Binary_Brain

Solution to prevent copyright infringement/piracy/plagiarism of NCERT text books.

PS No. RK785

Organization: Department of School Education & Literacy (DoSEL), Ministry of Education.

Piracy is an issue of concern as printing has become cheaper and OCR has become easier. Many pirates simply scan the books, recognize text and print the documents. Apart from being unfair to authors' hardwork and illegalities it also harms the reader because there is no guarantee the pirated book's content is correct and not lost in the duplicating process.
In this project we will be discussing methods that we can use to differentiate between original and pirated books.

Ideas:

[UPDATED 14th April] Plagiarism Checker:

We will perform plagiarism checking and find out if there are any similarities between content in a book and NCERT book. This plagiarism check first needs content in text format. We have NCERT books content but for other publisher content we need to perform OCR.
We have mentioned Pytesseract module further in the document and also shown a demo. Apart from that we will be trying to implement our own DL models wherever we think it might be necessary.
Plagiarism checking can be also said as string matching, and we have many algorithms for the very purpose. For example

1. Rabin Karp
2. KMP algorithm
3. String Matching with Finite Automata
4. Boyer-Moore Algorithm

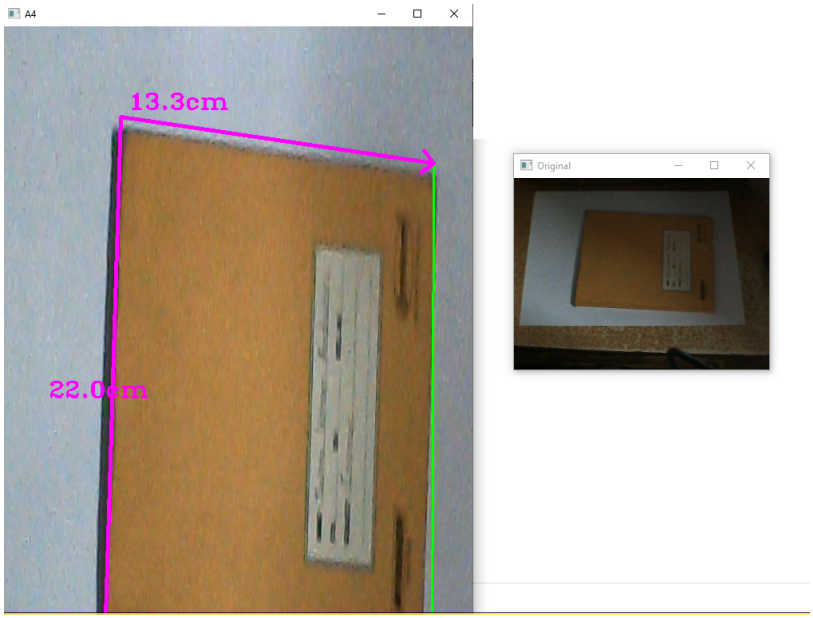
Now we won't be going into working of these algorithms as they are already available on internet in much detail but in addition to simple string matching we will be using a dictionary and grammar knowledge to find cases of plagiarism.
For eg there might be chances that some sentences are converted from active to passive and vice versa to avoid plagiarism. Also by using Natural language Processing we can detect if there was attempt for plagiarism.
This can be achieved by use of GPT-3 NLP model which is extremely powerful tool currently used and is trained from data over the internet. One good example of GPT-3 is Github Co-pilot uses it.
We have already implemented few projects using GPT-2 and thus we believe that we will be able to detect plagiarism in efficient way using these methods.

<u>Length and Width detection</u>

First, let's see how we differentiate original and duplicate books on the basis of length and width.

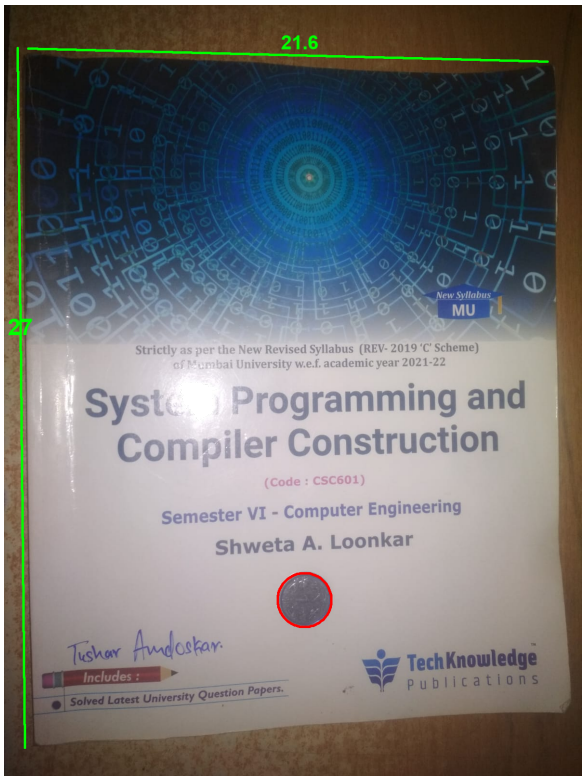
Approach 1: In this we can simply ask the user for dimensions of the book and compare it with data of the original book that we have.
Approach 2: In the second approach we can ask the user to place the book on a white A4 size sheet and then to take a snap from the mobile phone. We already know the size of the A4 sheet (210 x 297 mm). So by using that as a scale we can easily predict the size of a book. There are 2 disadvantages in this method.

1. The size of selected paper (A3 or A4) must be bigger than the size of book i.e. book should be inside the page/
2. Availability of page can be issue of concern



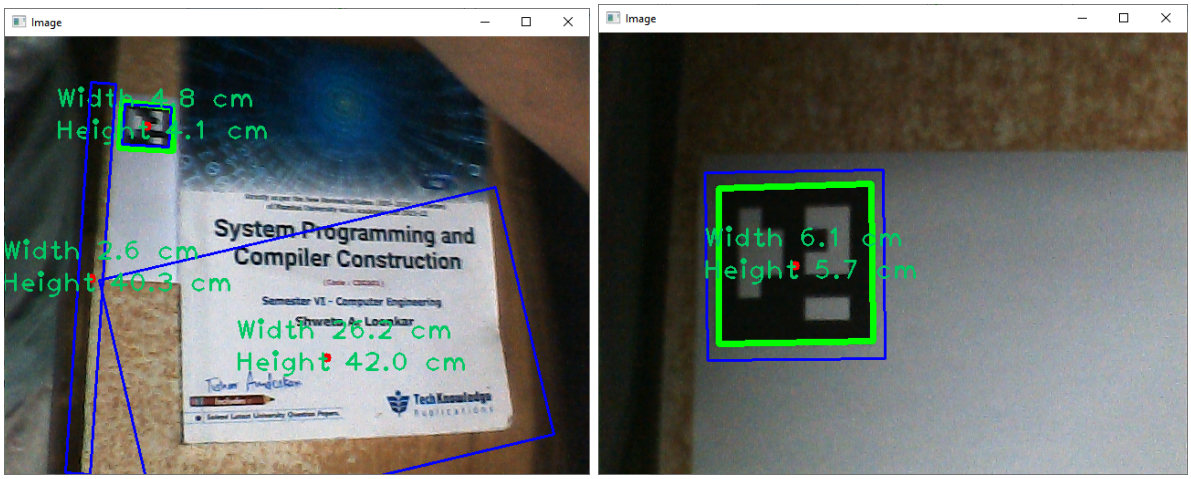
Source : Implemented by us (Binary_Brain).

Approach 3: This approach is similar to the previous one but instead of using A4 page we can use a ₹1 coin which can be placed on the book and then we can predict size of book. Here to make this method more general meaning - use of any coin like ₹5 or ₹10 or even international coins we will be training a deep learning algorithm to recognise which coin it is and then since we know the diameter of it we can calculate the dimensions of book.



Source : Created by us (Binary_Brain).

Approach 4: We can also use aruco markers as they are extremely easy to detect and would be very efficient but the buyer needs to take a printout of the exact size. If we are able to make Approach 3 as efficient as this one then there won't be a need for aruco marker method. Here demo doesn't look much appreciative but it will be improved.



Source : Implemented by us (Binary_Brain).

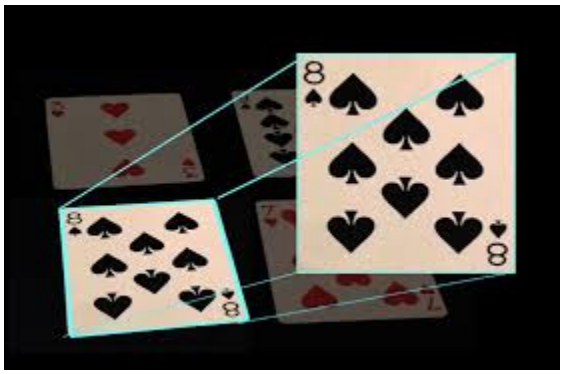
Cover Page similarities

The first and foremost thing to do in this is would be to apply wrap perspective to the images (similar to apps like camscanner) to remove unnecessary background and also to straighten the image.

Naive Method:

One simple method would be that after applying the wrap perspective we can read image value pixel by pixel.

- We need to calibrate the pixels so that two images are of the same size and superimposed correctly over each other.
- There would be a difference between brightness and contrast and that can be adjusted by taking relative differences between colors.



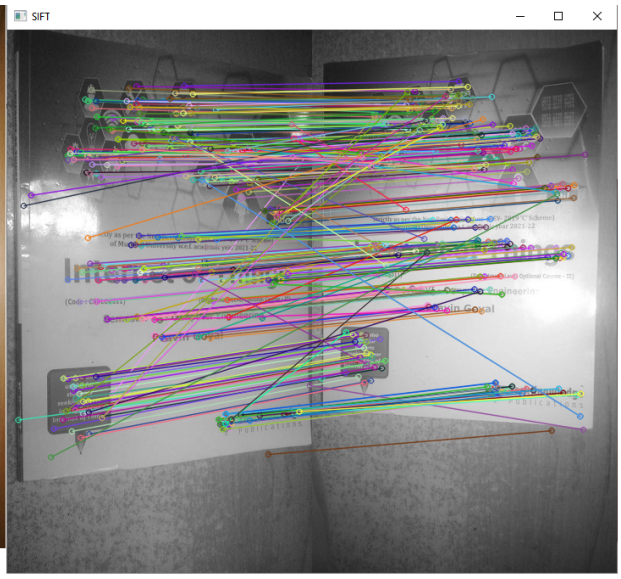
Wrap Perspective, Source: answers.opencv.org

Use of Algorithms

There are many algorithms that are used in Image Processing to detect if two images are similar. We have discussed few of them below

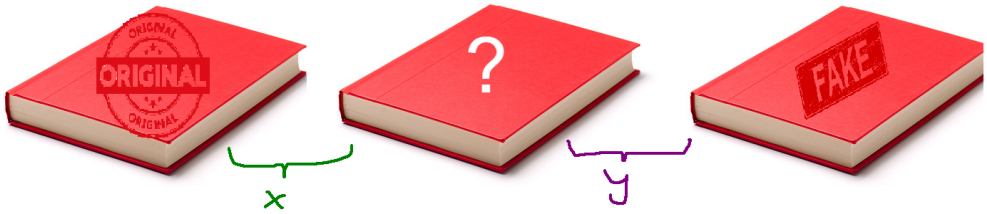
1. SIFT : SIFT (Scale-invariant feature transform) is a algorithm to detect, describe, and match local features in images. Here features mean edges, corners, blobs, ridges etc. Major advantages of this algorithm as quoted on wikipedia are as follows.
    - Locality: features are local, so robust to occlusion and clutter (no prior segmentation)
    - Distinctiveness: individual features can be matched to a large database of objects
    - Quantity: many features can be generated for even small objects
    - Efficiency: close to real-time performance
    - Extensibility: can easily be extended to a wide range of different feature types, with each adding robustness

Input Images (same book different angle)



Output Image

2. SURF : This is another algorithm which is similar to SIFT used to detect features and predict resemblance between images.
   Quoted from wikipedia -> It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.
3. ORB: Oriented FAST and Rotated BRIEF is yet another free to use algorithm for the same purpose.
4. Custom Deep learning: This would be one another approach that we would be implementing. We would be collecting copies of both original and pirated books and training them. A point to note is that we won't be just finding similarities between the original book and the book in the store, but we would be comparing it with both original and pirated copy.



Since we will be training both fake copies and one genuine copy we can calculate similarities and can conclude our unknown book is more analogous to which one of them

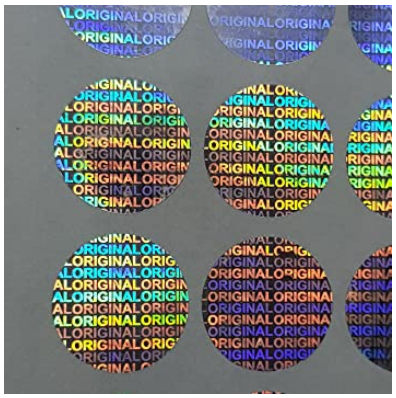
Source : Created by us (Binary_Brain).

Watermark Detection

According to NCERT circular (available on https://ncert.nic.in/pdf/publication/informationtocustomers/StopPiracy.pdf) we can see that on every 8th page the NCERT logo is printed as a watermark.
Watermark detection won't need much advanced algorithm though we have read many papers that use Convolution Networks, we can simply identify it by adjusting the image features and brightness, contrast etc. The images will be compared and we can make inferences.

Hologram Stickers

If the book contains holograms we can use Convolutional Neural Networks (CNN) or any suitable algorithm to find if it matches with the original copy.
Here we need to ask user to upload hologram from various angles so that we can predict that if it is authentic or fake


Source: Amazon.in

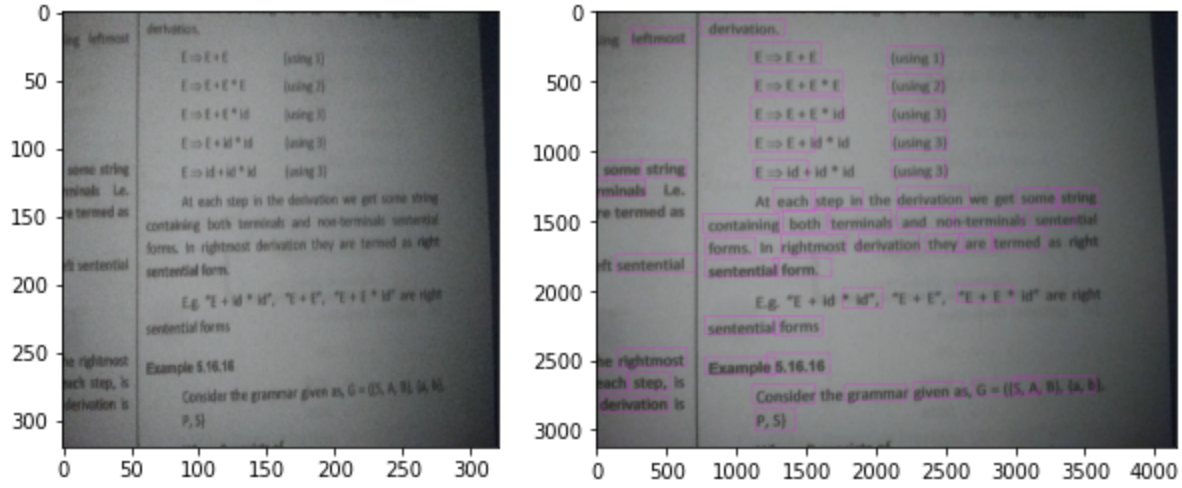
Text Formatting

In pirated books Text formatting is usually poor and even the fonts used are different from the original one.
Text formatting part can be divided into two parts:

1. Check if the paragraphs start and end word the same as the original copy of the same edition.
2. Font detection.

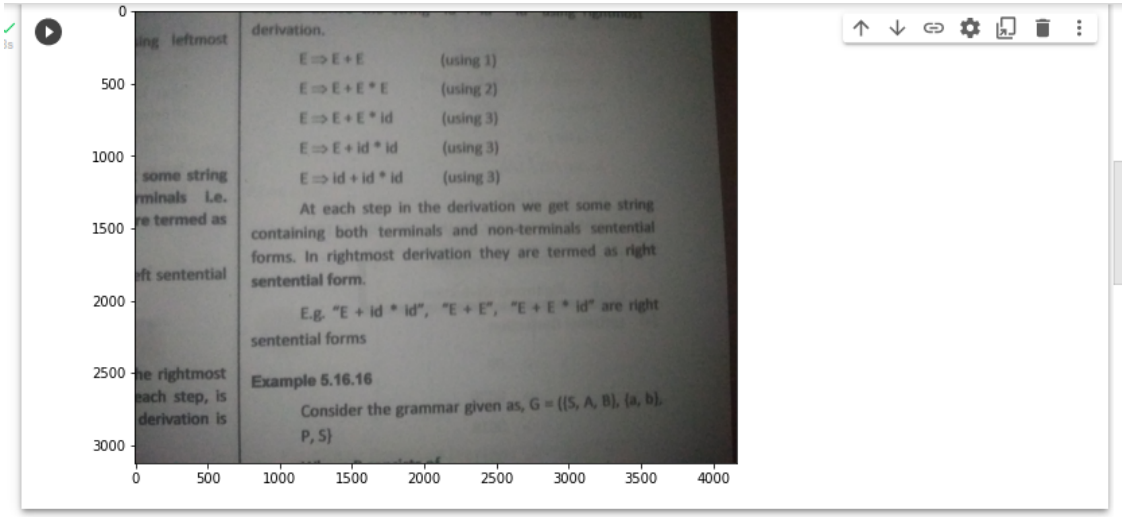
For text detection we can use the EAST algorithm which stands for - Efficient and Accurate Scene Text Detector. Then for text recognition we can use Google's Tesseract.
One another algorithm for Text Detection can be CRAFT (Character-Region Awareness For Text detection)

Result of East algorithm



Source : Implemented by us (Binary_Brain).

Now for font detection we can use PyTesseract Library which supports multiple languages. Below is Demo of pytesseract text recognition algorithm



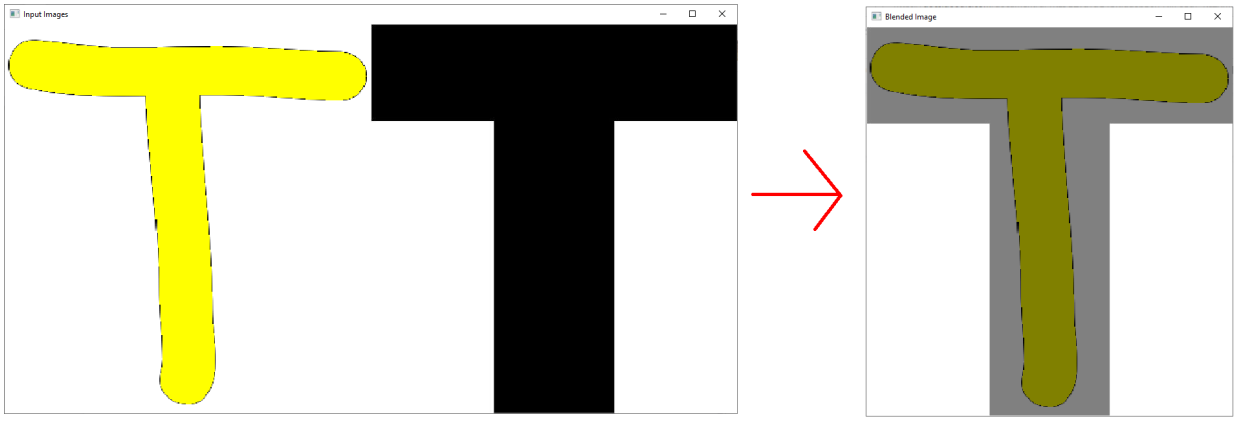
```
[7]  text_out = pytesseract.image_to_string(img)
```

```
[8]  text_out
```

```
'   \n\n=a) | lua ee "eR\n\n  \n  \n   \n    \n    \n  \n \n \n \n  \n  \n   \n\n \n  \n   \n  \n
\n \n\n \n\nte (using 1)\nee (using 2\neE *id (using 3)\nFeo E+ id * id using 3\nEF <> id + id * i
d using 3)\n\nAt each step in the derivation we\n\n \n\ncontaining both terminals and non-termin\n
forms. In rightmost derivation they are %\n\n \n\nsentential form.\n\nEg “E+id * id”, “+E "oe\n\n
\n\nsentential forms\n\nExample 5.16.16\n\x0c'
```

Source : Implemented by us (Binary_Brain).

Next part would be the detection of the font. In this what we can do, we can take any random letter like ‘E’ or ‘T’ and then superimpose that on font of the original



Two Texts are superimposed

Source : Implemented by us (Binary_Brain).

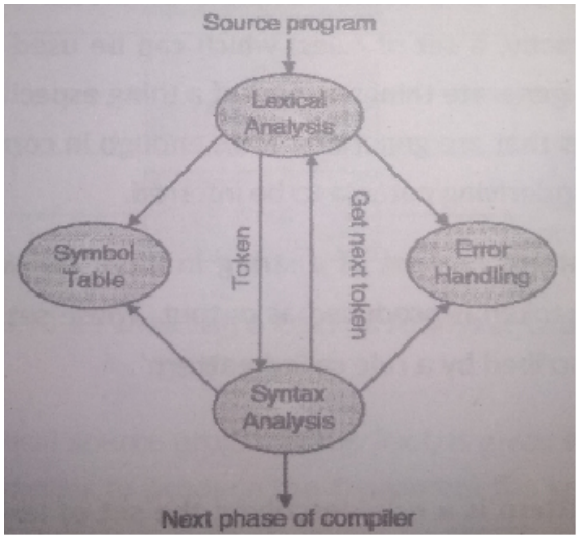
Blur Detection

Another peculiar thing that is noticed in the pirated copies are the blurry images of diagrams, flowcharts or any images. This is because the images are scanned copy of the original ones and pirates don't have real image available with them
Thus this can be a really good approach to tackle this problem.

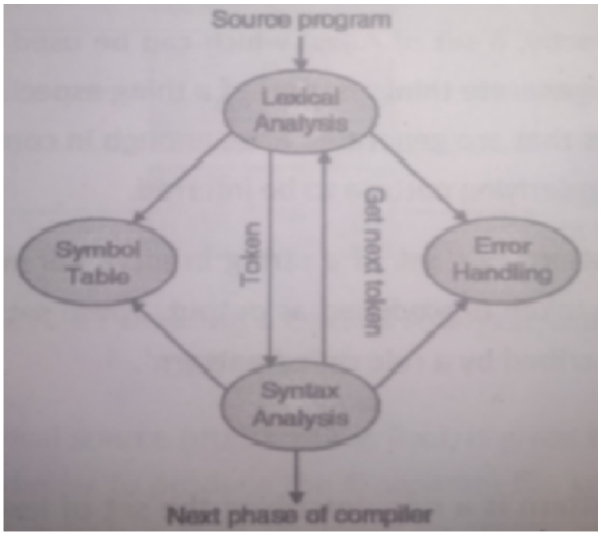
Laplacian Filter is an edge detection technique that is used in computer vision to detect edges. An image with more blurriness will have less laplacian values than that of sharp images.

Though in this method the camera quality will effect the results but given that in 2022 most of the smartphones' cameras are of decent quality (above 5MP) this won't be a big issue in practical scenario.

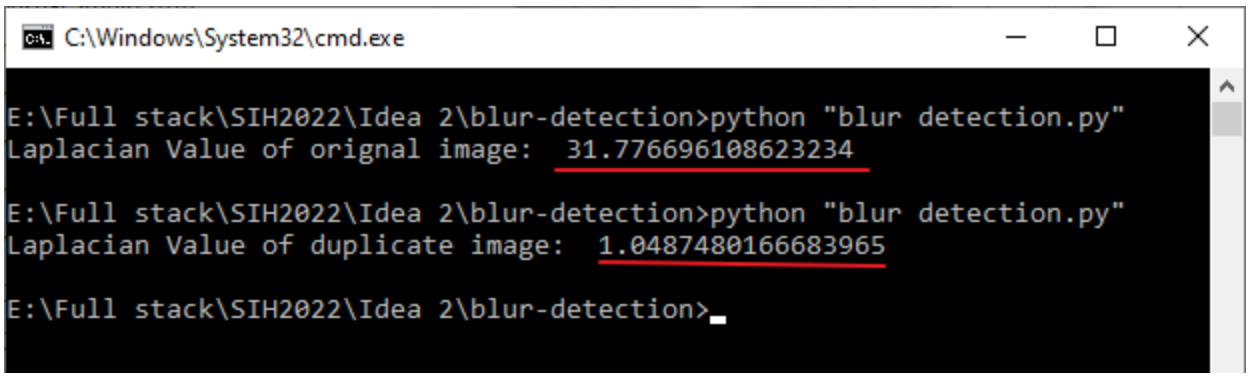
Demo of Blur Detection



Original



Duplicate



Web Portal / Progressive Web App for end users.

We would be creating a Responsive PWA that can run smoothly on both PCs and Smartphones.

- Frontend would be made using React
- For the backend we will be using Flask and Node depending on the need. REST APIs will be created
- For Database we will be using Google Firestore realtime database to get results quickly and give good experience to end users.
- We will be also deploying it on Azure (or any other cloud platform depending on the requirements)

The web portal will be extremely easy to use and user friendly.

The genuinity of the book can be checked in the form of steps and at the end we will show results. It will be somewhat similar to the quizzes where each slide will have a particular test for eg. first check length and width then next step would be cover page checker etc.

With that users can also report for pirated copies and shops will also have a rating system. If a shop is constantly selling fake books then more people will report and the trustability score will go down. Similarly if a shop is selling original books the rating will increase. This will help other users in the vicinity to get ideas about the correct shop to buy genuine books.

All the ideas mentioned above are tried and tested by us on our local machine and found to be promising. We will be implementing more ideas like paper quality prediction by putting phone's LED behind the paper and taking a snap etc. Images attached have a proper source and the one created by us is mentioned under our team name.
Our Team members consist of Web developers, Machine Learning devs and also Image Processing developers. We have created many personal projects and have done a decent number of internships so we are 100% confident that the given problem statement can be successfully completed by us and we can create a platform for end users to detect pirated books.