Authored By: Shiby Surendranath & Mentored By: Douglas Sherk   Date: Jan 24, 2021
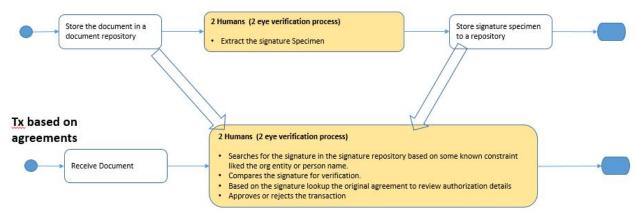
Problem Statement

Hard documents with wet signatures are still the norm when establishing contract agreements between entities. The signatures of authorizers of any future changes or transactional requests related to these agreements need to be vetted against the wet signatures in the original contract agreement. The vetting processing is currently manual in most organizations and typically involve the following activities:

Relevant Business domains
1) Financial services when processing applications/contracts between corporate entities, check processing, claims processing.
2) Elections (Voting paper ballots) processing

Process Improvement Opportunities
1. The process of signature verification is tedious, specifically around forgery detection.
2. Matching a signature against a signature repository is tedious
3. Reduce the number of humans involved in the verification process by 1 by having the system complement a human.
4. The process of extracting and cleansing the signature is tedious and error prone.

Solution Scope

- This solution will attempt to complete the human activity by providing the following information:
  - ***Forgery Classification***- Detect if the signature is a forgery
  - ***Signature Matching*** -
    - Match the signature from a list of signatures on file.
    - The list will be constrained to an organizational entity. The organizational entity could have multiple signers and each signer could have multiple signatures on file.

■ The output is a list of entities which match the signature with a prediction accuracy expressed as %. Only the highest match signature per person will be included as part of the list.

○ ***Signature Extraction*** - If time permits will demonstrate a signature extraction from a predefined simple template document and use this for matching.

Out of scope

To keep the goal attainable each of the scenarios ***will be independently processed.*** Integrating all 3 scenarios to a single unified seamless solution is out of scope of this effort.

● User interface
   a. Upload the artifacts to be matched to a source folder.
   b. Execute the program which will pick up the content in the source folder
   c. Write the output to a target folder in JSON format.
   d. Signature extraction - will output the signature specimen to the target folder with some metadata extracted from the document.

Solution

Supervised learning will be used to train multiple models for all three scenarios (forgery detection, signature matching and signature extraction).

At the most basic level it is a classification type supervised learning problem. As this is related to computer vision Deep Learning techniques like convolutions and pooling may be used.

Forgery Classification

Limitation - the number of forged signatures is rare and limited in the real world compared to genuine signatures. The performance of this model during training will need to be biased for forgery.

Inputs and Outputs

	Data type	Additional Comments
Data Input		
"Signature Specimen" to be analyzed.	*Image*	
Data Output		
forged	*boolean*	*True if forged*
confidence	*decimal*	*Percentage accuracy of the prediction*

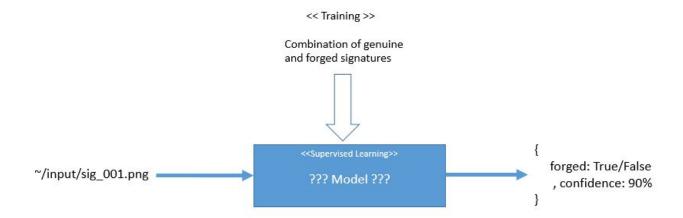| Data Distribution Considerations |
| --- |
| An even distribution of images of signature specimens for good and forged signatures All signature specimens will be done on a white or **green** background only. - Document background color (white, skyblue) - Background noise (Need to manually add some of this noise) - Signing outside a box or the lines crossing a signature - Sheet is crumbled after signature - Documents with watermarks and signatures on it (add a couple of samples - White doc with gray water mark - Signing over printed text in the document - Signature Color (Black, blue, green and red ink samples) - Varying resolution DPI scans. **Constraints** ○ Only .png files will be supported. ○ The kaggle dataset scan resolution is unknown at this time. This might have an impact on the results due to lack of consistency. |

DataSet

- Acquired a collection of signature specimens from Kaggle.
- The data is already classified as good versus forged.
- The signature specimen is aggregated by person.
- *Additional Data wrangling needed*

Design

Authored By: Shiby Surendranath & Mentored By: Douglas Sherk   Date: Jan 24, 2021

Yet to be determined

Signature Matching

Inputs and Outputs
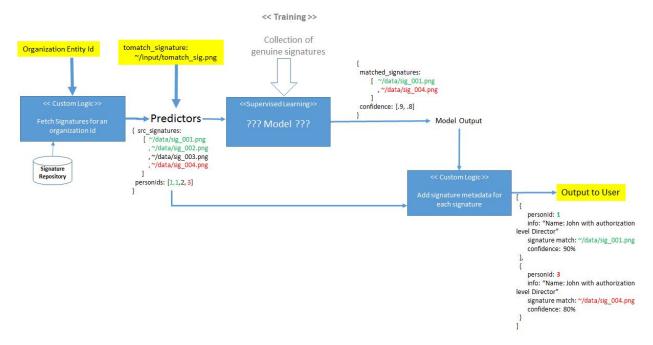
	Data type	**Additional Comments**
Data Input		
Signature Specimens to match against	*Image[]*	*A list of signature specimens for an organizational entity. Each specimen name will be unique*
Signature Specimens person ids	*int[]*	*A unique id maintained for each person, whose signature specimen is in the "to match" pool. A single person id could be matched to multiple signatures.*
Signature Specimen Metadata of each signature in the collection	*String[]*	*Metadata containing person name and any associated authorization levels rules.*
Signature Specimen to match	*Image*	*The signature specimen to be matched*
Data Output		
Matched Signature Specimen Collection id	*dict*	*Key - the image identifier.* ● *Include only signatures with a match accuracy >= 80%* ● *For each person include only signatures with the highest accuracy match.* *Value - the level of accuracy of the match.*
Data Distribution Considerations		

DataSet

Label the Kaggle dataset with organization and person names. Will distribute the signatures across a distinct combination of entities and personnel.

Design

Yet to be determined

Signature Detection

Inputs and Outputs

	Data type	**Additional Comments**
Data Input		
Same as signature match		The signature extraction is limited to know signatures in the repository. This solution does not account for signature extraction of a new signature.
Template	*String*	This narrows down on where in the document to look for the signature.
Data Output		
Signature Specimen	*Image*	
Person Name	*String*	*Name of the person from the document.*
Data Distribution Considerations		

DataSet

Do not have a dataset for this yet. Looking at maybe manually creating a small dataset to prove the concept.

Design

Estimate of resources needed for project execution

Preliminary estimate for processing hardware requirements is:
CPU - 8 Cores
GPU - GTX 1080 Ti
Memory - 16 GB
Hard Disk - 50GB