MIT Academy of Engineering

School of Computer Engineering

TY Project Phase 2 (Review 1) 2021-22

Online Examination

Guide:

Mrs. Diptee Ghusse ma’am

Students Name:

Tushar Gavkhare (SCET57)

Tanisha Nazare (SCET59)

Sharwari Ambade (SCET65)

Yeeshant Dahikar (SCET73)

Index

Problem statement

- Online examination Portal using Automated sign recognition and Devanagari handwritten checking using OCR.

Objectives to be achieved

- Build a platform which will contain both the features: Online examination and handwritten devanagari paper checking.
- Online examination: Online examination using sign recognition in order to decrease cheating and increase performance (next question, previous question, MCQ selection, etc.)
- Handwritten devanagari paper checking: Devanagari handwritten character recognition for paper checking using OCR.

Algorithms

- **Bounding Box Algorithm:** A bounding box in image processing is an imaginary rectangle that serves as a point of reference for object detection and creates a collision box for that object.
- **Image Cropping Algorithm:** Cropping is one of the important operations of the image processing to remove unwanted portions of an image as well as to add required features to an image.
- **Sort Contours Algorithm:** Sort contours from left-to-right and top-to-bottom or right-to-left and bottom-to-top.
- **PDF to JPG Algorithm:** Convert the pdf to jpg for image processing.
- **Folder to Zip Algorithm:** Convert the folder of jpg to zip file.

Algorithms

- **HOG:** HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data.
- **Gabor Filter:** Features are extracted directly from grayscale character images by Gabor filters which are specially designed from statistical information of character structures.
- **LBP:** LBP, OR Local Binary Pattern, encode local texture information, which you can use for tasks such as classification, detection, and recognition.

Libraries

- **pypdf2**: It is a python library used for performing major tasks on PDF files such as extracting the document-specific information, encrypting and decrypting the PDF files, etc.
- **glob**: glob (short for global) is used to return all file paths that match a specific pattern.
- **os:** The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.
- **shutil:** Shutil module offers high-level operation on a file like a copy, create, and remote operation on the file.

Libraries

- **pymupdf:** PyMuPDF is a Python binding for MuPDF – a lightweight PDF viewer, renderer, and toolkit, which is maintained and developed by Artifex Software, Inc. MuPDF can access files in PDF, XPS, OpenXPS, CBZ, EPUB and FB2 (e-books) formats.
- **fitz:**Version of Pymupdf
- **numpy:** NumPy is a Python library used for working with arrays.
- **imutils:** A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

Libraries

- **cv2:** It is an open-source library that can be used for image processing and performing computer vision tasks.
- **pillow:** Pillow is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving images.
  - Image:The Image module provides a class with the same name which is used to represent a PIL image
  - ImageFont:The ImageFont module defines a class with the same name. Instances of this class store bitmap fonts, and are used with the PIL.
  - ImageDraw:The ImageDraw module provides simple 2D graphics for Image

Techniques

KNN:

- Compute the Euclidean distance between the test data point and all the training data .
- Sort the calculated distances in ascending order .
- Get the k nearest neighbors by taking top k rows from sorted array
- Find the majority class of these rows .
- Return predicted class.
- Finding accuracy score to make sure the prediction is correct or not.

Techniques

SVM:

- A multi-class classifier where decision tree SVM (DTSVM) is used and in it, a problem of multi-class classification is decomposed into a series of ones of binary classification.
- Multiclass SVMs are usually implemented by combining several two class SVMs.
- Two approaches common in practice are one-versus-all method and one-versus-one method.
- The one versus-all method represents the earliest and most common multiclass approach used for SVMs.
- Each class is trained against the remaining N-1 classes that have been collected together. The character takes-all strategy is used for final decision.
- The character class corresponds to the class with highest output function.
- For one classification N binary classifiers are needed.

Techniques

ANN:

- We decide the number of layers and number of neurons in each layer for our ANN model with the number of neurons being proportional to the number of variables.
- We have an input node which is the image we give the model and the output node which is the character that the model recognizes.
- This models includes characteristics like when the input node is given an image it activates a unique set of neurons in the first layer starting a chain reaction that would pave a unique path to the output node.
- The activated neurons send signals to every connected neuron in the next layer. This directly affects which neurons are activated in the next layer.
- In the next layer, each neuron is governed by a rule on what combinations of received signals would activate the neuron and the ones who don’t receive right signals they remain grey.
- Steps are repeated for all remaining layers until we are left with the output node and then The output node deduces the correct char based on signals received from neurons in the layer directly preceding it.

Techniques

Ensemble Classifier:

- Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- A base model is created on each of these subsets.
- Each model is learned in parallel from each training set and independent of each other.
- The final predictions are determined by combining the predictions from all the models.

Models

1) **SVM -** The main objective of the SVM supervised algorithm to identify the arrangement of independent hyperplanes linearly concerning extracted features of single characters. The optimal hyperplane selected and arranged in such a way that separate feature space in different classes with maximum distance.One-vs-all is a heuristic method for using binary classification algorithms for multi-class classification.It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

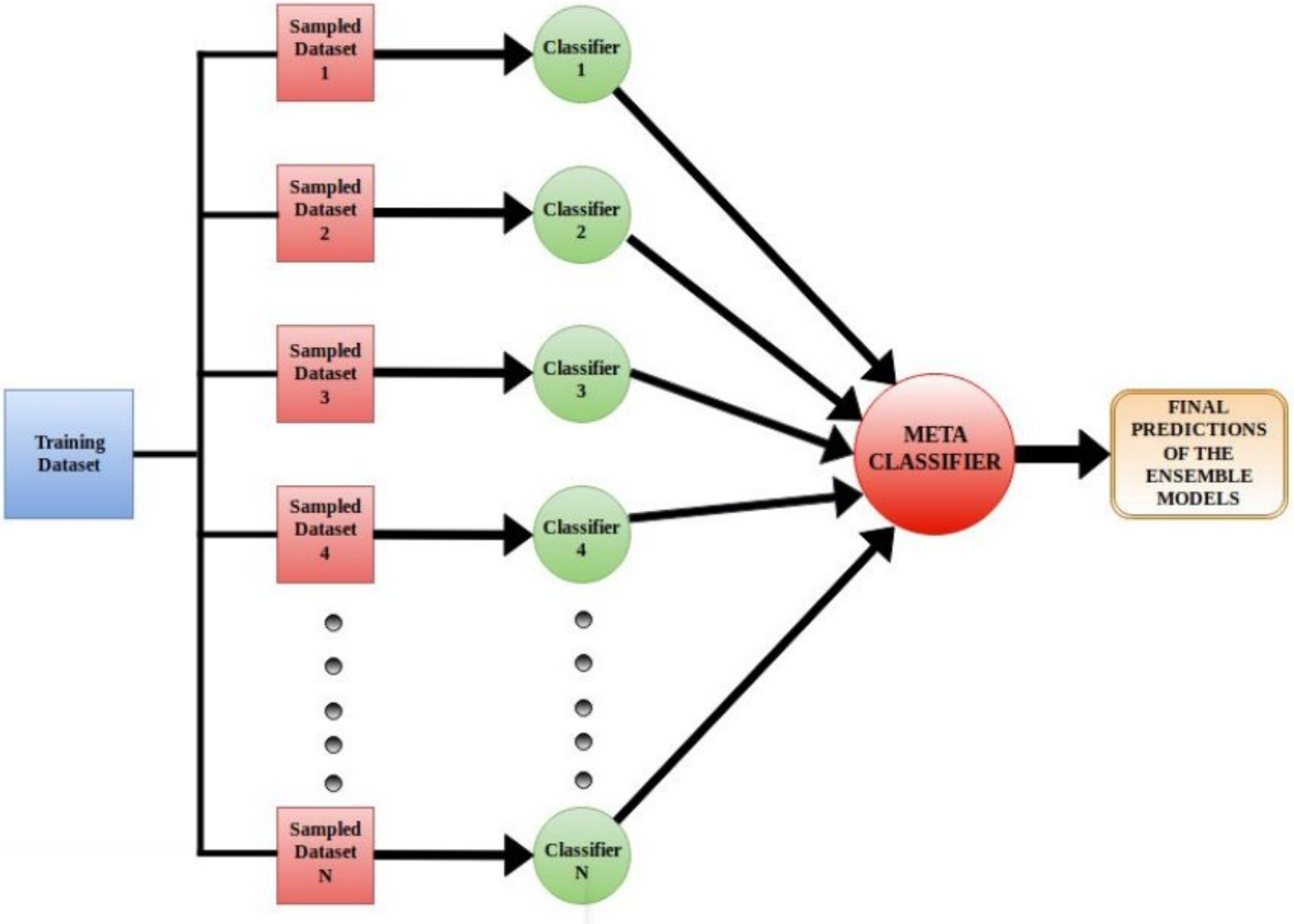
Models

2) **KNN** - K-nearest neighbor classifier is one of the first supervised classifier, non-linear and non-parametric classifier for performing the pattern classification task. The k nearest neighbor algorithm (KNN) used for classification in which the input data set contains the k nearest training examples in the feature space. A handwritten character classified by a majority choice of its neighbors to match the category of most familiar among its k nearest. The Euclidean distance function used in KNN for computing distance between neighbors.

Models

3) **ANN -** ANN is an adaptive system that modifies its structure based on information during the learning phase. The ANN translates handwritten images into pixels (a language it understands). Black pixels are given the value “0” and white pixels the value “1”. Each pixel in an image is called a variable.

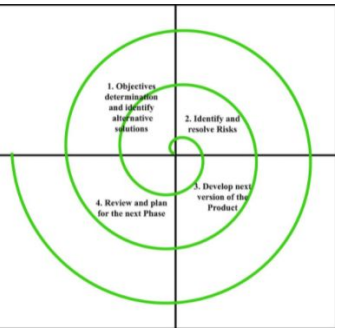
4) **Ensemble Classifier -** This technique combine various classifiers to enhance performance scores relative to an individual classifier. Instead of using a single classifier, this method combines several weak classifiers together to enhance the recognition precision.


Training Dataset
Sampled Dataset 1
Sampled Dataset 2
Sampled Dataset 3
Sampled Dataset 4
Sampled Dataset N
Classifier 1
Classifier 2
Classifier 3
Classifier 4
Classifier N
META CLASSIFIER
FINAL PREDICTIONS OF THE ENSEMBLE MODELS

Software Model

Spiral Model:

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.



Methodology

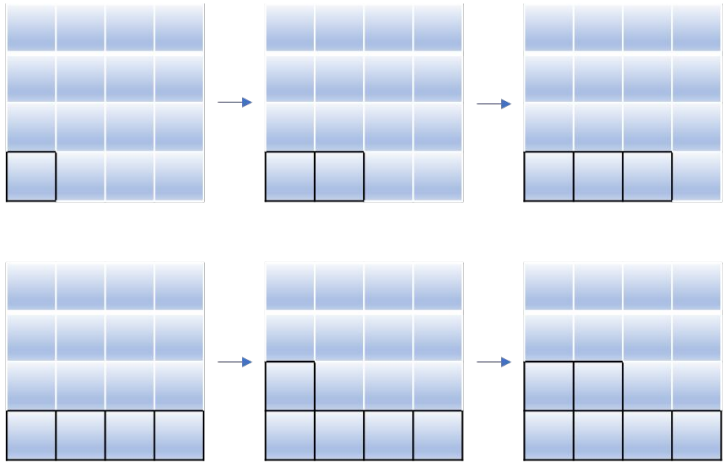
Main MODULE	PART	CRUX
Module1: OCR detection	1- PDF uploading	Colab/Jupyter upload, PDF paging, Aligning, Segmentation, Database collection
	2- PDF to text conversion	Segment to paragraph, paragraph to line, line to word, word to character recognition
	3- Answer key Comparison	Uploading answer key, Answer key conversion, Answer set creation, Rubrics, Grading
Module2: Hand recognition	1- Image Uploading	Image sequence, Arrangement, Filtering, Clustering
	2- Hand recognition	Segmentation, Hand recognition, Sign recognition, Movement recognition(Img_seq), Collection of database
	3- Task assigning	Task creation, Task selection, Task completion, Grading

Mathematical Understanding

1) To crop the image of Devanagari character :

```
for i in range(19):
  os.mkdir("/content/barakhadi/" + vyanjan[p][18-i])
  for j in range(12):
    for l in range(len(pdfs)):
      output[l] = pdf.PdfFileWriter()
      pages[l].cropBox.lowerLeft = (0 + 49*j, 0 + 42*i)
      pages[l].cropBox.upperRight = (49 + 49*j, 42 + 42*i)
      output[l].addPage(pages[l])
```

- Here we are cropping the devanagari characters in squared shape which are inside the boxes.
- The main objective is to make handwritten character dataset fast and easily
- We are going with path that is lower left box to upper right box
- We first crop lower row with every columns by traversing x axis and then second row (by incrementing j value means traversing y axis)



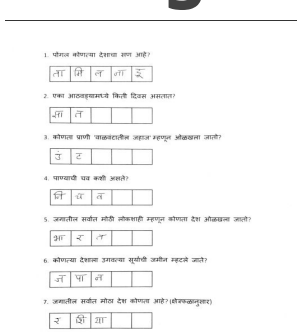
Mathematical Understanding

2)To detect boxes and crop :

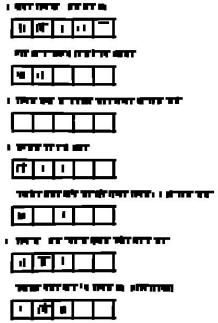
•To detect boxes we will use morphological operations.
•For that We will define rectangular kernel with the length based on the width of the image.
•We will define two kernels: 1) Kernel to detect horizontal lines. 2) Kernel to detect vertical lines.

```
# Morphological operation to detect verticle lines from an image
  img_temp1 = cv2.erode(img_bin, verticle_kernel, iterations=1)
  verticle_lines_img = cv2.dilate(img_temp1, verticle_kernel, iterations=1)
  cv2.imwrite("verticle_lines.jpg",verticle_lines_img)
# Morphological operation to detect horizontal lines from an image
  img_temp2 = cv2.erode(img_bin, hori_kernel, iterations=3)
  horizontal_lines_img = cv2.dilate(img_temp2, hori_kernel, iterations=3)
  cv2.imwrite("horizontal_lines.jpg",horizontal_lines_img)
```

•Morphology is a tool for extracting image components
•Mathematical morphology is set theory, it can apply directly to binary images.
•The usual operators (intersection, union) can be applied to them



After thresholding and inverting image



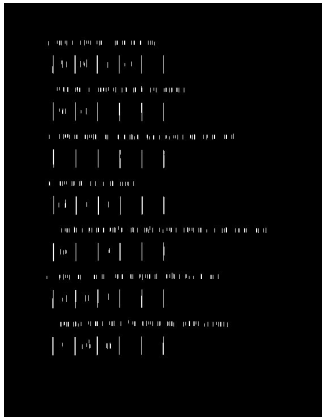


Image containing vertical lines

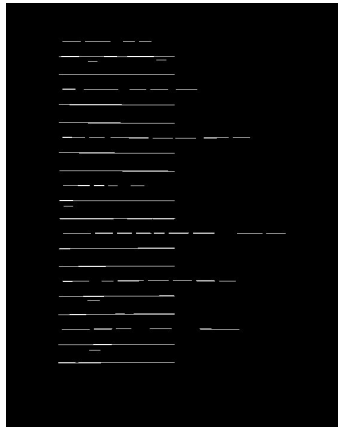
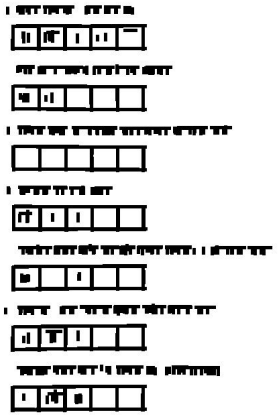


Image containing horizontal lines

How does addWeighted Function Work?

```
# Weighting parameters, this will decide the quantity of an image to be added to make a new image.
  alpha = 0.5
  beta = 1.0 - alpha
# This function helps to add two image with specific weight parameter to get a third image as summation of two image.
  img_final_bin = cv2.addWeighted(verticle_lines_img, alpha, horizontal_lines_img, beta, 0.0)
  img_final_bin = cv2.erode(~img_final_bin, kernel, iterations=2)
  (thresh, img_final_bin) = cv2.threshold(img_final_bin, 128, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
```

- By changing the alpha value, which will range from 0 to 1, we can easily transform from one image to another.
- The value of these weights ranges from 0 to 1, and then we can have the desired view of images as per our need.
- Here we have specified the alpha value as 0.5 and keeping the same value for the beta. The gamma value here is 0.
- As the alpha and beta values are the same, we can see the output as below, where we will see that the images are blended well

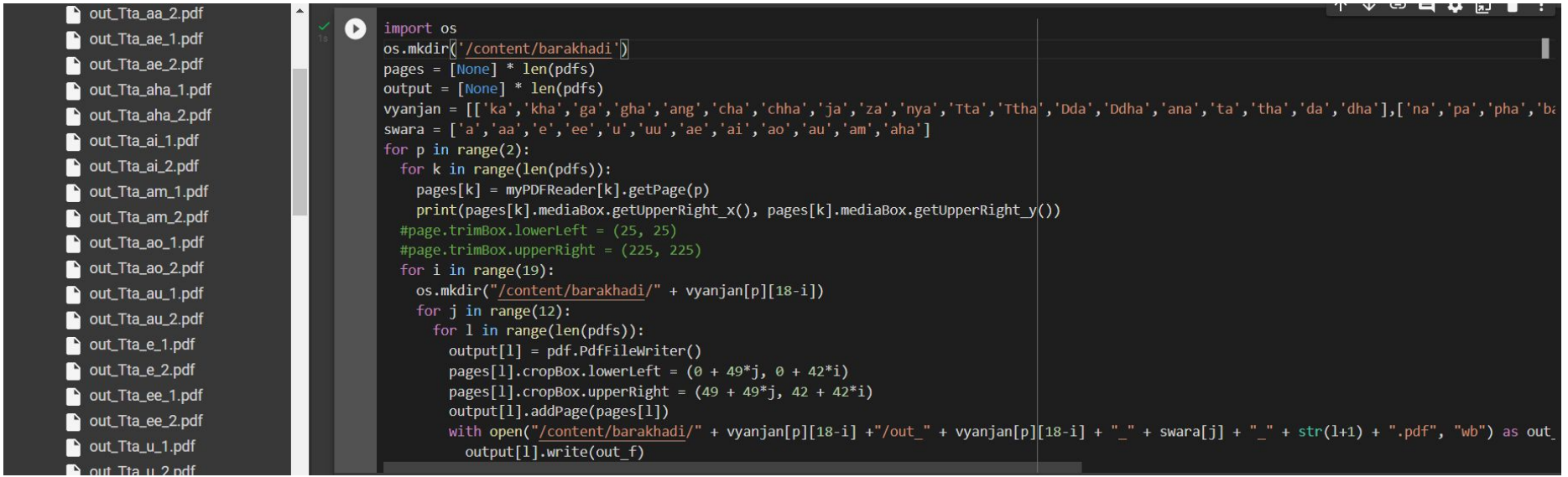


```
    if ((w > 20 and h > 25) and (w < 40 and h < 40)):
      #print("rectangle " + str(w) + " " + str(h) +" " + str(idx) + "c" + str(c))
      idx += 1
      new_img = img[y:y+h, x:x+w]
      xprev = x
      cv2.imwrite(cropped_dir_path + str(idx) + '.jpg', new_img)
      #encodeImage(cropped_dir_path + str(idx) + '.jpg',idx)
box_extraction("output_que.jpg", '/content/cropped/')
```

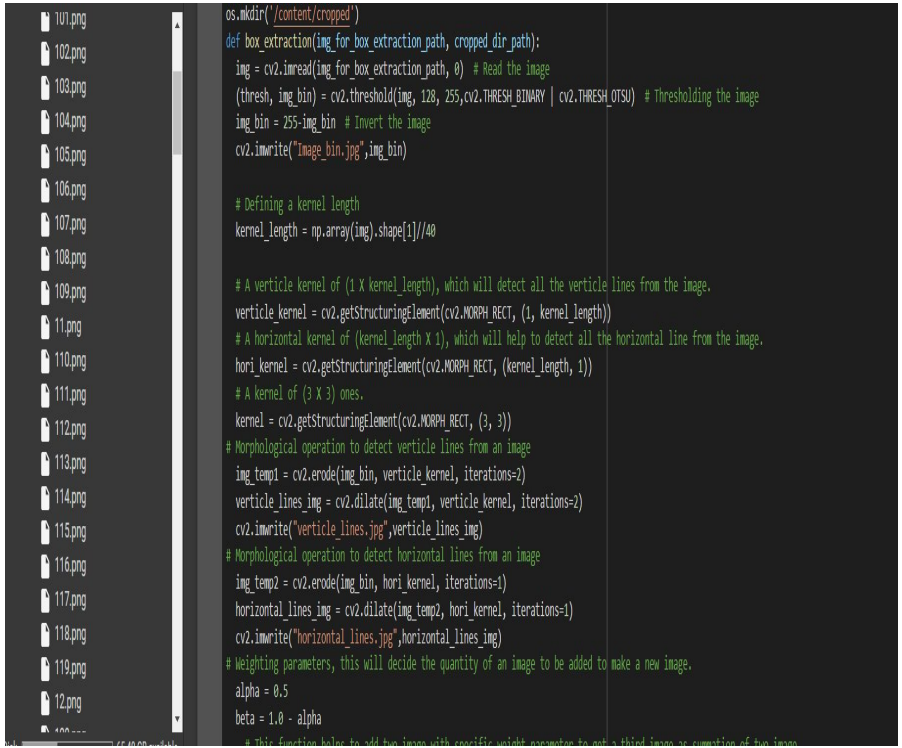
- We are finding the position of all the boxes and cuts the rectangular parts and saves them in a folder.
- If the processed image meets the conditions about width and height given in code then we will crop the box and and save in the folder.

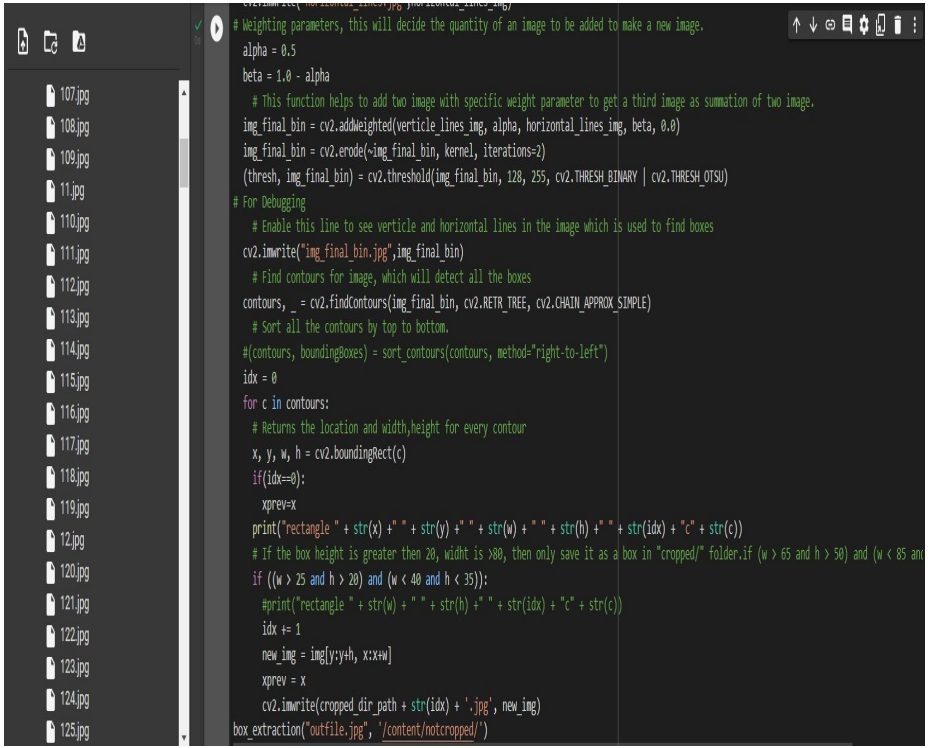
Extracted Images

Implementation: Coordinate Cropped character

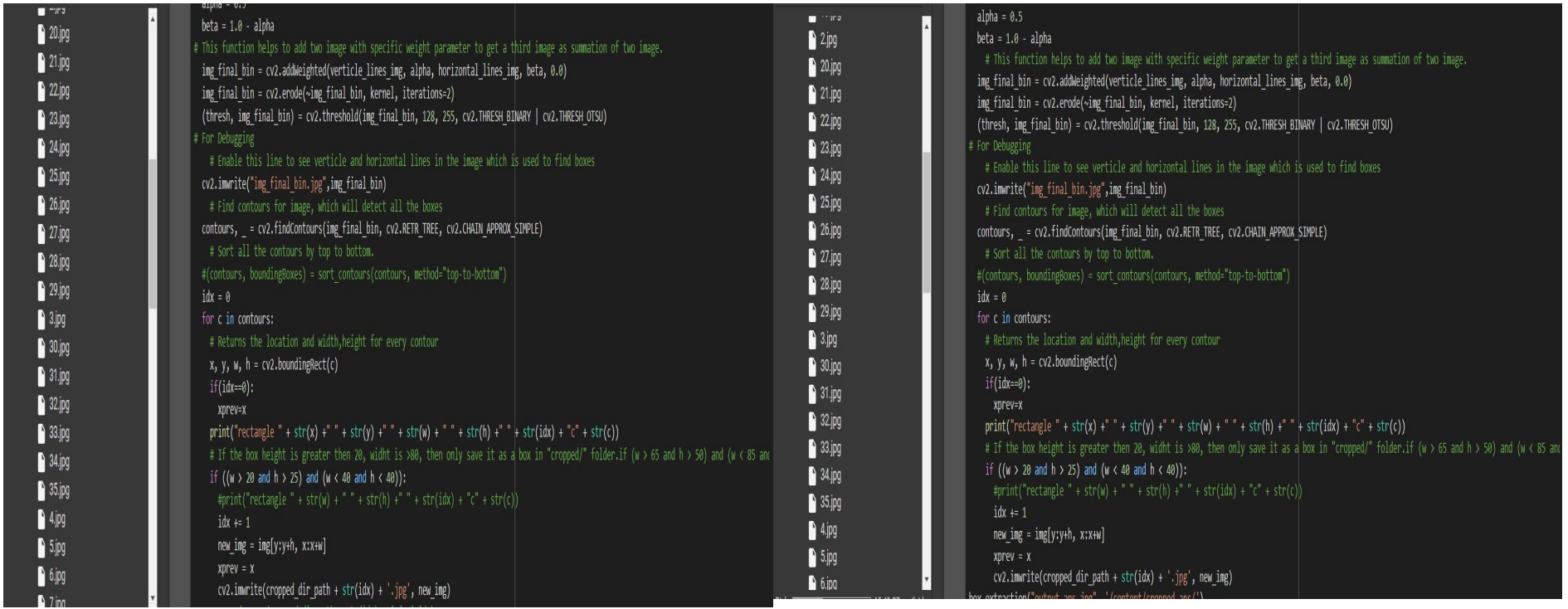


Implementation: Bound Boxes dataset





Implementation: Question paper and answers



Implementation: Hog Feature Extraction

```
[ ]  print(resized_img.shape)

     #creating hog features
     fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2,2), visualize=True, multichannel=True)
     plt.axis("off")
     plt.imshow(hog_image, cmap="gray")

     # save the images
     plt.imsave("resized_img.jpg", resized_img)
     plt.imsave("hog_image.jpg", hog_image, cmap="gray")

     # axarr[0].imshow(img)
     # axarr[1].imshow(resized_img)
     # axarr[2].imshow(hog_image, cmap = "gray")
     # axarr[3].imshow(img)
```

Connected to
Python 3 Google Compute Engine backend

RAM: 0.93 GB/12.69 GB Disk: 41.88 GB/107.72 GB

```
(620, 614, 3)
(128, 128, 3)
```

THANK YOU