AI FOR CANCER DETECTION

Dakshita Reebadiya

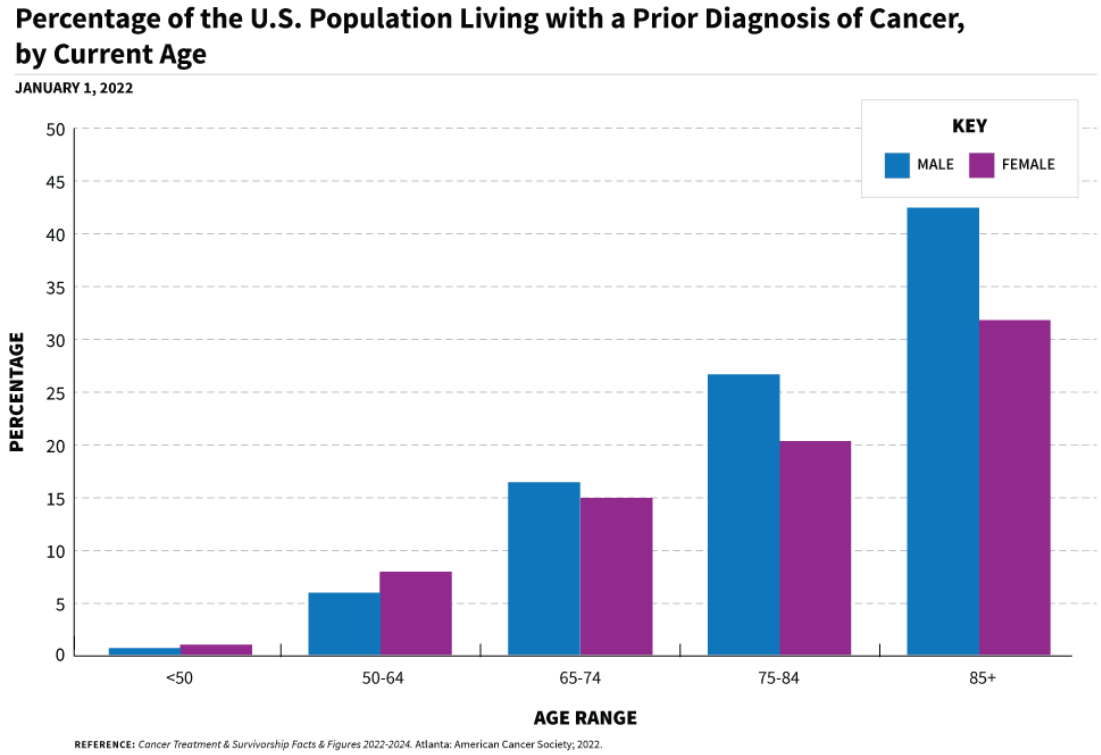
14th December, 2022

Abstract

Tradition clinical methods consume too much time to analyze and prognosis cancer. The number of cancer patients is increasing day by day. Hence, cancer patients must be analyzed as fast as possible. A biopsy takes around ten days for prognosis reports. Some types of cancers grow and spread exponentially. So, it is not well to wait for a long time for a prognosis. Such situations can be overcome using Artificial Intelligence (AI). Machine automation based on AI is one solution for such a critical condition. AI empowers machines by training them with past data, learning from it, and predicting future values. Here, we proposed Deep Learning (DL)-based Convolutional Neural Network (CNN) algorithm to process images and classify future data for cancer patients. For that, a case study on lung cancer is shown with implementation having an accuracy of 86%.

1.0 Introduction

Nowadays, Cancer is highly causing disease worldwide. The Number of people having Cancer is increasing exponentially. Due to that, Cancer is considered one of the prime reasons for death. According to research, the Number of cancer cases in 2010 was 18.7 million, which jumped to 26.3% growth in 2019. Similarly, death caused by a tumor increased by 20.9% in 2019 compared to 2010. The following graph shows the growth of cancer cases worldwide.



Cancer is caused by genetic changes, where a normal cell transforms into a cancerous cell passing through various stages, i.e., from a damaged cell to malignant Cancer. Cancer is diagnosed by physical, imaging, laboratory, or biopsy tests. A tumor can be detected significantly in different organs such as brain, breast, lungs, skin, mouth, kidney, and blood using medical imaging analysis. Abnormal cells can be of two types: benign and malignant. Malignant cells are cancerous and are very difficult to detect at the initial stage. Hence, prognosis accuracy is low.

As per the details above, Digital pathology (DP) can give more accurate prognoses, as the model is trained using historical data. Deep learning (DL)-based algorithm Convolutional Neural Network (CNN) analyses the Computed Tomography (CT) images and learns the behaviour and patterns of a tumor in a specific organ. For explicit knowledge about detection using CNN, a Lung-cancer case study is performed, and implementation is shown further.

2.0 Customer needs Assessment

As a cancerous cell grows abnormally, it takes time to prognosis the next stage period using current techniques. Apart from that accuracy of the prognosis is also only 60%. Hence, rapid detection of the behaviour and pattern is highly recommended. Here, we have recommended a DL-based CNN algorithm to predict the cancer type and behaviour of abnormal cells from CT images. The model is fed with historical CT records and trained using the CNN algorithm. The model learns through the data, detects tumor, and find the pattern, then validates using a validation dataset to check the accuracy. Detailed descriptions and implementation are given in corresponding sections.

3.0 Revised Needs Statement and Target Specifications

The following are vital factors in getting in-depth information about cancer aggressiveness and for particular medical assistance:

1. When cancer is detected.
2. How accurate cancer diagnosis is.
3. In which stage cancer is.

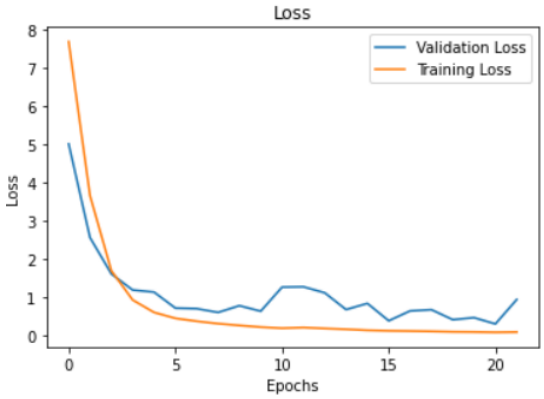
AI gives accurate results in such critical situations compared to human experts by analyzing a pre-provided dataset of a cancerous and non-cancerous patient. DNN is a powerful tool that uses complex computations to detect malignant tissues from a large set of images captured during the biopsy. Additionally, CNN can classify tumor subtypes as having a minor difference.

4.0 External Search

AI has the ability to maximize the speed of analysis and improve accuracy and flexibility for clinical decision-making. AI can go beyond humans' ability by finding certain critical behaviours, and complex genetic patterns by analyzing extensive records. One scientist group has created AI based model which uses patients' mammogram images and predicts the approximate possibility of developing breast tumors in the next five years. It also recommends a period to screen for breast cancer. According to previous research, we have recommended DL-based CNN to prognosis existing cancer and predict potential growth. Dataset is taken from Kaggle, and results are roughly calculated for research work.

4.1 Benchmarking

As cancer is a prime factor for the high rate of death worldwide, cancer detection at an early stage is most required. Hence, we require high model accuracy (>98%). The result must have a negligible loss, as shown in the below diagram.



4.2 Applicable patents

Convolutional Neural Network for Cancer Detection – Patent No: US 9,739,783 B1 Issue Aug. 22, 2017

US9739783B1 - Convolutional neural networks for cancer diagnosis - Google Patents

4.3 Applicable Standards

1. Required high accuracy as the model is predicting tumor.
2. Existing research work is also recommended to survey before deploying the model.
3. The Healthcare dataset contains sensitive data about the patient.
4. Patient's confidential data must be secured.
5. Model training requires high computation power, availability, and connectivity to actual word data for model upgradation and reliability.

4.4 Applicable Constraints

1. As the model predicts results on such a prime illness – cancer- it requires much research to get accurate data and outcomes.
2. The dataset contains sensitive and private user information, so it must be confidential.
3. It is hard to get confidential real word data to train a model.
4. Doctors are required to train with new technology to diagnose patients.

4.5 Business Opportunity

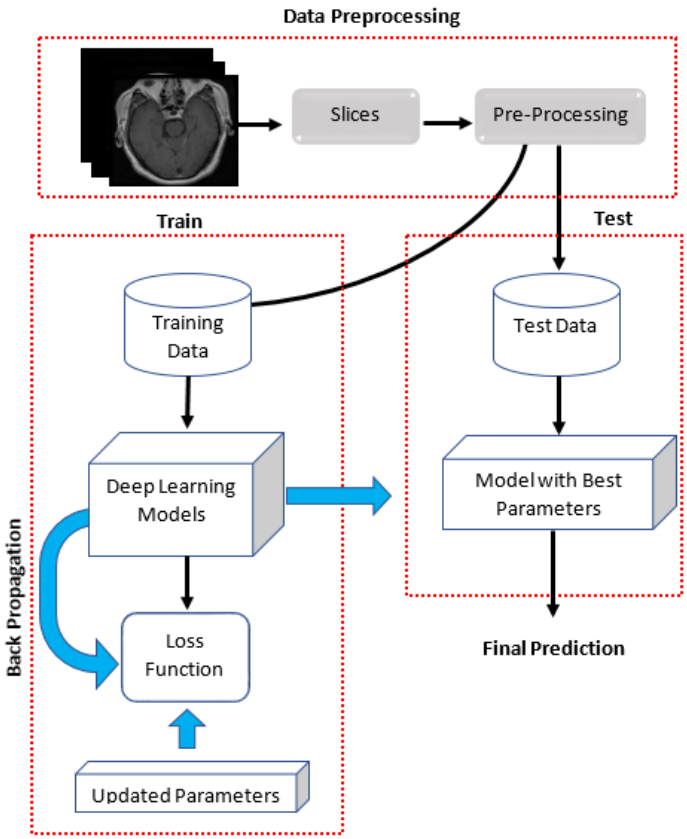
Current manual systems should take less time to show results. Apart from that, it is tough to accurately recognize and accurately predict a tumor growth period in the initial phase. System automation with past datasets can give rapid results and solutions for a particular treatment. Hence, DP has a high opportunity as it uses the DL-based algorithm CNN to classify the cancer type by complex computations and critical analysis on a vast dataset.

5.0 Concept Generation

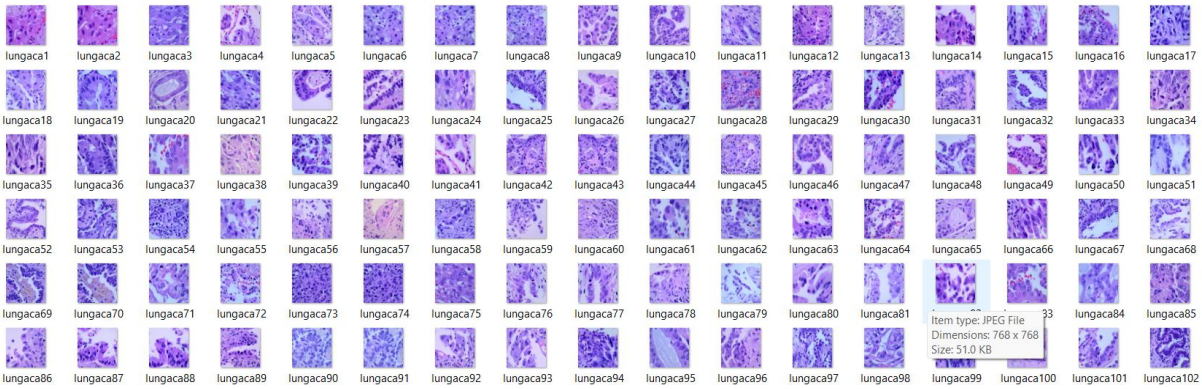
We used the DL-based CNN algorithm for cancer classification and prognosis, which requires an image dataset. Images are CT-scan reports which show where the tumor is and the periodical growth of the tumor. Initially, the tumor seems normal, so it is tough for humans to prognosis it. The effect of cancer spreads to the body exponentially. As the tumor grows, it passes through some stages as described below:

1. **Stage I:** Cancer is in the initial phase and affects a minimal area or a cell of a particular organ. This cancer is primarily curable by removing the affected part.
2. **Stage II (Early-Stage Cancer):** Affected cell start growing and showing more abnormality, but in this stage, other cells are not yet affected by the tumor.
3. **Stage III:** In this stage, the cancerous cell overgrows in size and starts affecting other parts of that organ.
4. **Stage IV (Metastatic-Stage Cancer):** This stage is the final stage where cancer has affected other parts of the body and is almost impossible to cure.

The type of cancer is named based on where cancer started, i.e., breast, brain, skin, lung, and kidney. So, the model can be developed for a particular type of cancer. Here, we have studied lung-based cancer caused by smoking and exposure to asbestos or radon. Dataset is fed to the model in two parts: the training dataset and the testing dataset. The training dataset is used to train a model for learning and analyzing cancer, while the testing dataset is used to validate the outcomes and accuracy of the model.



The dataset is taken from Kaggle named as "*lung_colon_image_set"* which contains 25000 images.

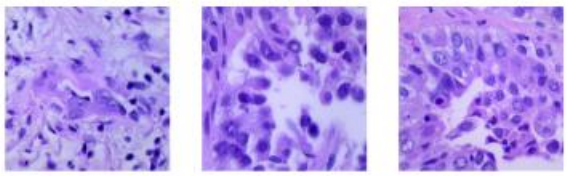


The software tool is required to develop a model to meet the necessary functionality of the model. Here, Implementation is performed on Anaconda Navigator using Python. We used various libraries for database pre-processing and CNN development, such as pandas, numpy, glob, sklearn, keras, tensorflow.

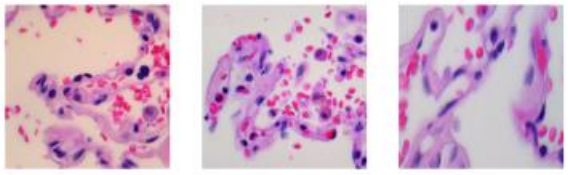
Class categories of dataset:

```
In [4]: for cat in classes:
            img_dir = f'{path}/{cat}'
            images = os.listdir(img_dir)

            fig, ax = plt.subplots(1, 3, figsize=(10,5))
            fig.suptitle(f'Images for {cat} category . . . .', fontsize=20)

            for i in range(3):
                k = np.random.randint(0, len(images))
                img = np.array(Image.open(f'{path}/{cat}/{images[k]}'))
                ax[i].imshow(img)
                ax[i].axis('off')
            plt.show()
```

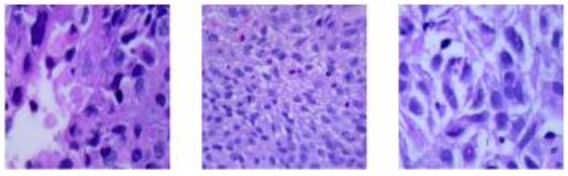
Images for lung_aca category . . . .



Images for lung_n category . . . .



Images for lung_scc category . . . .



Splitting Dataset for training and testing the model:

```
IMG_SIZE = 256
SPLIT = 0.2
EPOCHS = 10
BATCH_SIZE = 64
```

```
X = []
Y = []

for i, cat in enumerate(classes):
    images = glob(f'{path}/{cat}/*.jpeg')

    for image in images:
        img = cv2.imread(image)

        X.append(cv2.resize(img, (IMG_SIZE, IMG_SIZE)))
        Y.append(i)

X = np.asarray(X)
one_hot_encoded_Y = pd.get_dummies(Y).values
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, one_hot_encoded_Y,test_size = SPLIT,random_state = 2022)
print(X_train.shape, X_test.shape)
```

Implementation of Sequential Model:

```
model = keras.models.Sequential([
    layers.Conv2D(filters=32,
                  kernel_size=(5, 5),
                  activation='relu',
                  input_shape=(IMG_SIZE,IMG_SIZE,3),
                  padding='same'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(filters=64,
                  kernel_size=(3, 3),
                  activation='relu',
                  padding='same'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(filters=128,
                  kernel_size=(3, 3),
                  activation='relu',
                  padding='same'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(3, activation='softmax')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      2432

 max_pooling2d (MaxPooling2D  (None, 128, 128, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 128, 128, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 64, 64, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 32, 32, 128)      0
 2D)

 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 256)               33554688

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dense_1 (Dense)             (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 batch_normalization_1 (Batc  (None, 128)              512
 hNormalization)

 dense_2 (Dense)             (None, 3)                 387

=================================================================
Total params: 33,684,291
Trainable params: 33,683,523
Non-trainable params: 768
```

Model Deployment:

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_accuracy') > 0.90:
            print('\n Validation accuracy has reached upto 90% so, stopping further training.')
            self.model.stop_training = True

es = EarlyStopping(patience=3,monitor='val_accuracy',restore_best_weights=True)
lr = ReduceLROnPlateau(monitor='val_loss',patience=2,factor=0.5,verbose=1)
```

```
history = model.fit(X_train, Y_train,validation_data = (X_test, Y_test),batch_size = BATCH_SIZE,epochs = EPOCHS,verbose = 1,call
```

```
Epoch 1/10
188/188 [==============================] - 1019s 5s/step - loss: 0.3769 - accuracy: 0.8442 - val_loss: 2.2958 - val_accuracy:
0.4950 - lr: 0.0010
Epoch 2/10
188/188 [==============================] - 902s 5s/step - loss: 0.2515 - accuracy: 0.9033 - val_loss: 0.5879 - val_accuracy: 0.
8460 - lr: 0.0010
Epoch 3/10
188/188 [==============================] - 834s 4s/step - loss: 0.1921 - accuracy: 0.9233 - val_loss: 0.3755 - val_accuracy: 0.
8600 - lr: 0.0010
Epoch 4/10
188/188 [==============================] - 1316s 7s/step - loss: 0.1528 - accuracy: 0.9386 - val_loss: 0.9038 - val_accuracy:
0.7857 - lr: 0.0010
Epoch 5/10
188/188 [==============================] - ETA: 0s - loss: 0.1227 - accuracy: 0.9525
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
188/188 [==============================] - 786s 4s/step - loss: 0.1227 - accuracy: 0.9525 - val_loss: 6.4131 - val_accuracy: 0.
3447 - lr: 0.0010
Epoch 6/10
188/188 [==============================] - 803s 4s/step - loss: 0.0876 - accuracy: 0.9692 - val_loss: 0.4282 - val_accuracy: 0.
8560 - lr: 5.0000e-04
```

Results:

The accuracy of model is 86%.

```
Y_pred = model.predict(X_test)
Y_test = np.argmax(Y_test, axis=1)
Y_pred = np.argmax(Y_pred, axis=1)
```

```
94/94 [==============================] - 28s 287ms/step
```

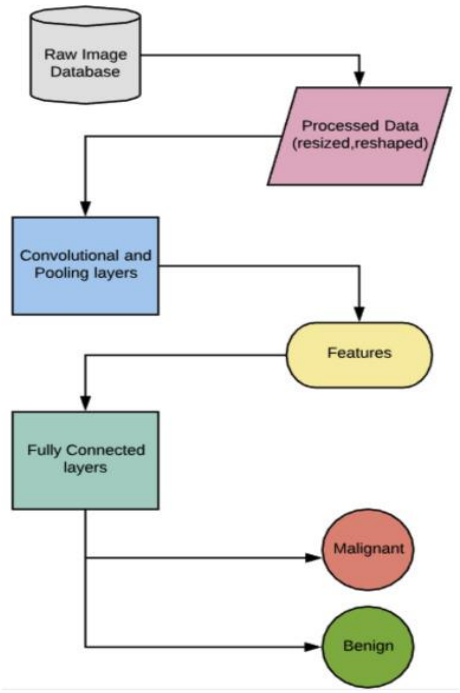
```
metrics.confusion_matrix(Y_test, Y_pred)
```

```
array([[ 605,   58,  324],
       [   2,  974,    1],
       [  35,    0, 1001]], dtype=int64)
```

```
print(metrics.classification_report(Y_test, Y_pred,target_names=classes))
```

```
              precision    recall  f1-score   support

    lung_aca       0.94      0.61      0.74       987
      lung_n       0.94      1.00      0.97       977
    lung_scc       0.75      0.97      0.85      1036

    accuracy                           0.86      3000
   macro avg       0.88      0.86      0.85      3000
weighted avg       0.88      0.86      0.85      3000
```

6.0 Concept Development



As the data grows day by day exponentially, we need high storage capacity. Cloud storage is the best place to store data. Cloud service is centralized. Hence, real-time data can be stored there automatically and used by authenticated users whenever required.

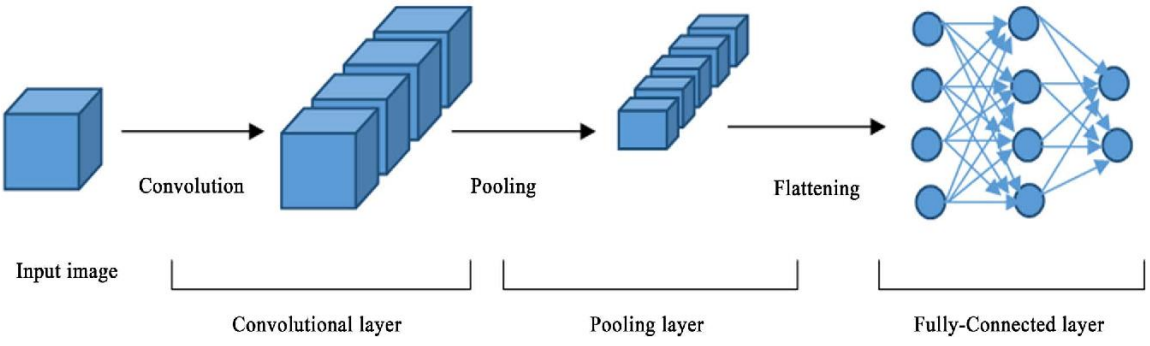
7.0 Final Report Prototype

Introducing DP for cancer detection is far better than humans. Because the machine performs in a fraction of a second without getting exhausted like human-being, it can process an abandoned amount of data repeatedly until it gets an accurate outcome. Machine automation can detect minor infections through imaging analysis. Hence, doctors can start immediate treatment for the patients based on the probability of infection.

CNN Model Explanation:

CNN is a feed-forward neural network used for face detection, handwriting recognition, speech recognition, image classification, voice recognition, etc. CNN has three main layers: Convolutional Layer (CL), Pooling Layer (PL), and Fully-connected Layer (FL). CL and PL are for pre-processing, and FL is for deploying the model.

1. **Convolutional Layer:** The prime functionality of CL is to extract features from the image using one or more filters or feature maps created by the previous layers. And then, it creates a feature map as an output.
2. **Pooling Layer:** The task of the pooling layer is to reduce the dimensions of the feature map. For that, it sums up surrounding outputs considering remaining features. The most commonly used methods are Max Pooling and Average Pooling. Max pooling selects the max value from the feature map, while Average Pooling averages the value of the feature map.
3. **Fully-connected Layer:** This layer is the final step of CNN used for the classification of images. Feature map from pooling layer is converted to vector and fed to FL as the input layer. FL computes the data and gives outcomes.



8.0 Conclusions

DL-based CNN algorithm works well for cancer detection in DP. DP introduced machine automation for accurate and instant outcomes. Current manual task by human consumes time for processing data and analyzing the image. At the same time, the machine is able to process the same thing multiple times in a second. The proposed model gives 86% accuracy. For better results, we require high accuracy (>98%).