The InceptionV3 model (model 4): a pre-trained convolutional neural network that is fine-tuned for image classification tasks. In our case, the last 30 layers of the pre-trained model were unfreezing, and new trainable layers were added. These layers include a GlobalAveragePooling2D layer, Batch Normalization layer, and several Dense layers with LeakyReLU activation function and dropout layers with varying rates. The model also includes a Dense output layer with 2 units and SoftMax activation function, which uses L2 regularization with a value of 0.01.

CNNSVM: The first layer is a Conv2D layer with 32 filters, a 3x3 kernel, a stride of 2, and ReLU activation. This layer is followed by a MaxPooling2D layer with a pool size of 2x2 and a stride of 2. The second layer is another Conv2D layer with 64 filters, a 3x3 kernel, and ReLU activation. This layer is also followed by a MaxPooling2D layer with a pool size of 2x2 and a stride of 2. The third layer is a Conv2D layer with 128 filters, a 3x3 kernel, and ReLU activation. This layer is also followed by a MaxPooling2D layer with a pool size of 2x2 and a stride of 2. Next, there is a Flatten layer, which flattens the output of the previous layer into a 1D array, and a Dense layer with 128 units and ReLU activation. To prevent overfitting, a Dropout layer with a rate of 0.5 is added. Finally, there is an output layer with a single unit, a linear activation function, and L2 regularization. The model is compiled with the Adam optimizer, hinge loss function, and accuracy metric.

CNN Master Model (model2) Architecture was after final tuning: The model starts with an Input Layer that specifies the input shape of the data. It then includes three convolutional blocks, each consisting of a Conv2D layer followed by a MaxPool2D layer and a Batch Normalization layer. The first block has 25 filters, a filter size of 5x5, and uses "same" padding. The second block has 50 filters, a filter size of 5x5, and uses "same" padding. The third block has 70 filters, a filter size of 3x3, and uses "same" padding for the first two layers and "valid" padding for the last layer.

After the convolutional layers, there is an ANN block that flattens the output of the last Conv2D layer and adds two fully connected layers with 100 units each, ReLU activation functions, and Dropout regularization with a rate of 0.25.Finally, there is an output layer with a single unit and a sigmoid activation function, which produces a probability score for binary classification.

Alexnet: A Sequential model in TensorFlow/Keras for image classification using the Alex Net architecture, which was one of the pioneering deep neural networks in computer vision. The model consists of several layers: Conv2D: Applies a 2D convolution operation to the input data, which helps to extract features from the images. This layer is used five times with different parameters, including filter size, activation function, stride, padding, and number of filters. Batch Normalization: Normalizes the output of the previous layer to avoid the problem of internal covariate shift, which can slow down the training process. MaxPooling2D: Performs

max pooling operation on the output of the convolutional layer, which reduces the spatial size of the feature maps and helps to avoid overfitting. This layer is used three times with different parameters for pool size and stride. Flatten: Flattens the output of the previous layer into a 1D array, which can be used as input to the fully connected layers. Dense: Adds a fully connected layer with a specified number of units and activation function, which helps to learn the non-linear mappings between the input and output. This layer is used twice with different parameters for units and activation function. Dropout: Regularizes the model by randomly dropping out some of the units during training, which can help to avoid overfitting. Dense (output layer): Adds a fully connected layer with three output units and SoftMax activation function, which produces a probability score for multiclass classification.

Accuracy:

Accuracy was evaluated using several techniques:

1- Model. Evaluate: This method is provided by Keras, for evaluating the model on training data and test data, the metrics specified during model compilation are used to calculate the evaluation results, including accuracy.
2- Accuracy: evaluation metric provided by the scikit-learn, it is the ratio of correctly classified samples to the total number of samples.
3- Precision: it evaluates the proportion of correctly predicted positive samples out of all samples predicted as positive, evaluates the proportion of correctly predicted positive samples out of all samples predicted as positive. It focuses on the quality of positive predictions. Precision is calculated as TP / (TP + FP), where TP is the true positive and FP is the false positive.
4- Recall (Sensitivity/True Positive Rate): out of all the actual positive samples it measures the proportion of correctly predicted positive samples. Recall is calculated as TP / (TP + FN), where FN is the false negative. It focusses on the model's ability to find positive samples.
5- F1-Score (F-Measure): The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It is calculated as 2 * (precision * recall) / (precision + recall).

6- Confusion Matrix: The confusion matrix is a table that summarizes the model's predictions versus the true class labels. It provides detailed information on the model's performance for each class, including true positives, false positives, false negatives, and true negatives. The scikit-learn library provides the confusion matrix function to compute the confusion matrix.
7- Classification Report: The classification report is a text summary that includes precision, recall, F1-score, and support (number of samples) for each class. It provides a comprehensive evaluation of the model's performance across different classes. The scikit-

learn library provides the classification report function to generate the classification report.

Table 1.1 Table of accuracy for different models with different augmentation techniques

	Without Augmentation	Augmentation On all data	Augmentation On only Train
CNN Main model (model 2)	Accuracy: 0.9276018099547512 Precision: 0.928 recall: 0.928 F-Measure: 0.928	Accuracy: 0.9411764705882353 Precision: 0.942 recall: 0.941 F-Measure: 0.941	Accuracy: 0.9363636363636364 Precision: 0.937 recall: 0.936 F-Measure: 0.936
Inception3V (model 4)	Accuracy: 0.8914027149321267 Precision: 0.898 recall: 0.891 F-Measure: 0.891	Accuracy: 0.9788838612368024 Precision: 0.979 recall: 0.979 F-Measure: 0.979	Accuracy: 0.9 Precision: 0.905 recall: 0.900 F-Measure: 0.900
Alexnet	Accuracy: 0.9577677224736049 Precision: 0.958 recall: 0.958 F-Measure: 0.958	Accuracy: 0.9562594268476622 Precision: 0.957 recall: 0.956 F-Measure: 0.956	Accuracy: 0.9318181818181818 Precision: 0.932 recall: 0.932 F-Measure: 0.932
CNNSVM	Accuracy: 0.9095022624434389 Precision: 0.910 recall: 0.910 F-Measure: 0.909	Accuracy: 0.8929110105580694 Precision: 0.893 recall: 0.893 F-Measure: 0.893	Accuracy: 0.9181818181818182 Precision: 0.918 recall: 0.918 F-Measure: 0.918

1.7- The Used Tools in The Project (SW and HW):

Hardware:

This Project didn't need any special hardware but just personal computers.

Software:

1- Models:

Table 1.2 Table of Models Tools

Language	**Python**: is one of the foremost well-known and broadly utilized programming dialects for machine learning, its wealthy biological system of libraries and systems planned particularly for machine learning assignments, and broad documentation make it a well-known choice for a wide run of applications and ventures.
IDE	**PyCharm**: popular Integrated Development Environment (IDE) created by JetBrains that gives comprehensive bolster for machine and profound learning ventures. Whereas PyCharm is essentially known for its strong Python advancement capabilities, it gives a few highlights and integrative that make it well-suited for machine and profound learning errands as well
	Kaggle: well-known online stage for information science and machine learning competitions, as well as a community of information researchers and machine learning devotees. It gives a wide run of datasets and challenges, at the side instruments for information investigation, include building, and demonstrate improvement. Kaggle too offers an coordinates improvement environment (IDE) for running code, making and sharing scratch pad, and collaborating with other users. With its tremendous collection of datasets and assets, Kaggle could be a profitable device for anybody working on machine learning ventures.

Libraries	**SKLearn**: In scikit-learn (sklearn), you'll be able calculate precision utilizing the exactness score function. The precision score function may be a helpful way to assess the execution of a classification show by comparing the anticipated names with the genuine names (ground truth), and numerous other ways and capacities to calculate the exactness with.
	Keras: Famous open-source deep learning library that gives a high-level interface for building and training neural networks. It is built on best of TensorFlow and gives an easier, more natural API for building deep learning models. Keras permits you to effectively make and prepare deep learning models, with back for different sorts of layers, actuation capacities, and optimization calculations. It too incorporates pre-trained models for common assignments such as picture classification and common language preparing. With its user-friendly interface and effective highlights, Keras is a good choice for building deep learning models.
	TensorFlow: Capable open-source deep learning library developed by Google. It is one of the foremost common and broadly utilized frameworks for building and preparing profound learning models. TensorFlow permits you to form custom layers, loss functions, and optimization schedules, giving you full control over the show design and training process.

2-WEB:

Table 1.3 Tools of Web

IDEs for web	Visual Studio Code (VS Code): is a lightweight, adaptable IDE that provides excellent support for web development. It was created by Microsoft. To customize the IDE to meet your unique needs, a variety of extensions are available. It supports a variety of popular web development frameworks and programming languages.
Languages	PyCharm: JetBrains' PyCharm is best known as a Python IDE, but it also offers robust support for HTML, CSS, and JavaScript web development. It is well-liked by web developers who use Python as part of their stack because it has intelligent code completion, debugging, and version control integration.
	HTML: The building blocks of a web page are HTML. Through a variety of elements and tags, it provides the structure and content of a webpage. It outlines a webpage's structure, headings, paragraphs, images, links, and other content.
	CSS: The presentation and arrangement of HTML elements on a webpage can be managed with CSS, which also improves a website's visual appeal.
	JavaScript: JavaScript is a flexible programming language that enables programmers to include dynamic functionality and interactivity in web pages.
Libraries	Flask: A micro web framework for Python called Flask enables you to create web applications quickly and with little code. It is intended to be small and straightforward, offering only the necessities for building web applications without imposing any particular architecture or structure. Small to medium-sized projects and prototyping are perfect for Flask.

1.8- Report Organization:

Chapter Two: Related Work: in chapter two, we will talk about other related works of our project and will show the differences that we did to achieve our goals .we will declare title of papers , the authors of these methods, Published year, the classification method and dataset they use in their project, and then the result of their project like the accuracy

Chapter Three: System Analysis: In chapter three, we will talk about project specification. it will describe the purpose and goals of the website, the user roles and scenarios, List the functional requirements, user and system requirements and performance details such as speed, security, account creation, password generation, reliability, environment, usability and flexibility, and then illustrate a use case diagram that emphases our program.

Chapter Four: System Design: The System Design Document has some diagrams like System Component Diagram (which is useful for understanding the system's structure and relationships between components),System Class Diagrams(which is widely used to visualize the structure of object-oriented systems),Sequence Diagrams(which is helpful in understanding the flow of messages and the order of events during the execution of a use case),Project ERD(It depicts the logical structure of a system and helps in understanding the relationships between different entities and their attributes) and then System GUI Design(that creating the visual and interactive elements of a software application or system that enable users to interact with the system easily and efficiently.

Chapter Five: Implementation and Testing: in this chapter we will show screen shots and some of details of training and testing of some machine learning techniques as well as the main deep learning models (CNN, CNNSVM, Alexnet and inception3v) and then Screenshots of the system running and samples of the applied test cases.

Chapter 2: Related Work

Related work

Paper 1

Title: Detecting and Classifying Fetal Brain Abnormalities Using Deep Learning
Published year: April-2020

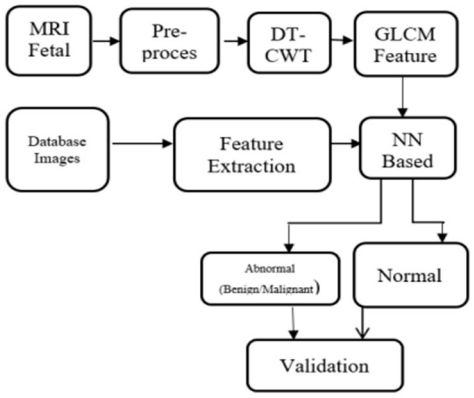


Fig 2.1 Block diagram of Proposed Algorithm for (Detecting and Classifying Fetal Brain Abnormalities Using Deep Learning, 2020)

Methodology:

1- preprocessing: convert RGP to grayscale by adjusting resolution of the image as required and applying e bilateral and Gaussian filter.
2- Feature extraction: using DWT to extract the localization, GLCM to find out the textures in an image.
3- Edge detection It's fundamental tool for image segmentation
4- applying CNN
5 - Software system using python and ML, DL libraries like CUDA, OpenCV and TensorFlow.
Dataset: include 12 fetuses (22.9–34.6 weeks post menstrual age). Images were acquired on a Philips achieve 3T scanner at the University center (UMC) Utrecht, Netherlands.

Paper 2

Title: Recognition and Classification of Fetal Brain Abnormalities
Published online: 20 April 2021

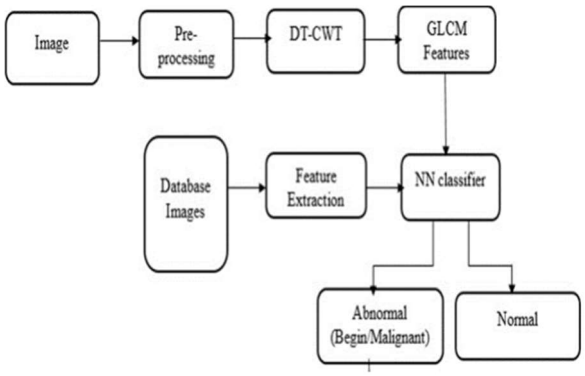


Fig2.2 Block diagram of Proposed Algorithm for (Recognition and Classification of Fetal Brain Abnormalities, April 2021)

Methodology:

1- preprocessing: resize raw images, turn it to grayscale and remove any noisy data
2- segmentation
3- feature extraction: image go into DT-CWT to extract features then go to GLCM phase to extract texture like Energy, Contrast, Entropy etc.
4- CNN classifier
5- (Graphical User Interface)
Dataset: fetal brain atlas, it includes 300 images, (120 normal and 180 abnormal) with GA.
Results: CNN to classify between normal and abnormal with accuracy around 92%-93% and it's calculated using accuracy, sensitivity and specificity.

Results: the approach is CNN is used to classify between normal and abnormal with accuracy of 97.2% with CNN, and 88.5% with SVM.	

Paper 3

Title: Fetal Brain Abnormality Classification from MRI Images of Different Gestational Age.
Published: 12 September 2019.
Methodology: framework in 5 phases:
1-segmentation: segment the brain using 5 phases.
2-enhancement process using local and global filters.
3- feature extraction: several methods were used here including DWT, GLCM and Gabor filters.
4- feature reduction using PCA
5- classification: using several methods like KNN, QDA, Random Forest, Naïve Bayes, Radial Basis Function (RBF) Network, Ensemble Classifiers.
Dataset: from Atlas of Fetal MRI 227 (113 healthy and 114) unhealthy from week 16 to week 39.
Results: calculated using Classification Accuracy, Precision, Sensitivity and Area Under the Receiver Operating Characteristic Curve.

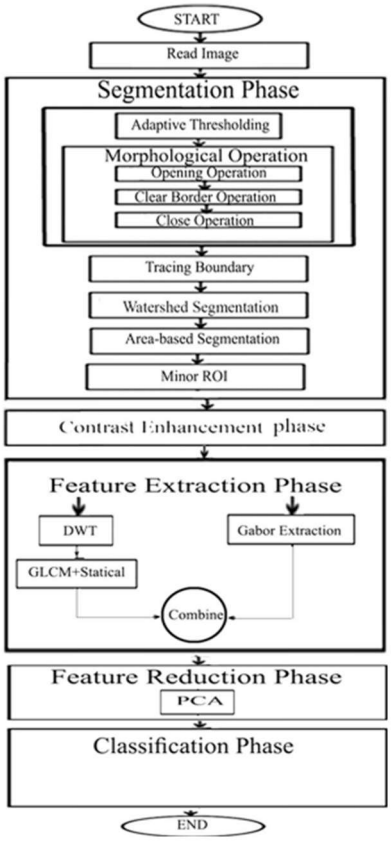


Fig 2.3 A Diagram describing Proposed Algorithm (Fetal Brain Abnormality Classification from MRI Images of Different Gestational Age, 2019)

Paper 4

Title: Deep Learning Techniques for Automatic Detection of Embryonic Neurodevelopmental Disorders.
Published: 7 January 2020.
Methodology:
Preprocessing: Data Augmentation.
Framework consist 4 stages with 3 experiments:
1-Transfer Learning Stag.
2- deep feature extraction.
Experiment 1 (end-to-end DL): using several pretrained DCNNs, pretrained Alexnet, pretrained Googlenet and pretrained Resnet50, each of them was used individually.4096, 1024, and 2048 features were generated from each DCNN respectively.
3- feature reduction: using PCA
4- classification stages
Experiment 2: each DCNN trained using SVM classifiers.
Experiment 3: concatenate every two deep features for every image to form one vector.
Dataset: Atlas of fetal MRI (113 were healthy and 114 had neurodevelopmental disorders) GA between week 16 and week 39.
Results: Accuracy evaluated using Accuracy, Sensitivity and Sensitivity and highest accuracy achieved using this framework was 88.6%.

Table 2.2. Classification accuracy of individual classifiers constructed using several ways for accuracy calculations (Fetal Brain Abnormality Classification from MRI Images of Different Gestational Age, 2019)

Classifier Model	ACC (std)	P (std)	S (std)	AUC (std)
QDQA	91%(0.0092)	99%(0.0052)	83%(0.00516)	98.5%(0.0023)
K-NN	95.6%(0.0056)	97%(0.0067)	94%(0.024585)	99%(0.0067)
RBF Network	93%(0.006)	93.6%(0.005587)	93%(0.0066)	93.3%(0.0082)
Naïve Bayes	91.2%(0.0104)	92.2%(0.008072)	91.6%(0.0098)	97.2%(0.0077)
Random Forest DT	90.31%(0.001)	91%(0.008634)	90.3%(0.0086)	96.3%(0.0099)

Table 2.4. Classification accuracy of individual classifiers constructed using different combinations of feature extraction methods (Fetal Brain Abnormality Classification from MRI Images of Different Gestational Age, 2019)

Classifier Model	Feature Subset 1 (DWT Features	Feature Subset 2 (Gabor Features + PCA)	Feature Subset 3 (GLCM + Statistical Features)	Feature Subset 4 (DWT + GLCM + Statistical Features)	Feature Subset 5 (GLCM and Statistical Features Extracted from DWT)	Feature Subset 6 (Gabor Features + Feature Subset 5+ PCA)
QDQA	71%	89%	50%	50%	74%	91%
K-NN	62%	94.3%	63%	63%	65%	95.6%
RBF Network	70%	92%	67%	71%	71%	93%
Naïve Bayes	71%	90%	63%	72%	73%	91.2%
Random Forest DT	70%	86.3%	71.8%	74%	72.2%	90.31%

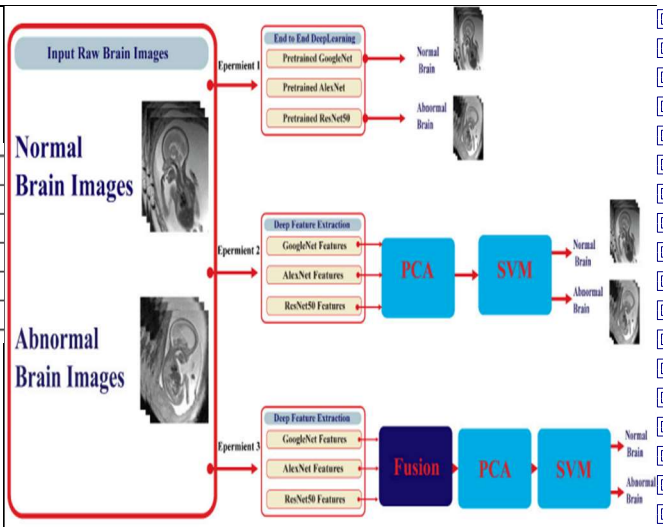


Fig 2.4 A block diagram of the propose Algorithm (Deep Learning Techniques for Automatic Detection of Embryonic Neurodevelopmental Disorders, 2020)

Table 2.3 accuracy of the proposed algorithm for (Deep Learning Techniques for Automatic Detection of Embryonic Neurodevelopmental Disorders, 2020)

The proposed framework	Deep features: GoogleNet + AlexNet + ResNet 50	Linear SVM	84.2%
		Quadratic SVM	87.7%
	Deep features: AlexNet + ResNet 50	Linear SVM	87.2%
		Quadratic SVM	88.6%
	Deep features: GoogleNet + ResNet 50	Linear SVM	86%
		Quadratic SVM	83.8%
	Deep features: GoogleNet + AlexNet	Linear SVM	82%
		Quadratic SVM	86.8%

- **the main difference between our work and the other papers:**

1-Larger Dataset: as we mentioned before we used dataset from 3 different places and we increased the data many times. The largest data used in a paper is around 300 images, we used 1105 image in both classes.

2- Report to classify some of abnormalities: we were able to reach to a data for different abnormalities, we gathered around 6 brain diseases to generate an additional output for the abnormal class images and which abnormality the image might belong to.

3- Different methods for preprocessing and different classifiers: out methods were much simpler than other papers, simple preprocessing with some models we were able to reach a high accuracy.

4- Web application: it's the GUI for our application and it displays more then a model to test them in the application.

5- Additional Precaution: we add third class to our dataset called unknown that if any image that doesn't belong to normal or abnormal will be classified as unknown, of course unknown category can be anything other than the 2 classes but it have 500 images and it's achieved acceptable accuracy for most of our models.

6- Higher Accuracy: accuracy achieved by any of the last version of the data (with or without augmentation or with augmentation only on training) accuracy was above 90 or slightly under 90 in some models with only using deep learning models and few preprocessing techniques.

Table 2.5. Comparison of different papers Accuracy

	Our Paper	Paper1	Paper2	Paper3	Paper4
Accuracy	97.9 %	97.2 %	92%-93%	95.6%	88.6%

Chapter 3: System Analysis

3.1 Project specification

3.1.1. Functional requirement

System Requirements:

1) Login: If the user already registers before and his/her name and password saved in database before, the user will log in to the app.
2) System should be able to verify the user name and password and display error message if any of them is wrong.
3) Sign up: If this is the first time for the user to use the website, he/she will register and her/his name, password and email will be saved inside database.
4) Home: This is the main page of the website where the buttons of navigation through the system lay, each button express a part in the website.
5) In classification page, input of system should be an image that is selected by the user that must be classified.
6) the output should be the classification of the image and a report explain what is type of the disability if the image is abnormal.
7) If the image is not MRI or it's a random image the system should be able to detect that.

User Requirements:

1) Login: user need to be able to login to the system if they already have an email.
2) User should be able to sign up if they don't have an account in the website.
3) User should be able to upload the image they want to classify.
4) User is able to choose from different models in classification.

non-functional requirement:

1- speed: Our website can quickly apply classification to MRI images.

2- Security: Store patient medical records in secure databases. Firewalls may be used to protect their databases from unauthorized access. Examples of typical software security measures are provided below.

3- Account creation: In order to access applications that store information and display profiles, users may be asked to create accounts through our system. Account access is typically granted by our security system when users enter the proper username and password.

4- Reliability: the probability of the prediction is generated through models with high accuracy tested many times so it's reliability.

5- Environment: Our system depends on the number of epochs used to run our model and applies specific filters to images, which could impact its speed or accuracy.

6- Flexibility: user won't have to learn complex guide to be able to use it.

3.2. Use case Diagrams

in use case diagram user can make signup to system and system can interact with function by verify user name and password in database to make login and explore website then upload image, chose model to classify image and clear the container if he wants and system generate the result.

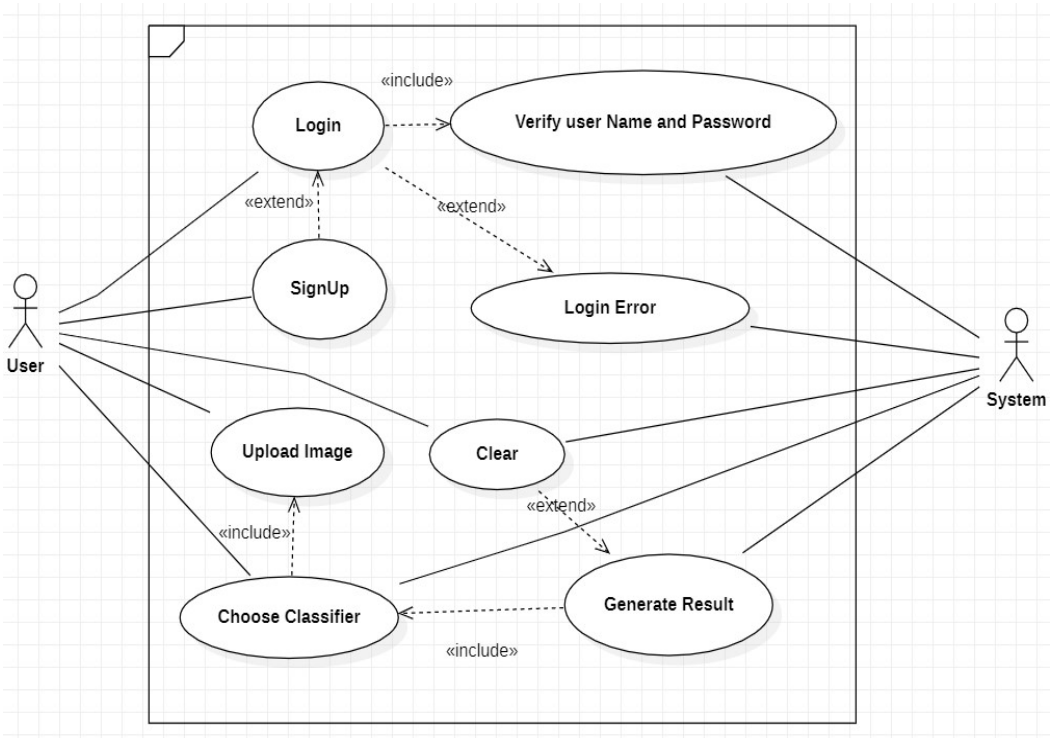


Fig 3.1 Use Case Diagram

Chapter 4: System Design

System Component Diagram

it's showed the structure and relationships of the components within a system. web app GUI component includes authentication and register that take user info to database get connect with rest of page, upload image that connect with image classification.

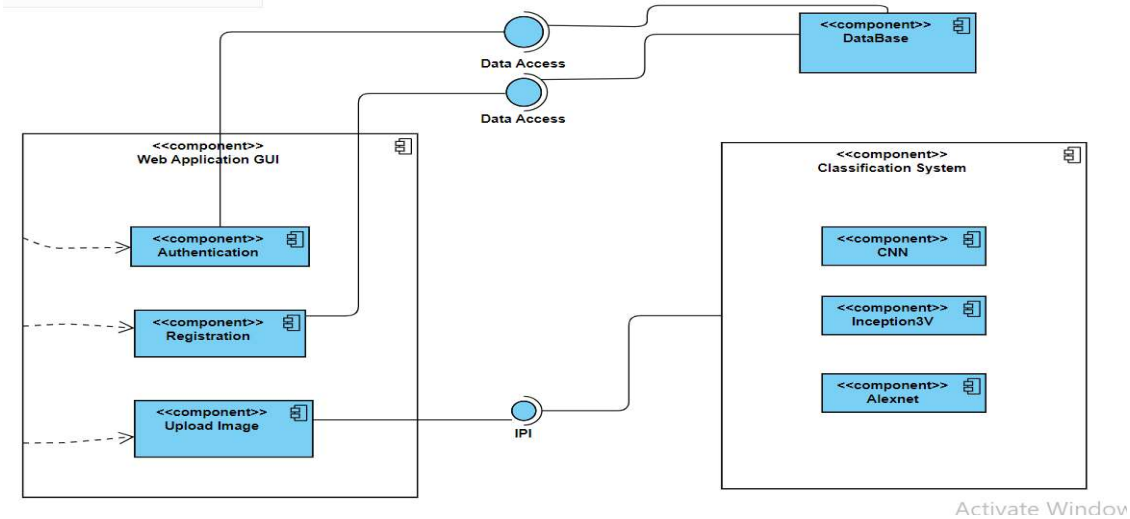


Fig 4.1 System Component Diagram

System Class Diagrams

it's show the static structure and relationships of classes within a system .the user class has username and password and email that give user permission to connect with the website within login and sign up methods and system class has image attribute that user entered to classify within four model CNN ,inception3v and Alexnet that prepare deep learning and predict the classification of the image.

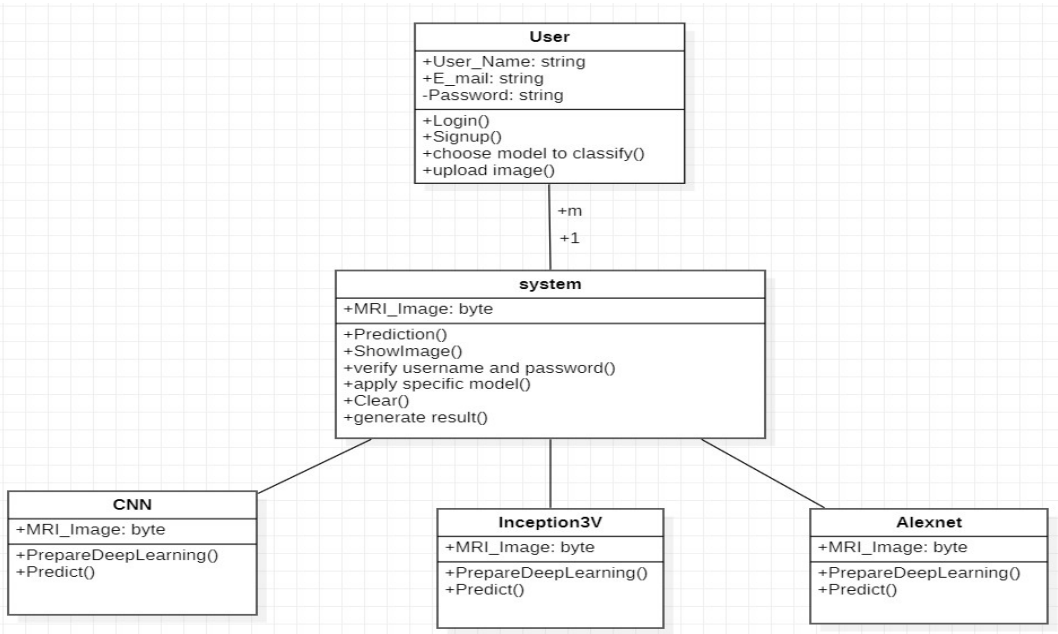


Fig 4.2 System Class Diagram

Sequence Diagram

that shows the interactions and communication between object in a system .doctor is the lifeline that interact with all component by login function but should be registered before (username and password should be same with database) and then can upload image in GUI and chose model that predict and return the classification of image and report in case of abnormal image.

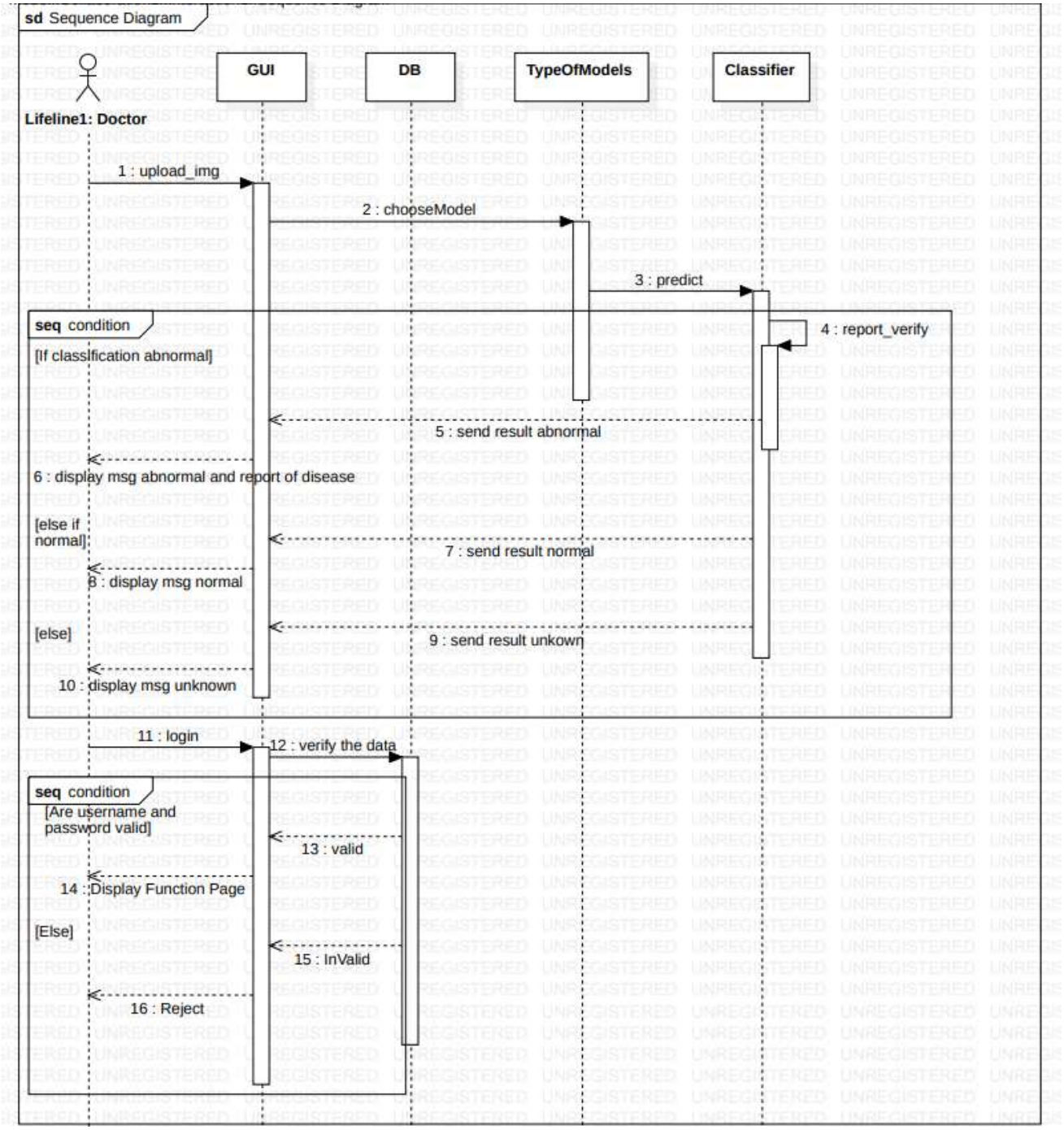


Fig 4.3 Sequence Diagram

Project ERD

in ERD we can describe relationship between databases .user database can connect to images database by id (primary key).

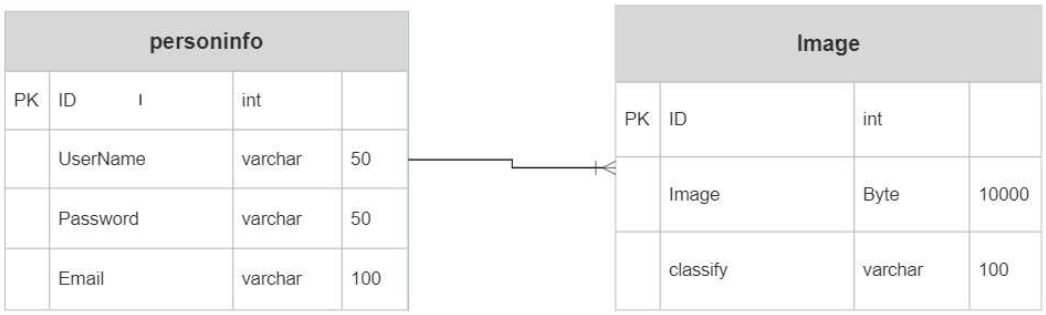


Fig 4.4: ERD Diagram

System GUI Design

Login

User must have an account on our website to be able to interact with it and if have not ,then he/she must sign up firstly to store his/her information in the database then the account is done then the user can interact after that with the website through login (sign in).

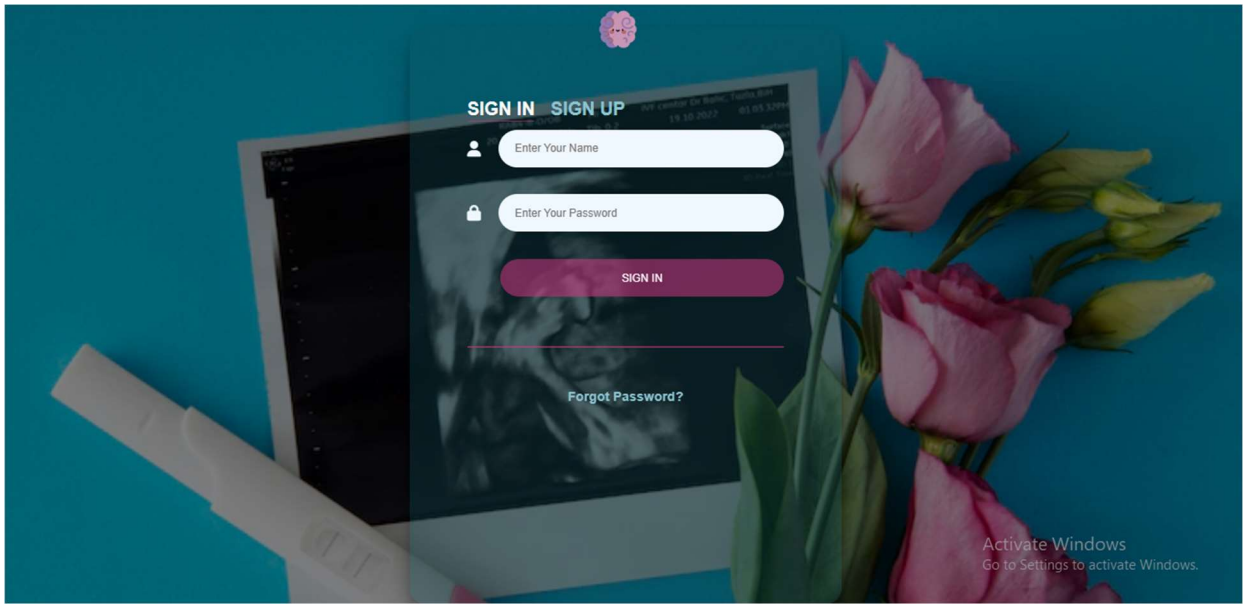


Figure 4.5 (System Login GUI)

Signup page

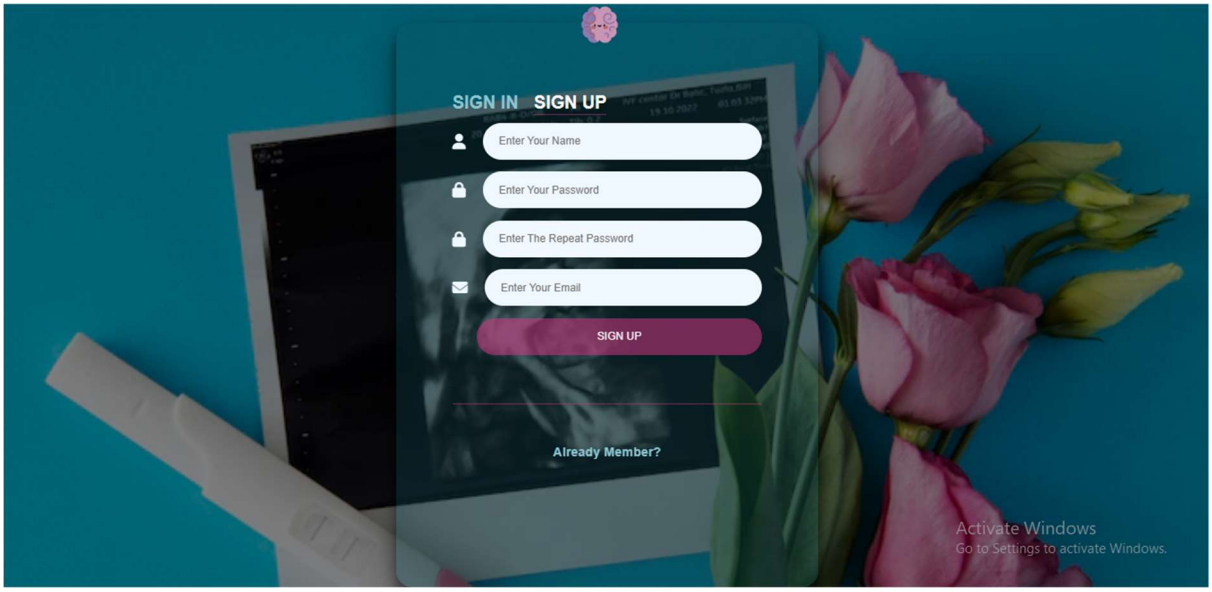


Figure 4.6 (System Signup GUI)

Home Page

After login and sign up, the user enter home page that has buttons as: FUNCTIONS will go to Classification Page)function page(, ABOUT will go to section About which is fast idea of our project, LOGIN will go to login page, SIGN UP will go to sign up page , EXPLORE will go to section Explore.

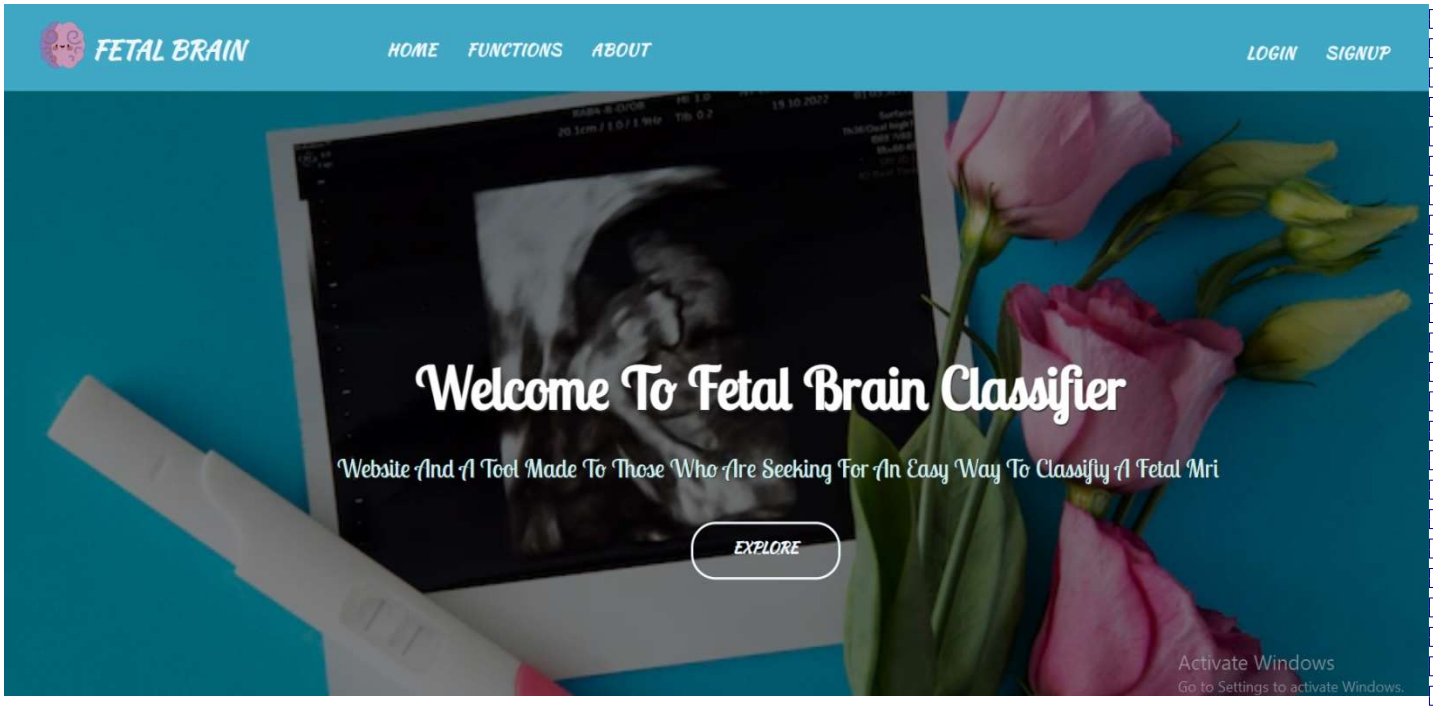


Figure 4.7 (System Home GUI)

Home page: Explore Section

Explore Section which includes 3 buttons: CONTACT will go to contact section, CLASSIFY YOUR MRI will go to Classification Page (functions page), and HELP will go to section Help

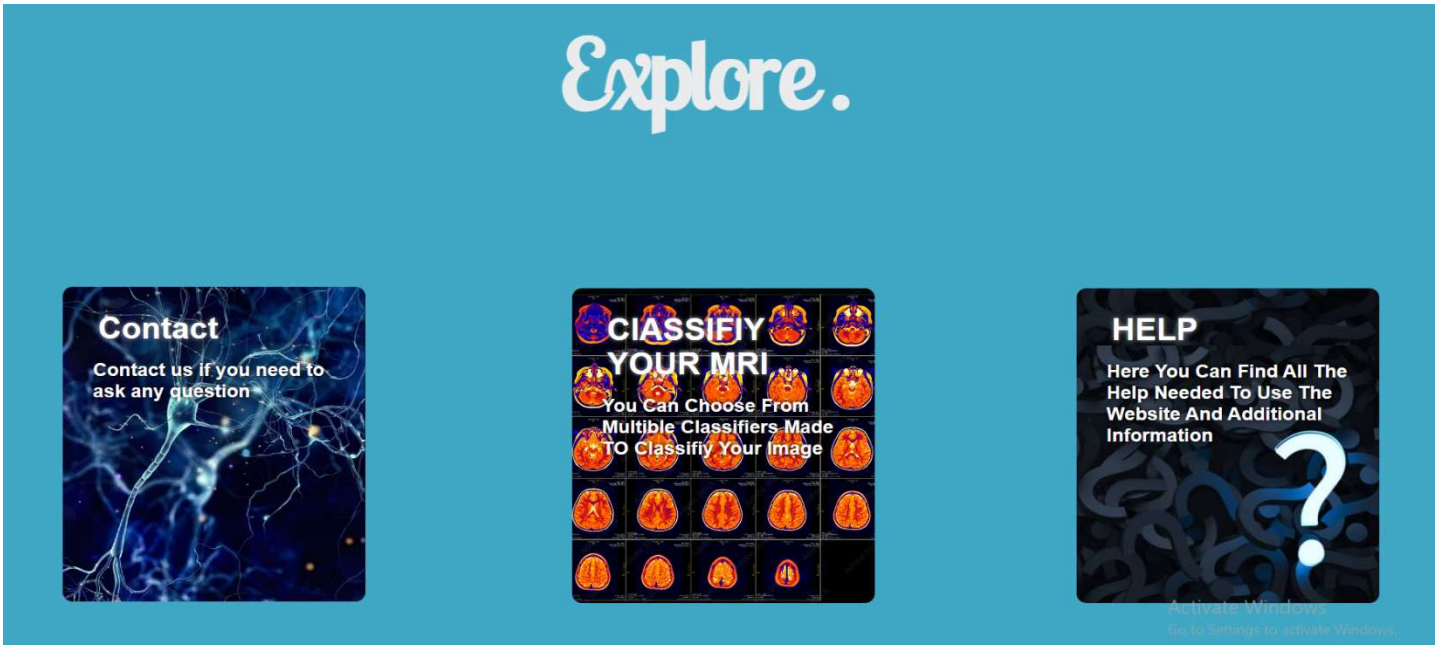


Figure 4.8 (System Home GUI (Explore))

Home Page: About Section

section About which is fast idea of our project.

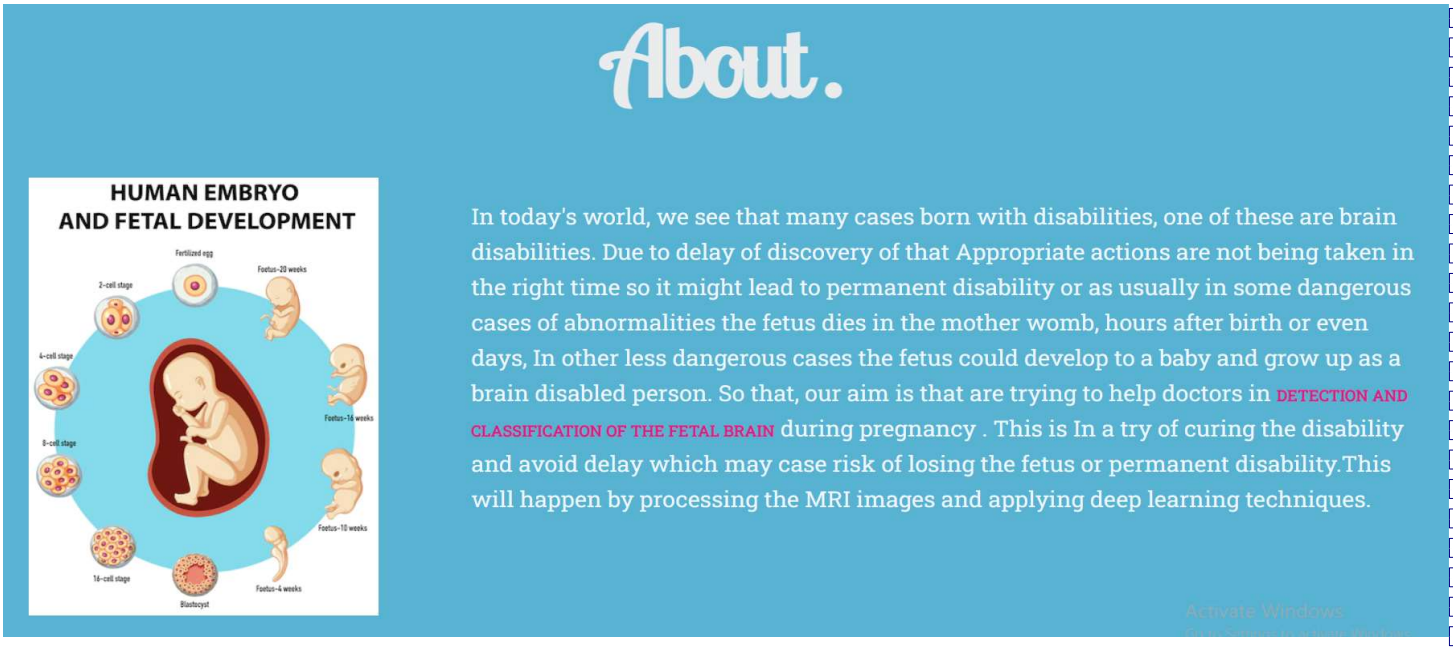


Figure 4.9 (System Home GUI (About))

Home Page: Help Section:

section Help which explain to user how to use the website.

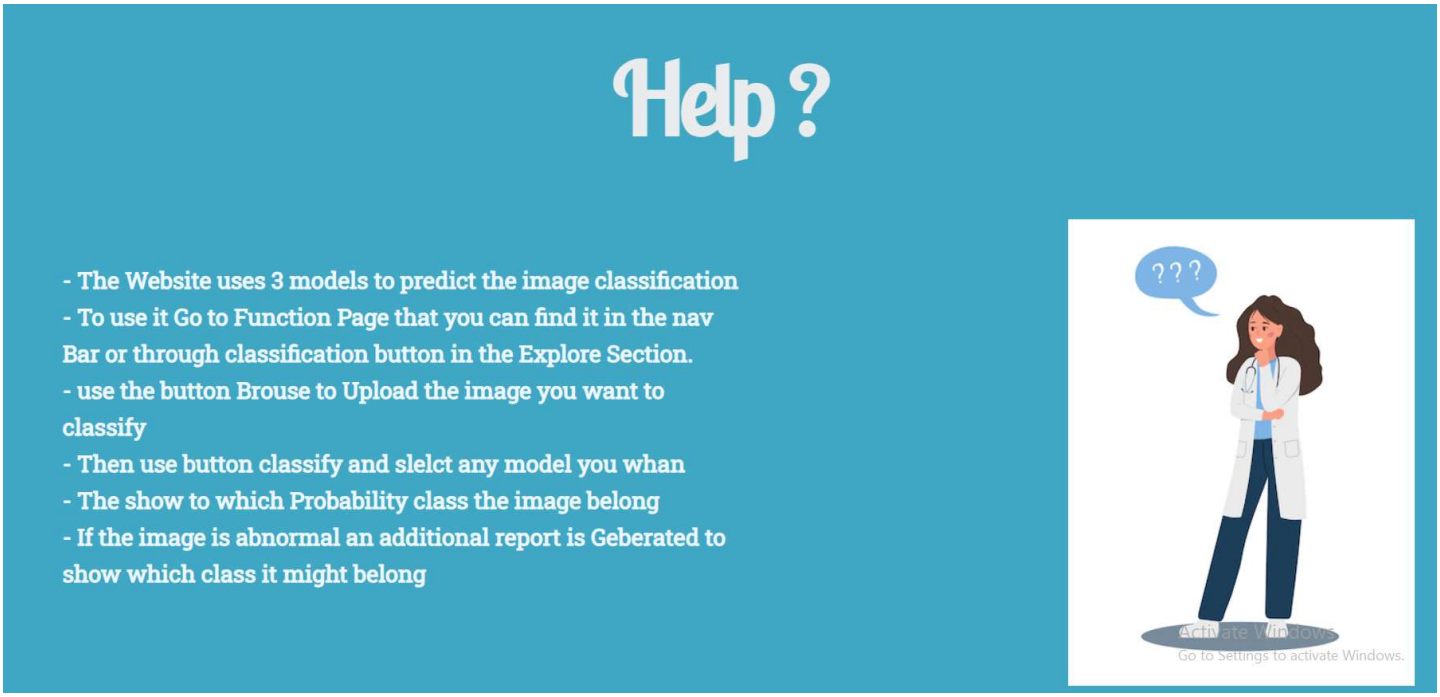


Figure 4.10 (System Home GUI (Help))

Contact section

Contact Section which contains the contact email if the user faces any problem and needs (solution) help.

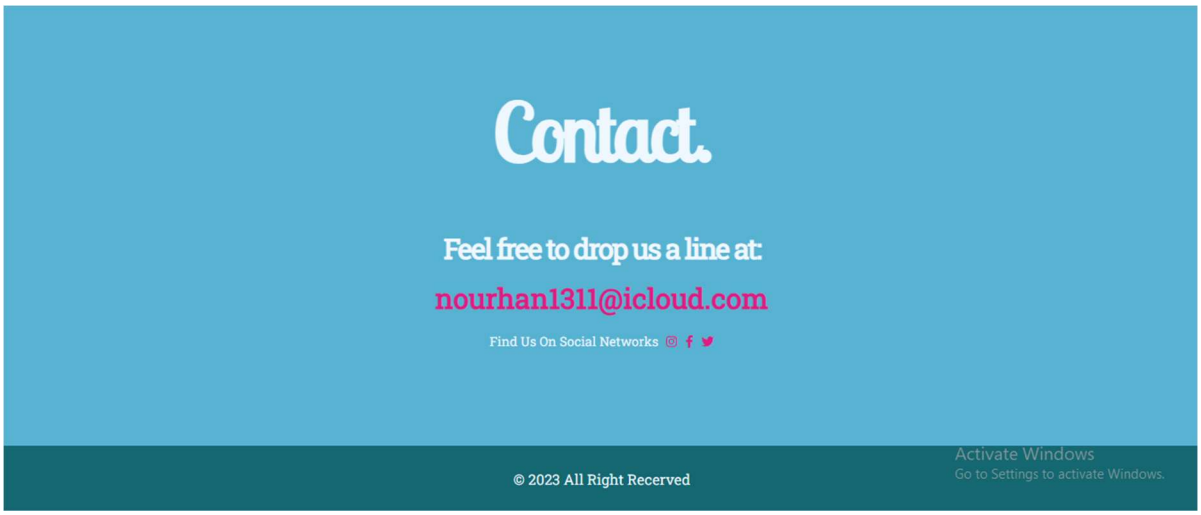


Figure 4.11 (System Home GUI (Content))

Classification Page

This is the button where you can choose which model you want to use for classification.

Fig 4.12 Button of Models

This is an example of the classification process.

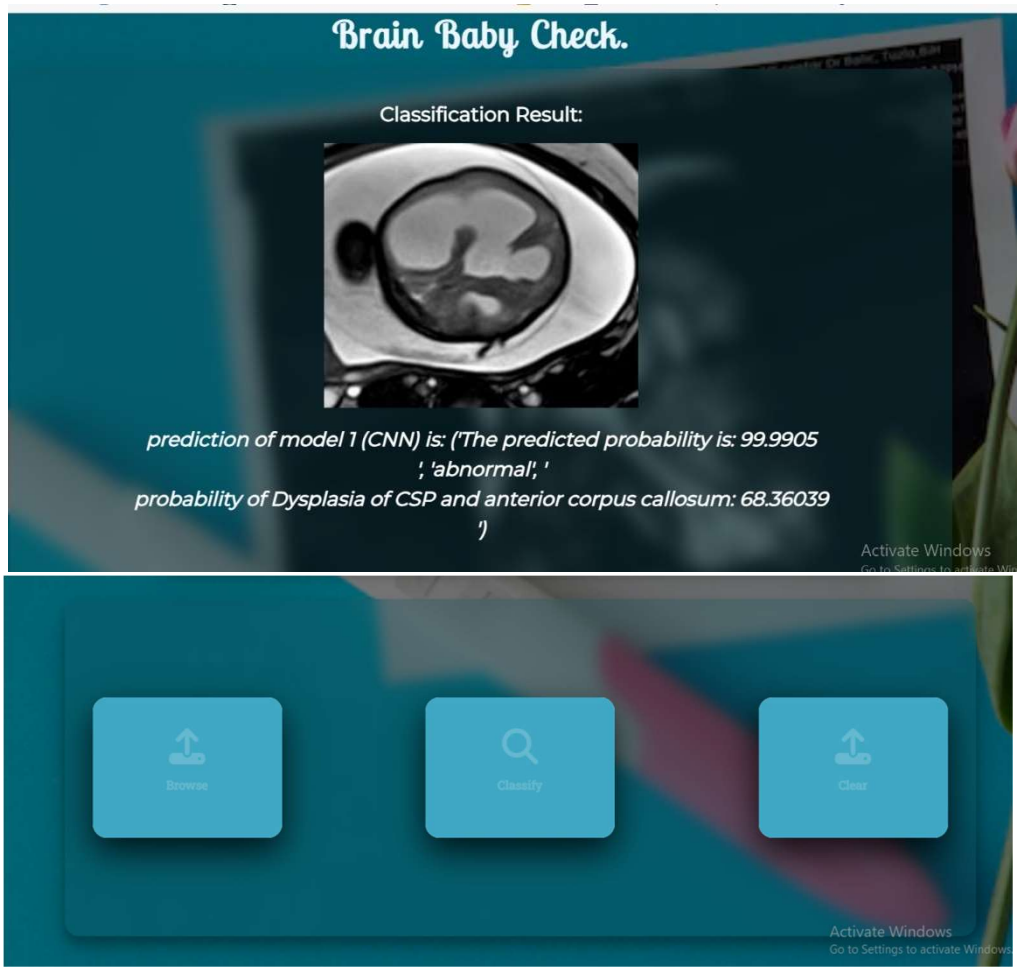


Figure 4.13 (System Function GUI (Classification Page and a test case))

Chapter 5: Implementation and Testing

Deep Learning

Experiment 1

We only applied the CNN model on the data that we downloaded from the atlas

The data was gray scaled and resized to 300 * 300

Normal = 258, abnormal = 462

Description of the model

the model consists of 3 Conv2D layers, 3 MaxPooling2D layers, 1 Flatten layer, and 2 Dense layers.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(xtrain.shape[1:])))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()


model.compile(optimizer='adam', loss=tf.keras.losses.binary_crossentropy,
metrics=['accuracy'], run_eagerly=True)
history = model.fit(xtrain, ytrain, epochs=10, batch_size=32,
validation_data=(xtest, ytest))
```

```
Accuracy: 0.3402777777777778
Precision: 0.116
recall: 0.340
F-Measure: 0.254
```

Experiment 2

We add some piece of code to adjust the results

```
predictions1 = model.predict(xtest)
for i in range(len(predictions1)):
    if predictions1[i] >= 0.5:
        predictions1[i] = 1
    else:
        predictions1[i] = 0
```

the results:

Accuracy: 0.6805555555555556
Precision: 0.675
recall: 0.681
F-Measure: 0.637

Experiment 3

We tried 3 values of gamma for intensity adjustment but it was a fail.
intensity is a parameter that control the brightness of the images.

```
# for gamma in [0.5, 1.2, 2.2]:        # Trying 3 gamma values.
for gamma in [2.2]:       # Trying one at a time
# #     # Apply gamma correction.
    gamma_corrected = np.array(255 * (img / 255) ** gamma, dtype='uint8')
    x.append(gamma_corrected)
    y.append(classes[cls])
```

The result for the 3 values:
Accuracy: 0.6597222222222222
Precision: 0.435
recall: 0.660
F-Measure: 0.397

Experiment 4

We applied Bm3d filter to reduce noise and improve the visual quality of the image.

```
BM3D_denoised_image = bm3d.bm3d(img, sigma_psd=0.2,
stage_arg=bm3d.BM3DStages.ALL_STAGES)
```

Accuracy: 0.6388888888888888
Precision: 0.632
recall: 0.639
F-Measure: 0.589

```
BM3D_denoised_image = bm3d.bm3d(img, sigma_psd=0.1,
stage_arg=bm3d.BM3DStages.ALL_STAGES)
```

Accuracy: 0.7361111111111112
Precision: 0.726
recall: 0.736
F-Measure: 0.676

Experiment 5

We tried the unsharp mask filter that seem good for the data as it's sharpened the edges making the details clearer.

Result unsharp 1
Accuracy: 0.6875
Precision: 0.670
recall: 0.688
F-Measure: 0.619

Result unsharp 2
Accuracy: 0.631944
Precision: 0.641
recall: 0.632
F-Measure: 0.599

Result unsharp 3
Accuracy:0.71527
Precision: 0.702
recall: 0.715
F-Measure: 0.653

Later we only used the parameters of the unsharp 3

Experiment 6

Result unsharp 1 + BM3D
Accuracy: 0.645833
Precision: 0.610
recall: 0.646
F-Measure: 0.540

Result unsharp 2 + BM3D
Accuracy: 0.68055555
Precision: 0.675
recall: 0.681
F-Measure: 0.637

Result unsharp 3 + BM3D
Accuracy: 0.6597222 22
Precision: 0.658
recall: 0.660
F-Measure: 0.619

Experiment 7

Only augmentation on all the data + resize 200*200

train accuracy = [0.6922383308410645, 0.5213457942008972]
28/28 [==============================] - 6s 207ms/step - loss: 0.6918 - accuracy: 0.5266
test accuracy = [0.691818118095398, 0.5265536904335022]
28/28 [==============================] - 6s 204ms/step
Accuracy: 0.5265536723163842
Precision: 0.277
recall: 0.527
F-Measure: 0.345

Augmentation on all the data + unsharp 3

Accuracy: 0.6655367231638418
Precision: 0.681
recall: 0.666
F-Measure: 0.664

Experiment 8, 9

We figured out the data itself must be having something wrong so we decided to delete some of it. We deleted the really noisy images and the images that had the fetal appears in far positions were. We only leaved the images that were most of it were the brain only appears and also most accurate with the least amount of noise. so the

normal became from 258 -> 123

And the abnormal 462 -> 204

We run the model with unsharp 3

Epochs = 20 and batch size = 32

Accuracy: 0.7424242424242424
Precision: 0.744
recall: 0.742
F-Measure: 0.719

Validation accuracy was going up and down in each epoch and definitely there was overfitting so we figured that augmentation may solve that.

But all the loss was under 1
We run again with 30 epoch and 40 epochs
30

Accuracy: 0.7121212121212122
Precision: 0.731
recall: 0.712
F-Measure: 0.664
40
Accuracy: 0.7575757575757576
Precision: 0.758
recall: 0.758
F-Measure: 0.749
And all the loss was under 1

We also did augmentation on training data only with each image is repeated 14 times. We selected a random portion from the data with test size = 0.2

Results:

The loss became more then 1 in just the fourth epoch.

Accuracy: 0.7126436781609196
Precision: 0.759
recall: 0.713
F-Measure: 0.684

The images were very few, let's just try to add more from the old data and try the same process.

Experiment 10

Normal = 163

Abnormal = 299

In 30 epochs with only unsharp mask + test size = 0.2

train accuracy = [0.00035134857171215117, 1.0]

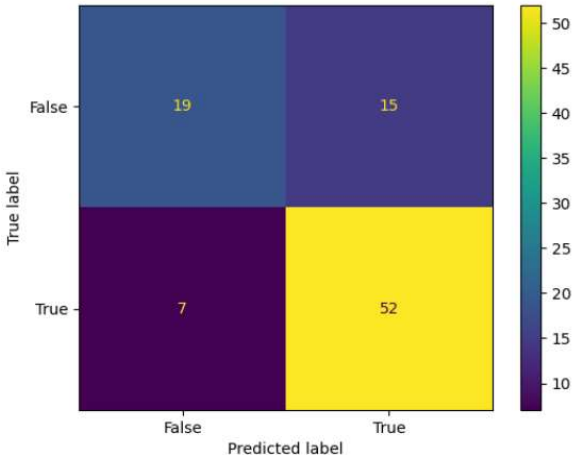


Fig 5.1 Accuracy Matrix for exp. 10

3/3 [==============================] - 0s 155ms/step - loss: 1.5315 - accuracy: 0.7634
test accuracy = [1.5314661264419556, 0.7634408473968506]
3/3 [==============================] - 0s 144ms/step
Accuracy: 0.7634408602150538
Precision: 0.760
recall: 0.763
F-Measure: 0.729

Test size 0.2 is definitely better then test size = 0.3

train accuracy = [0.00029840448405593634, 1.0]
5/5 [==============================] - 1s 133ms/step - loss: 1.6335 - accuracy: 0.7050
test accuracy = [1.6335450410842896, 0.7050359845161438]
5/5 [==============================] - 1s 127ms/step
Accuracy: 0.7050359712230215
Precision: 0.698
recall: 0.705
F-Measure: 0.665

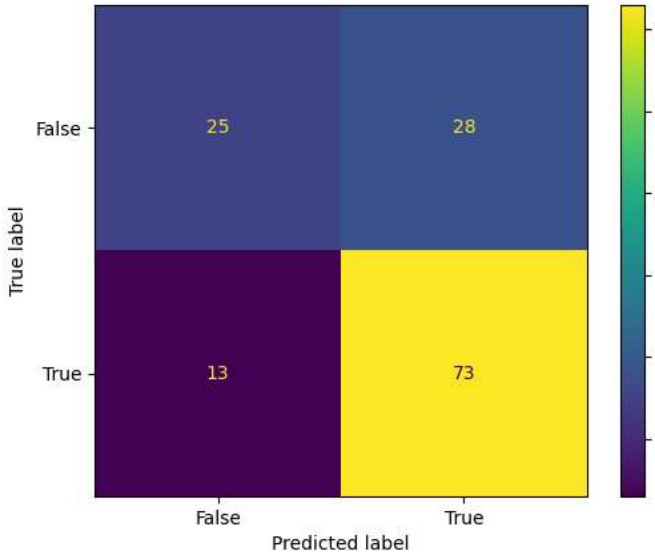


Fig 5.1.2 Accuracy Matrix for Exp 10

It's obvious from the past accuracy matrix that that data is biased and there is something wrong with the first class as it's half the second one and the model doesn't seem to have enough data to train on and the data itself is divergent. Meaning there isn't much similarity between the images in the same class and due to lack of accuracy; this all lead to the matrix that we see in the last graph.

Augmentation is mainly to reduce overfitting and increase the amount of data to adjust the biased classes to make them close to each other.

We did multiple augmentations on this data each has a different thing:

- Augmentation on whole data.

```
    datagen = ImageDataGenerator(
    rescale=1. / 255,
    rotation_range=45,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
)
```

batch size for normal = 7, abnormal = 4

note: batch size is the number that one image in the old class will be multiplied in the new class. So, each image in the normal class will be increased 7 times in different positions corresponding to the ImageDataGenerator.

Experiment 11

Normal = 1111

Abnormal = 1193

Test size = 0.3, batch = 32, epochs = 30

train accuracy = [0.004818768240511417, 0.9944168925285339]
22/22 [==============================] - 4s 186ms/step - loss: 2.9303 - accuracy: 0.6402
test accuracy = [2.9303221702575684, 0.6401734352111816]
22/22 [==============================] - 4s 179ms/step
Accuracy: 0.6401734104046243
Precision: 0.641
recall: 0.640
F-Measure: 0.640

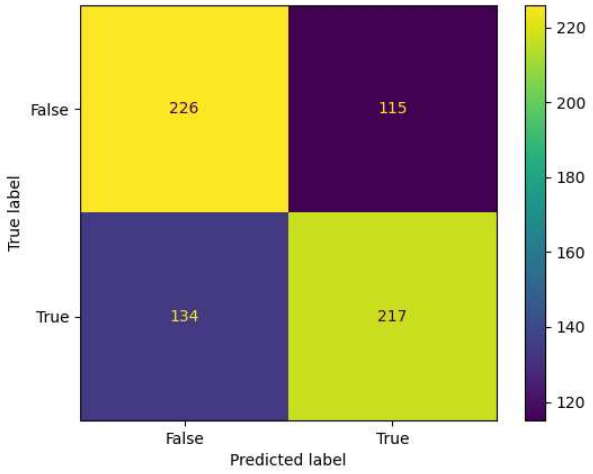


Fig 5.2 Accuracy Matrix for Exp. 11

Test size = 0.2

train accuracy = [8.01835922175087e-05, 1.0]
15/15 [==============================] - 2s 162ms/step - loss: 2.8629 - accuracy: 0.6377
test accuracy = [2.862898349761963, 0.6377440094947815]
15/15 [==============================] - 2s 159ms/step
Accuracy: 0.6377440347071583
Precision: 0.639
recall: 0.638
F-Measure: 0.637

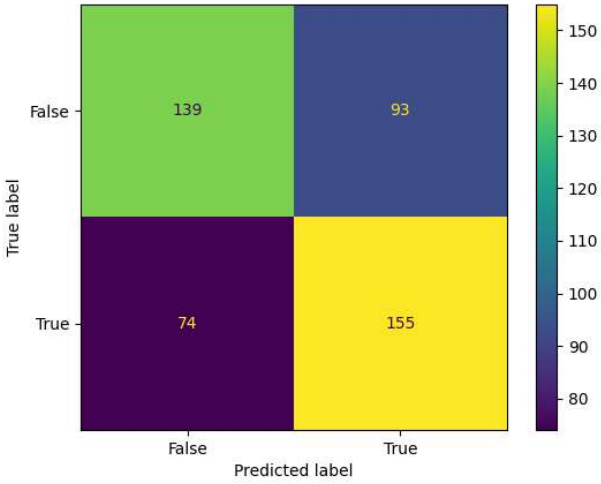


Fig 5.2.2 Accuracy Matrix for Exp. 11

We tried different CNN models on the same data to choose the best from them

train accuracy = [0.05281385779380798, 1.0]
15/15 [==============================] - 2s 127ms/step - loss: 0.8799 - accuracy: 0.7592
test accuracy = [0.8798708915710449, 0.7592191100120544]
15/15 [==============================] - 2s 128ms/step
Accuracy: 0.7592190889370932
Precision: 0.759
recall: 0.759
F-Measure: 0.759

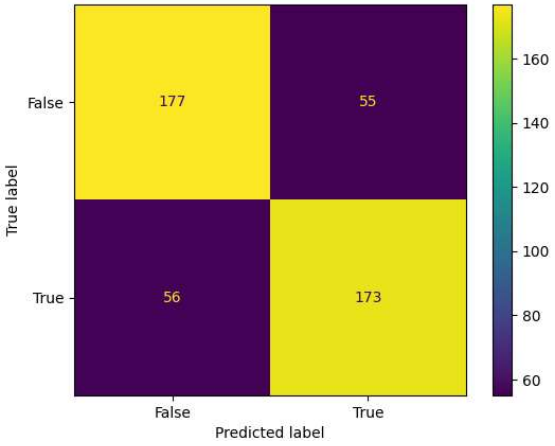


Fig 5.3 Accuracy Matrix for Exp. 12 CNN

So, by only model tuning accuracy raised to more than 10 %

Testing another model

train accuracy = [0.07199587672948837, 1.0]
15/15 [==============================] - 3s 183ms/step - loss: 0.9179 - accuracy: 0.7570
test accuracy = [0.9178623557090759, 0.7570499181747437]
15/15 [==============================] - 2s 162ms/step
Accuracy: 0.7570498915401301
Precision: 0.757
recall: 0.757
F-Measure: 0.757

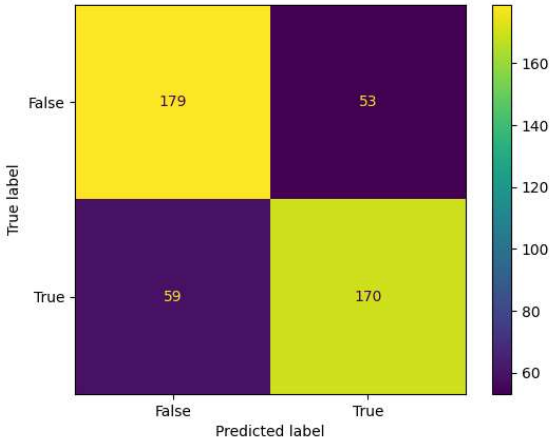


Fig 5.3.2 Accuracy Matrix for Exp. 12 CNN2

From that we will work with only last 2 CNN models

- Augmentation on only training

It has been done with the same parameters of the last augmentation on the same data but after splitting train = 70 % and test = 30%

Table 5.1 accuracy table for Exp. 12

First Model	Second Model
train accuracy = [0.04795157164335251, 1.0] 5/5 [==========] - 1s 115ms/step - loss: 1.3723 - accuracy: 0.6835 test accuracy = [1.3723320960998535, 0.6834532618522644] 5/5 [==========] - 1s 106ms/step Accuracy: 0.6834532374100719 Precision: 0.676 recall: 0.683 F-Measure: 0.643	train accuracy = [0.00016261650307569653, 1.0] 5/5 [==========] - 1s 118ms/step - loss: 2.3121 - accuracy: 0.6978 test accuracy = [2.3121178150177, 0.6978417038917542] 5/5 [==========] - 1s 114ms/step Accuracy: 0.697841726618705 Precision: 0.701 recall: 0.698 F-Measure: 0.672

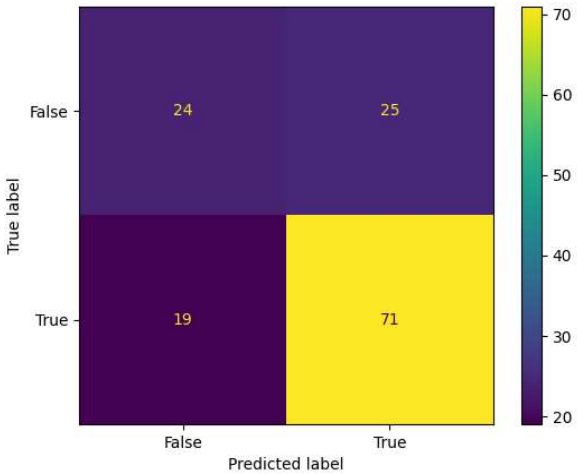
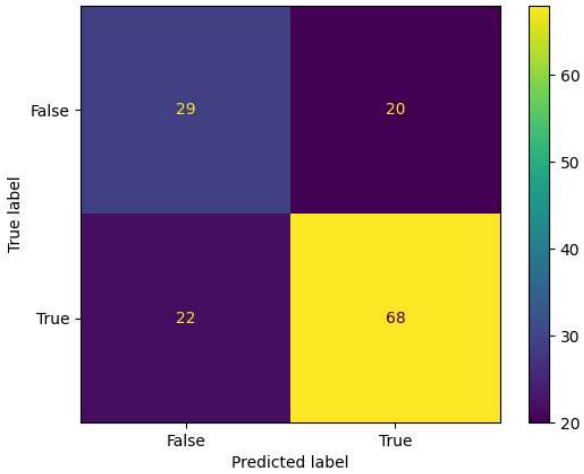

	

Fig 5.3.3 Accuracy Matrix for Exp. 12 CNN1	Fig 5.3.4 Accuracy Matrix for Exp. 12 CNN2

augmentation with the parameters used in the paper on whole data

normal batch size = 7, abnormal = 4

parameters: rescale = 1. / 255, rotation_range = 45, width_shift_range = 0.3, height_shift_range = 0.3, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True, vertical_flip = True

Table 5.1.2 accuracy table for Exp. 12

First Model	Second Model
test size = 0.2 train accuracy = [0.08908713608980179, 1.0] 15/15 [==========] - 2s 132ms/step - loss: 1.2768 - accuracy: 0.6594	test size = 0.2 train accuracy = [5.279739707475528e-05, 1.0] 15/15 [==========] - 3s 194ms/step - loss: 1.9466 - accuracy: 0.6508

test accuracy = [1.2768038511276245, 0.6594359874725342] 15/15 [==============================] - 2s 128ms/step Accuracy: 0.6594360086767896 Precision: 0.660 recall: 0.659 F-Measure: 0.659	test accuracy = [1.946559190750122, 0.650759220123291] 15/15 [==============================] - 3s 180ms/step Accuracy: 0.6507592190889371 Precision: 0.651 recall: 0.651 F-Measure: 0.651

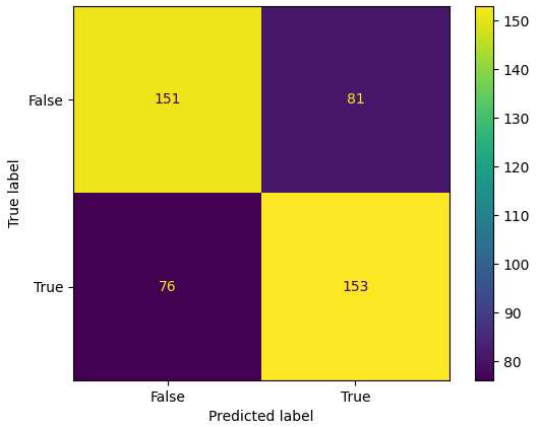
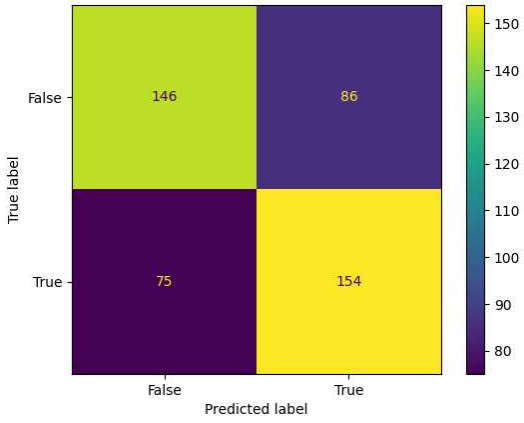

	

Fig 5.3.5 Accuracy Matrix for Exp. 12 CNN1	Accuracy Matrix for Exp. 12 CNN2

Fig 5.3.6

last augmentation is an augmentation on the whole data with a parameter that was recommended for the type of MRI

rescale=1. / 255, rotation_range=20, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, brightness_range= [0.9, 1.1]

normal batch size = 7, 30 epochs
abnormal = 4
test size = 0.2

Table 5.1.3 accuracy table for Exp. 12

First Model	Second Model
train accuracy = [0.06581839919090271, 1.0] 15/15 [==============================] - 3s 206ms/step - loss: 0.9062 - accuracy: 0.7419 test accuracy = [0.9062339067459106, 0.7418655157089233] 15/15 [==============================] - 3s 196ms/step	train accuracy = [5.149940261617303e-05, 1.0] 15/15 [==============================] - 3s 214ms/step - loss: 1.5280 - accuracy: 0.7072 test accuracy = [1.5279864072799683, 0.7071583271026611] 15/15 [==============================] - 3s 209ms/step

Accuracy: 0.7418655097613883
Precision: 0.742
recall: 0.742
F-Measure: 0.742

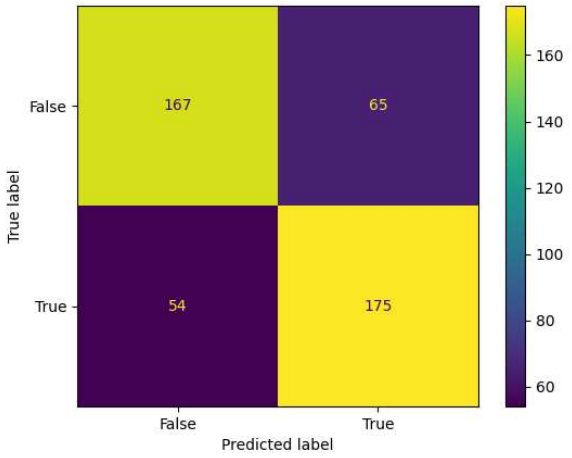


Fig 5.3.7 Accuracy Matrix for Exp. 12 CNN1

Accuracy: 0.7071583514099783
Precision: 0.707
recall: 0.707
F-Measure: 0.707

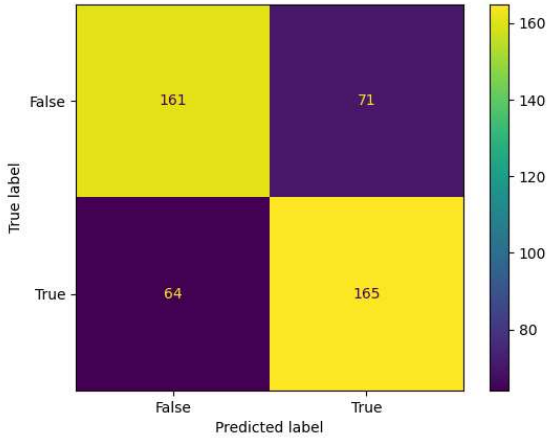


Fig 5.3.8 Accuracy Matrix for Exp. 12 CNN2

Experiment 13 Segmentation

We tried to applicate a segmentation technique that appear in one of the papers but there were too many steps and not all of them were available on the internet,

The first step was Adaptive Thresholding and on its own it has several techniques, we tried some of them and tested their effect on the accuracy

test 1 thresh-BINARY
Accuracy: 0.7502824858757062
Precision: 0.750
recall: 0.750
F-Measure: 0.750

test 1.1 thresh-BINARY + unsharp mask
Accuracy: 0.7796610169491526
Precision: 0.781
recall: 0.780
F-Measure: 0.779
##
test 2 ADAPTIVE_THRESH_GAUSSIAN_C
Accuracy: 0.7570621468926554
Precision: 0.760
recall: 0.757

F-Measure: 0.757

test 3 THRESH_TOZERO
Accuracy: 0.784180790960452
Precision: 0.786
recall: 0.784
F-Measure: 0.783
##
test 4 THRESH_TOZERO_INV
Accuracy: 0.784180790960452
Precision: 0.784
recall: 0.784
F-Measure: 0.784

test 4.1 THRESH_TOZERO_INV + unsharp mask
Accuracy: 0.7966101694915254
Precision: 0.797
recall: 0.797
F-Measure: 0.796

But after applying morphological operations which is the second step containing many steps with many parameters, we could not able to complete them.

We even tried some code for segmentation and it did work on some images but we discovered something important; the images was different angles on the brain, and not all of them are foxed on the brain only some of them showed the fetal body, they were too different that one algorithm couldn't handle them.

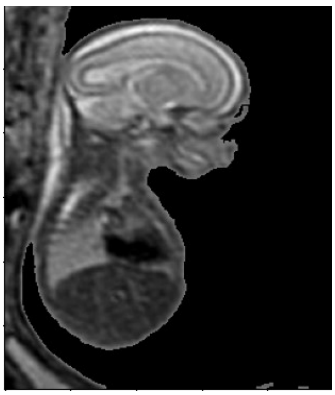


Fig.5.4.1


Fig. 5.4.2


Fig.5.4.3

Fig. 5.4.1, Fig. 5.4.2, Fig. 5.4.3 show result of segmentation

Here you can see it's accepted in some images but other are not so we gave up on segmentation step.

After all the past preprocessing and Experiments, there were no doubt that we need to increase the data if we want to improve the accuracy.

From the above experiments we notice that the normal class is responsible for the incompatibility of the accuracy for the accuracy matrix.

One of the most important draw backs in this project as a total is the difficulty for reaching to a data. And this drawback appears in all the papers that worked before on the same topic. Not to mention that nearly all the brain data available on the internet are 3D image data. Anyway, lucky us we came across to nearly exactly what we needed to solve this.

Stanford Lucile Packard Children's Hospital provided a large diverse single-center dataset of 741 developmentally normal fetal brain MRI.

Data Description: folders 1 to 741, subdirectories for each fetal brain MRI in .jpg format, consisting of a sequence for each of the 3 planes (axial, sagittal, coronal).

What we did is start to collect some images from every folder from the 3 planes and adding it to the normal class to increase it. We just deleted a couple of images from the old class to become 149 and we increased that from the normal data to reach to 336 images. We tasted all that with the abnormal 299.

Experiment 14

Test-size = 0.2, epochs = 30, batch-size = 32

Table 5.2 accuracy table for exp.14

Model 1 CNN	Model 2 CNN
train accuracy = [0.04304877296090126, 1.0] 4/4 [==========] - 1s 210ms/step - loss: 0.4365 - accuracy: 0.8661 test accuracy = [0.4364560842514038, 0.8661417365074158] 4/4 [==========] - 1s 207ms/step Accuracy: 0.8661417322834646 Precision: 0.866 recall: 0.866 F-Measure: 0.864	train accuracy = [0.000751218874938786, 1.0] 4/4 [==========] - 1s 257ms/step - loss: 0.5557 - accuracy: 0.8583 test accuracy = [0.5556982755661011, 0.8582677245140076] 4/4 [==========] - 1s 234ms/step Accuracy: 0.8582677165354331 Precision: 0.858 recall: 0.858 F-Measure: 0.856

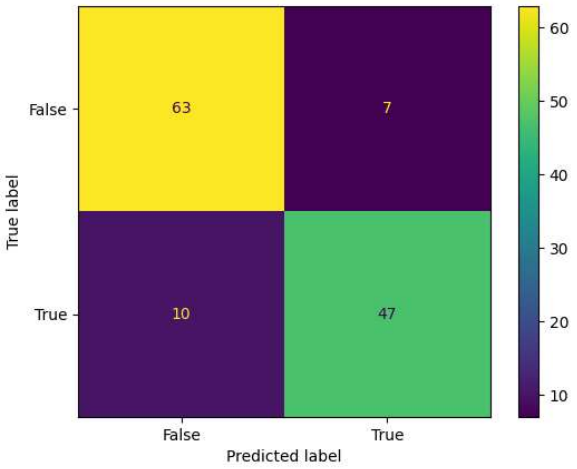


Fig.5.5.1 Accuracy Matrix of exp. 14 CNN1

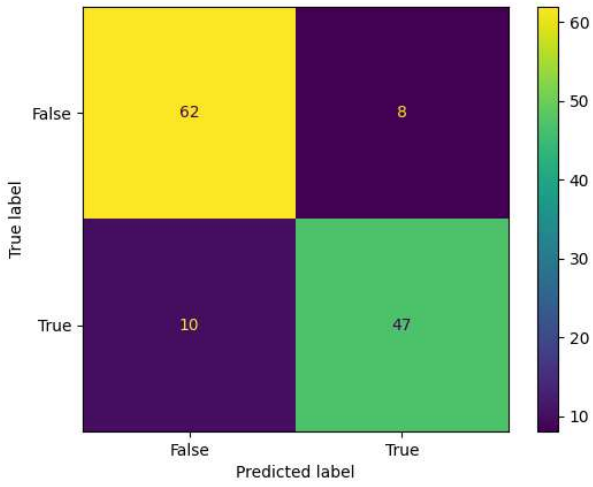


Fig.5.5.2 Accuracy Matrix of exp. 14 CNN2

Model 3 Resnet50 (Pretrained)

train accuracy = [1.2434186935424805, 0.5314960479736328]
4/4 [==============================] - 6s 2s/step - loss: 1.1842 - accuracy: 0.5748
test accuracy = [1.1841778755187988, 0.5748031735420227]
4/4 [==============================] - 7s 2s/step
Accuracy: 0.5748031496062992
Precision: 0.760
recall: 0.575
F-Measure: 0.411

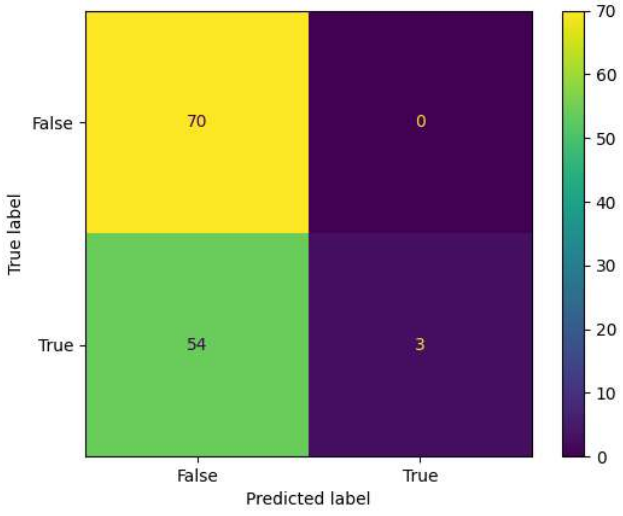


Fig 5.5.3 Accuracy Matrix of exp. 14 Resnet50

Model 4 Alex net

train accuracy = [0.014606599695980549, 0.998031497001648]
4/4 [==============================] - 1s 252ms/step - loss: 0.4947 - accuracy: 0.8583
test accuracy = [0.4946829080581665, 0.8582677245140076]
Accuracy: 0.8582677165354331
Precision: 0.861
recall: 0.858
F-Measure: 0.858

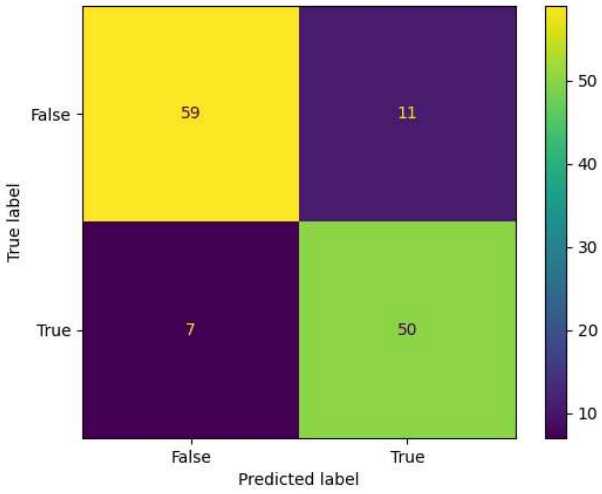


Fig 5.5.4 Accuracy Matrix of exp. 14 Alexnet

Note: it's obvious from the last experiment which models are we sticking to.

In addition, we tried efficientnetB7but it took 5 minutes to complete 1 epoch, the accuracy was low and the loss was big, so we passed it.

We did augmentation on the whole data with Batch-size = 5 for both classes

Normal = 1664

Abnormal = 1491

train accuracy = [1.5759258531033993e-05, 1.0]
20/20 [==============================] - 4s 221ms/step - loss: 0.7927 - accuracy: 0.8463
test accuracy = [0.7926719188690186, 0.8462757468223572]
20/20 [==============================] - 4s 212ms/step
Accuracy: 0.8462757527733756
Precision: 0.847
recall: 0.846
F-Measure: 0.846

Seems like augmentation in 30 epochs. Doesn't do any improvement, even after 40 epochs it 0.85 it.

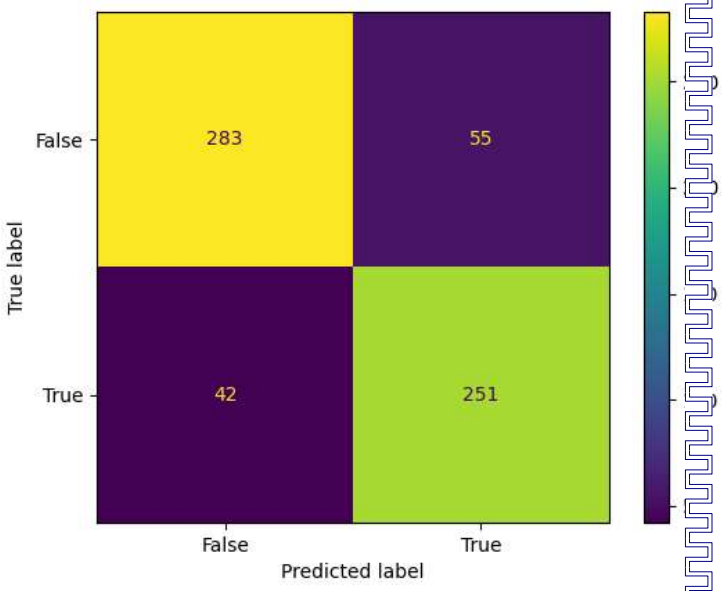


Fig 5.5.5 Accuracy Matrix of exp. 14 CNN2

We also did augmentation on the train data only, one with test-size = 0.2, the other with test-size = 0.3. but here we something we didn't do on the other augmentations on the train data. We used to choose manually random portion of the data -by hand- in to add it to test folder and do the augmentation on the train folder only.

we made a code that choose by pc a random portion of the data and divide it then move it to test class.

> Augmentation parameters: rescale=1. / 255, rotation_range=20, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.1, zoom_range=0.1, horizontal_flip=True, vertical_flip=True, brightness_range= [0.9, 1.1]
> Batch-size for every image = 7

Note: Train was in total = 3259 with test = 126

These are the results in 30 epochs and with augmentation on train test-size = 0.2

Table 5.2.2 accuracy table for exp.14

Model 2 CNN	InceptionV3 (Pretrained)
train accuracy = [0.0007445571827702224, 1.0] 4/4 [==========] - 1s 221ms/step - loss: 1.4435 - accuracy: 0.8254 test accuracy = [1.4434905052185059, 0.8253968358039856] 4/4 [==========] - 1s 212ms/step Accuracy: 0.8253968253968254 Precision: 0.825 recall: 0.825 F-Measure: 0.824	train accuracy = [0.008941326290369034, 0.9993863105773926] 4/4 [==========] - 3s 757ms/step - loss: 0.7162 - accuracy: 0.8095 test accuracy = [0.7161605954170227, 0.8095238208770752] Accuracy: 0.8095238095238095 Precision: 0.813 recall: 0.810 F-Measure: 0.809

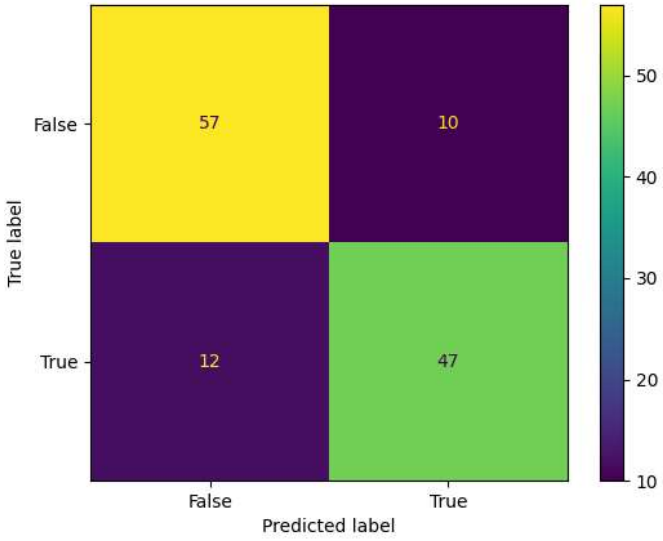


Fig 5.5.6 Accuracy Matrix of exp. 14 CNN2

Accuracy fluctuating in every epoch and the loss is really large.

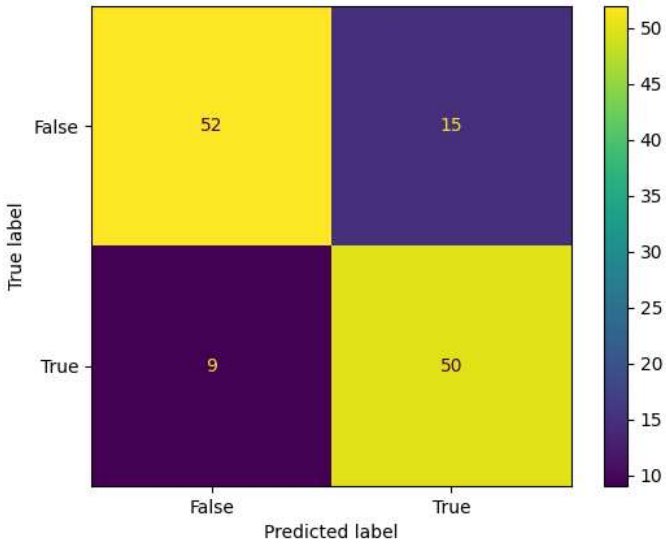


Fig 5.5.7 Accuracy Matrix of exp. 14 InceptionV3

Definitely less accuracy but the loss was always lower than 1

We used pretrained inception model put we tuned it to match our data and reduce overfitting.

Note: The results made sense since the train size is way bigger then test size. These are the results in 40 epochs and with augmentation on train test-size = 0.3 Batch-size for every image during augmentation = 5

With total train data = 1994, test = 189

train accuracy = [3.5334764106664807e-05, 1.0]
6/6 [==============================] - 2s 258ms/step - loss: 1.5128 - accuracy: 0.7566
test accuracy = [1.5128262042999268, 0.7566137313842773]
6/6 [==============================] - 1s 242ms/step
Accuracy: 0.7566137566137566
Precision: 0.758
recall: 0.757
F-Measure: 0.756
And total run took really long time

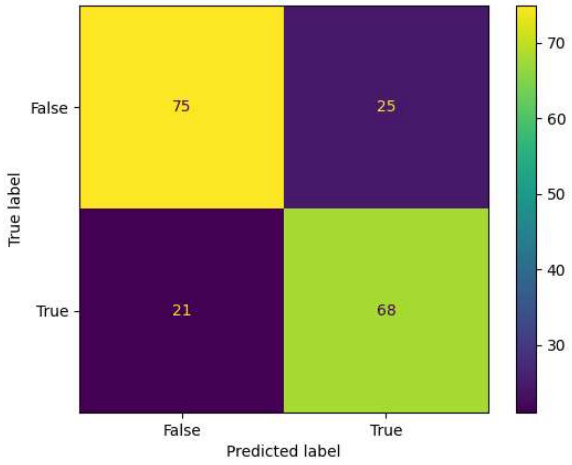


Fig 5.5.8 Accuracy Matrix of exp. 14 CNN2

There is one thing that got us worried about the past data. The thing is the normal data was not trimmed like the abnormal one. Meaning that a big part of the womb appears so we got worried that the distance and other unnecessary features might affect the classification.

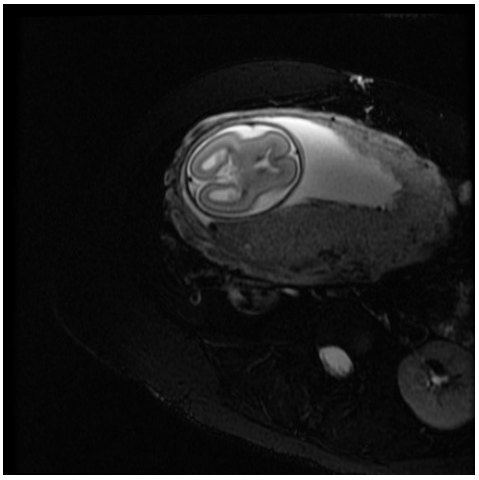


Fig 5.6

Fig 5.6.2

Fig 5.6, Fig.5.6.2 Normal Image Before Trimmed

Due to that we needed to trim the data ourselves making it look like the abnormal class the classifier is more accurate and we also add more data to the class. On the other hand we also re-add more images in the abnormal class, we trimmed some noisy images that a lot of the fetal body and womb appears on it and we flipped some images to make it match the images in the other class so the model is fair enough in classification.

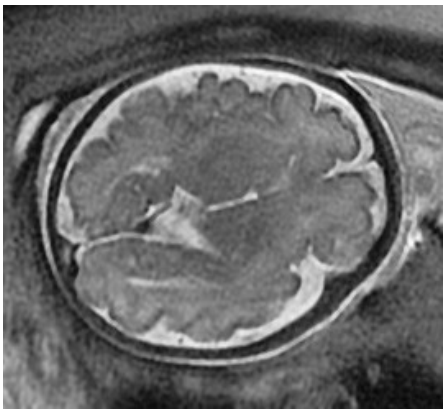


Fig.5.6.3 Normal Image After Trimmed

Experiment 15

Normal class became = 374

Abnormal = 353

The accuracy of the run was between 84 – 89 %

We used PCA for image compression and to fasten the run and adjust the loss because in some surprisingly and suddenly epochs the loss might go up and the accuracy goes down.

So, this is the run in 30 epochs with PCA, test-size = 0.2

train accuracy = [3.833045411738567e-05, 1.0]
5/5 [==============================] - 0s 19ms/step - loss: 0.5848 - accuracy: 0.8699
test accuracy = [0.5847597718238831, 0.8698630332946777]
5/5 [==============================] - 0s 11ms/step
Accuracy: 0.8698630136986302
Precision: 0.870
recall: 0.870
F-Measure: 0.870

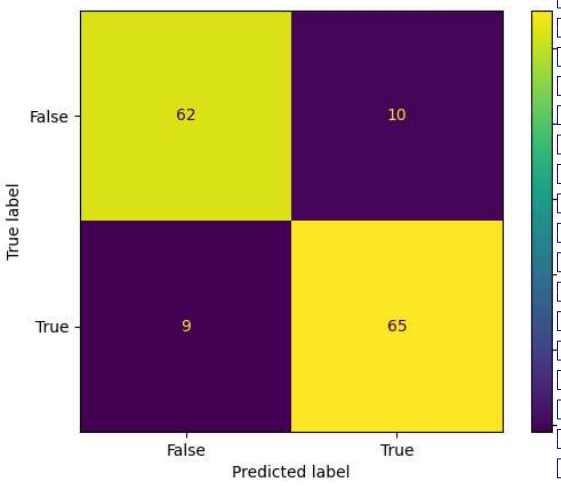


Fig 5.7 Accuracy Matrix of exp. 15 CNN2

Augmentation was done in also 2 ways, on the whole data and the train only.

The whole data with batch-size = 3, we just wanted to make each class reach for around 1000.

Both classes were 1057 so the total data = 2114

40 epochs

Table 5.3 accuracy table for exp.15

Model 1 CNN	Model 2 CNN
train accuracy = [0.0241712499409914, 1.0] 14/14 [==========] - 0s 32ms/step - loss: 0.6863 - accuracy: 0.8322 test accuracy = [0.6863283514976501, 0.8321512937545776] 14/14 [==========] - 0s 15ms/step Accuracy: 0.8321513002364066 Precision: 0.835 recall: 0.832 F-Measure: 0.832	train accuracy = [4.903240915155038e-05, 1.0] 14/14 [==========] - 0s 25ms/step - loss: 1.0507 - accuracy: 0.8369 test accuracy = [1.0506523847579956, 0.8368794322013855] 14/14 [==========] - 0s 18ms/step Accuracy: 0.8368794326241135 Precision: 0.837 recall: 0.837 F-Measure: 0.837

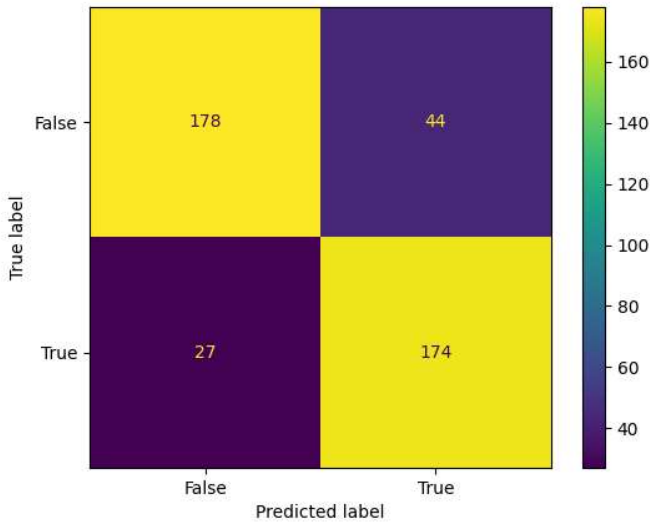


Fig 5.7.2 Accuracy Matrix of exp. 15 CNN1

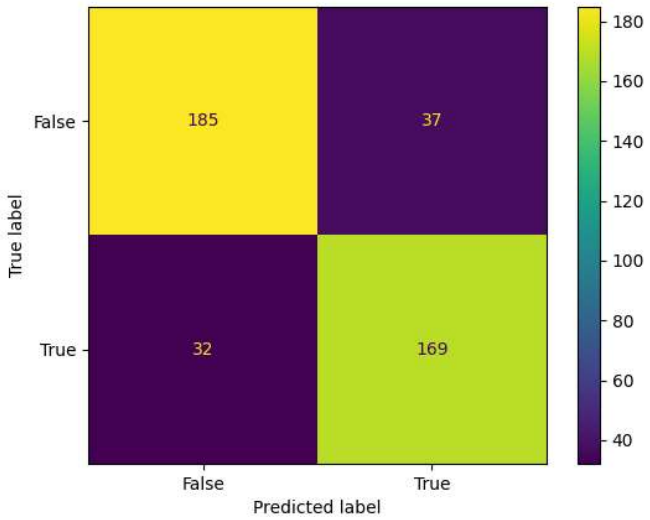


Fig 5.7.3 Accuracy Matrix of exp. 15 CNN2

Accuracy got less and loss got bigger.

Let's now try augmentation on only train data with test-size = 0.3, batch size in this augmentation was 20, so total train data = 10216, and test = 217

I used PCA in a try to reduce the run but it was a failure because the accuracy was really low.

Batch-size = 128, epochs = 40

train accuracy = [0.00017842056695371866, 1.0]
4/4 [==========] - 0s 31ms/step - loss: 9.4983 - accuracy: 0.4147
test accuracy = [9.498292922973633, 0.41474655270576477]
7/7 [==========] - 0s 19ms/step
Accuracy: 0.4147465437788018
Precision: 0.416

recall: 0.415
F-Measure: 0.415
Even with Batch-size = 64, epochs = 30 accuracy is still the same.
In conclusion, PCA is a good image compression technique to make the for-feature reduction, sometimes it fastens the run without effecting the accuracy and sometimes it lowers it. We focused on the accuracy itself it's important to keep it high even if the run will take a long time because Fetal brain abnormality is a complicated task to acknowledge and the abnormalities are different. It might reduce features important to detect wither there is abnormality or not. Or maybe the simplicity of the image after PCA didn't match the complexity of the models. Plus, PCA doesn't work with pretrained models. So, we declined it.

Note: We tried a combination of dwt and GLCM for feature extraction but using them needed experience and the result accuracy was really low.

When we were searching for other data to make the report for the type of abnormality, we found some abnormal data so we add it to the abnormal class and also add normal data to the normal class.

Experiment 16

This is the last version of the data set after we found some abnormalities on radiopaedia. And we also increased the normal so the 2 classes are close in numbers

Normal became = 557

Abnormal became = 548

Accuracy using model 2 CNN in 30 epochs

train accuracy = [3.1569266866426915e-05, 1.0]
7/7 [==============================] - 1s 163ms/step - loss: 0.4787 - accuracy: 0.9050
test accuracy = [0.47869786620140076, 0.9049773812294006]
7/7 [==============================] - 1s 145ms/step
Accuracy: 0.9049773755656109
Precision: 0.909
recall: 0.905
F-Measure: 0.905

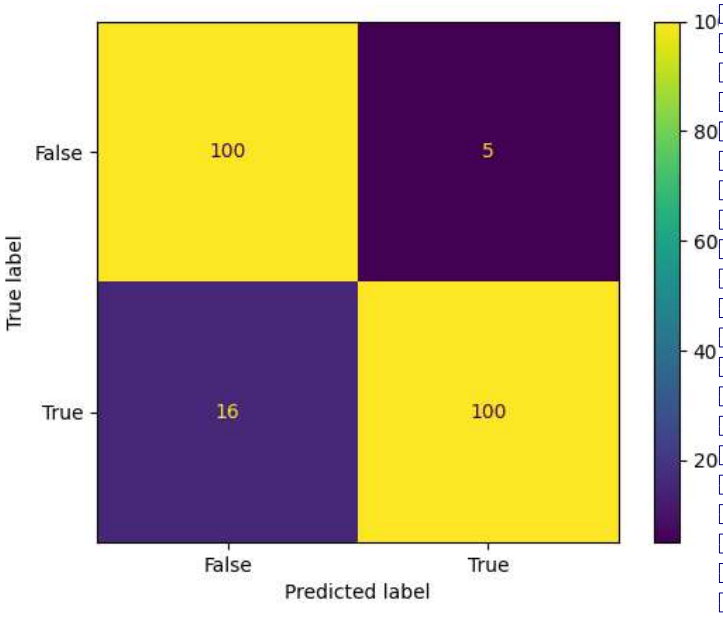


Fig 5.8 Accuracy Matrix of exp. 16 CNN2

We also add normalization to the code and resizes images to 150 * 150 and this is the new accuracy using CNN model 2

There were no fluctuations and the running in stable and the accuracy is decent. Here is the accuracy using different Models:

Table 5.4 accuracy table for exp.16

Model 1 CNN	Model 2 CNN
train accuracy = [0.02319718338549137, 1.0] 7/7 [==========] - 1s 131ms/step - loss: 0.3725 - accuracy: 0.9276 test accuracy = [0.37249326705932617, 0.9276018142700195] 7/7 [==========] - 1s 124ms/step Accuracy: 0.9276018099547512 Precision: 0.928 recall: 0.928 F-Measure: 0.928	train accuracy = [2.5468742023804225e-05, 1.0] 7/7 [==========] - 1s 156ms/step - loss: 0.3218 - accuracy: 0.9367 test accuracy = [0.3218283951282501, 0.9366515874862671] 7/7 [==========] - 1s 138ms/step Accuracy: 0.9366515837104072 Precision: 0.938 recall: 0.937 F-Measure: 0.937

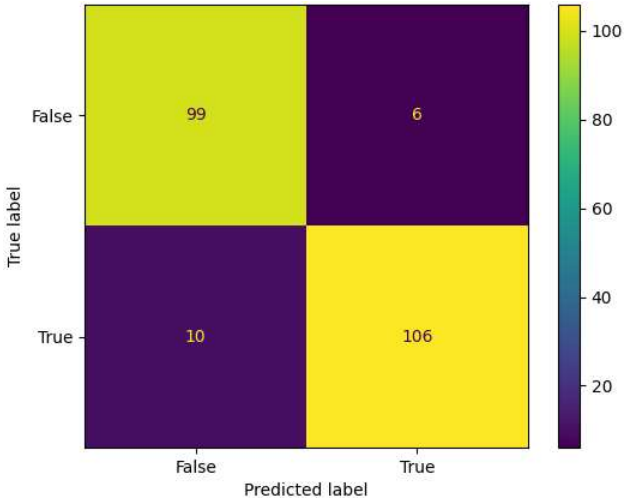
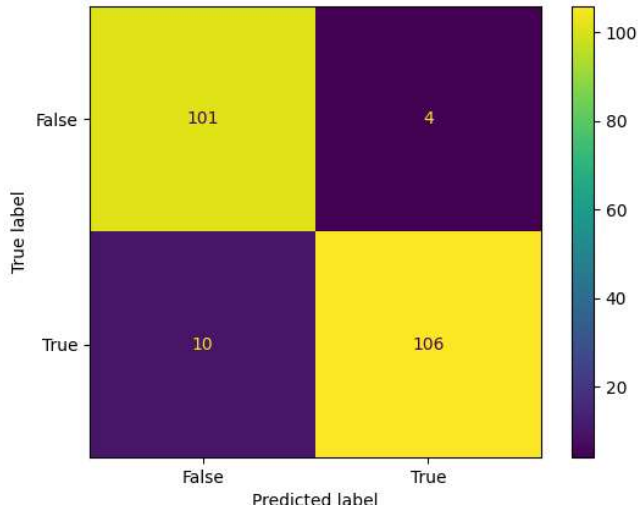

	

Fig 5.8.2 Accuracy Matrix of exp. 16 CNN1	Fig 5.8.3 Accuracy Matrix of exp. 16 CNN2
Model 3 inception3V (pretrained)	**Model 4 inception3V (pretrained)**
Accuracy: 0.8235294117647058 Precision: 0.824 recall: 0.824 F-Measure: 0.823	Accuracy: 0.8914027149321267 Precision: 0.898 recall: 0.891 F-Measure: 0.891

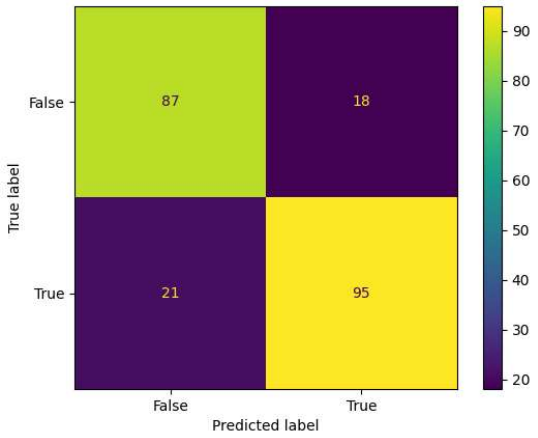


Fig 5.8.4 Accuracy Matrix of exp. 16 inception3V (3)

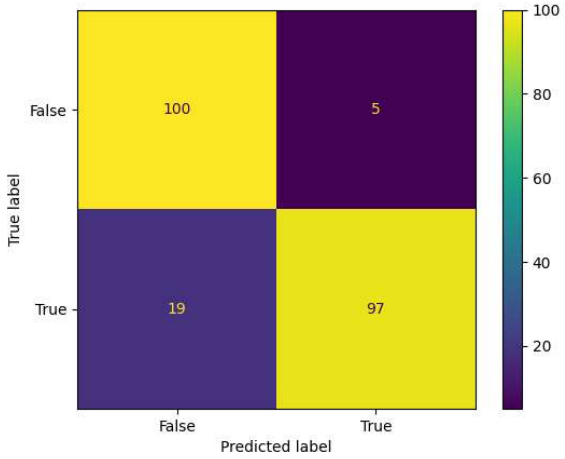


Fig 5.8.5 Accuracy Matrix of exp. 16 inception3V (4)

Model Alex Net

train accuracy = [0.0002266874216729775, 1.0]
11/11 [==============================] - 3s
288ms/step - loss: 0.1249 - accuracy: 0.9578
test accuracy = [0.12492857128381729,
0.9577677249908447]
Accuracy: 0.9577677224736049
Precision: 0.958
recall: 0.958
F-Measure: 0.958

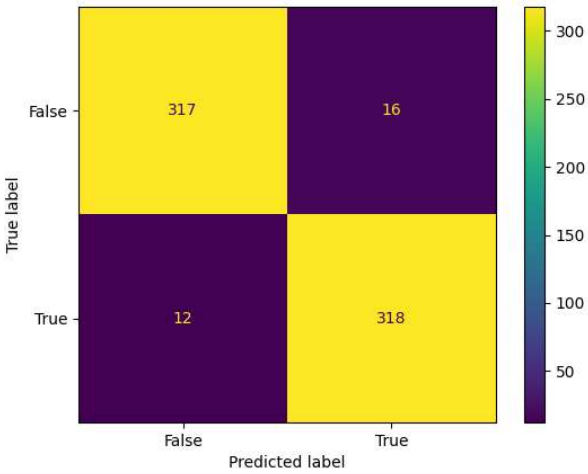


Fig 5.8.6 Accuracy Matrix of exp. 16 Alexnet

Model CNNSVM

train accuracy = [0.008021362125873566, 1.0]
7/7 [==============================] - 0s
43ms/step - loss: 0.3300 - accuracy: 0.9095
test accuracy = [0.3300335109233856,
0.9095022678375244]
7/7 [==============================] - 0s
41ms/step
Accuracy: 0.9095022624434389
Precision: 0.910
recall: 0.910
F-Measure: 0.909

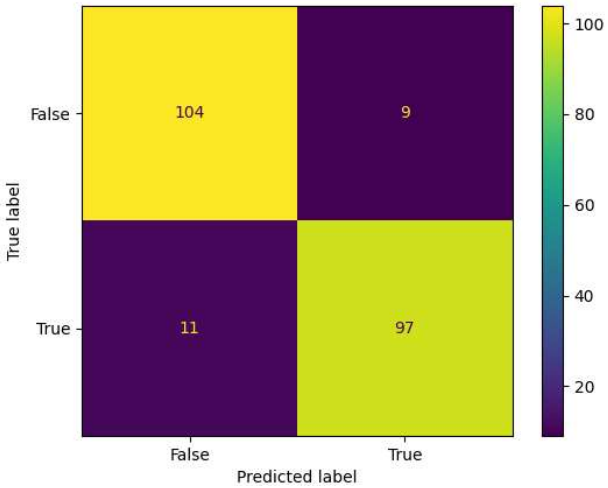


Fig 5.8.7 Accuracy Matrix of exp. 16 CNNSVM

We did augmentation on the whole data with batch-size = 4

Parameters:

datagen = ImageDataGenerator(rescale=1. / 255, rotation_range=20, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.1, zoom_range=0.1, horizontal_flip=True, vertical_flip=True, brightness_range=[0.9, 1.1], fill_mode='nearest')

results

train accuracy = [0.39137524366378784, 0.9186508059501648]
28/28 [==============================] - 4s 152ms/step - loss: 1.0303 - accuracy: 0.8245
test accuracy = 1.0302766561508179, 0.8244620561599731]
28/28 [==============================] - 4s 143ms/step
Accuracy: 0.8244620611551529
Precision: 0.835
recall: 0.824
F-Measure: 0.822

After tuning the parameters of the augmentation

Batch- size here is only = 3

datagen = ImageDataGenerator(rescale=1. / 255, rotation_range=20, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.1, zoom_range=0.1, horizontal_flip=True, fill_mode = 'nearest')

Accuracy when Batch-size = 128

Data size = 3311

train accuracy = [0.00012914894614368677, 1.0]
6/6 [==============================] - 3s 455ms/step - loss: 0.3713 - accuracy: 0.9231
test accuracy = [0.3712729513645172, 0.9230769276618958]
21/21 [==============================] - 3s 142ms/step
Accuracy: 0.9230769230769231
Precision: 0.923
recall: 0.923
F-Measure: 0.923

results in 30 epochs, batch-size = 64.

Table 5.4.2 accuracy table for exp.16

Model 1 CNN	Model 2 CNN
train accuracy = [0.02166377194225788, 1.0] 11/11 [==========] - 3s 275ms/step - loss: 0.1925 - accuracy: 0.9412 test accuracy = [0.19253228604793549, 0.9411764740943909] 21/21 [==========] - 3s 143ms/step Accuracy: 0.9411764705882353 Precision: 0.942 recall: 0.941 F-Measure: 0.941	train accuracy = [4.438757969182916e-05, 1.0] 11/11 [==========] - 3s 270ms/step - loss: 0.2843 - accuracy: 0.9382 test accuracy = [0.2843115031719208, 0.9381598830223083] 21/21 [==========] - 3s 146ms/step Accuracy: 0.9381598793363499 Precision: 0.938 recall: 0.938 F-Measure: 0.938

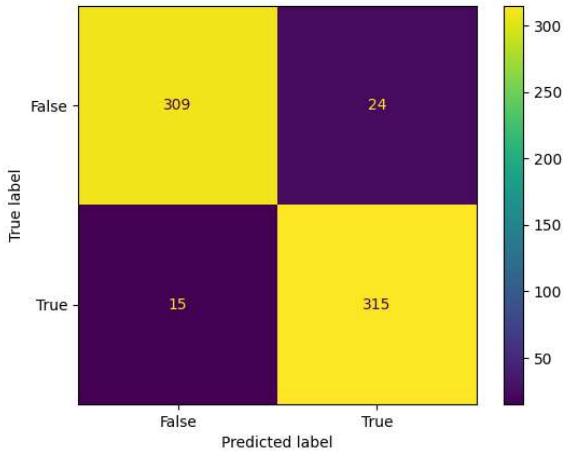
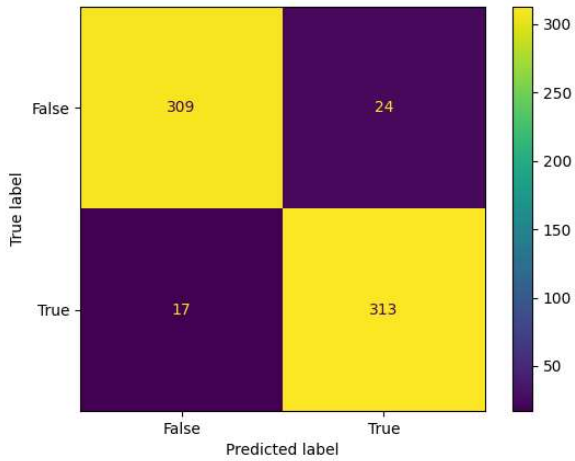

	

Fig 5.8.8 Accuracy Matrix of exp. 16 CNN1	Fig 5.8.9 Accuracy Matrix of exp. 16 CNN2 Note: accuracy was fixed around 93 since epoch 22
Model 4 inception3V (pretrained)	Model 5 inception3V (pretrained)
Accuracy: 0.9788838612368024 Precision: 0.979 recall: 0.979 F-Measure: 0.979	Accuracy: 0.9441930618401206 Precision: 0.945 recall: 0.944 F-Measure: 0.944

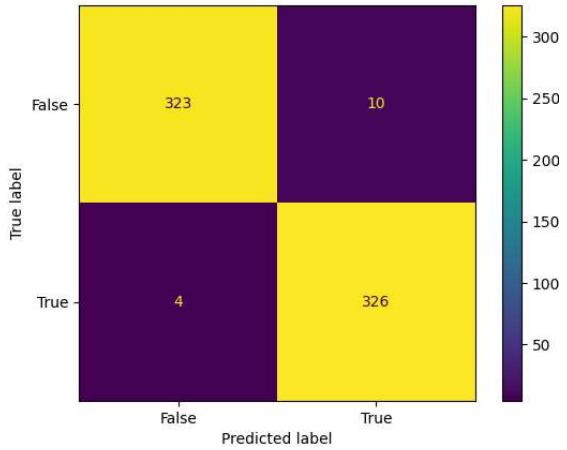


Fig 5.8.10 Accuracy Matrix of exp. 16 Inception3V (4)

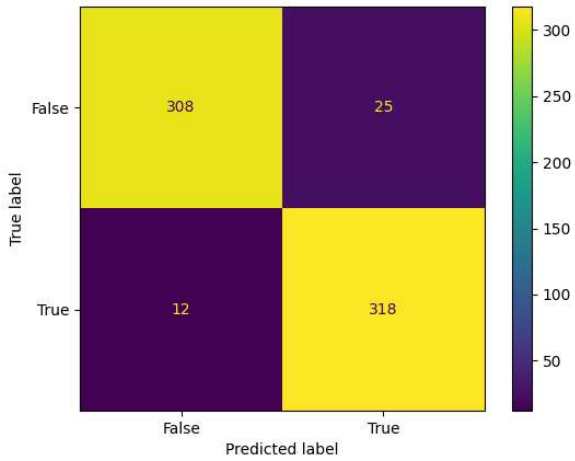


Fig 5.8.11 Accuracy Matrix of exp. 16 Inception3V (5)

Model Alex Net

train accuracy = [0.0005074588116258383, 1.0]
11/11 [==============================] - 3s 257ms/step - loss: 0.1386 - accuracy: 0.9563
test accuracy = [0.1386023312807083, 0.9562594294548035]
Accuracy: 0.9562594268476622
Precision: 0.957
recall: 0.956
F-Measure: 0.956

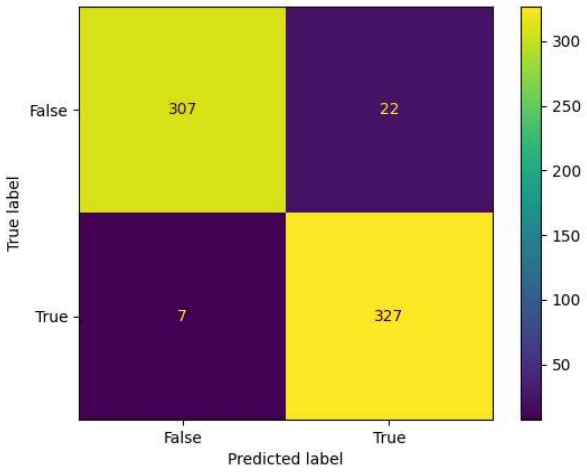


Fig 5.8.12 Accuracy Matrix of exp. 16 Alexnet

Model CNNSVM

train accuracy = [0.007088968530297279, 1.0]
11/11 [==============================] - 1s 66ms/step - loss: 0.3076 - accuracy: 0.8929
test accuracy = [0.30760928988456726, 0.8929110169410706]
21/21 [==============================] - 1s 37ms/step
Accuracy: 0.8929110105580694
Precision: 0.893
recall: 0.893
F-Measure: 0.893

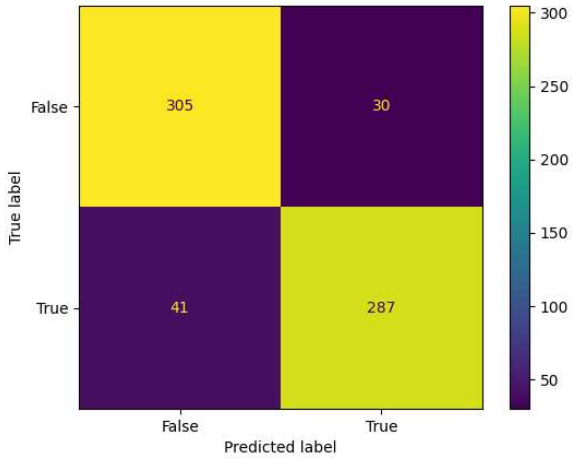


Fig 5.8.13 Accuracy Matrix of exp. 16 Alexnet

Augmentation with only on training data with batch-size =5 in augmentation.

Table 5.4.3 accuracy table for exp.16

Model 4 inception3v	Model CNN 1
ain accuracy = [0.002291244687512517, .9990919232368469] /4 [==========] - 3s 684ms/step - oss: 1.3533 - accuracy: 0.8818 est accuracy = [1.3533425331115723, .8818181753158569] /7 [==========] - 5s 527ms/step ccuracy: 0.8818181818181818 recision: 0.883 ecall: 0.882 -Measure: 0.882	train accuracy = [0.057570770382881165, 0.9961407780647278] 4/4 [==========] - 0s 105ms/step - loss: 0.4045 - accuracy: 0.9182 test accuracy = [0.40449368953704834, 0.918181836605072] 7/7 [==========] - 1s 60ms/step Accuracy: 0.9181818181818182 Precision: 0.920 recall: 0.918 F-Measure: 0.918

We increased batch size to 7

Data size train = 6171, test = 220

Batch-size in run = 64, epochs = 30

Table 5.4.4 accuracy table for exp.16

CNN 2 Model	Alex Net
train accuracy = [6.360341649269685e-05, 1.0] 4/4 [==========] - 2s 383ms/step - loss: 0.4907 - accuracy: 0.9364 test accuracy = [0.49066269397735596, 0.9363636374473572] 7/7 [==========] - 2s 259ms/step Accuracy: 0.9363636363636364 Precision: 0.937 recall: 0.936 F-Measure: 0.936	train accuracy = [9.573027637088671e-05, 1.0] 4/4 [==========] - 1s 226ms/step - loss: 0.3222 - accuracy: 0.9318 test accuracy = [0.3221839666366577, 0.9318181872367859] Accuracy: 0.9318181818181818 Precision: 0.932 recall: 0.932 F-Measure: 0.932

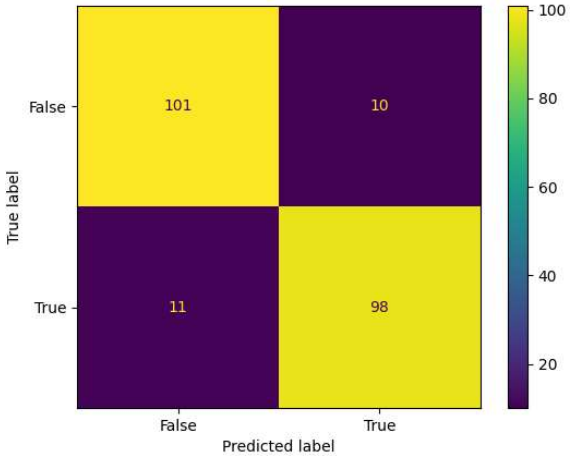


Fig 5.8.14 Accuracy Matrix of exp. 16 CNN2

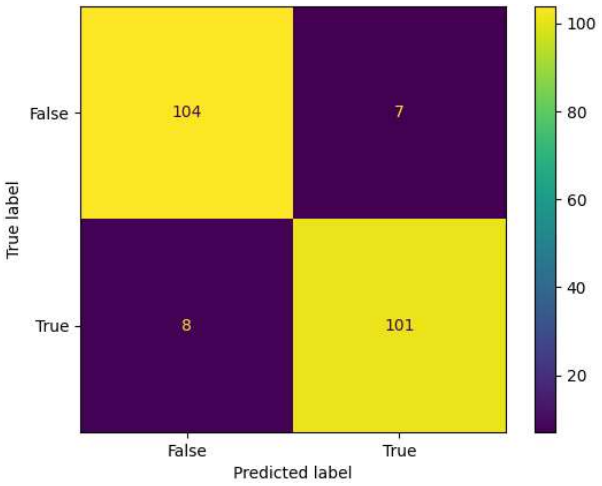


Fig 5.8.15 Accuracy Matrix of exp. 16 Alexnet

Inception3V model 4 (Pretrained)	Inception3V model 5 (Pretrained)
Accuracy: 0.9 Precision: 0.905 recall: 0.900 F-Measure: 0.900	Accuracy: 0.8772727272727273 Precision: 0.882 recall: 0.877 F-Measure: 0.877

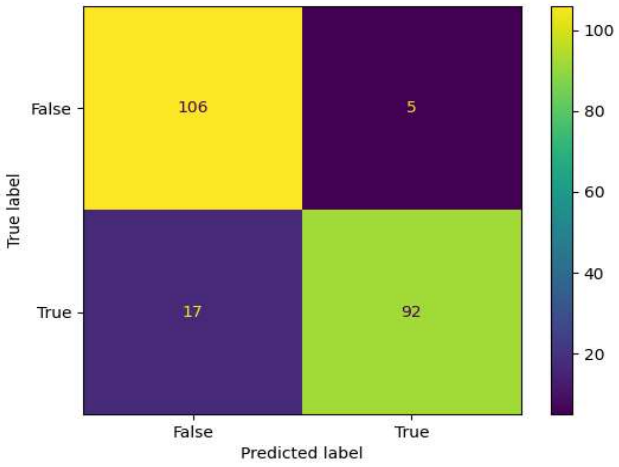


Fig 5.8.16 Accuracy Matrix of exp. 16 inception3V (4)

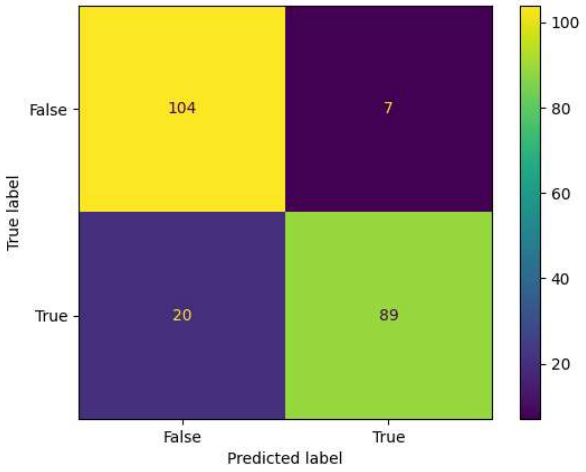


Fig 5.8.17 Accuracy Matrix of exp. 16 inception3V (5)

CNNSVM

Accuracy: 0.9181818181818182
Precision: 0.918
recall: 0.918
F-Measure: 0.918

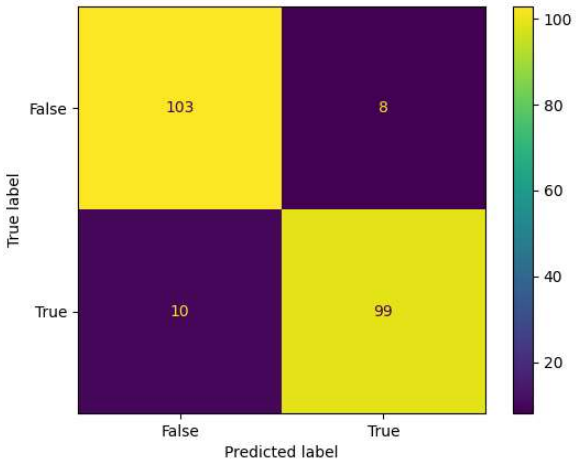


Fig 5.8.18 Accuracy Matrix of exp. 16 CNNSVM

After all the models the time came to convert the highest accuracy of them to file.h5 to use it in the backend of the web.

We decided to use Model Inception 4 as our model for testing and prediction the images in our website as it scored the highest accuracy.

We ran it again on only 15 epochs and this is the accuracy

Accuracy: 0.9547511312217195
Precision: 0.956
recall: 0.955
F-Measure: 0.955
We saved it as "model4inception.h5" to use it later in testing.

An important phase of the project was to make the report for the abnormal class using a bunch of abnormalities. The usual problem was the un availability of the data, we spent around 3 weeks searching all the websites, we called around 6 radiology centers but either there is no data or data is not allowed to be shared. We searched some atlases and books related to the subject to search for data in it like '' Imaging of Fetal Brain and Spine'' for B. S. Rama Murthy and ''MRI of the fetal Brain'' for Catherine Garel and many other books and reach papers but came out with nothing. Until we decided to search for some abnormalities by name. we found a paper that have a collection of fetal brain diseases and we started to search for images for every brain abnormality. Radiopaedia.org is what we discovered and it's a radiology reference for many types of radiology and it's loaded with many cases. We found some data and downloaded images for 6 fetal brain diseases. Agenesis of the corpus callosum, Dandy walker malformation with corpus callosum dysgenesis, Dysplasia of CSP and anterior corpus callosum, Fetal hydrocephalus, intraventricular hemorrhage and Mild unilateral ventriculomegaly.

These images for every category were different in their number in each class, we used augmentation to make each class reach to 100 images, slightly above or below 100. And we ran the models.

Most of the models were high accuracy but didn't classify right when we tested them also the accuracy was high it except Inception3v the 4's model with 12 epochs and 16 batch size, we tested the model on some images and most of them were classified right, so we adopted it.

Its accuracy is

Test Loss: 0.40018847584724426
Test Accuracy: 0.9842519760131836
4/4 [==============================] - 3s 436ms/step
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.97	0.98	32

1	1.00	1.00	1.00	17
2	0.95	0.95	0.95	19
3	1.00	1.00	1.00	22
4	1.00	1.00	1.00	10
5	0.96	1.00	0.98	27
accuracy			0.98	127
macro avg	0.99	0.99	0.99	127
weighted avg	0.98	0.98	0.98	127

we wanted to handle a case in which if the user entered unknown image of anything other than the MRI normal or abnormal. So, we collected some random images from 2 datasets for paintings, sculptures, cartoon etc. around 500 and added them to class called unknown and add them with the augmented normal abnormal data.

The total accuracy got lower because we deleted a step of the preprocessing (Z-normalization) which increases the accuracy around 3%. But all in all, the total accuracy for 3 of 4 models was good.

Accuracy after adding class unknown

Table 5.4.5 accuracy table for exp.16

CNN model 1	Alexnet
40 epochs Accuracy: 0.9083769633507853 Precision: 0.914 recall: 0.908 F-Measure: 0.926	30 epochs Accuracy: 0.930628272251309 Precision: 0.934 recall: 0.931 F-Measure: 0.939

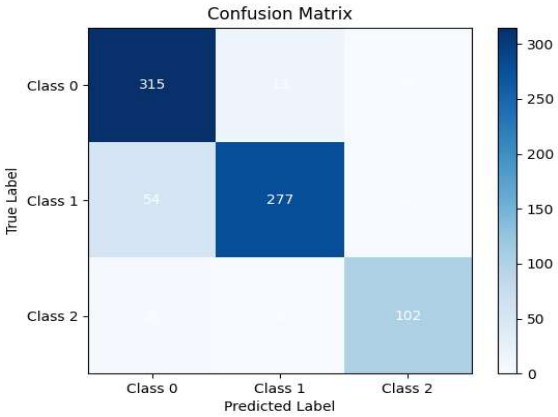
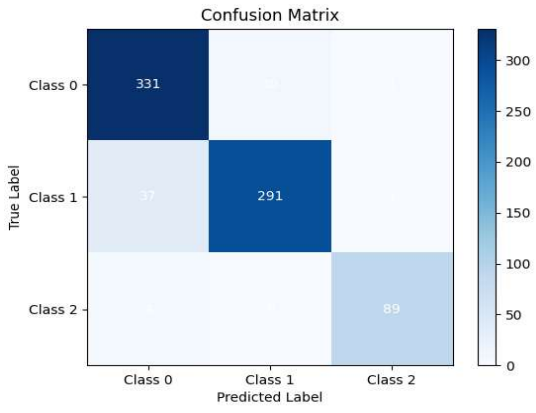
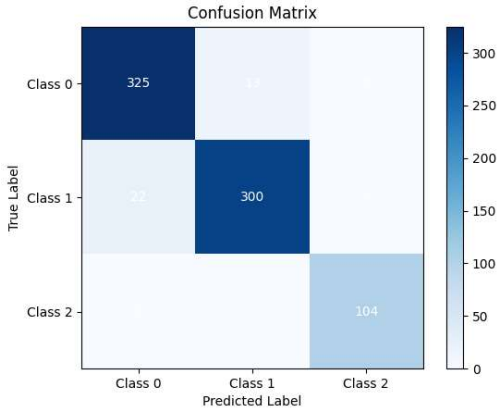

	

Fig 5.8.19 Accuracy Matrix of exp. 16 CNN1	Fig 5.8.20 Accuracy Matrix of exp. 16 Alexnet

Inception3v model 4	CNN Model 2
20 epochs Accuracy: 0.9541884816753927 Precision: 0.954 recall: 0.954F-Measure: 0.965  5.8.21 Accuracy Matrix of exp. 16 Inception3V (4)	40 epochs train accuracy = [0.4377495050430298, 0.4269331693649292] 12/12 [==============================] - 1s 46ms/step - loss: 0.4359 - accuracy: 0.4437 test accuracy = [0.4358675181865692, 0.44371727108955383] 24/24 [==============================] - 1s 24ms/step Accuracy: 0.443717277486911 Precision: 0.197 recall: 0.444 F-Measure: 0.205
CNNSVM	
train accuracy = [0.7029256224632263, 0.5041540861129761] 11/11 [==============================] - 1s 43ms/step - loss: 0.7029 - accuracy: 0.5038 test accuracy = [0.7029247879981995, 0.5037707686424255] Accuracy: 0.5037707390648567 Precision: 0.254 recall: 0.504 F-Measure: 0.335	

Fig

Flask and Web

The functionality of the models in web have 3 functions

Function 1 is to display the probability of the first 2 main classes, it uses whatever model you give to it.

```
def predict_label(image_path, model):
    result1 = ''
    result2 = ''
    result3 = ''
    # Define the classes
    classes = {'normal': 0, 'abnormal': 1, 'unknown': 2}
    # Load the image
```

```
    img_path = image_path
    img = cv2.imread(img_path)
    img = cv2.resize(img, (150, 150))
    img = unsharp_mask(img, radius=20, amount=1)   # unsharp masking
    x = np.expand_dims(img, axis=0)
    # Make the prediction
    prediction = model.predict(x)
    predicted_class = np.argmax(prediction)  # to return the index of the max
prob with refare to the indx of class
    probability_normal = prediction[0][0]
    probability_abnormal = prediction[0][1]
    probability_unknown = prediction[0][2]
    class_label = list(classes.keys())[
        list(classes.values()).index(predicted_class)]  # extract class label
by it's index
    if class_label == 'normal':
        result1 = f"The predicted probability is: {round((probability_normal
* 100), 4)}<br>"
        result2 = class_label
    elif class_label == 'abnormal':
        result1 = f"The predicted probability is:
{round((probability_abnormal * 100), 4)}<br>"
        result2 = class_label
    else:
        result1 = f"The predicted probability is: {round((probability_unknown
* 100), 4)}<br>"
        result2 = class_label
    return result1, result2
```

Function 2 and it's for the type of abnormality, it displays it if the past function returns abnormal and it only use one model to predict the abnormality type

```
def predict_abnormality(image_path, model_1):  # it take the model and the
path so that i can yse it to
    # Load the saved model                        #make models options
    m = ''
    prob = []
    model = load_model('Inception3v_6categories.h5')
    result1, result2 = predict_label(image_path, model_1)
    if result2 == 'abnormal':
        img_path = image_path
        img = cv2.imread(img_path)
        img = cv2.resize(img, (150, 150))
        img = unsharp_mask(img, radius=20, amount=1)   # unsharp masking
        # Calculate the mean and standard deviation of the training images
        mean = np.load('mean.npy')
        # Load std from .npy file
        std = np.load('std.npy')
        # Normalize the input image using z-score normalization
        x = (img - mean) / std
        x = np.expand_dims(img, axis=0)
        # Make the prediction
        prediction = model.predict(x)
        probabilities = prediction[0]
        for p in probabilities:
            prob.append(p * 100)
        max_value = max(prob)   # to return only max prob
```

```
        max_index = prob.index(max_value)  # extract indx of this max prob to
return it's type of abnormality
        if max_index == 0:
            m = f"<br>probability of Agenesis of the corpus callosum:
{round(prob[0], 5)}<br>"
        elif max_index == 1:
            m = f"<br>probability of Dandy walker malformation with corpus
callosum dysgenesis: {round(prob[1], 5)}<br>"
        elif max_index == 2:
            m = f"<br>probability of Dysplasia of CSP and anterior corpus
callosum: {round(prob[2], 5)}<br>"
        elif max_index == 3:
            m = f"<br>probability of Fetal hydrocephalus: {round(prob[3],
5)}<br>"
        elif max_index == 4:
            m = f"<br>probability of intraventricular hemorrhage:
{round(prob[4], 5)}<br>"
        elif max_index == 5:
            m = f"<br>probability of Mild unilateral ventriculomegaly:
{round(prob[5], 5)}"
        return result1, result2, m
    elif result2 == 'normal':
        return result1, result2
    else:
        return result2
```

Web Test Cases:

Test Case 1: We tasted an abnormal image with Inception 3v and the results was correct/.

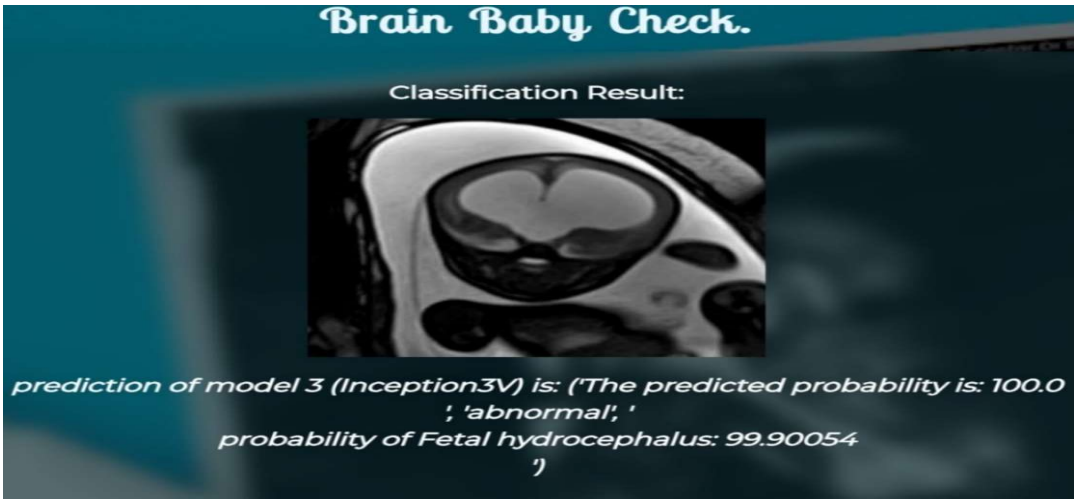


Fig 5.9 testcase 1 using Inception3V abnormal case

Test Case 2: Normal image tested with CNN correct results

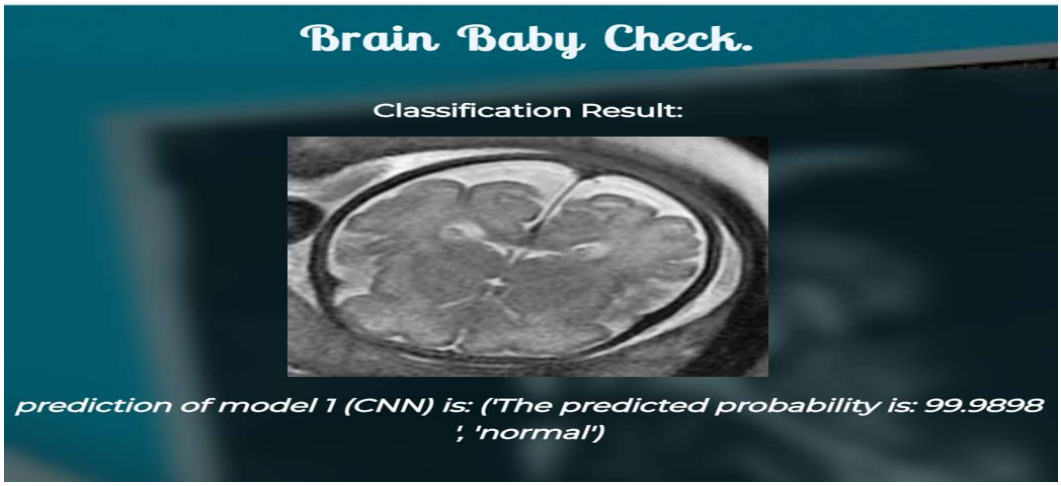


Fig 5.9.2 testcase 2 using CNN normal case

Test Case 3: Random image is tested by Alexnet and the results are correct

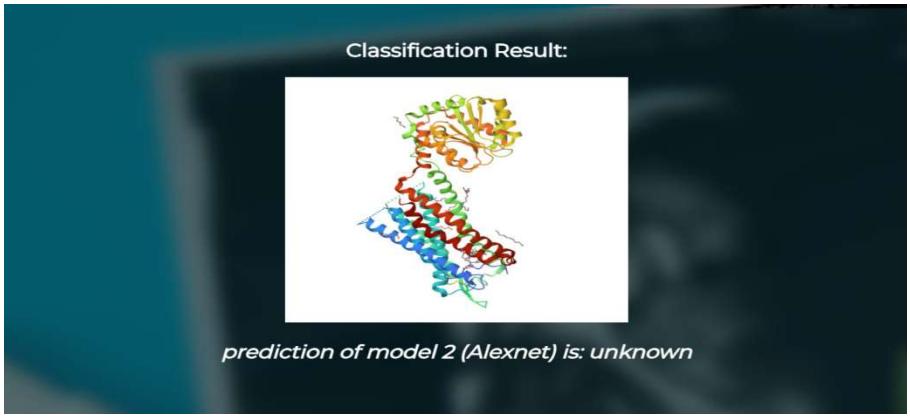


Fig 5.9.3 testcase 3 using Alexnet unknown case

Conclusion:

Magnetic resonance imaging (MRI) Significantly improved imaging of the fetal brain and uterus, so for medical investigation of the brain it we may consider it as a crucial tool.

In recent times, fetal MRI is a non-invasive, modern tool to monitor the development of the fetal brain. MRI provides high contrast resolution for the brain texture.

DL techniques are usually used to classify and detect tumors and brain abnormalities for fetal MRI photos with none surgical interventions at early stage.

Early detection will indicate many things as possible treatments that can be taken, how the mother will manage the pregnancy.

Moreover, early discovery can improve quality of diagnosis and the follow up plans.

Deep Learning Techniques had shown its capability to classify MRI images as long as the data is available and in decent amount. The more the data is large the more accuracy of models is high.

We provided a technique for identifying and categorizing fetal brain abnormalities. For this, we employed deep learning methodologies and algorithms, such as the CNN, Alexnet, CNNSVM, Inception3V algorithm. Our proposed work, in contrast to other studies that achieved the same goals using Deep learning techniques and algorithms, has demonstrated higher accuracy and more effective outcomes. Our accuracy, which is higher than any prior work, was between 97.9%

Future Work:

In the future, we aim to add all fetal brain diseases to the classification instead of just six types when we obtain more data. We also aim to improve the website and try to differentiate between users to be either doctors or regular users, and attempt to connect them through the website in case a doctor is needed. We were also able to predict and detect various brain anomalies using images from various gestational ages. In this case, the CNN classifier performed better than any other algorithm in terms of lowering computational costs and accelerating image processing. In the upcoming work, we want to increase efficiency. Additionally, we will expand the data sets. Additionally, we will categorize various abnormal diseases like tumors and cerebellar hypoplasia.

References:

[1] Fetal MRI: Brain. Available online: https://radnet.bidmc.harvard.edu/fetalatlas/brain/brain.html

[2] Shen, L., Zheng, J., Shpanskaya, K., McKenna, E.S., Atluri, M., Guimaraes, C.V., Dahmoush, H., Halabi, S.S. & Yeom, K.W. (2021). Fetal Brain MRI from Stanford Lucile Packard Children's Hospital. Stanford Digital Repository. Available at: https://purl.stanford.edu/sf714wg0636

[3] Laura Meeker on 24 Nov 2021, Fetal brain tumors, Radiopaedia.org, Available: https://radiopaedia.org/search?lang=us&modality=MRI&page=1&q=fetal+brain&scope=cases

-Wikimedia Foundation. (2022, September 4). Maternal–fetal medicine. Wikipedia. https://en.wikipedia.org/wiki/Maternal%E2%80%93fetal_medicine

- Vihar Kurama. (2021).ML-based Image Processing. Nononets. https://nanonets.com/blog/machine-learning-image-processing/

- National Institutes of Health. U, S. Department of Health & Human Services Magnetic Resonance Imaging (MRI) https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri

- Saraswathi, E. (2021). Recognition and Classification of Fetal Brain Abnormalities. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(9), 2534-2540. https://turcomat.org/index.php/turkbilmat/article/view/3737

- Attallah, O., Sharkas, M. A., & Gadelkarim, H. (2020). Deep Learning Techniques for Automatic Detection of Embryonic Neurodevelopmental Disorders. Diagnostics (Basel, Switzerland), 10(1), 27. https://doi.org/10.3390/diagnostics10010027

- Attallah, O., Sharkas, M. A., & Gadelkarim, H. (2019). Fetal Brain Abnormality Classification from MRI Images of Different Gestational Age. Brain sciences, 9(9), 231. https://doi.org/10.3390/brainsci9090231

- Song, L., Wang, Q., Liu, T., Li, H., Fan, J., Yang, J., & Hu, B. (2022). Deep robust residual network for super-resolution of 2D fetal brain MRI. Scientific Reports, 12(1), 406.

https://www.nature.com/articles/s41598-021-03979-1

- Jagadeesh, P., Subashini, S., & Lakshmi, R. V. Detecting and Classifying Fetal Brain Abnormalities Using Deep Learning.

https://ijresm.com/Vol.3_2020/Vol3_Iss4_April20/IJRESM_V3_I4_61.pdf