

**EAI6010 Applications of Artificial Intelligence**

**Group 4**

Debpriyo Roy

Atharva Kousadikar

Praveen Pandey

Harsh Sindhwa

**Final Project**

**FALL 2019**

**Professor: Sergiy Shevchenko**

**Introduction**

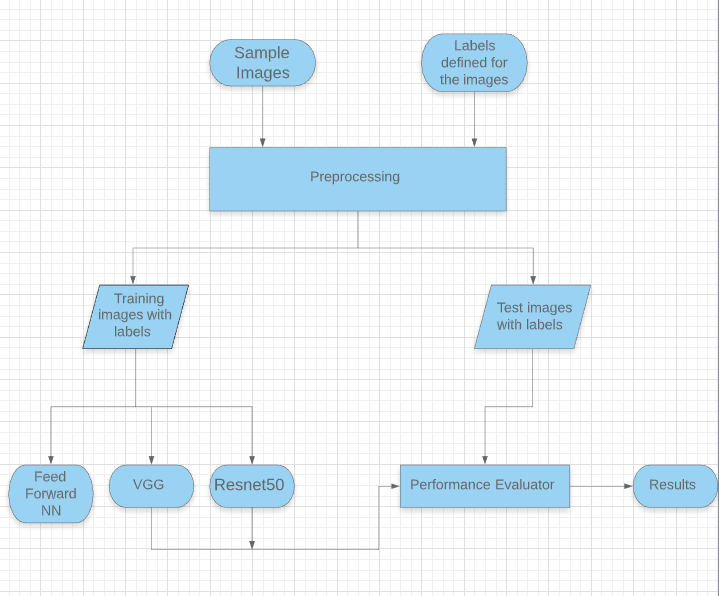
Document image classification is an important step in Digital Libraries and other document image analysis applications. There is great diversity in document image classifiers: they differ in the problems they solve, in the use of training data to construct class models, and in the choice of document features and classification algorithms. We survey this diverse literature using three components: the problem statement, classifier architecture, and performance evaluation. We emphasize techniques that classify single-page typeset document images without using OCR results. Developing a general, adaptable, high-performance classifier is challenging due to the great variety of documents, the diverse criteria used to define document classes, and the ambiguity that arises due to ill-defined document classes.

**Procedural Aspects under Modelling:**

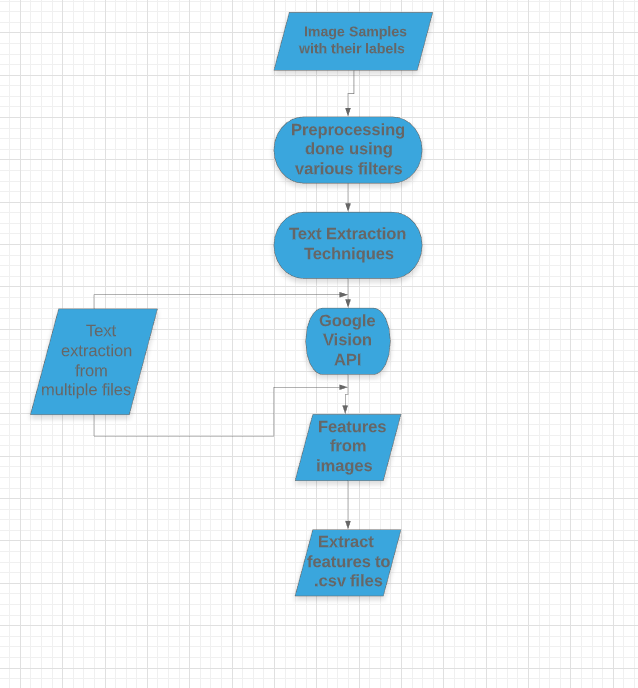
As part of this project, we were trying to solve 2 problem statements (i.e) Document(forms) Classification and Document fields Extraction. Three different components of a document classifier include:

1. Problem Statement: The problem statement for a form classifier has two aspects: the document space and the set of classes. The former defines the range of input forms, and the latter defines the output that the classifier can produce.
2. The Classifier Architecture: Different classification algorithms such as Feedforward Neural Network, Resnet, VGG were applied on the trained data set in order to capture the features from the images and classify accurately on the unseen data.
3. Evaluation Metrics: Performance evaluation is a critically important component of a document classifier. It involves challenging issues, including difficulties in defining standard data sets and standardized performance metrics, the difficulty of comparing multiple document classifiers, and the difficulty of separating classifier performance from pre-processor performance.

In order to represent the flow in the form of a block diagram we have created the three component-based flow chart which is as follows:



Problem Statement 2: Fields Extraction from forms



**Labeling and pre-processing:**

Based on our group discussion and understanding, we differentiated the forms into 4 different categories. Some have blocks for fields, some have lines, and some are empty and some of them are handwritten so we put them in separate sub-folders. We created a directory path in which we saved the path of our folders writes code for labeling. We write a code that labeled the images from 0 to 3 on the basis of their types and saved the labels in an array Y. Keras provides the load\_img() function that can be used to load the image files directly as an array of pixels. The pixel data needs to be converted to a NumPy array for use in Keras. We can use the img\_to\_array() Keras function to convert the loaded data. We used both in-built functions load\_img and img\_to\_array functions and stored the pixels of all images in a list X.

**Image classification:**

For Image classification, we explored different algorithms, since the dataset is very small, and the classification task is an easy problem as all the four types of forms have distinguished features. For example, some forms have blocks, some have lines, and some are empty. That’s why we start the classification by choosing a simple model. We created a 4-layer feedforward neural network that consists of a dense input layer, two hidden layers, and one fully connected output layer. Also, we want to explore how a feedforward neural network model performs in this image classification task. We use this pre-trained model as a feature extractor in our model, we can preprocess the pixel data for the model by using the preprocess\_input() function in Keras,

The input layer is a dense layer of 64 nodes that takes input in the form of an array having the shape of (224,224,3) its output goes to another dense layer with 128 nodes and an activation function “Relu”. The output layer requires the vector, so we use a flatten layer that converts array or tensor into a vector. Finally, we pass the vector to the output which is a fully connected dense layer with 4 nodes followed by a ‘softmax’. For a given example, softmax gives us the probabilities of four categories. (Image 2.1)

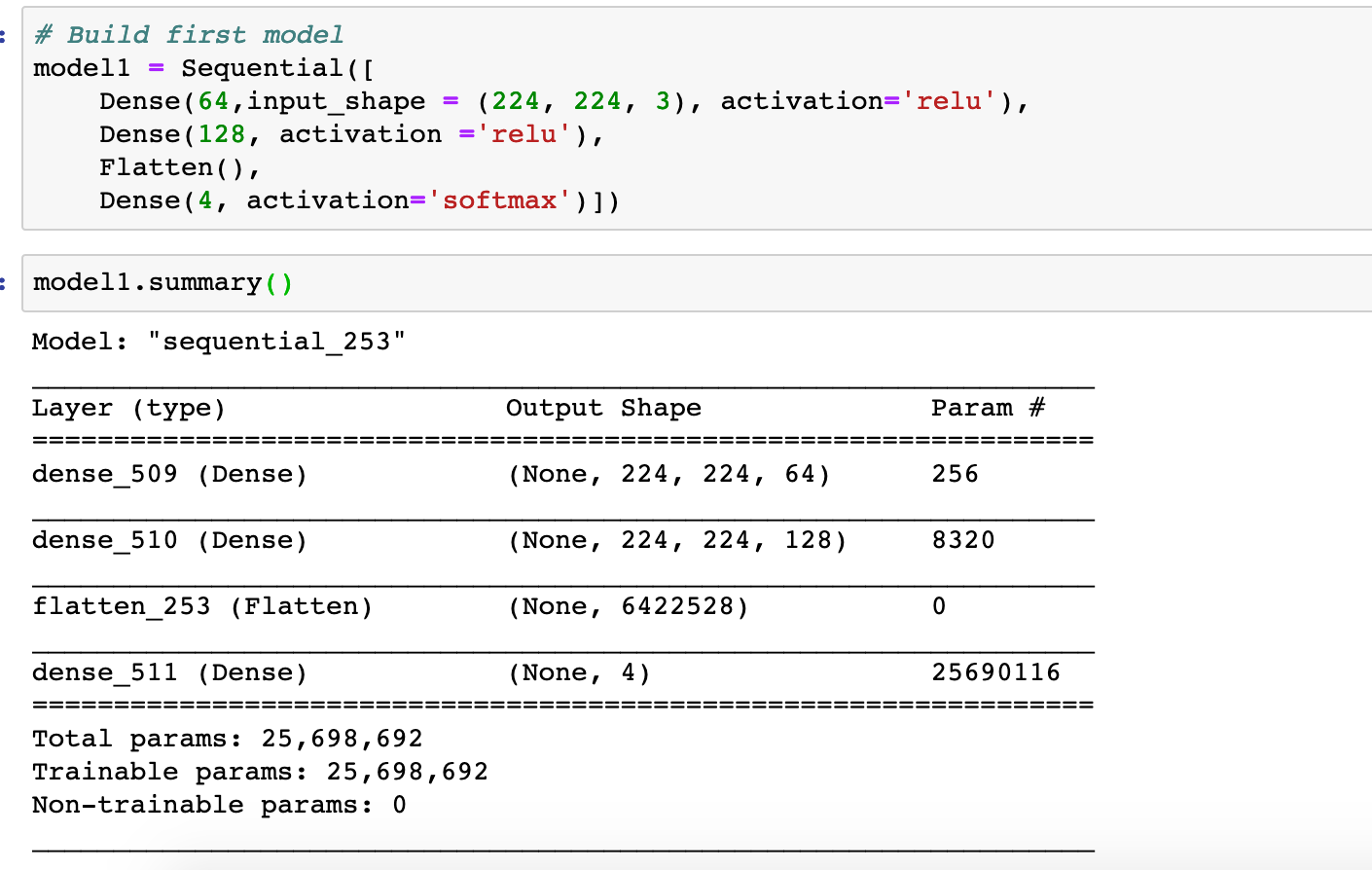


Image 2.1: Model summary

We split our pre-processed input data into train and test with a proportion of 80:20 and inspect that the test data contains the examples of all types of forms. Then we fit our model1 on train data and evaluate the results on test data. We got the training accuracy of 100 percent in the first epoch (refer image 2.2) and the test accuracy of 83%. (refer image 2.3)

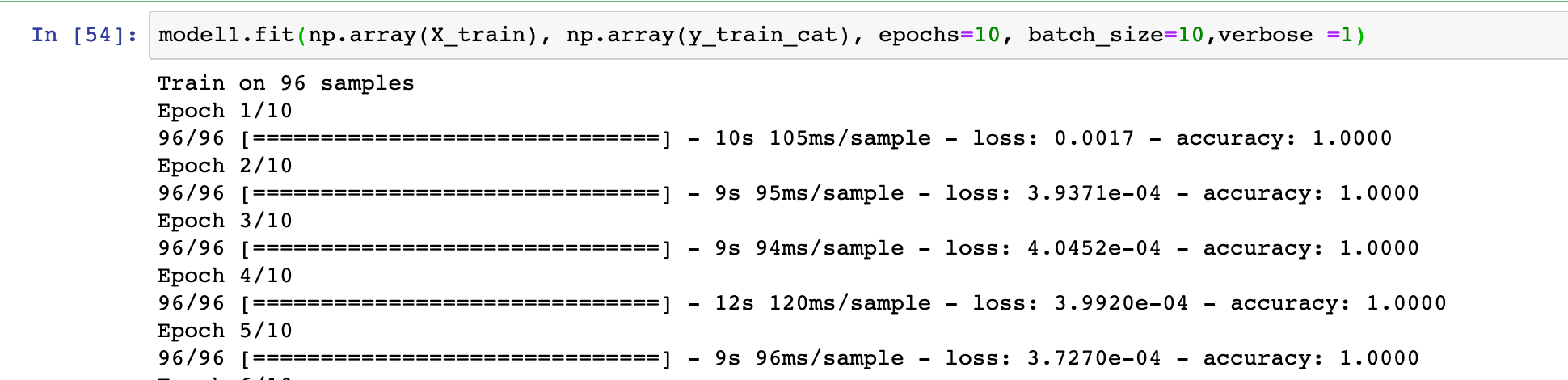


Image 2.2: Train accuracy

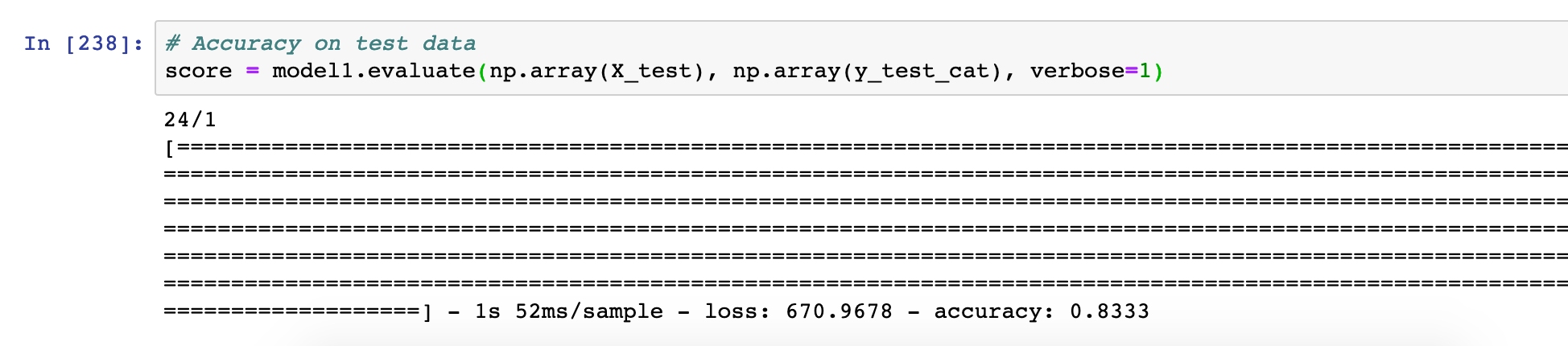


Image 2.3

We fine-tune the model by varying the hyperparameters like the varying number of nodes, learning rate, batch size, etc but they could not achieve more than 95% accuracy. Also, this variation shows that we cannot trust the model performance.

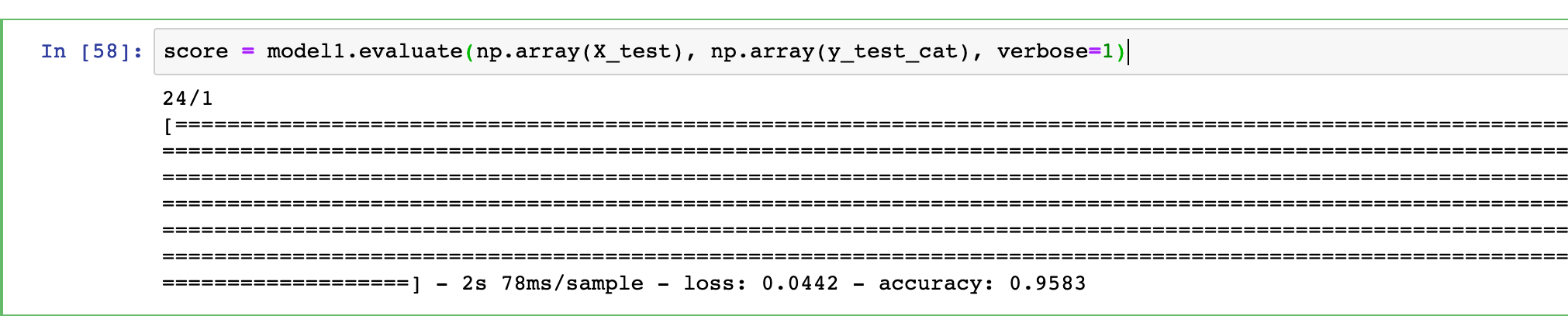


Image 2.4

We also trained a convolution neural network but since our dataset is very small the model performance was poor. So, we decided to move forward with the state-of-the-art deep learning models of image classification like VGG16 and ResNet50.

**Image classification using Transfer Learning**

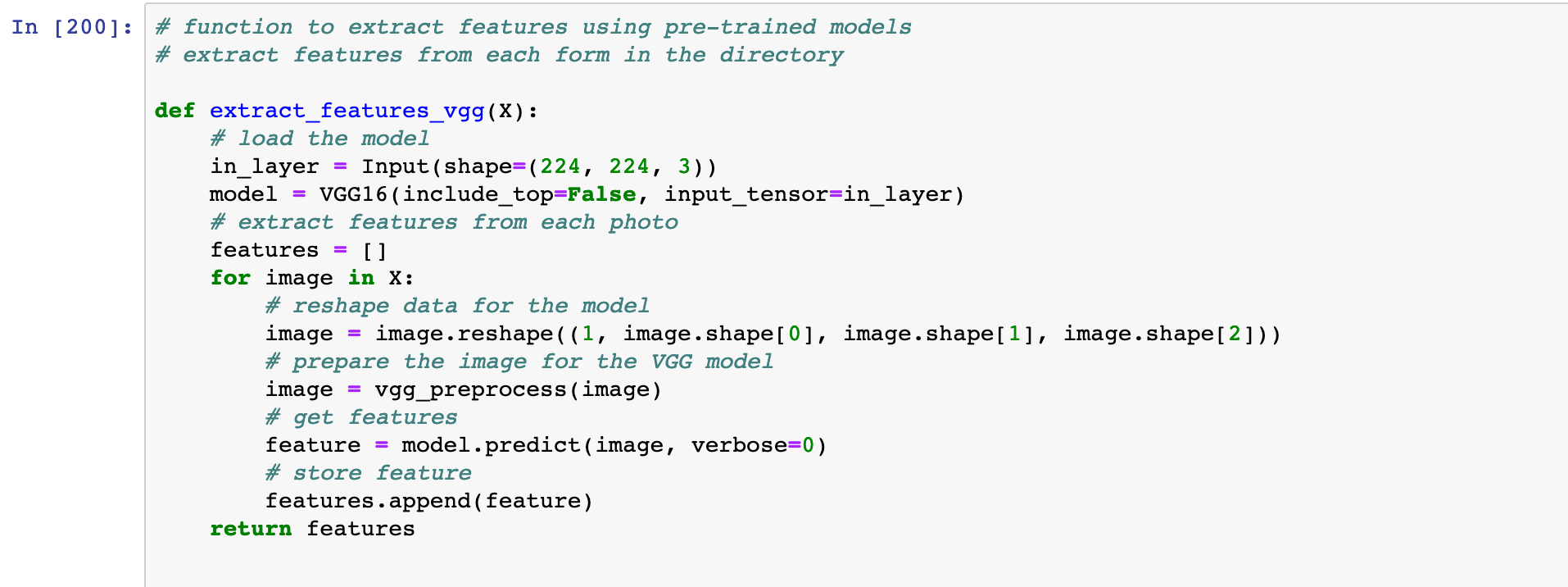
Transfer learning is a technique that makes use of the knowledge gained while solving one problem and applying it to a different but related problem. We use transfer learning for our image classification task. The VGG16 and ResNet50 models are pre-trained for image classification tasks on more than 1,000,000 images for 1,000 categories. These models can be used as the basis for transfer learning in computer vision applications. These models have learned how to detect generic features from the given images and achieved state of the art performance and remain effective on the specific image recognition task for which they were developed. These models can be used as a classifier and as a feature extractor. Since we already mentioned that our dataset is very small for these complex models, we decide to use these models as feature extractors instead of a classifier.

**VGG16 and ResNet50 as Feature Extractor**

Transfer learning speeds up training by enabling us to reuse existing pre-trained image classification models, only retraining the top layer of the network that determines the classes an image can belong to [2]. The features extracted from the pre-trained model is integrated into a new model, but the layers of the pre-trained model are frozen during training. We build a new classification model that uses the extracted features as input and gives the probabilities of 4 classes.

To extract the feature, we create a function for both these pre-trained models that extract the features. (refer image 2.5 and 2.6) The default input shape for Keras API to use any pre-trained model is (224,224,3) and we need to provide the input shape to the input layer. When loading a given model, the “include top” argument can be set to False, in which case the fully-connected output layers of the model used to make predictions is not loaded, allowing a new output layer to be added and trained.

We performed this for all the images in and extracted the important features of all images and saved them in an array so that we can use them as input to our new classification model. We may predict the probabilities using the same model, but it may lead to overfitting as this is a very small dataset. So, we build and use our new classification model.

 Image 2.5: Feature extractor for VGG16

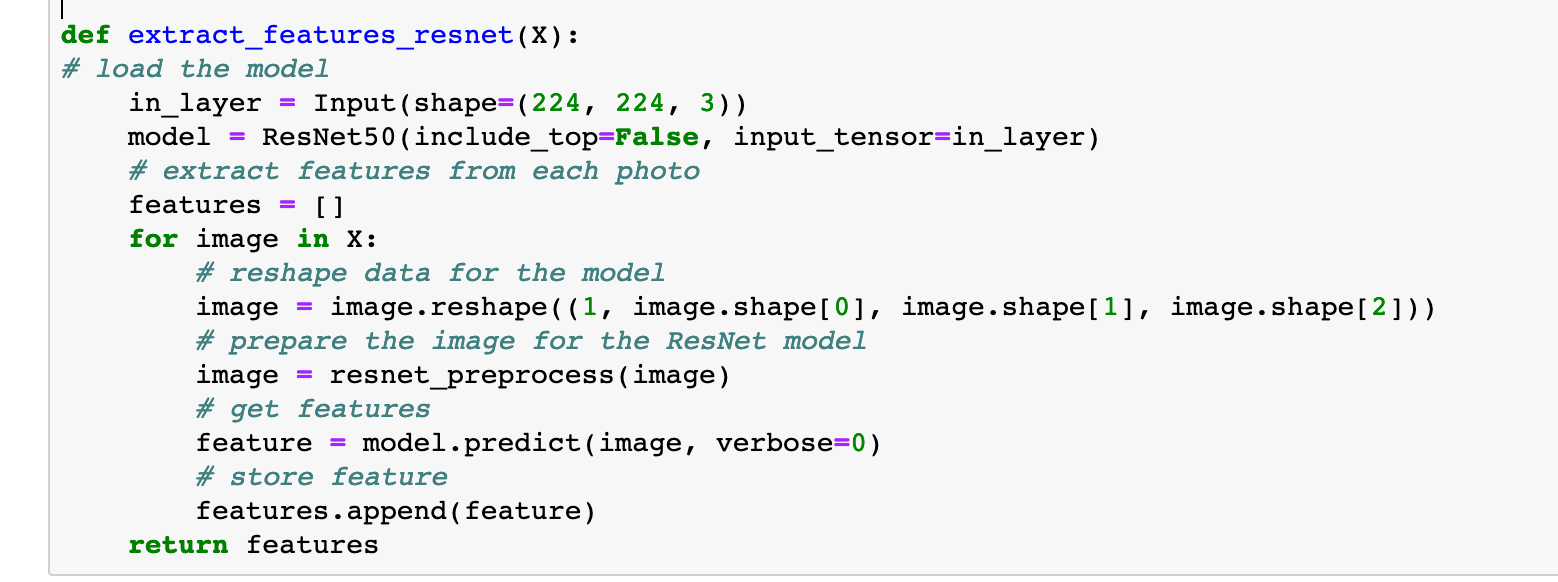
****

Image 2.6: Feature extractor for ResNet50

**Build a New Model for classification**

We used a new model with a dense fully connected layer and an output layer with 4 nodes and ‘SoftMax’ to give probabilities for every example image. We used the “Adam” optimizer with a learning rate of 0.001.

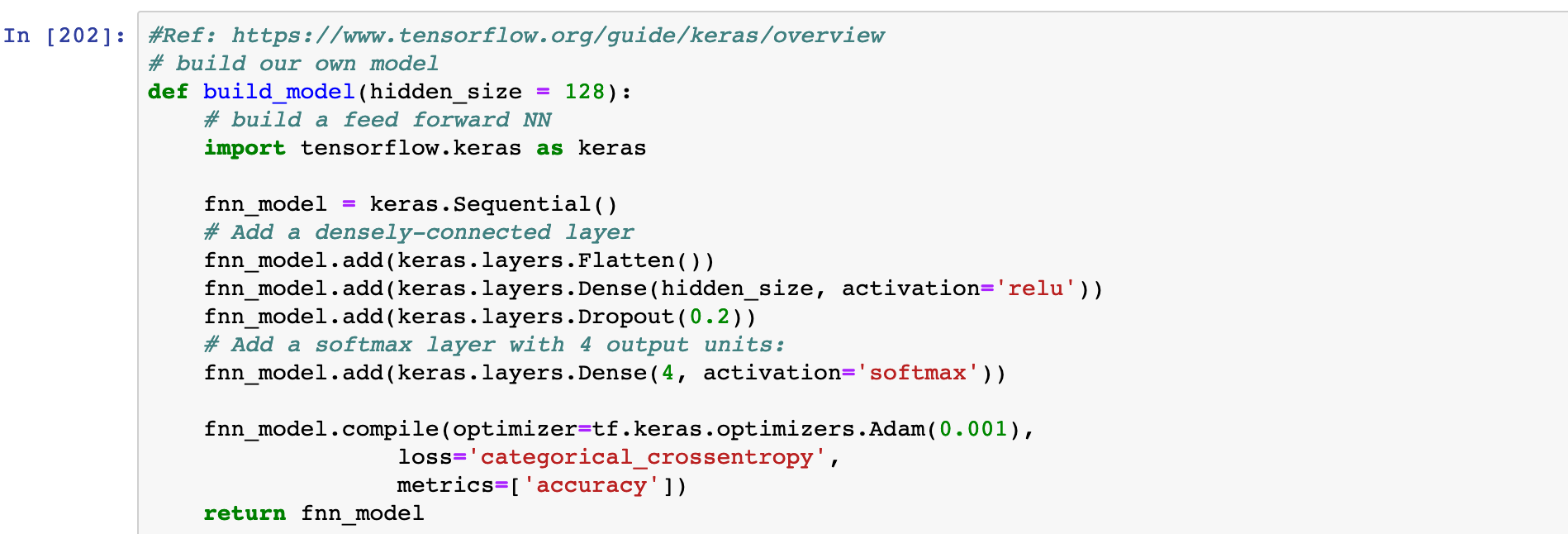
****

Image 2.7: A new model to integrate the features

**Classification images into different categories from extracted features**

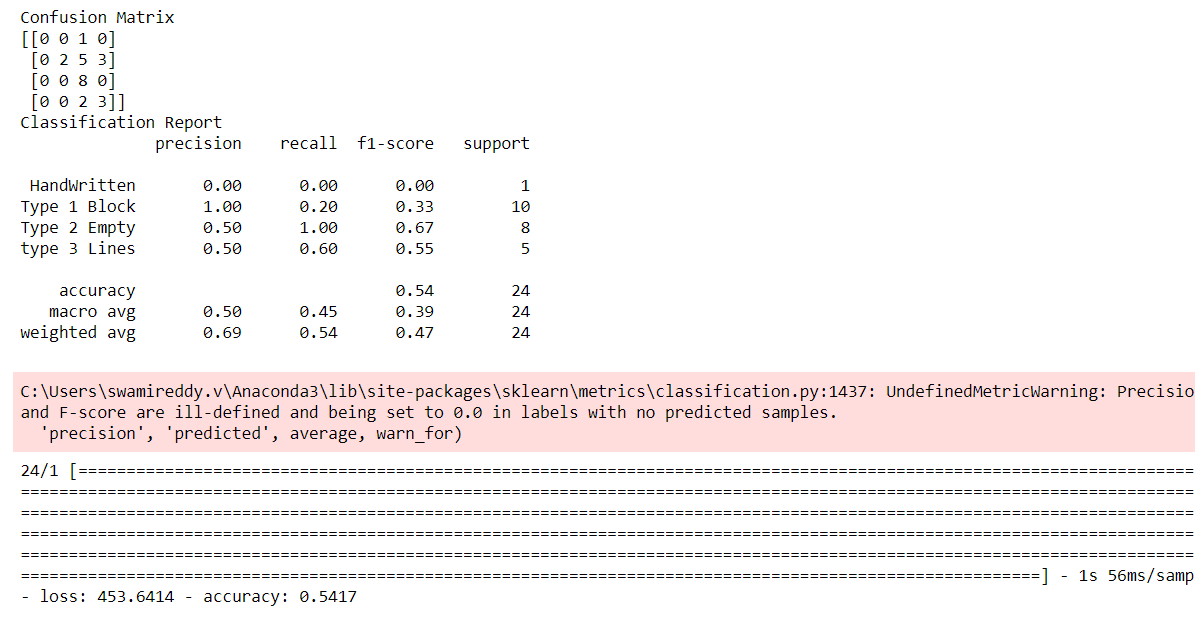
We extracted the features using Feature extractor function which extracts the outputs of the last max-pooling layer of the VGG or ResNet50 models. Now we pass these features as input to our classification model and get the predicted probabilities for 4 classes. We fit the model on train data and evaluate the performance on test data.

**Evaluation of Classification Models:**

**Feedforward neural network:**

This is the simple model with 4 layers to classify the images. The accuracy of this model is about 87%. So, there is a chance of misclassification with simpler models. We can manually increase the layers, but it would have a chance of overfitting the model and it miss-classifies when a new form is inputted to the model.

Accuracy, recall score, F1 score, and other metrics are shown in the below picture for the Feed Forward Neural Network model

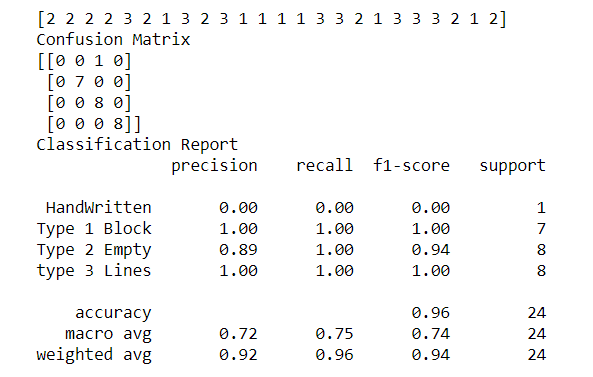


**VGG Modeling:**

As we know that VGG 16 is a pre-trained model we have 16 layers and the top layer is not included in the model and extract the features and input it to our softmax layer to overcome the overfitting issue. This VGG model in a comprehensive model and is trained on thousands of images which leads it to overfit on small data sets

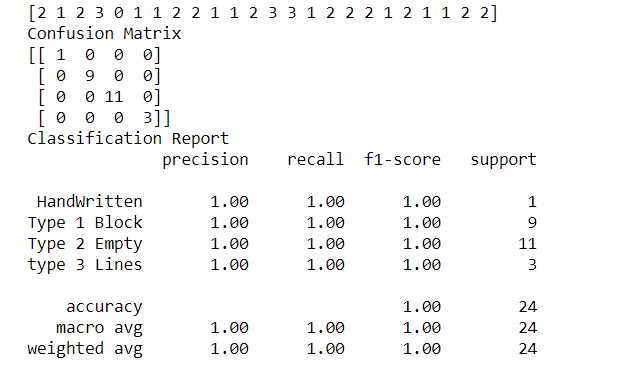
**Test set I:**

In the test set, we have images that are randomly selected for training and testing purposes. When we check the accuracy for the test set it comes out around 96% that is 4% image is miss-classified in this case. Below are the recall score, Accuracy, F1-score and other metrics for the VGG 16 classification model on Test set I.



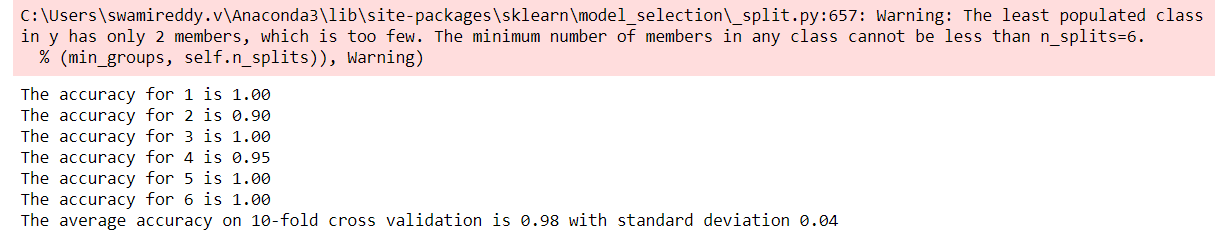
**The test set II:**

We are testing the same model on the second set of test data to see if the model is overfitting. This time we are having an accuracy of 100% which tells that the model is overfitting due to less count of observations. So, in the future, if a new form is given there is a high chance of overfitting as this performs perfectly on known images and fails on unseen images. Below are the evaluation metrics for the VGG 16 model test data.



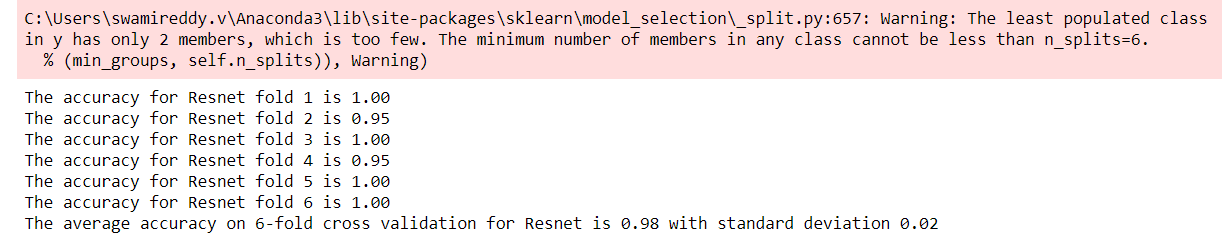
**VGG 16 model with K-fold validation:**

To come over the issue of overfitting we are coming with the K-fold validation. In this, we have the 6-fold which means we have 6 different random picks for each fold and the accuracy is calculated for each test output. Once we have the six different accuracies the average of them is the final accuracy of the model. We can trust this unlike the previous model as we have an accuracy of 98% for six different sets of trains and tests. Below are the accuracies for each fold and the average of them for overall accuracy.



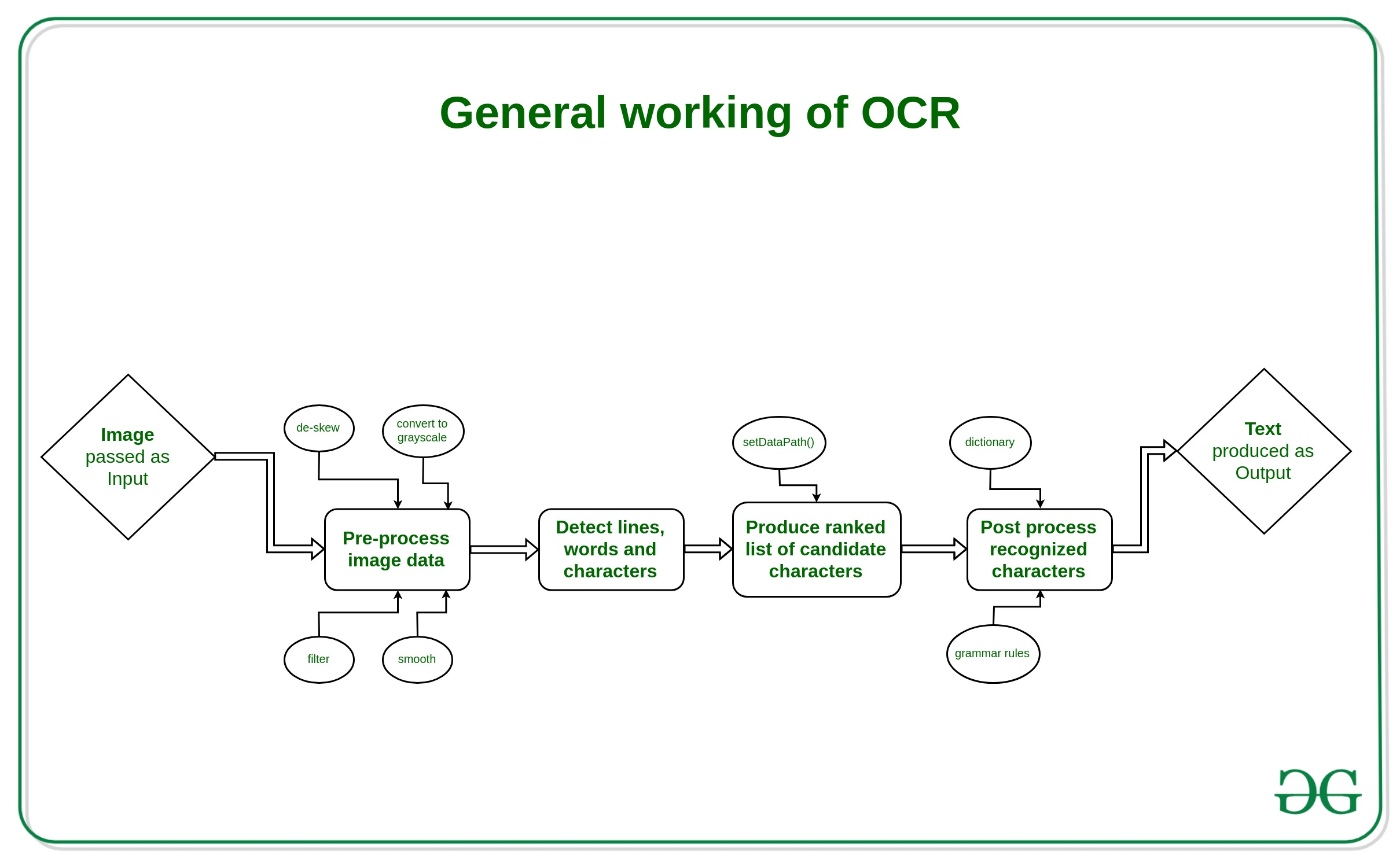
**ResNet 50 Model:**

As explained in modeling ResNet 50 has 50 layers and is trained over 1 million images. So, we are using it to extract features and build our final layer for classifying 4 categories. We are training our model on train data and trying it on test data. We are having an accuracy of 98% which is similar to the VGG 16 K-fold model. We conclude that the VGG 16 and ResNet 50 models are suitable models for classification of the Forms. Below are our evaluation metrics for the ResNet 50 Model.



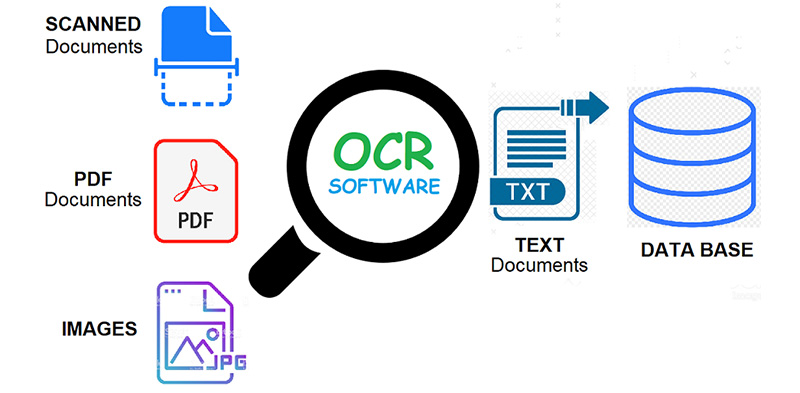
**Text Extraction using Tesseract:**

Tesseract is a method of extracting text from the images and was developed as proprietary software by Hewlett Packard Labs. But Tesseract became a more robust model in the 3.X version and was added to many programming languages. This is based on the traditional computer vision algorithms. We can install the Tesseract package using a handy command-line tool called "tesseract".

[**https://media.geeksforgeeks.org/wp-content/uploads/20190530190230/General-working-of-OCR-3.jpg**](https://media.geeksforgeeks.org/wp-content/uploads/20190530190230/General-working-of-OCR-3.jpg)

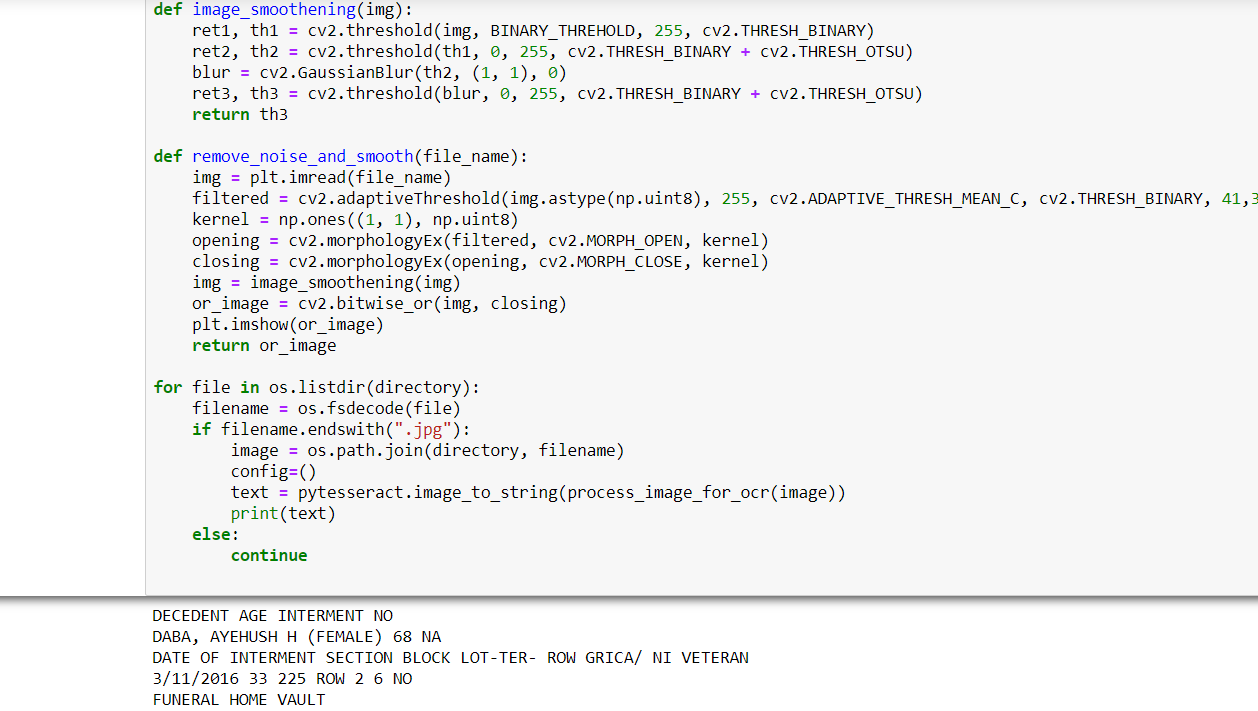
**Optical Character Recognition (OCR):**

In the process flow, we first input the image to the model and we then extract features. Then we pre-process the images for having the accurate text. In this, we first crop the images as we only need 20% of the image and remove the remaining as this may lead to producing some noise in our extracted text. We then detect the lines, characters, words, and phrases tag them. We smoothen the images so that the image is clear for the model to extract the correct text. Then the Post process of the recognized characters and text is done then the text from the images are given as output.



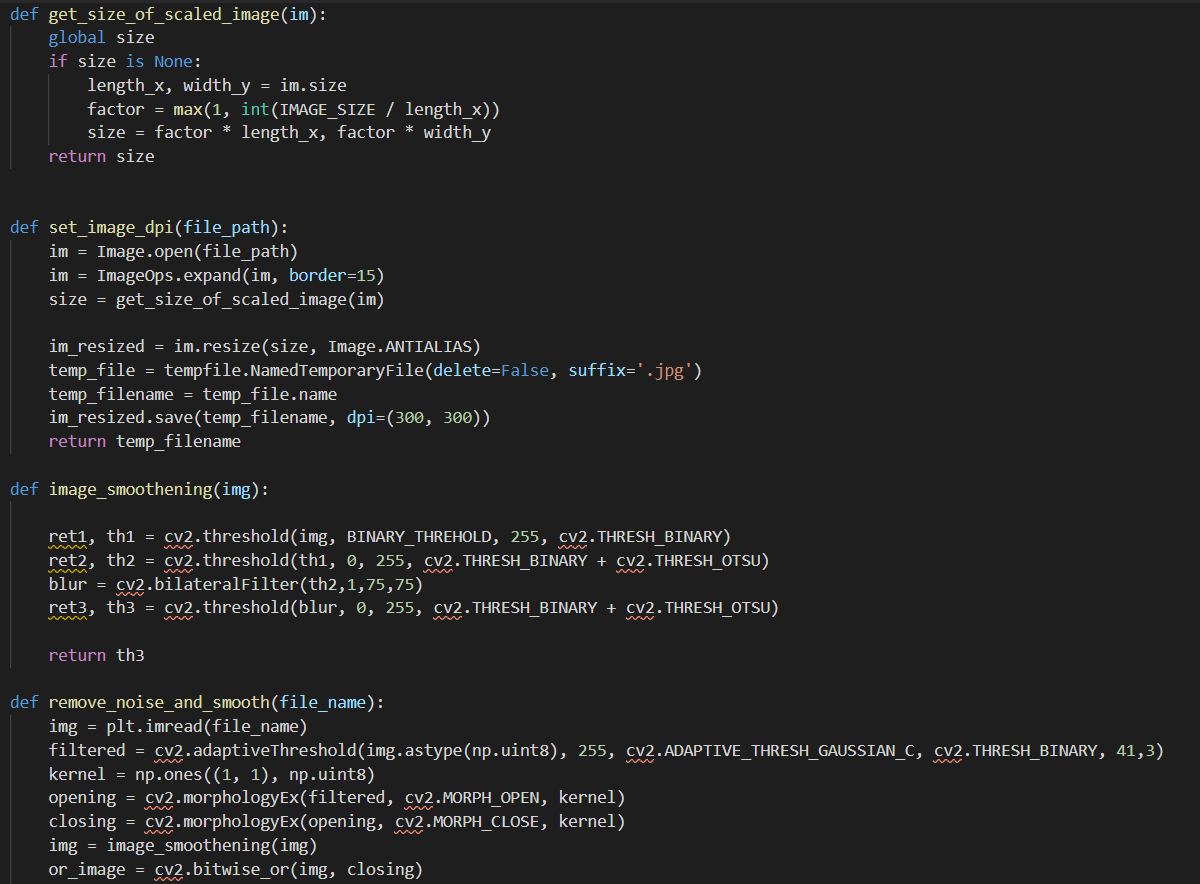
[**https://www.funavry.com/portfolio/optical-character-recognition/**](https://www.funavry.com/portfolio/optical-character-recognition/)

****

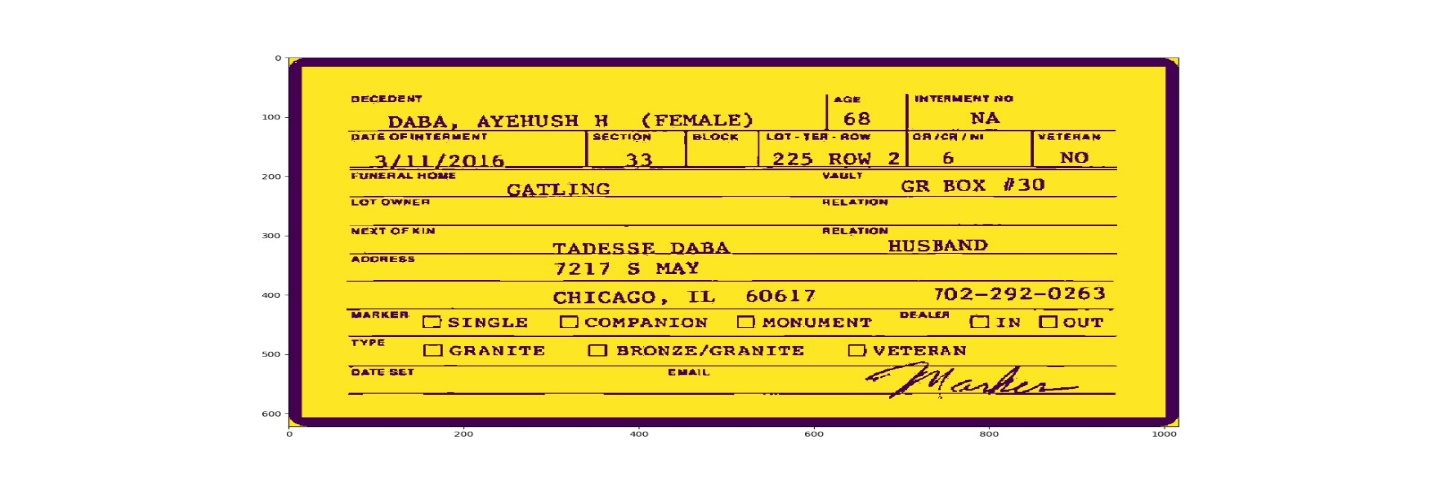
****

**Image Enhancement:**

Image enhancement is a technique in which we can improve the image by using various operations. In this we have made four functions for getting better image. Scaling, Image dpi, smoothening and noise removal is done by creating different functions.



We have used bilateral filter for removing noise in the images because it has higher precision than rest filters. Image smoothening function enhances the image by setting the threshold value. Scaling function scales down the image according to our desire. DPI function is also used to get a better view of the image.



**Google Vision API:**

Google Cloud provides two computer vision apps that help you recognize the photos with industry-leading prediction precision, utilizing machine learning.

* Auto ML Vision.
* Vision API.

**Vision API**

The Vision API of Google Cloud offers strong pre-trained machine learning models across the REST and RPC APIs. Assign labels to photographs and rate them easily into millions of predefined categories. Detect items and expressions, interpret typed and handwritten text, and create useful metadata through your database of pictures.

**It's benefits:**

#### **Detect objects automatically**

Detect and identify several objects including where every object is positioned inside the image. Using Vision API and AutoML Vision to learn more about target recognition.

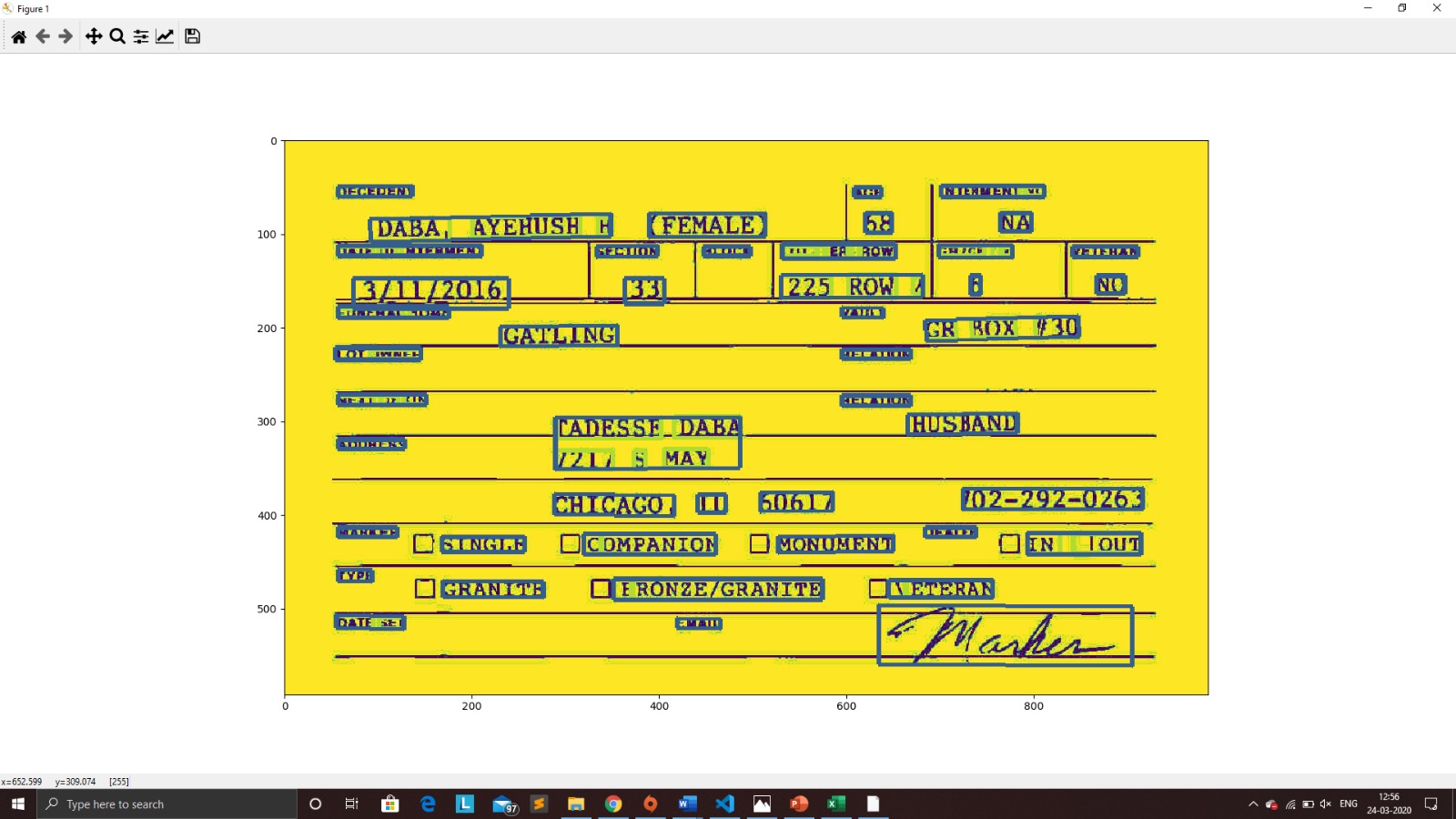
**Document Classifier and Text extractor Application**

We also developed a UI for classifying the documents and extracting the text from this user interface, making two different buttons one for classifying and the other for extracting the text from the pictures.

#### **Understand text and act on it**

Vision API utilizes OCR to identify text in more than 50 languages and various styles of files inside the images. It's all part of Data Knowing AI, helping you to easily scan millions of records and simplify company workflows.

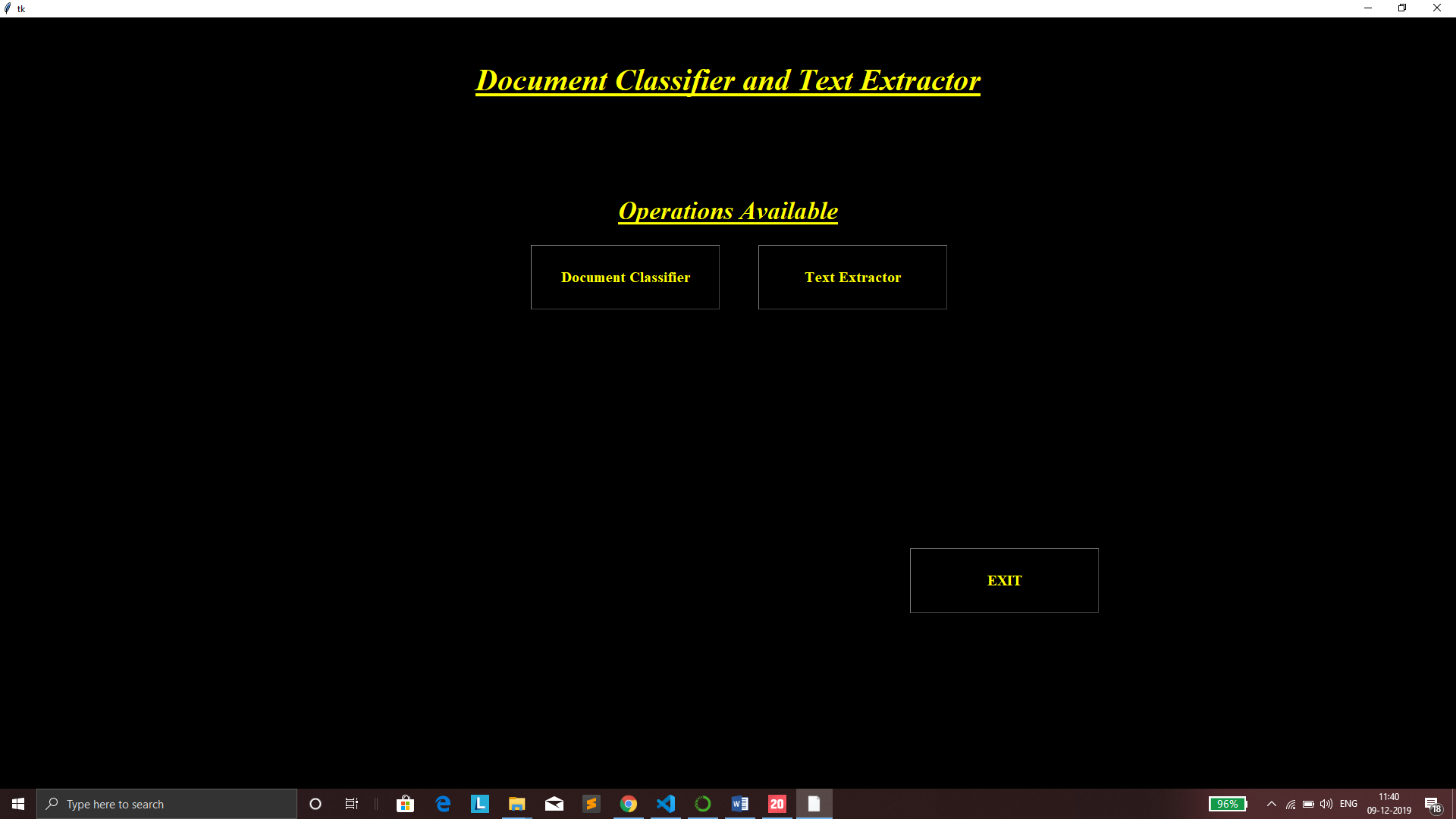
We have used Google Vision API to detect and get Bounding boxes for each element in an image.



With the help of Google Vision API, we are extracting text by getting bounding boxes surrounded for each element.

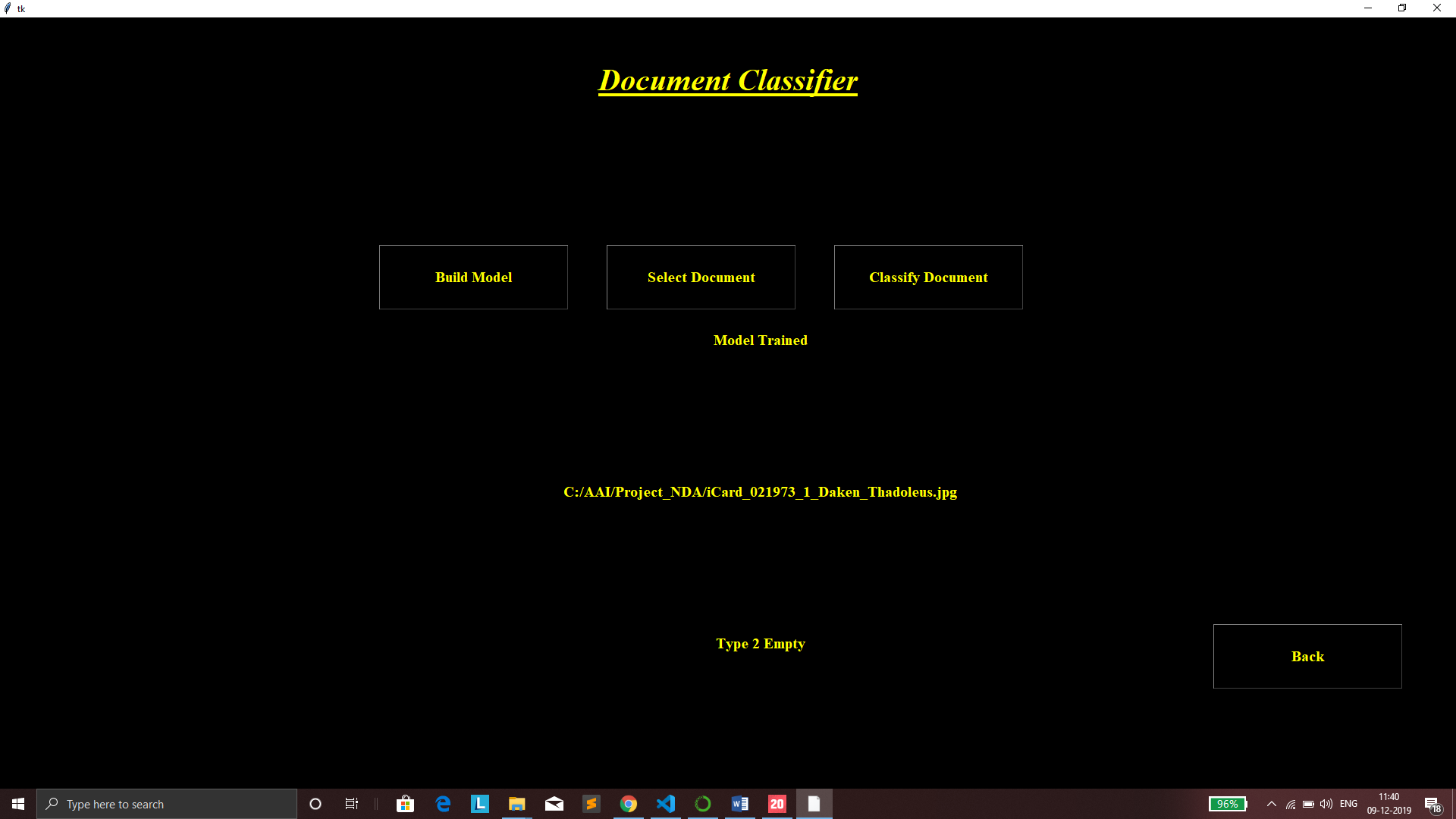
The bounding box is a rectangular box that can be determined by the xx and yy axis coordinates in the upper-left corner and the xx and yy axis coordinates in the lower-right corner of the rectangle.

**Document Classifier and Text Extractor:**

****

**Document Classifier:**

In the document classifier tab, we have an option to build the model first and then we need to select the document which we need to classify. After selecting the document, using the classifier which we build using the VGG and resnet we can classify the images.

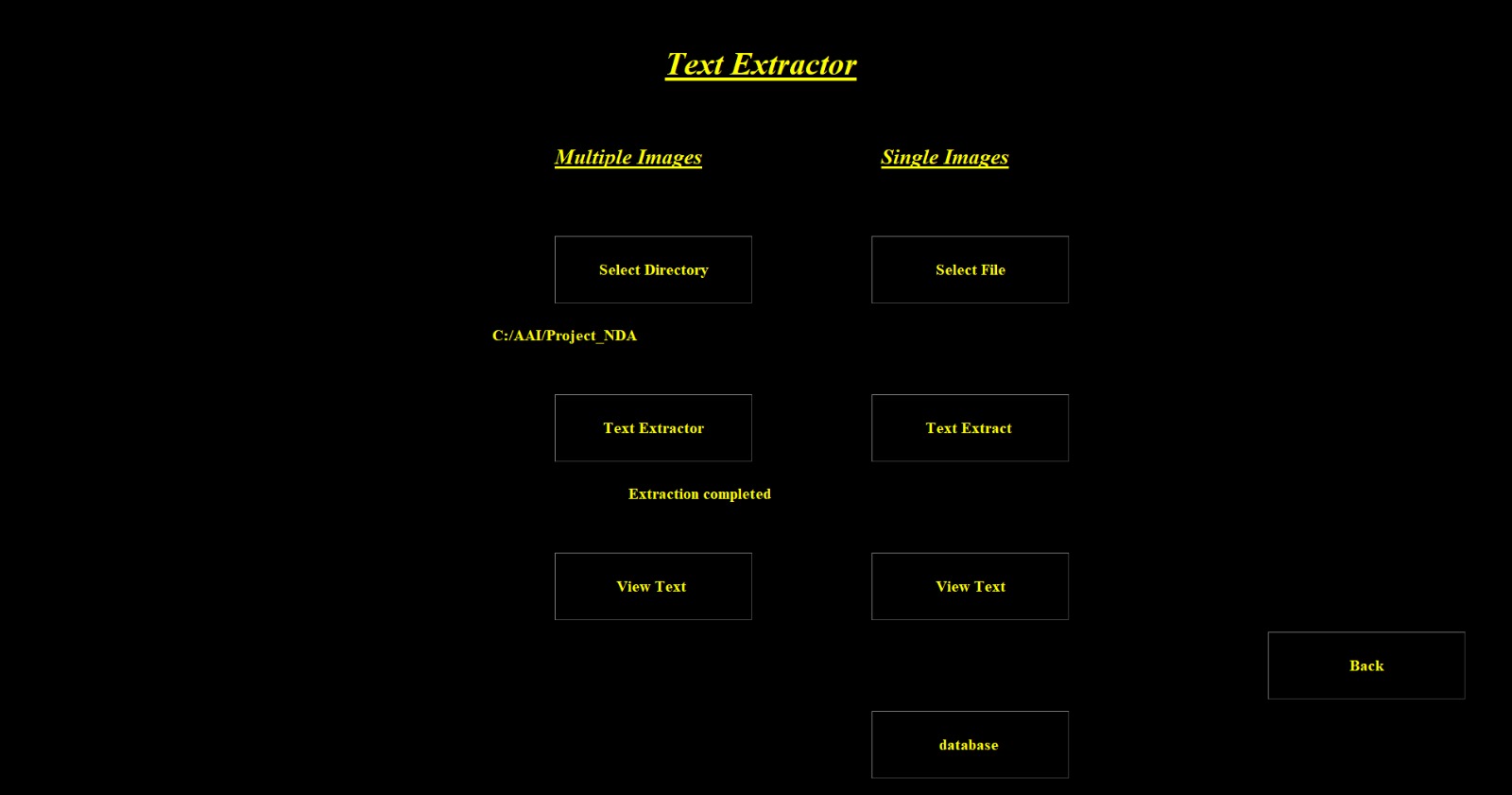
****

Here, in text extractor tab we can extract the text for multiple images or else for the single image, depending upon the requirements from the user. We need to select file of the from the given button and then from the text extract button we are able to extract the data from the image using tesseract and opencv. And later we have saved these data into a csv file using the database button.

**Text Extractor:**

In the text extractor, as mentioned we are using Google vision API, to get the bounding boxes around the texts.

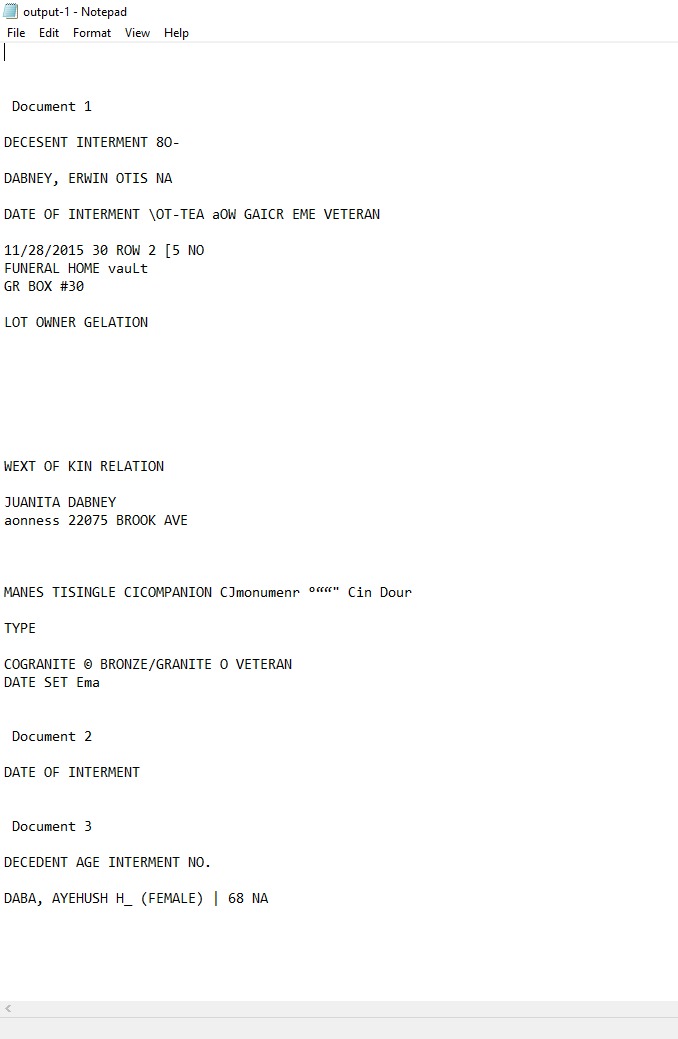
Inside the Text Extractor tab, you get a chance to extract texts from multiple images or for single image.



**Multiple Images:**

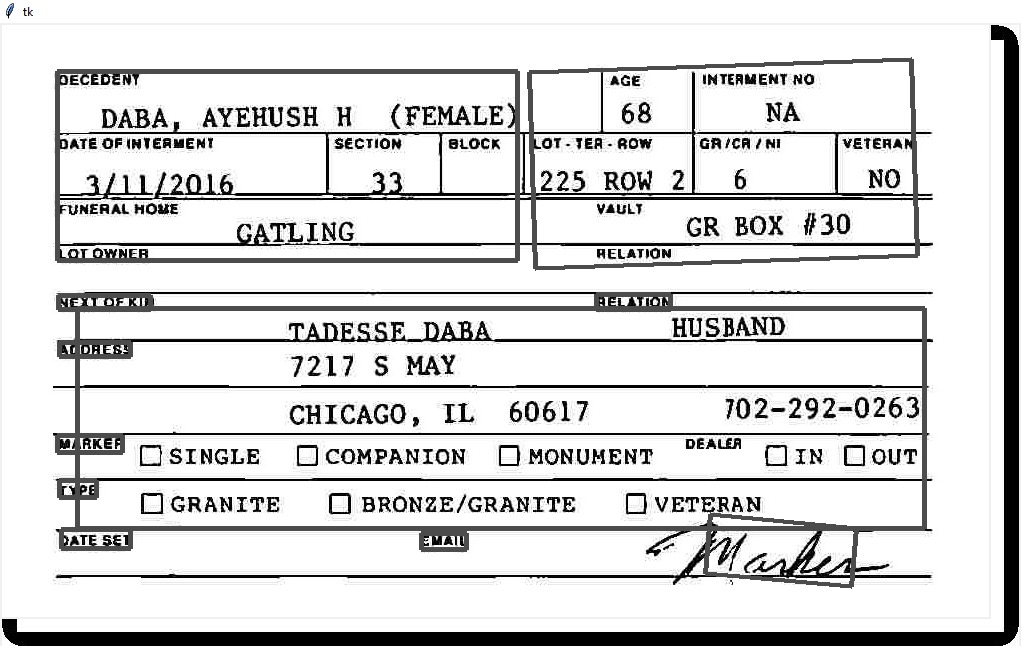
By pressing Multiple Image tab, you are providing file in which images has been stored and then by pressing text extractor, we are processing this images through google vision, when the extraction is completed it will give you a pop up as “ Extraction Completed”. To view text, you need to press view text tab.

**View Text output:**



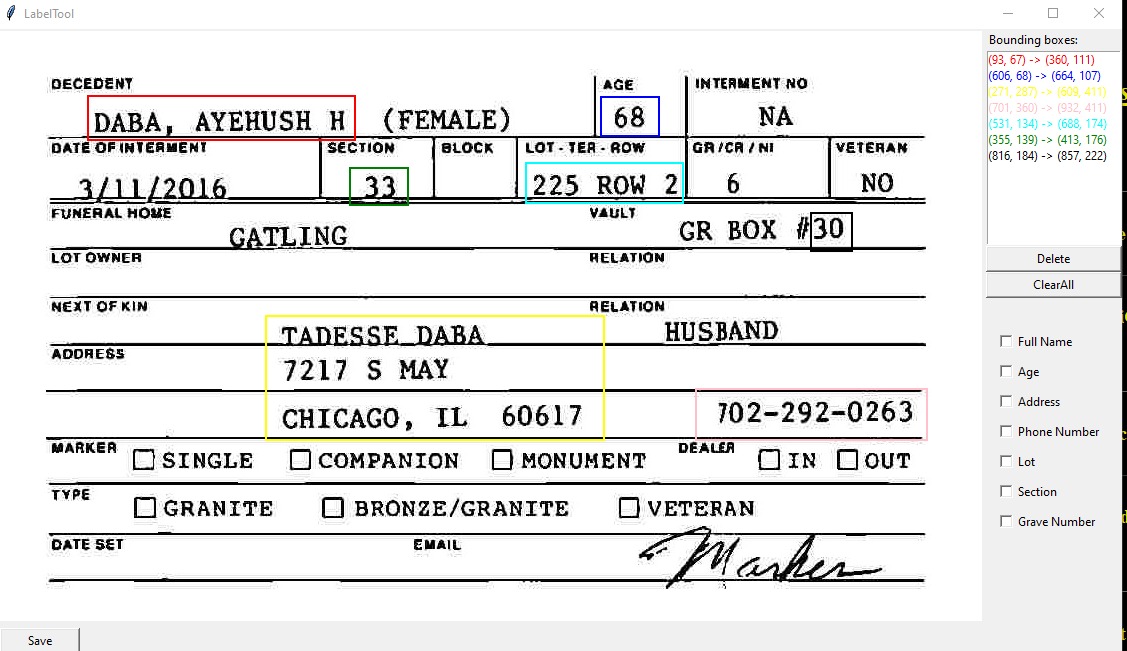
**For Single Image:**

Likewise, for single image, we created the same process, you select the image and when the extraction is completed, you will get pop up as, “Extraction completed”.



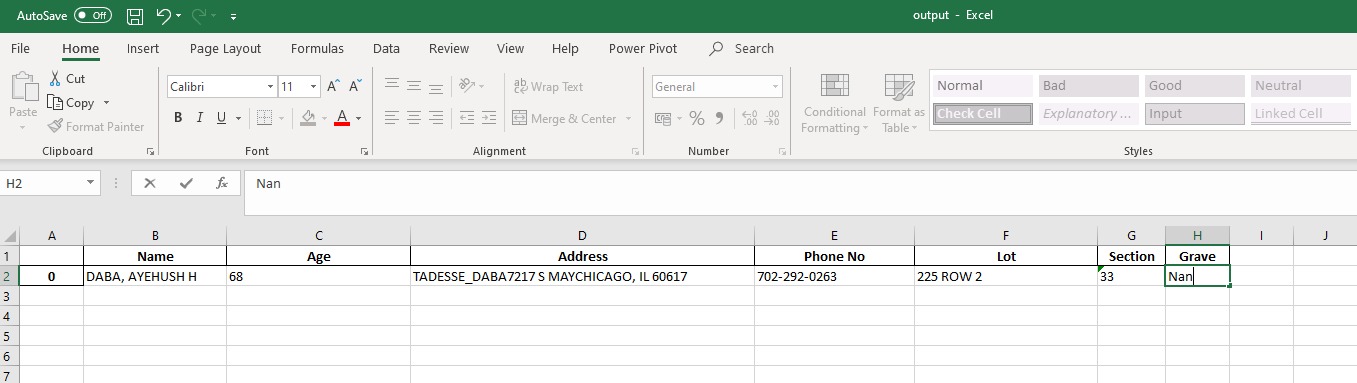
Now, in View Text, we are giving user a freedom to extract the information per their needs,

But user must draw boxes in around the text in order of textboxes given in the image.



Here, by ticking the boxes given at the right side of image, we are extracting that much texts and appending them in an excel sheet.

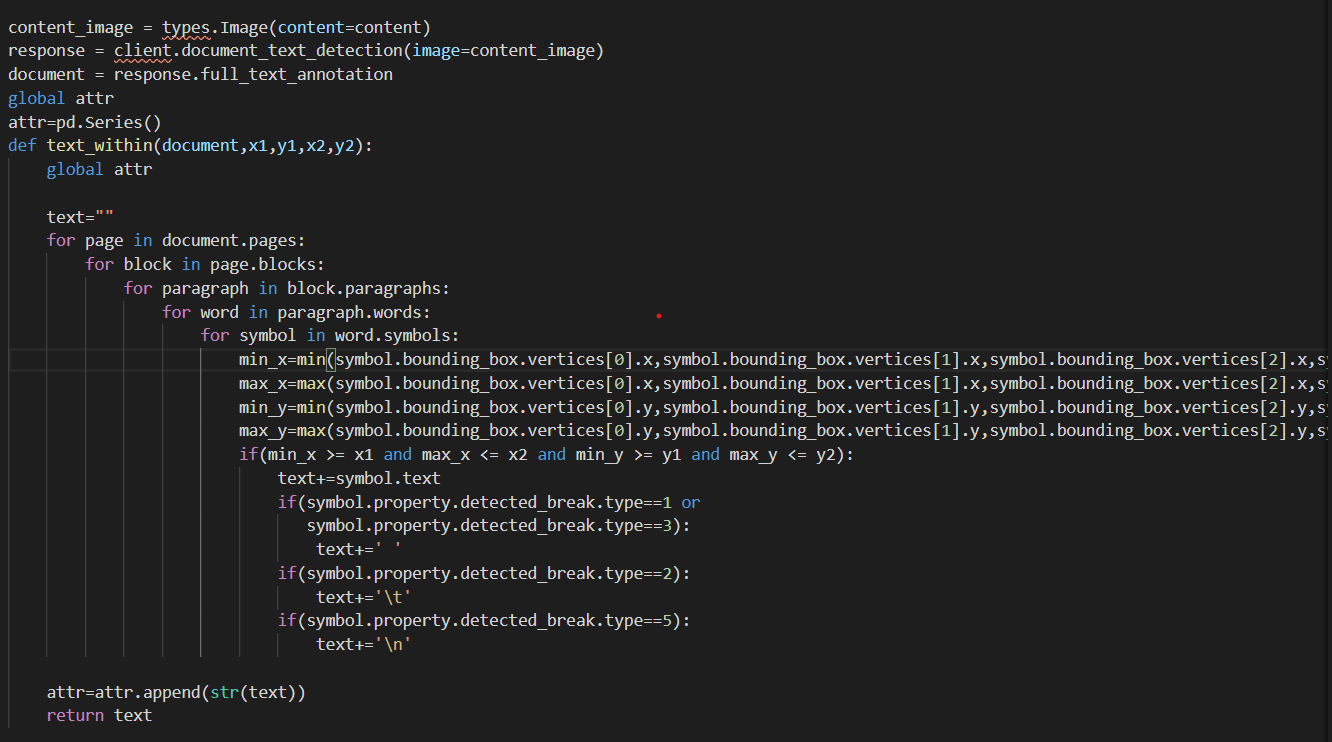
By clicking Data base Tab, you will get an excel sheet, which will have attributes selected by a user:



**Algorithm - How we are extracting with the help of Bounding boxes:**

In the entire process, we used tesseract on batch file and google vision API on single images, by taking multiple tests, we came to conclusion that google vision API is working better than tesseract and so we proceeded with google vision API.

First, we are getting the coordinates given by the google vision and we are passing those coordinates in the function, which takes the coordinates and returns the word**.**

****

**Conclusions:**

Using simpler models like FNN there is a chance of miss-classification with Handwritten forms. That is the reason we are using more robust models like VGG 16 AND ResNet 50 to classify our forms. Both the models VGG 16 and ResNet 50 models are performing well in classifying forms either of them can be considered for future classification of new models. They both have an accuracy of 98%. Major information was extracted from the images using tesseract, opencv and Google Vision API. We used tesseract and Google vision API and came to know after testing that Google Vision API is pretty much accurate than tesseract and we have extracted feature successfully and appended in the csv file.

**References:**

1. Brownlee, J. (2019, August 7). How to Prepare a Photo Caption Dataset for Training a Deep Learning Model. Retrieved from <https://machinelearningmastery.com/prepare-photo-caption-dataset-training-deep-learning-model/>.
2. Radhakrishnan, P. (2019, October 26). What is Transfer Learning? Retrieved from <https://towardsdatascience.com/what-is-transfer-learning-8b1a0fa42b4>.
3. Brownlee, J. (2019, September 2). Transfer Learning in Keras with Computer Vision Models. Retrieved from <https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>.
4. Vaibhaw,S. (June 6, 2018). Deep Learning based Text Recognition(OCR) using Tesseract and OpenCV. <https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/>
5. <https://www.tensorflow.org/tutorials/images/transfer_learning>
6. Adrian, R., (2018). OpenCV OCR and text recognition with Tesseract
7. Gidi. S., (2019). Introduction to OCR with tesseract and OpenCV