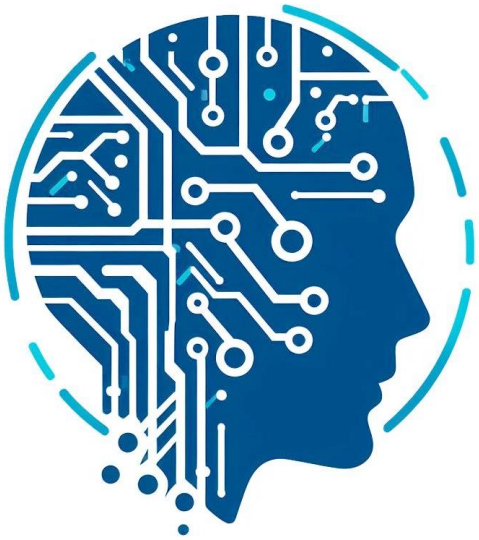



# GSoC 2024 Proposal

# Automating Text Recognition and Transliteration of Historical Documents with convolutional - recurrent architectures

# Table Of Contents

# Text Recognition with CRNN

## HumanAI


### Shashank Shekhar Singh

**Name:** Shashank Shekhar Singh
**Major:** Mechanical Engineering
**Degree:** Bachelor of Technology
**Year:** Sophomore
**University:** Indian Institute of Technology (BHU), Varanasi
**GitHub:** @Shashankss1205
**College Mail:** shashankshekhar.singh.mec22@itbhu.ac.in
**Private Mail:** shashankshekharsingh1205@gmail.com
**Phone:** (+91) 8303130207
**LinkedIn:** Profile
**Resume:** Link
**Nationality:** India
**Address:** SH 1/1A -1-12, Unchawa Lodge, Gilat Bazar, Varanasi, UP
**Time zone**: Indian Standard Time (UTC +5:30)


# About Me

Hi! I am Shashank Shekhar Singh, a Mechanical Engineering sophomore at the prestigious Indian Institute of Technology BHU (Varanasi), India. I am a Machine Learning Engineer with in-depth knowledge of **Model selection, Development, and Deployment**, some of my other skills include Web development using MERN, app development using Flutter and competitive programming. I love open-source development and building scalable solutions catering towards arts and humanities field development. This has motivated me to contribute towards HumanAI organization.

# My Personal Projects

1. **Visual Question Answering System | Inter IIT Preparation | Kaggle**

- Constructed a VQA model utilizing Keras and TensorFlow frameworks.
- Utilized NLP components such as stemming, tokenization, and Google News bin pretrained weights.
- Incorporated the VGG16 Model for ImageNet CNN for computer vision tasks.

2. **Dark Pattern Detection with Transformer-based Models | Dark Pattern Buster Hackathon' 2023 | GitHub**

- Engineered a Python script leveraging fine-tuned transformer-based models (BERT, XLNet, RoBERTa) to detect and categorize dark patterns in textual data.
- Extended functionality with a web plugin for visual highlighting of deceptive design elements, enhancing user interface analysis and empowerment tools.

3. **NLP-based Question Answering System for Tabular Data | NLP.py | GitHub**

- Developed a web interface using Vanilla JavaScript.
- Implemented a database backend using SQLite and integrated a prompting NLP tool, GPT-3 API.
- Engineered a pipeline for question input, database injection, and SQL generation for streamlined operation.

4. **NLP-based Coding Automation with OpenAI GPT-3.5 | Byte Synergy | GitHub**

- Implemented an NLP-based chatbot model and a React application to automate website development coding processes.

# My Open-Source Contributions

1. **Text Detection and Recognition with CRAFT | Open-Source Project | HumanAI | GitHub**
- Developed software utilizing the Character-Region Awareness for Text detection (**CRAFT**) in PyTorch for precise text bounding box detection.
- Integrated CNN-RNN for accurate word recognition capability.

2. **Gravitational Lensing Image Generation using DDPM Models | Open-Source Project | ML4SCI | GitHub**
- Developed a Denoising Diffusion Probabilistic Model in TensorFlow and Keras for high fidelity Image generation integrated with Gaussian Diffusion approach.

3. **Gravitational Lensing Image Classification using Auto Encoder-Decoder | Open-Source Project | ML4SCI | GitHub**
- Developed an **Auto Encoder-Decoder Model** in TensorFlow and Keras for high accuracy image classification using the **ResNet50** model coupled with **Gaussian and Rotational augmentation** to achieve the same.

4. **Interpolation to find unknown values in space-exploration | Open-Source Project | OpenAstronomy | GitHub**
- Integrated **linear** and **polynomial interpolation**, **PyTorch-based neural networks (ML/DL)**, and **support vector regression model** for accurate prediction of flux measurements at desired wavelengths.

# Project Abstract

This project aims to address the challenge of text recognition from historical Spanish printed sources dating back to the seventeenth century, a domain where existing Optical Character Recognition (OCR) tools often fail due to the complexity and variability of the texts. Leveraging hybrid end-to-end models based on a combination of multiple architectures, such as CNN-RNN, our research seeks to develop advanced machine learning techniques capable of accurately transcribing non-standard printed text.

# Project Goals

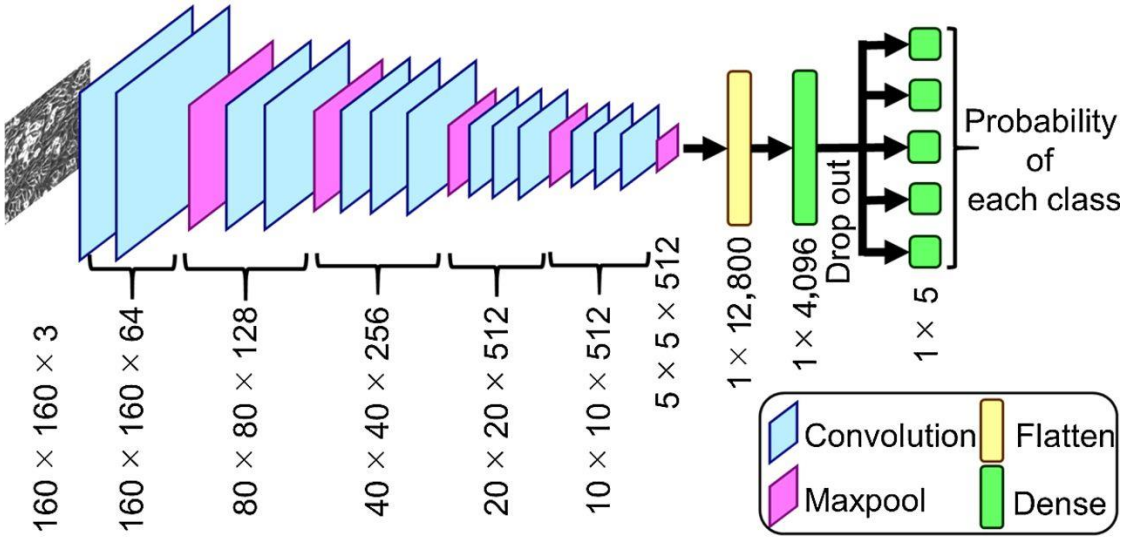
**1. Development of Hybrid End-to-End Models:** The primary goal of this project is to **design, implement, and fine-tune** hybrid end-to-end models based on **CRNN** architectures for text recognition. By combining the strengths of architectures such as recurrent neural networks (RNN) or convolutional neural networks (CNN), the models aim to effectively capture both local and global features in the historical Spanish printed text, enhancing accuracy and robustness in transcription.

**2. Achieving High Accuracy:** The ultimate objective is to train machine learning models capable of extracting text from seventeenth-century Spanish printed sources with **at least 80% accuracy**. This entails extensive experimentation, hyperparameter tuning, and dataset curation to ensure the models generalize well across **various styles, fonts, and degradation levels** present in historical documents. Achieving this goal will signify a significant advancement in text recognition, particularly in the context of preserving and analyzing ancient textual artifacts.

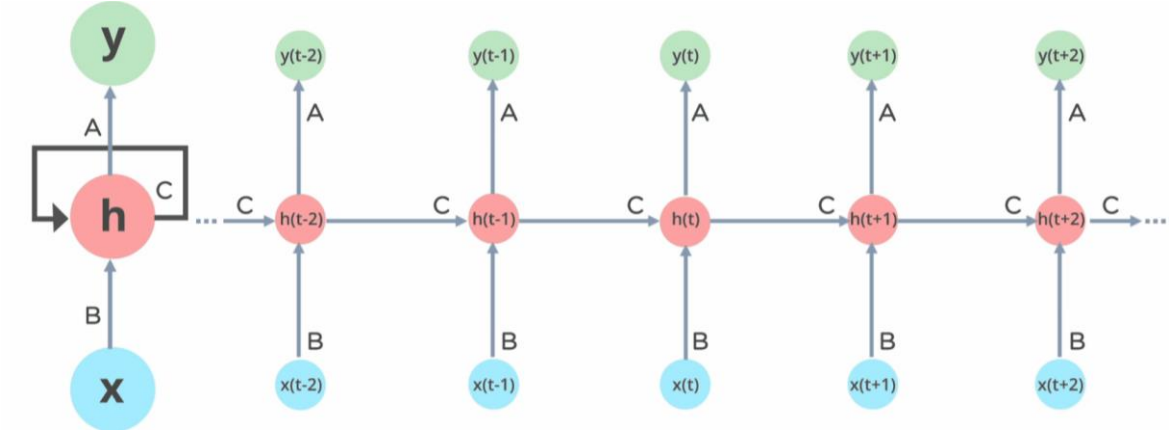
# The CRNN Model

The Convolutional Recurrent Neural Networks is the combination of two of the most prominent neural networks. The CRNN (convolutional recurrent neural network) involves **CNN** (convolutional neural network) followed by the **RNN** (Recurrent neural networks).

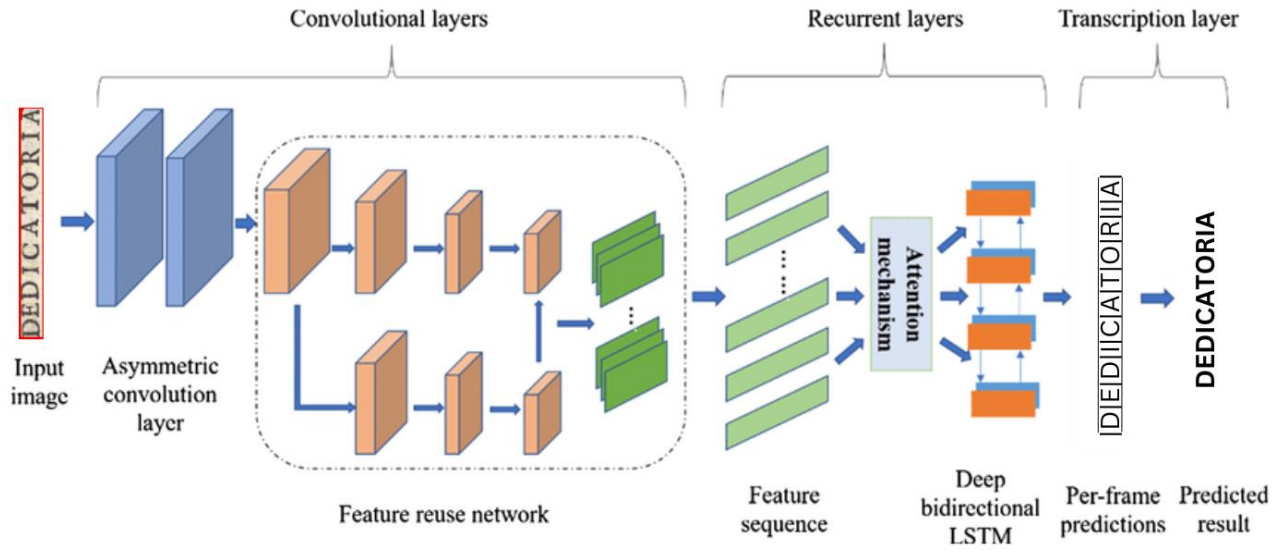
## Convolutional Neural Network




## Recurrent Neural Network



# CRNN Architecture




# My Approach towards attempting the Evaluation Test

## 1) Understanding the problem statement

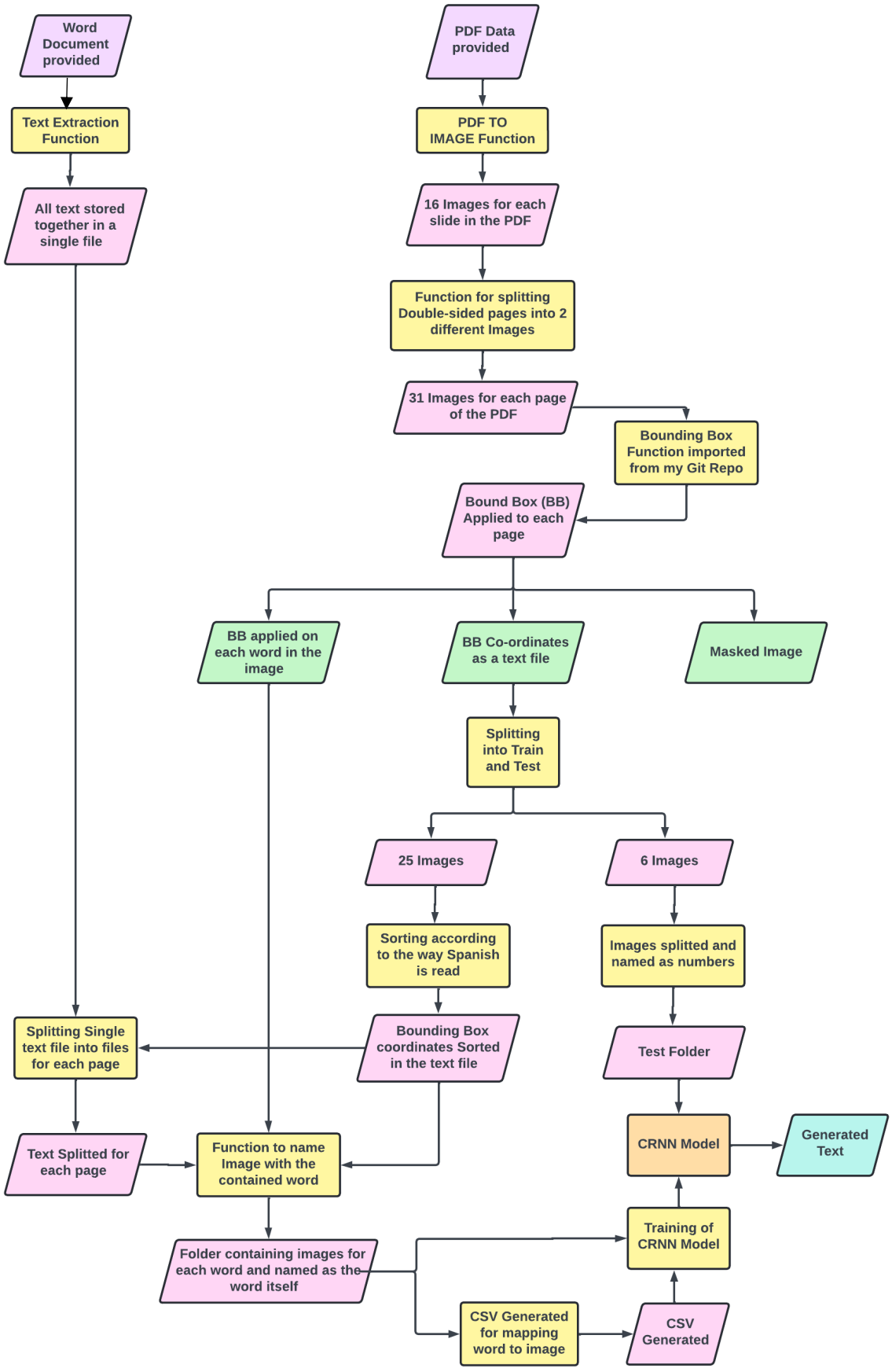
My first step was to look over the problem statement and understand the working and architecture of the Convolutional Recurrent Neural Network (CRNN) in greater detail.

## 2) Importing the dataset

My next step was to glance at the dataset provided and do the pre-processing required to convert it into desirable format for CRNN Model.

The dataset consists of **2 files** named as:

1) **Padilla - 1 Nobleza virtuosa_testTranscription.docx**
2) **Padilla - Nobleza virtuosa_testExtract.pdf**


Word Document provided
Text Extraction Function
All text stored together in a single file
PDF Data provided
PDF TO IMAGE Function
16 Images for each slide in the PDF
Function for splitting Double-sided pages into 2 different Images
31 Images for each page of the PDF
Bounding Box Function imported from my Git Repo
Bound Box (BB) Applied to each page
BB applied on each word in the image
BB Co-ordinates as a text file
Masked Image
Splitting into Train and Test
25 Images
6 Images
Sorting according to the way Spanish is read
Images splitted and named as numbers
Splitting Single text file into files for each page
Bounding Box coordinates Sorted in the text file
Test Folder
CRNN Model
Generated Text
Text Splitted for each page
Function to name Image with the contained word
Training of CRNN Model
Folder containing images for each word and named as the word itself
CSV Generated for mapping word to image
CSV Generated

# 3) Exploring and pre-processing the dataset

Now, I went on to perform the pre-processing of my dataset as follows:

## 1) PDF to IMAGE function

I created this function to generate an image for each slide in the PDF provided since images are easy to process using python libraries.

```python
def pdf_to_images(pdf_path, output_folder):
    # Open the PDF
    pdf_document = fitz.open(pdf_path)

    # Iterate over each page in the PDF
    for page_number in range(len(pdf_document)):
        # Get the page
        page = pdf_document.load_page(page_number)

        # Render the page as a Pixmap
        pixmap = page.get_pixmap()

        # Save the Pixmap as a PNG image
        image_path = os.path.join(output_folder, f'page_{page_number + 1}.png')
        pixmap.save(image_path)

    # Close the PDF
    pdf_document.close()
```

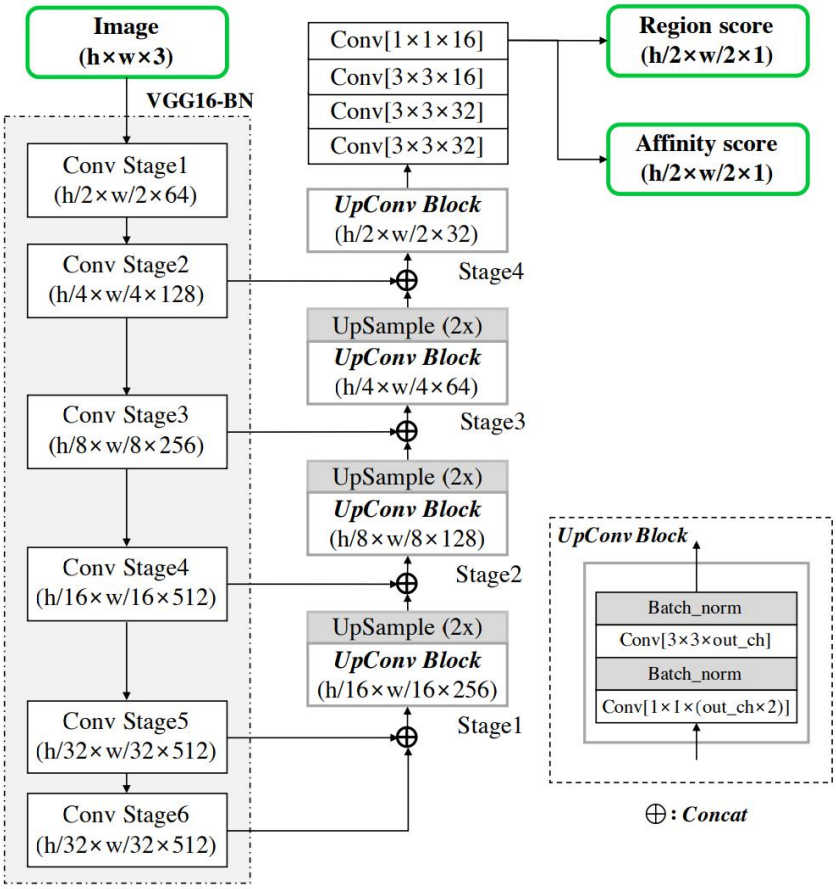
## 2) Splitting Double-sided pages into 2 different images

I created this function to split an image containing a double-sided page into 2 images since processing individual pages is comparatively easier.

```python
def split_and_save_image(image_path, output_folder, last_image_number):
  # Read the image
  img = cv2.imread(image_path)

  # Get image width
  _, width, _ = img.shape

  # Determine filename based on width and last image number
  if width < 350:
    filename = f"image_{last_image_number}.png"
    output_path = os.path.join(output_folder, filename)
    cv2.imwrite(output_path, img)
    last_image_number += 1
  elif width > 450:
    left_half = img[:, :width // 2]
    right_half = img[:, width // 2:]
    filename = f"image_{last_image_number}.png"
    output_path = os.path.join(output_folder, filename)
    cv2.imwrite(output_path, left_half)
    last_image_number += 1
    filename = f"image_{last_image_number}.png"
    output_path = os.path.join(output_folder, filename)
    cv2.imwrite(output_path, right_half)
    last_image_number += 1

  return last_image_number
```

## 3) Bounding Box Function

For detection of Bounding Boxes for each word present in a page, I chose the **CRAFT** (Character Region Awareness for Text Detection) Model. A link to the Official implementation of the research paper can be found here.




**Ground Truth Label Generation:** CRAFT generates ground truth labels for both the **region score** and **affinity score** by encoding the probability of the character center with a **Gaussian heatmap**. This heatmap representation allows for flexibility in representing regions that are not rigidly bounded, such as irregularly shaped text.

**Character Bounding Box Generation:** The ground truth bounding boxes for **character regions** are approximated and generated using a process that involves perspective transformation and **Gaussian map warping**. This

ensures that the model can effectively localize individual characters despite distortions induced by perspective projections in real-world images.

**Affinity Box Generation:** Affinity boxes, which represent the **space between adjacent characters**, are defined using the bounding boxes of adjacent character pairs. By connecting opposite corners of each character box and generating triangles, affinity boxes are created with the centers of these triangles as corners. This process enables the model to understand the relationships between characters and group them into text instances accurately.

### 4) Splitting the Bounding box Folder

I split the folder containing text files for bounding boxes Co-ordinates into Train and Test, where **Train contains 25 text files** and **Test contains 6 text** files intentionally left blank for the purpose of Testing of the model.

```
for image in range(count_files_in_folder(image_folder)- 6): #last 6 pages are for testing
```

### 5) Sorting Function for Bounding Boxes

This function was a breakthrough for mapping the **extracted images for words present in each page** to the **actual text contained in the image** for training my CRNN model.

1. **Grouping Bounding Boxes**: The bounding boxes were organized into groups based on their vertical proximity, with a **maximum difference in the y-coordinate of 10 pixels**. This grouping approach aimed to cluster all words and bounding boxes belonging to the **same sentence together**.
2. **Sorting Groups**: The groups were then **sorted in ascending order of their y-coordinates.** This sorting was essential as **Spanish text is typically read from top to bottom**, ensuring that sentences were arranged in a logical sequence.
3. **Sorting Bounding Boxes Within Groups**: Within each group, the individual bounding boxes representing words were **sorted based on**

**their x-coordinates.** This sorting method ensured that words within a sentence were arranged from **left to right**, aligning with the left-to-right reading orientation of Spanish text.

```python
def process_bounding_boxes(file_path):
    with open(file_path, "r") as file:
        lines = file.readlines()

    # Parse bounding box coordinates
    bounding_boxes = []
    for line in lines:
        coords = list(map(int, line.strip().split(',')))
        bounding_boxes.append(coords)

    # Sort bounding boxes based on y_min value
    bounding_boxes.sort(key=lambda box: box[1])

    # Group bounding boxes based on difference between max and min y_min values
    grouped_boxes = []
    current_group = []
    for box in bounding_boxes:
        if not current_group:
            current_group.append(box)
        else:
            min_y = min(current_group, key=lambda x: x[1])[1]
            max_y = max(current_group, key=lambda x: x[1])[1]
            if box[1] - min_y <= 10:
                current_group.append(box)
            else:
                grouped_boxes.append(current_group)
                current_group = [box]

    # Append the last group
    if current_group:
        grouped_boxes.append(current_group)

    # Sort each group based on x_min value
    for group in grouped_boxes:
        group.sort(key=lambda box: box[0])

    return grouped_boxes
```

## 6) Extracting Text from the DOC file provided

I extracted all the useful text from the DOC file removing any extra information which was given for guiding through the problem statement. Along with this, I also proceeded to **remove punctuation** as guided in the problem statement itself. There was **no need to work upon** anything related to the letter **'f' and 's'** being used synonymously, because that was **tackled by my ML model itself.**

```python
def remove_punctuation(text):
    # Define a translation table that maps punctuations to None
    translation_table = str.maketrans("", "", string.punctuation)
    # Remove punctuations using translate() method
    return text.translate(translation_table)

def save_pages_to_text(docx_file, output_file):
    document = Document(docx_file)
    all_text = ""
    for paragraph in document.paragraphs:
        text = paragraph.text.strip()
        # Remove punctuations from the text before appending to the all_text variable
        text_without_punctuation = remove_punctuation(text)
        # Check if the line starts with "PDF p"
        if not text.startswith("PDF p"):
            all_text += text_without_punctuation + "\n"

    # Write all_text to the output file
    with open(output_file, "w") as file:
        file.write(all_text)
```

## 7) Generating Input files for the CRNN Model

Then I went on to create the "**Input data for my CRNN model"** i.e. **Images saved with the name of the word contained in the image.**

```python
def extract_bounding_boxes(image_path, bounding_boxes_file, text_file, output_folder):
    # Read the main image
    main_image = cv2.imread(image_path)

    # Create the output folder if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Read bounding box coordinates from the text file
    with open(bounding_boxes_file, 'r') as f:
        bounding_boxes_data = f.read().split(';')
    bounding_boxes_data = bounding_boxes_data[1:]
    print(bounding_boxes_data) #first value is for page number so skip it
    # Read text data from the text file
    with open(text_file, 'r') as f:
        text_data = f.read().split('\n')
    print(text_data)

    for indx in range(len(text_data)):
        words = text_data[indx].split(' ')
        print(words)
        bounding_box_coords = bounding_boxes_data[indx].strip().split('\n')
        print(bounding_box_coords)
        for cnt in range(min(len(words),len(bounding_box_coords))):
            coordinates_list = [int(coord) for coord in bounding_box_coords[cnt].split(',')]
            x_min, y_min, x_max, y_min, x_max, y_max, x_min, y_max = coordinates_list

            # Extract the bounding box from the main image
            bounding_box = main_image[y_min:y_max, x_min:x_max]

            # Save the bounding box as a separate image
            output_path = os.path.join(output_folder, f'{words[cnt]}.png')
            cv2.imwrite(output_path, bounding_box)

            print(f'Saved bounding box for "{words[cnt]}" as {output_path}')

def apply_extraction_to_folder(image_folder, bounding_box_folder, text_folder, output_folder):
    # Iterate over each image in the image folder
    # for image_filename in os.listdir(image_folder):
    for image in range(count_files_in_folder(image_folder)- 6): #last 6 pages are for testing
        image_filename = 'res_image_' + str(image+1) + '.jpg'
        print(image_filename)
        if image_filename.endswith('.png') or image_filename.endswith('.jpg'):
            # Get the shared number before the extension
            image_base_name = os.path.splitext(image_filename)[0]
            bounding_box_filename = image_base_name + '_sorted.txt'
            text_filename = image_base_name + '_actual.txt'
            # Check if the corresponding bounding box file exists
            bounding_box_path = os.path.join(bounding_box_folder, bounding_box_filename)
            actual_text_path = os.path.join(text_folder, text_filename)
            if os.path.exists(bounding_box_path):
                image_path = os.path.join(image_folder, image_filename)
                # Apply bounding box extraction to each image
                extract_bounding_boxes(image_path, bounding_box_path, actual_text_path, output_folder)
```

### 8) Defining CTC Loss for the CRNN Model

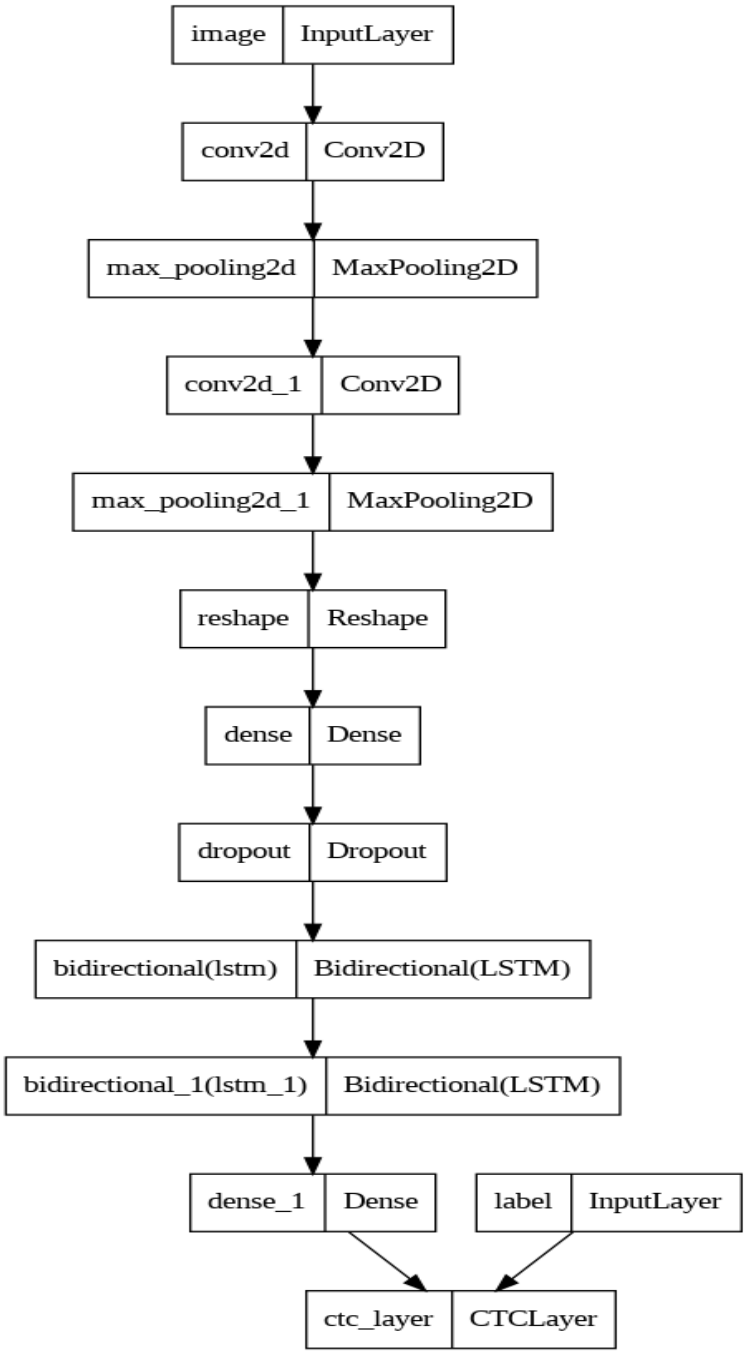
Next step was to define the loss function for my CRNN model, which was the **CTC (Connnectionist Temporal Classification Loss),** Reference link to a famous article about CTC loss can be found here.

```python
class CTCLayer(layers.Layer):

    def __init__(self, **kwargs) -> None:
        super().__init__(**kwargs)

        self.loss_fn = keras.backend.ctc_batch_cost

    def call(self, y_true, y_pred):

        batch_len = tf.cast(tf.shape(y_true)[0], dtype='int64')

        input_len = tf.cast(tf.shape(y_pred)[1], dtype='int64') * tf.ones(shape=(batch_len, 1), dtype='int64')
        label_len = tf.cast(tf.shape(y_true)[1], dtype='int64') * tf.ones(shape=(batch_len, 1), dtype='int64')

        loss = self.loss_fn(y_true, y_pred, input_len, label_len)

        self.add_loss(loss)

        return y_pred
```

# 4) Training the CRNN Model

Next, I went on to explore various architectures and implementations within the **CRNN Model framework**, with difference in number of layers and neurons per layer. Finally, I chose a simple model architecture of the CRNN Model, as my initial hit to the problem statement. My CTC loss has already approached 1 with a little bit of fine-tuning.

```
Epoch 199/200
82/82 [==============================] - ETA: 0s - loss: 1.0556
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which
WARNING:tensorflow:Can save best model only with val_loss available, skip
82/82 [==============================] - 3s 31ms/step - loss: 1.0556
Epoch 200/200
80/82 [============================>.] - ETA: 0s - loss: 1.0865
WARNING:tensorflow:Early stopping conditioned on metric `val_loss` which
WARNING:tensorflow:Can save best model only with val_loss available, skip
82/82 [==============================] - 4s 45ms/step - loss: 1.0905
```




# 5) Testing the CRNN Model

Next, I went on to test my trained model on the test dataset, i.e. the images for the last 6 pages that were intentionally left blank for testing purposes.

```
decoded_predictions = decode_pred(inference_model.predict(test_ds))

# Print decoded predictions in groups of 8 words per line
words_per_line = 8
for i in range(0, len(decoded_predictions), words_per_line):
    print(' '.join(decoded_predictions[i:i+words_per_line]))
```

```
54/54 [==============================] - 1s 12ms/step
pca poe vertivar pndo pedido botral avta edo
pienova buestre zida rosve p greos uiquada pete
o So nueos amize prezga dosla pueeys digrer
dadlos vust baaendo para esle a grave tecemi
comilos dacar intinetes dros precdo didcandolas liteate
dixo haas Rera niso Caro mu dalo tatros
deras Miubra se diad paes Portes tiio o
adimera ubrra cion qude mampecio anmicieia porto ramto
poo edcundados ernras bie muan relco sr vuea
buen empto puos pres qulira criñion ros presiras
tles l daeio ea Don contrario ferveación en
agigos No o éala puedsion rejor sazo Ser
f Coro bueeien Chritiiao no apor nerias ablges
```

# Learnings from the assignment test

1. **Data Preprocessing Challenges**: **Extracting text** from historical documents stored in **PDF and DOC formats** posed initial challenges. Preprocessing involved converting these documents into a machine-readable format suitable for training convolutional-recurrent architectures. Techniques such as PDF parsing and document conversion were crucial in this stage.
2. **Understanding Model Complexity**: Implementing convolutional-recurrent architectures requires a **deep understanding of both** convolutional neural networks (**CNNs**) and recurrent neural networks (**RNNs**). These architectures are powerful but complex, and understanding their intricacies is essential for effective implementation.

3. **Handling Historical Document Characteristics**: Historical documents often exhibit various characteristics such as **degraded text quality, unusual fonts, and non-standard layouts**. Adjusting model architectures and preprocessing techniques to handle these challenges is crucial for accurate text recognition and transliteration.
4. **Training Challenges**: Training convolutional-recurrent architectures can be computationally intensive, especially when dealing with large datasets of historical documents. Optimizing training procedures, choosing appropriate hyperparameters, and leveraging hardware acceleration are essential for efficient training.
5. **Model Interpretability**: While achieving high accuracy in text recognition and transliteration is important, it is equally essential to understand what the model has learned. This understanding aids in interpreting historical texts and verifying the transliteration accuracy against ground truth.
6. **Generalization and Robustness**: Ensuring that the implemented model **generalizes well to new, unseen historical documents** is crucial. Techniques such as **data augmentation, transfer learning,** and **cross-validation** can enhance the model's robustness and performance on diverse historical texts.
7. **Conditional Architectures for Specialized Tasks**: As part of future development, exploring **conditional convolutional-recurrent architectures** tailored for specific tasks, such as transliteration with language constraints or handling historical document styles, can further improve performance and adaptability.
8. **Continuous Learning and Improvement**: Continuous learning, experimentation with new techniques, and incorporating feedback from domain experts are essential for ongoing improvement and refinement of the system.

# Workflow and Detailed Execution Strategy

In this section, I will describe the strategy and process I will use for my project. The specific steps for carrying out the project will be outlined in this section. Additionally, I will work closely with my mentors to review and improve the plan's effectiveness throughout the project's duration.

As discussed above, this project will be divided into 2 Major parts. I will explain each of them in greater detail and how I plan to achieve the same:

## 1) Development of Hybrid End-to-End Models

As discussed above, I will implement CRNN Model during the project timeline. My working methodology will remain like the test assignment, but I will practice the following guidelines:
1) I will make a **documentation** for the model that I am working upon.

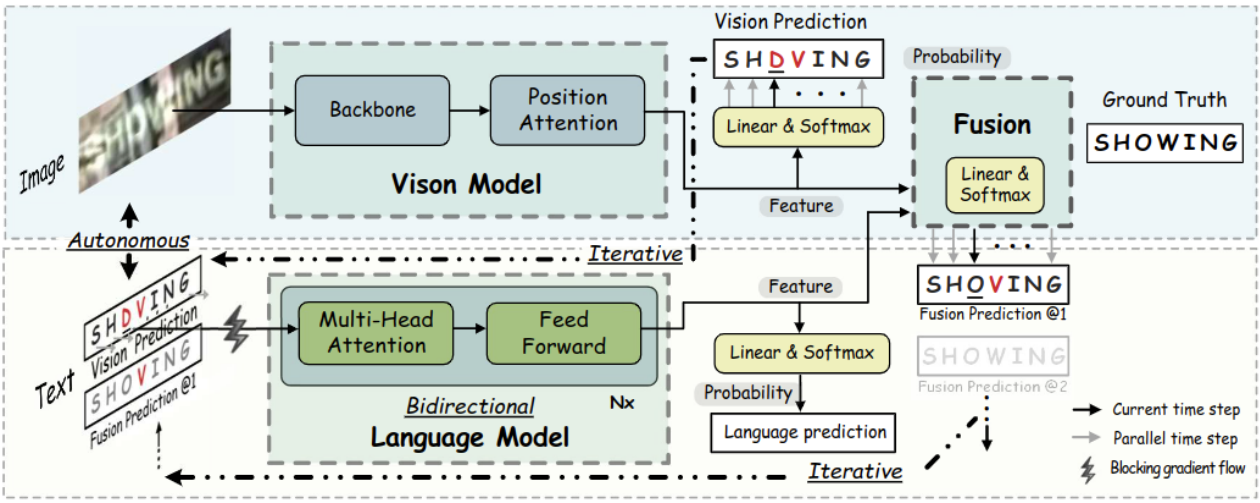
2) I will also design my code structure in a way that is **reproducible**, **easy to understand** and abide by the **DRY principle**.

I have already implemented a **CRNN Model** for **Optical Character Recognition** as part of my test assignment for the **HumanAI Organization's** Project (Link to the GitHub hosted code for the same can be found here.)

## The Model

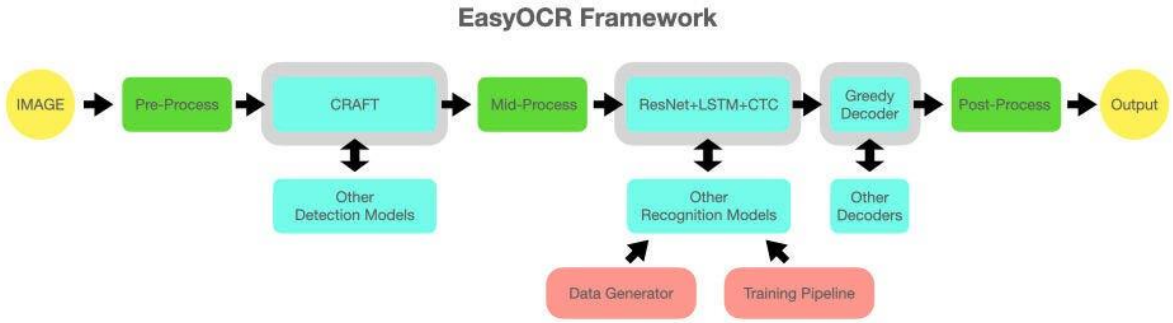
Defining the model and its architecture will be done like how I approached my test assignment, discussed above.
One of the possible improvements to my model can be brought about by

adding a **parallel Transformer/GPT model** to test for the existence of the predicted word. This would lead to higher accuracies and better performance. A sample architecture is drawn below for reference.

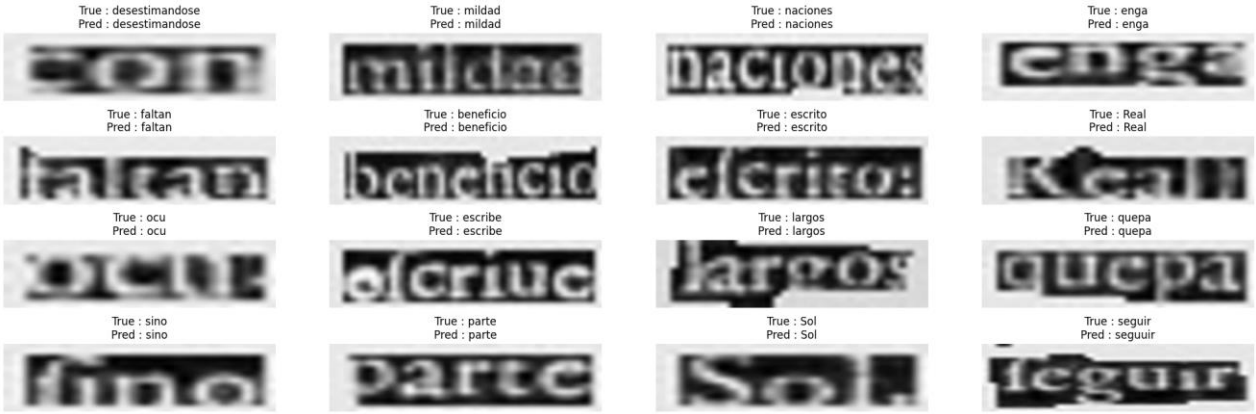



A similar model development is seen in case of the famous Python library EasyOCR used for the purpose of text recognition in English.

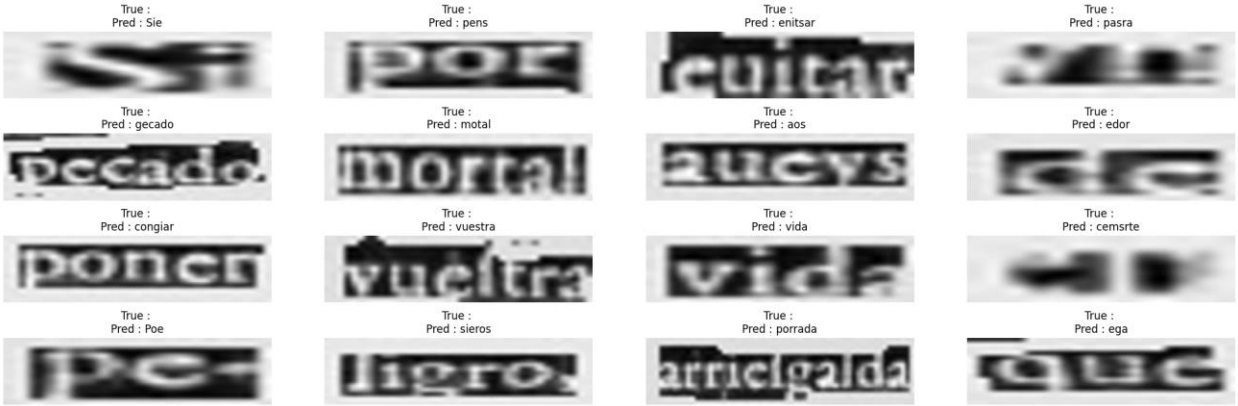
1. Image pre-processing is done, and the processed image is fed into a CRAFT detection model, which signifies the presence of text in a region.
2. The detected text is fed into a **CNN-RNN** like **convolutional to sequential model** for image to text recognition.
3. The output of the RNN model is then fed into a decoder to test for its existence, and a loss is calculated.
4. The most similar word is output as the prediction, if the predicted text from previous model doesn't exist in English vocabulary.




# Training my CRNN Model

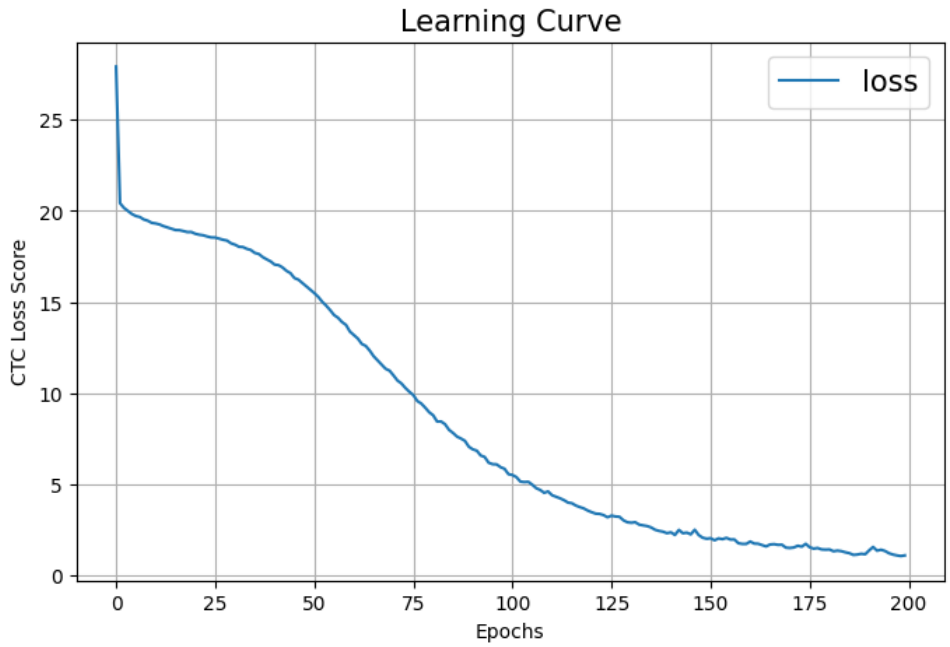



# Running and Testing my Model for Optical Character Recognition (Epoch 200)



# 2) Achieving High Accuracy

The ultimate objective is to train machine learning models capable of extracting text from seventeenth-century Spanish printed sources with at least **80% accuracy**. This entails extensive **experimentation, hyperparameter tuning, and dataset curation** to ensure the models generalize well across various styles, fonts, and degradation levels present in historical documents. Achieving this goal will signify a significant advancement in text recognition, particularly in the context of preserving and analyzing ancient textual artifacts. To accomplish this, **I plan to generate many augmentations of the original text samples.**




I would also like to experiment with the following during my project development phase:

1) **Hyperparameter Tuning**: Experimenting with various hyperparameters such as **learning rate, batch size, optimizer choice, and weight initialization methods** can significantly impact

the performance of the CRNN model. **Fine-tuning** these parameters through systematic experimentation is essential for optimizing model performance.

2) **Exploration of Sequential Models**: Investigating different sequential models like **Bidirectional LSTM, Vanilla RNN, Transformers, and their variants** can provide insights into their suitability for the task at hand. Comparing their performance in terms of accuracy, training time, and computational efficiency can guide the selection of the most appropriate architecture.
3) **Variation in Convolutional Layers**: Adjusting the **number of convolutional layers** and their **kernel sizes** can influence the model's ability to extract hierarchical features from input data. Experimenting with different architectures, including variations in the **number and depth of convolutional layers**, will help identify the optimal configuration for maximizing performance.
4) **Integration of Dropout Layers**: Incorporating **dropout layers** at various stages of the network can **mitigate overfitting and improve generalization** performance and help find the optimal balance between regularization and model capacity.
5) **Exploration of Pooling Techniques**: Assessing the impact of different pooling techniques such as **max pooling, average pooling, and global pooling** on the model's feature extraction capabilities can enhance performance. Experimenting with varying pool sizes and strides can help optimize the pooling strategy for the specific task.
6) **Integration of Group Normalization**: Experimenting with group normalization layers as an **alternative to batch normalization** can provide insights into their effectiveness in **stabilizing training and improving convergence speed**. Comparing the performance of different normalization techniques can inform the selection of the most suitable approach for the CRNN model.

7) **Upscaling and Down sampling Strategies**: Exploring different strategies for upsampling and downsampling, such as **bilinear interpolation, nearest neighbor interpolation, and transposed convolutional layers**, can impact the model's ability to capture spatial information and handle varying input resolutions effectively.
8) **Regularization Techniques**: Besides dropout layers, experimenting with other regularization techniques such as **L1 and L2 regularization, weight decay,** and **early stopping** can further improve the generalization performance of the CRNN model.
9) **Transfer Learning and Pre-trained Models**: Leveraging pre-trained models and transfer learning techniques, especially for the convolutional layers, can accelerate training and improve performance, particularly when dealing with limited annotated data. Fine-tuning pre-trained models on domain-specific datasets can help capture relevant features for the CRNN task.

# Why HumanAI?

Choosing HumanAI for Google Summer of Code 2024 aligns with my passion for both machine learning and the domains of arts and humanities. Some factors that contribute to my decision are as follows:

1. **Interest in Machine Learning:** During my programming journey for the 2 years in my college I have developed immense interests in Machine Learning and have undertaken various projects, including **Visual Question Answering systems** and **RNN, GPT, Transformer, CNN** based models, honing skills in **Natural Language Processing, Computer Vision, and Deep Learning**. These experiences have equipped me with the technical proficiency required for HumanAI project, allowing me to contribute meaningfully to the organization's goals. Additionally, as an **active member of the Club of Programmers (Intelligence group)-Machine Learning Engineering at IIT BHU**, I engage in ML discussions and activities, further enriching my understanding and expertise in the field.
2. **Passion for Arts and Humanities:** I harbor a **profound passion for the arts and humanities**, igniting my curiosity and driving me to explore the fusion of **machine learning (ML) and social sciences.** My goal is to harness ML techniques to unravel the complexities of human behavior and societal dynamics, contributing to **societal advancement through HumanAI.** Active participation in the **Social Service Club (SCS)** at **IIT BHU** has provided me with invaluable insights into societal issues, intensifying my passion and motivation.
3. **Interactive and Supportive Environment:** Interacting with HumanAI 's members has provided valuable insights into the organization's culture and values. I have found the community to be welcoming, supportive, and eager to foster growth among its members. The

opportunity to engage in meaningful discussions, seek guidance from experienced mentors, and collaborate with like-minded individuals greatly appeals to me. I believe that the interactive and supportive environment within HumanAI will not only facilitate my personal and professional development but also enhance the overall GSoC experience.

4. **Alignment with Long-term Goals:** Participating in GSoC with HumanAI is not merely a short-term endeavor but a steppingstone towards my long-term goals. I envision a future where I continue to contribute to cutting-edge research at the **intersection of machine learning and social science domains**, leveraging technology to tackle pressing challenges and drive meaningful innovation. HumanAI provides an ideal platform to nurture these aspirations, offering unparalleled opportunities for learning, growth, and impact.

# Why am I an ideal contributor?

1. **Dedicated Time and Commitment:** I am fully committed to dedicating my time and effort to this project for the next 12 weeks (about 3 months). I have **no other major commitments**, ensuring my full focus on meeting and exceeding project expectations within the given time limit.
2. **Active Participation in Open Source:** I am deeply involved in open-source communities, showcasing my teamwork and tech skills. I have made **7 pull requests** of which **3 have been merged** and have raised **6 issues**. Beyond code, I actively discuss, offer feedback, and solve problems collaboratively.
3. **Advanced Expertise in CV and NLP:** With my proficiency in **CV and NLP**, like Supervised Learning for image classification, text classification, image to text conversion, Self-Supervised Learning (SSL)

and Diffusion models, I am well-equipped to address the challenges of Optical Character Recognition. I excel in refining classification accuracy and generating realistic simulations, pushing the boundaries of text in image recognition with technical prowess and creative thinking.

4. **Proven Track Record in HumanAI:** With a **Successfully completed** task in HumanAI organization, I have established a strong record of accomplishment of contributing to machine learning. My past experiences have equipped me with valuable insights into the specific challenges and requirements of HumanAI initiatives, enabling me to hit the ground running and **make meaningful contributions from day one.**
5. **Engagement in Academic and Technical Communities:** As an active member of the **Club of Programmers (Intelligence Group)** as well as the **Social Services Club** at **IIT BHU**, I am deeply immersed in the intersection of Machine Learning and Social Sciences- Which are the founding principles of **RenAIssance** Project by HumanAI organization. By tapping into this rich pool of knowledge and expertise, I can approach the project from multiple angles, uncovering innovative solutions and driving meaningful progress.

# Project Timeline

I will devote 2 hours per day for 12 weeks (about 3 months) during my summer holidays and around 2-3 hours each day during the final week. This sums up to **175 hours** for the entire project. The work distribution has been written in a tabular form for better understanding.

| | | | |
|---|---|---|---|
| **Community Bonding and Onboarding (May 1, 2024 - May 26, 2024)** | | | |
| Pre-Coding | May 1- May 26 | 1) Exploring all the available Techniques and Models for Optical Character Recognition and **choosing the best CRNN Model** for the development.<br>2) Discussing my project with my mentors and the community to see if more **improvements** can be made in the plan of action.<br>3) I will learn all the etiquette for software development and testing followed by HumanAI to maintain compatibility.<br>4) I will start my **first blog** documenting my journey with HumanAI. | |
| **Project Period (May 27, 2024 - July7, 2024)** | | | |
| Week1 - Week3 | May 27- June 2 | Review relevant **research papers**, and additional ones suggested by mentors, focusing on AI techniques for OCR. | |
| | June 3- June 9 | Gain a thorough understanding of deep learning architectures suitable for character detection and recognition, with a focus on **Convolutional and Recurrent Neural Networks**. | |
| | June 10- June 16 | Develop and implement a **base model** with deep learning CRNN architecture tailored for character detection and recognition. | |

| | | | |
|---|---|---|---|
| Week 4-Week 6 | June17-June23 | Gather historical printed data, including Spanish printed data. **Augment the dataset** by incorporating relevant information from complementary sources, ensuring data diversity. | |
| | June24-June30 | **Train the model** on the preprocessed dataset, optimizing **performance metrics**, validating and adjusting **Hyperparameters.** | |
| | July31-July7 | Experiment with different configurations of **model layers and architectures** to assess their impact on performance. **Fine-tune model** architecture, based on experimental results to enhance performance. | |
| **Midterm Evaluations (July8, 2024 - July12, 2024)** | | | |

| **Project Period (July8, 2024 - Aug18, 2024)** | | | |
|---|---|---|---|
| Week 7-Week 9 | July 8-July 14 | **Document the development process**, including model architecture, model implementation and training procedures for reproducibility. | |
| | July 15-July 21 | Explore innovative approaches to enhance the novelty of the developed model, such as incorporating novel loss functions like **novel CTC loss or regularization techniques**. **Seek feedback** from mentors and experts to identify areas for further innovation and improvement. | |
| | July 22-July 28 | Perform **final optimizations** on the deep learning architecture, focusing on improving efficiency and scalability for handling large-scale historical datasets. | |

| | | | |
|---|---|---|---|
| Week 10 -Week12 | July 29- Aug 4 | Extend model application to **other historical data,** ensuring compatibility and scalability. Generate metrics for each dataset to evaluate the generalizability and robustness of the developed model across diverse data sources. | |
| | Aug 5- Aug 11 | Incorporate any necessary changes or **refinements** based on the results and feedback. Continuously iterate on the model to address emerging challenges and improve overall performance to make the **final model**. | |
| | Aug 12- Aug18 | **Complete the documentation** by including final model architecture, optimized hyperparameters, and performance metrics. | |
| Final Week | Aug 19- Aug 26 | 1) Finishing up the documentation work.<br>2) Complete any remaining work and prepare my final report.<br>3) Make my **2nd blog** for the completion of my GSOC'24 Project. | |
| **Final Evaluations (Aug19, 2024 - Aug26, 2024)** | | | |

# Post GSoC

As a dedicated contributor to the HumanAI organization's GSoC'24 project named **"Automating Text Recognition and Transliteration of Historical Documents with convolutional - recurrent architectures"**, my commitment to the project would not end with GSoC'24. An open-source project's success depends on its community's contributions and continuous improvement. Hence, I would like to continue to contribute to the project even after GSoC by staying in touch with the community, attending meetings, and collaborating with other contributors. I would ensure that the **codebase remains up to date** with the latest technologies, **fix bugs, and add new features** as per the project's needs. To accomplish this, I would work closely with the project's maintainers and other contributors, providing guidance and support wherever needed.