Signboard Translation from Vernacular Languages

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

by

17BEC0003 - Karveandhan P

17BEC0055 - Sachin K

17BEC0129 – Shivaani K

Under the guidance of

Prof. Sankar Ganesh S

School of Electronics Engineering

VIT, Vellore

May, 2021

DECLARATION

I hereby declare that the thesis entitled "Signboard Translation from Vernacular Languages" submitted by me, for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering to VIT is a record of bonafide work carried out by me under the supervision of Prof. Sankar Ganesh S.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date : 27

May 29, 2021

Karveandhan

Sachin

Shivaani

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Signboard Translation from Vernacular Languages" submitted by Karveandhan(17BEC0003), Sachin K(17BEC0055), Shivaani K(17BEC0129), **School of Electronics Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 02. 2021 to 30.05.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The thesis fulfills the requirements and regulations ofthe university and in my opinion meets the necessary standards for submission.

Place: Vellore — Sankar Ganesh S

Date: 27th May 2021 — **Signature of the Guide**

Internal Examiner — **External Examiner**

Dr. Prakasam P

Electronics and Communication Engineering

ACKNOWLEDGEMENTS

It is a privilege to express my sincerest regards to my project coordinator, Dr. Sankar Ganesh S, for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of my project. I deeply express my sincere thanks to my School Dean and HOD and the University Management for encouraging and allowing me to present the project on the topic "Signboard Translation from Vernacular Languages".

I also take this opportunity to thank all my lecturers who have directly or indirectly helped me in my project. Furthermore, I would like to express my gratitude and appreciation to all those who gave me the possibility to complete this report.

Last but not the least I express my thanks to my friends for their cooperation and support.

KARVEANDHAN P

SACHIN K

SHIVAANI K

Executive Summary

There are numerous challenges faced by people who travel to different cities/states within India. As there are more than 20 constitutional languages in India, people of one state will not be able to understand the language in the other states and hence reading and understanding the signboards while travelling to other states will be a major problem faced by them. A solution to this problem can create a whole new pleasant experience during the time of travel. Also, the tourist might recognize important monuments and places to visit without the help of any other native speakers. It can also help everyone to navigate and find essential locations such as Hospitals, Police Station, Railway Stations in case of emergency. They don't have to stop their vehicle and ask others for the direction. In order to find a solution for this language barrier problem, we have developed an efficient signboard transliteration system. The first step is to detect the presence of text in the signboard and to classify to its respective languages. To perform this, we make use of an object detection algorithm Yolov4 tiny and EAST algorithm. Then, to extract and recognize the text/string from the image we use OCR tools like PyTesseract and EasyOCR. To further transliterate the extracted text into the desired language, we make use of a Seq2Seq Encoder Decoder model with Attention mechanism.

CONTENTS **Page No.**

List of Figures

List of Tables

List of Abbreviations

CNN	-	Convolutional Neural Network
DL	-	Deep Learning
ML	-	Machine Learning
YOLO	-	You only look once (Algorithm)
V4	-	Version 4
OCR	-	Optical Character Recognition
EAST	-	Efficient and Accurate Scene Text Detector
ANN	-	Artificial Neural Network
FC	-	Fully connected (dense) layer
EO	-	Encoder output
H	-	Hidden state
X	-	Input to the decoder

Symbols and Notations

α_{ts}	Attention Weights
c_t	Context Vector
a_t	Attention Vector
h_t	Target Hidden State
h_s	Source Hidden State

1. INTRODUCTION

1.1. OBJECTIVE

The goal of this project is to transliterate the text written on a signboard to another language as desired by the user (Typically 1 language transliteration due to the complexity involved). We will first design a system which works for names (such as road names, city names, organisation names shop names etc.) which typically contain 1-2 words and are rarely longer than 4-5 words. We implemented this using Machine Learning algorithms which performs object detection, text recognition, optical character recognition tools and Seq2Seq modelling.

1.2. MOTIVATION

India has 22 constitutionally recognized languages written in 13 different scripts. An average traveler, on a business or pleasure trip, often gets confused by the various signboards written in an unfamiliar language in a new region. This often spoils the experience of visiting a new place and the traveler goes back with not-so-fond memories. This often leads to language tussles wherein people of region A may feel that people of region B are not considerate enough to display signboards in their language and vice versa. In reality, this is simply a logistics problem. It is just impossible to have every signboard in every city/town/village across the country written in 22 different languages. The real estate available on the signboard may allow the text to be written in 2-3 languages only. Hence there are bound to be many languages which will be left out. The resulting unfortunate, unpleasant and unproductive bitterness can easily be avoided by building better platforms. We believe that our project will allow people from different regions to read and understand text written in native languages as they are traveling across the country.

1.3. BACKGROUND

The Project can be split into three major divisions

- Text region detection
- Recognition of individual characters
- Natural Language Processing for translation/transliteration

With Deep Learning, Problem when considered as object detection, is mainly handled in two ways. The first approach is detecting word by word and then processing to get the characters in each word. The second approach is detecting each character first and then group them into words. Another Deep Learning approach is to classify pixel-wise if it belongs to a text or not and then use 8-Nearest neighbour approach to instance segment each character

The survey in [1] aims at summarizing and analysing the major changes and significant progresses of scene text detection and recognition in the deep learning era. In this article, new insights and ideas, recent techniques, benchmarks and future trends of text detection and recognition are discussed.

In [2], an approach based on colour channel selection for text recognition from scene images and video frames is proposed. In this, a colour channel is automatically selected and then selected colour channel is considered for text recognition. From each sliding window of a colour channel, the colour-channel selection approach analyses the image properties from the sliding window and then a multi-label Support Vector Machine classifier is applied to select the colour channel that will provide the best recognition results in the sliding window. This colour channel selection for each sliding window has been found to be more fruitful than considering a single colour channel for the whole word image.

In [4], a novel method for machine transliteration is proposed based on Deep Belief Networks, which despite not having competitive results can be an important additional cue for system combination setups. Common approaches use finite state machines or other methods similar to conventional machine translation. Instead of using conventional NLP techniques, the approach presented in this builds on a technique which is shown to work well for other machine learning problems.

2. PROJECT DESCRIPTION AND GOALS:

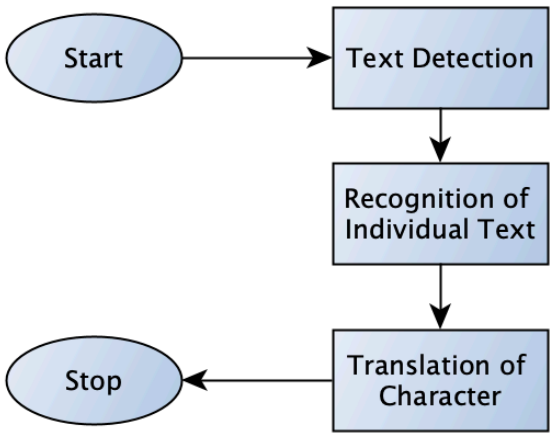

Fig 2.1: Block Diagram of Signboard Translation Process

Phase 1 (Detection of text from the image):

Text Recognition - Just like detecting real life objects, we detect text here with classification as Languages. The images involving text are annotated by drawing boundary boxes and labelling them to their respective classes (Languages).

There are numerous languages with which classification can be performed. Here, in this project we predominantly work on Hindi and English. So, we classify the images given into these two categories.

The Algorithms used in this process are:

- YOLO v4
- EAST

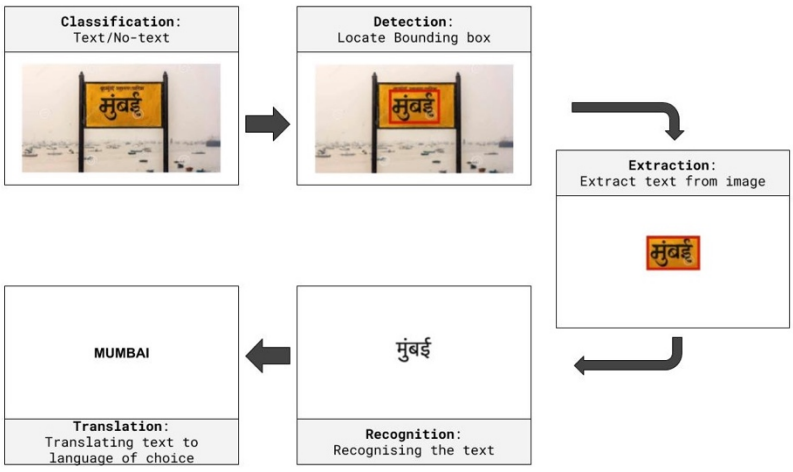

Fig 2.2: Step by step process involved in extracting the text and translating to the desired language.

Phase 2 (Recognising String):

- After detecting the region of text from the image, the text has to be extracted separately. In order to extract the text from the image, we used OCR tools such as

1. PyTesseract
2. EasyOCR

- PyTesseract is used for detecting the English text from the images.
- EasyOCR is used for detecting Multiple Language (English and Hindi for our Purpose) from the Image.

Phase 3 (Transliteration):

After Phase 2 of recognition, the text has to be transliterated to the desired language of user's interest. To implement this we make use of Encoder Decoder model with attention. A typical Seq2Seq model consists of an encoder and a decoder which are themselves two separate neural networks combined into a single giant network. Both encoder and decoder are typically GRU models. Due to the complexity involved and unavailability of datasets in the project, we were able to implement transliteration from English to Hindi only. As Hindi is predominantly spoken language in India, we prioritised English to Hindi transliteration.

3. TECHNICAL SPECIFICATION:

- Platform for Execution : Google Colab
- Source for Datasets : Ai4bharat
- Performing Image Processing Techniques : Roboflow
- Algorithms Involved in the Project :
    - Phase 1: Detection of text from the image : Yolo-v4 Tiny, EAST
    - Phase 2: Recognising String : PyTesseract, EasyOCR
    - Phase 3: Transliteration : Encoder Decoder Model with Attention
- Programming Language : Python

Deep learning:

For deep learning we should focus more on GPU/vRAM than CPU. Average System Specifications for two layer Inception model needs i5 processor +4gb RAM and also it takes about 24-30 mins to run .Higher Specifications are always nice or we can also use cloud platform. Clouds are always fast and have enough compute capability that we need.The inception model for deep dream for 2 years comes on within 1-2 secs in Google cloud, and it takes around 5-6 mins for 5+layers.

Machine learning:

Machine learning is a subset of artificial intelligence function that provides the system with the ability to learn from data without being programmed explicitly. For machine learning we need a lots of RAM and a fast CPU to run a program .Machine learning need a minimum RAM of 32gb. If your task is small u can fit in a complex sequential processing, you don’t need a big system. You can skip the GPU’s altogether. If your task is a bit intensive and has a manageable data a reasonably powerful GPU would be used.

4. DESIGN APPROACH AND DETAILS .

4.1 Design Approach / Materials & Methods

STAGE 1: Detection of Text in the Signboards

The most fundamental step of the project is to first detect the text in the signboard and draw a bounding box over the text. Secondly, this text can be classified into numerous languages. To perform this recognition and classification we have made use of Yolo-v4 tiny algorithm.

YOLO-v4 Tiny : You Only Look Once – v4 Tiny

Yolo divides the image into equal grids and determines the probability on an individual grid being an object or a centre of an object. And later makes use of parameters such as w – width, h - height, x – x-coordinate, and y – y coordinate, and the algorithm expands this initial co-ordinate to determine the complete co-ordinate of the object and matches it to its class by increasing the probability of the relevant class (which are Languages). Most methods the model to an image at multiple locations and scales. High scoring regions of the image are considered detections. Yolo, on the other hand, applies a single neural network to the full image. The network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

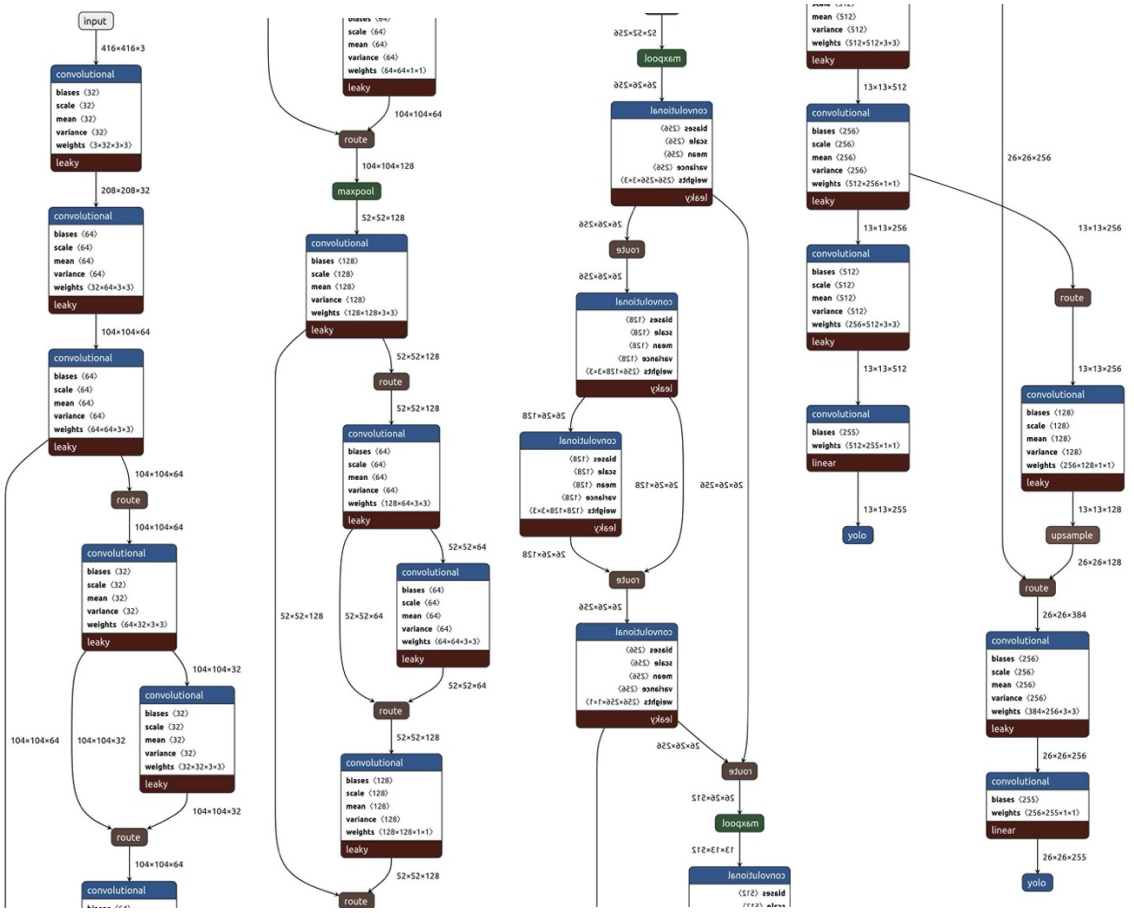

Fig 4.1: Yolov4 Tiny Architecture

We can use YOLOv4-tiny for much faster training and much faster detection. It has only two YOLO heads as opposed to three in YOLOv4 and it has been trained from 29 pre-trained convolutional layers as opposed to YOLOv4 which has been trained from 137 pre-trained convolutional layers.

Yolov4-tiny builds on the progress of yolov4, but emphasizes model speed and a smaller model size for inference even on smaller hardware. This model can be implemented when it has to be implemented on a constrained compute environment such as a mobile application, embedded device inference, constrained training resources. While our project aims to be a part of every citizens' smartphone, yolo-v4 tiny would be the best model to work on constrained environment like Mobile Phones with greater speed. YOLOv4-tiny is especially useful if you have limited compute resources in either research or deployment, and are willing to trade-off some detection performance for speed.

Since we need to extract and transliterate after text detection, we have made use of another well-established algorithm called EAST algorithm.

Stage 2: Text Detection and Recognition

EAST: An Efficient and Accurate Scene Text Detector

The core of text detection is the design of features to distinguish text from backgrounds. The EAST algorithm [6] is a scene text detection pipeline that has two stages.

- Firstly, the pipeline utilizes a fully convolutional network (FCN) model that directly produces word or text-line level predictions, excluding redundant and slow intermediate steps.
- Secondly, the produced text predictions which can be either rotated rectangles or quadrangles, are sent to Non-Maximum Suppression (a post-processing step to merge the nearby text detections) to yield final results.

The key component of the proposed algorithm is a neural network model, which is trained to directly predict the existence of text instances and their geometries from full images. The model is a fully-convolutional neural network adapted for text detection that outputs dense per-pixel predictions of words or text lines. This eliminates intermediate steps such as candidate proposal, text region formation and word partition. The post-processing steps only include thresholding and NMS on predicted geometric shapes.

EAST Algorithm description:

- The FCN can be decomposed into three parts: feature extractor stem, feature-merging branch and output layer.
- An image is fed into the FCN and multiple channels of pixel-level text score map and geometry are generated.
- One of the predicted channels is a score map whose pixel values are in the range of [0, 1].
- The remaining channels represent geometries that enclose the word from the view of each pixel.
- The score stands for the confidence of the geometry shape predicted at the same location.
- Two geometry shapes are used for text regions, rotated box (RBOX) and quadrangle (QUAD), and different loss functions are designed for each geometry.
- Thresholding is then applied to each predicted region, where the geometries whose scores are over the predefined threshold is considered valid and saved for later non maximum-suppression. Results after NMS are considered the final output of the pipeline.

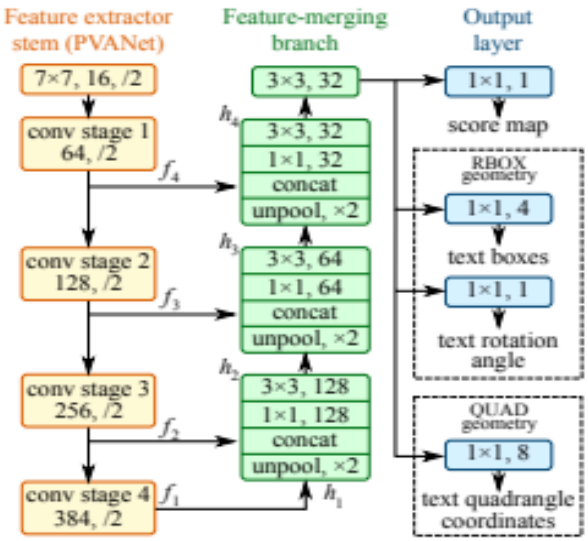

Fig 4.2: Structure of the text detection FCN

- In the feature-merging branch, we gradually merge the feature maps using,

$$g_i = \begin{cases} \text{unpool}(h_i) & \text{if} \quad i \leq 3 \\ \text{conv}_{3\times3}(h_i) & \text{if} \quad i = 4 \end{cases}$$

$$h_i = \begin{cases} f_i & \text{if} \quad i = 1 \\ \text{conv}_{3\times3}(\text{conv}_{1\times1}([g_{i-1}; f_i])) & \text{otherwise} \end{cases}$$

where g_i is the merge base, and hi is the merged feature map, and the operator $[\cdot;\cdot]$ represents concatenation along the channel axis. In each merging stage, the feature map from the last stage is first fed to an unpooling layer to double its size, and then concatenated with the current feature map. Next, a $\text{conv}_{1\times1}$ bottleneck cuts down the number of channels and reduces computation, followed by a $\text{conv}_{3\times3}$ that fuses the information to finally produce the output of this merging stage. Following the last merging stage, a $\text{conv}_{3\times3}$ layer produces the final feature map of the merging branch and feed it to the output layer.

- For a quadrangle $Q = \{p_i | i \in \{1, 2, 3, 4\}\}$, where $p_i = \{x_i , y_i\}$ are vertices on the quadrangle in clockwise order. To shrink Q, we first compute a reference length r_i for each vertex p_i as

$$r_i = \min(D(p_i, p_{(i \bmod 4)+1}), D(p_i, p_{((i+2) \bmod 4)+1}))$$

- The loss can be formulated as,

$$L = L_s + \lambda_g L_g$$

where L_s and L_g represents the losses for the score map and the geometry, respectively, and λ_g

weighs the importance between two losses.

- Loss for Score Map (L_s) can be formulated as,

$$L_s = \text{balanced-xent}(\hat{\mathbf{Y}}, \mathbf{Y}^*) \\ = -\beta \mathbf{Y}^* \log \hat{\mathbf{Y}} - (1-\beta)(1-\mathbf{Y}^*) \log(1-\hat{\mathbf{Y}})$$

where $\hat{Y} = F_s$ is the prediction of the score map, and Y^* is the ground truth. The parameter β is the balancing factor between positive and negative samples, given by

$$\beta = 1 - \frac{\sum_{y^* \in \mathbf{Y}^*} y^*}{|\mathbf{Y}^*|}.$$

- Loss for Geometries(L_g) can be formulated as,

Let all coordinate values of Q be an ordered set

$$C_{\mathbf{Q}} = \{x_1, y_1, x_2, y_2, \ldots, x_4, y_4\}$$

then the loss can be written as

$$L_g = L_{\text{QUAD}}(\hat{\mathbf{Q}}, \mathbf{Q}^*) \\ = \min_{\tilde{\mathbf{Q}} \in P_{\mathbf{Q}^*}} \sum_{\substack{c_i \in C_{\mathbf{Q}}, \\ \tilde{c}_i \in C_{\tilde{\mathbf{Q}}}}} \frac{\text{smoothed}_{L1}(c_i - \tilde{c}_i)}{8 \times N_{\mathbf{Q}^*}}$$

where the normalization term N_{Q^*} is the shorted edge length of the quadrangle, given by

$$N_{\mathbf{Q}^*} = \min_{i=1}^{4} D(p_i, p_{(i \bmod 4)+1}),$$

and P_Q is the set of all equivalent quadrangles of Q^* with different vertices ordering [6].

Procedure:

- Firstly, we load the already available trained EAST model in our code.
- Then, we resize the input image to be a factor of 32, since that was the convention used in the original EAST model.
- Then, we find the confidence scores to find how confident the model is that the particular pixel in the image is actually containing text.
- Most of the times, the texts are not written straight. So we find the angle in which the text box has to be drawn according to the text orientation.
- Then, we find the coordinates of the text detector box which is to be drawn around the text.
- Then, we use the optical character recognition tool called Pytesseract to extract the text from the image.

STAGE 3: Transliteration:

An encoder and a decoder, which are two distinct neural networks integrated into a single giant network, makes up a standard Seq2Seq model. Generally, both the encoder and the decoder are LSTM or GRU models. These Seq2Seq models are used in our project for Neural machine transliteration .The encoder network's task is to comprehend the input sequence and convert it to a smaller dimensional representation, which is then sent to the decoder network, which generates the output. An encoded sentence is fed into the encoder (in order to perform neural machine translation). We employ a technique known as Teacher Forcing, in which the real output (rather than the expected output) from the previous time stamp is fed into the current time stamp as input. A big flaw in a simple Seq2Seq model without attention mechanism is that it tends to forget the first part of the sequence as it progresses. It isn't useful while translating longer sentences. So, we make use of attention mechanism in our project.

After training this sequence to sequence model (*Seq2Seq*), we will be able to input an English word, such as "aakash"*, and return the Hindi translation: "*आकाश*"*.

Pre-processing:

- Add start as 0 and end as 1 token to each sentence.
- Clean the sentences by removing special characters.
- Create a word index and reverse word index (dictionaries mapping from word → id and id → word).
- Pad each sentence to a maximum length.

Encoder and decoder model:

We implement an encoder-decoder model with attention mechanism. The following diagram shows that each input word is assigned a weight by the attention mechanism which is then used by the decoder to predict the next word in the sentence. The following picture and formulas illustrate attention mechanism:

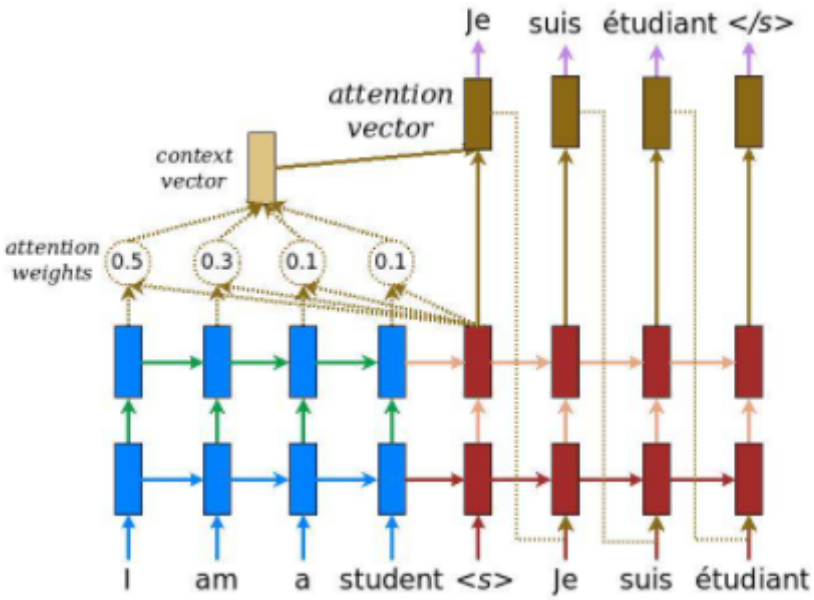

Fig 4.3: Encoder Decoder Model with Attention

The attention computation happens at every decoder time step. It consists of the following stages:

- The current target hidden state is compared with all source states to derive attention weights.
- Based on the attention weights we compute a context vector as the weighted average of the source states.
- Combine the context vector with the current target hidden state to yield the final attention vector
- The attention vector is fed as an input to the next time step (input feeding). The first three steps can be summarized by the equations below:

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]}$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]}$$

Here, the function score is used to compare the target hidden state h_t with each of the source hidden states h_s, and the result is normalized to produce attention weights. There are various choices for the scoring function; popular scoring functions include the multiplicative and additive forms such as,

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases}$$

We use Bahdanau attention for the encoder and decoder model.

Notations for Bahdanau attention:

- FC = Fully connected (dense) layer
- EO = Encoder output
- H = hidden state
- X = input to the decoder

Pseudo-code for Bahdanau attention:

- score = FC(tanh(FC(EO) + FC(H)))
- attention weights = softmax(score, axis = 1). Softmax by default is applied on the last axis but here we want to apply it on the 1st axis, since the shape of score is (batch_size, max_length, hidden_size). Max_length is the length of our input. Since we are trying to assign a weight to each input, softmax should be applied on that axis.
- context vector = sum(attention weights * EO, axis = 1). Same reason as above for choosing axis as 1.
- embedding output = The input to the decoder X is passed through an embedding layer.
- merged vector = concat(embedding output, context vector)
- This merged vector is then given to the GRU.

Training the encoder and decoder model:

- We pass the input through the encoder which returns the encoder output and the encoder hidden state.
- The encoder output, encoder hidden state and the decoder input (which is the start token) is passed to the decoder.
- The decoder returns the predictions and the decoder hidden state.
- The decoder hidden state is then passed back into the model and the predictions are used to calculate the loss.
- We use teacher forcing to decide the next input to the decoder.
- Teacher forcing is the technique where the target word is passed as the next input to the decoder.
- The final step is to calculate the gradients and apply it to the optimizer and backpropagate.

Transliteration:

- We design a function called ‘evaluate’ which is similar to the training loop, except that we don't use teacher forcing here.
- The encoder output is calculated only once for one input.
- The input to the decoder at each time step is its previous predictions along with the hidden state and the encoder output.
- Stop predicting when the model predicts the end token.
- And store the attention weights for every time step.

Accuracy:

The average accuracy of our model is calculated and is found to be 0.9055351731601727

4.2 Constraints, Alternatives and Trade-offs:

- This project predominantly involves Hindi and English languages. This can be further extended to multiple languages when there are proper datasets available. Whenever proper datasets are made available, the model can be trained and developed with those languages in order to serve other native speakers.

- To reach all the citizens this project can be developed into a Mobile Application which is user friendly and available in all languages. This increases the usability to a broader range of people over the country.

- Detection of vertically aligned and curved text in images are not precisely detected.

- In Yolo-v4 tiny, the rectangular bounding box might cover unnecessary information in the background apart from the text. To overcome this limitation, we used EAST algorithm which makes use of rotated bounding boxes based on the angle in which the text is aligned. Therefore, removing unnecessary information from the background.

5. SCHEDULE, TASKS AND MILESTONES:

February	March	April	May
Collection of Datasets.	Phase 1 : text detection using Yolo and East successfully implemented.	Phase 3: Transliteration	Final Thesis preparation.
Annotation of all the datasets.	Phase 2: Text Recognition from the image with boundary box using PyTesseract and EasyOCR	Preparation of Poster for the developed project.	
Beginning of Phase 1: Text Boundary Detection using Yolo and East.		Drafting the report for the implemented project.	

Table 5.1: Timeline of our project

6. PROJECT DEMONSTRATION

Sample of Training Data Set for Boundary Box Detection and Language Classification:

Fig 6.1: Sample Training Datasets for Yolo. (Green annotates Hindi, Pink annotates English)

Fig 6.2: Sample Output after detecting the text in the Signboard and the Language in which the text is written

Sample of Training Data for the model to perform Transliteration (Encoder Decoder Model with Attention):

```
<Name ID="1">
<SourceName>RAASAVIHAAREE</SourceName>
<TargetName ID="1">रासविहारी</TargetName>
</Name>
<Name ID="2">
<SourceName>DEOGAN ROAD</SourceName>
<TargetName ID="1">देवगन रोड</TargetName>
</Name>
<Name ID="3">
<SourceName>SHATRUMARDAN</SourceName>
<TargetName ID="1">शत्रुमर्दन</TargetName>
</Name>
<Name ID="4">
<SourceName>MAHIJUBA</SourceName>
<TargetName ID="1">महिजुबा</TargetName>
</Name>
<Name ID="5">
<SourceName>SABINE</SourceName>
<TargetName ID="1">सैबिन</TargetName>
</Name>
<Name ID="6">
<SourceName>BILL COSBY</SourceName>
<TargetName ID="1">बिल कॉस्बी</TargetName>
</Name>
<Name ID="7">
<SourceName>RISHTA KAGAZ KA</SourceName>
<TargetName ID="1">रिश्ता कागज़ का</TargetName>
</Name>
<Name ID="8">
<SourceName>Hatim</SourceName>
<TargetName ID="1">हातिम</TargetName>
</Name>
<Name ID="9">
<SourceName>SHREEMAYI</SourceName>
<TargetName ID="1">श्रीमयी</TargetName>
</Name>
<Name ID="10">
<SourceName>FARIHAH</SourceName>
<TargetName ID="1">फरीहाह</TargetName>
```

Table 6.1: Sample Input Data for Training the Model

Sample Output after performing transliteration and calculating its accuracy:

```
aadhi आधी आधी 1.0
aakash आकाश आकाश 1.0
aap आप आप 1.0
aayasha आयशा आयशा 1.0
aayee आई आई 1.0
abduh अबदस अबदह 0.75
aberhart एबरहरट अबरहारट 0.83333333333
abey अबय एबी 0.3333333333333333
abou अब अबो 1.0
abri एबरी अबरी 0.75
academy अकडमी एकडमी 0.8
accommodation एकोमडशन ककोमोडशन 0.857
acorn एकोरन अकॉरन 0.6
adams एडमस एडमस 1.0
adgaon अदगाव अदगाव 1.0
adlabs एडलबस एडलाबस 1.0
agro एगरो एगरो 1.0
ahlaad आहलाद अहलाद 0.8
air एअर आयर 0.3333333333333333
aisin एसिन ऐसिन 0.75
ajee अजी अजी 1.0
ajhai अझाई अजह 0.25
ajith अजीथ अजिथ 0.75
akaram अकरम अकरम 1.0
akele अकल अकल 1.0
akhri आखिरी अखिरी 0.8
akodia अकोडिया अकोडिया 1.0
akurdi अकरडी अकरदी 0.8
alas अलास अलास 1.0
albany अलबान एलबनी 0.6
alester एलसटर अलसटर 0.8
alex एलकस अलकस 0.75
alipur अलीपर अलीपर 1.0
alkali अलकली अलकली 1.0
alpesh अलपश अलपश 1.0
saiful सफल सफल 1.0
saigal सगल सगल 1.0
sainte सट सट 1.0
saleh सालह सलह 0.75
sama समा समा 1.0
sammy समी समी 1.0
sanawbar सनवबर सनवबर 1.0
sanitary सनिटरी सनितारी 0.833333333331
sanket सकत सकित 1.0
sap सप सप 1.0
sapera सपरा सपरा 1.0
sariyah सारियाह सरियाह 0.857142857142
sarovar सरोवर सरोवर 1.0
satta सटटा सतता 0.5
sattaar सततार सततार 1.0
satte सतत सतत 1.0
savu साव सव 0.6666666666666666
sawar सवर सवार 1.0
saymukta सयमकता समकता 0.8333333333
schuyler सयलर शवलर 0.5
scissor सीजर ससिजर 0.75
scorpio सकॉरपियो सकोरपियो 0.875
scottish सकॉटिश सकॉटिश 1.0
sebastien सबसटियन सबसटीयन 0.8571428
seblat सिबलाट सबलात 0.6666666666666
secunderabad सिकदराबाद सकयडरबाद 0.66
seedha सीधा सीधा 1.0
seekh सीख सीख 1.0
sei सइ सी 0.5
seiki सीकी सिकी 0.75
seinfeld सनफीलड सिफलड 0.6666666666
selleck सीलक सलक 0.75
senior सीनियर सनियर 0.833333333333333
serjeant सरजट सरजीट 1.0
seth सठ सठ 1.0
shaanxi शाकसी शासकी 0.8
```

Table 6.2: Sample Testing Data Output with predicted accuracy

7. Results and Discussion

```
  layer   filters  size/strd(dil)      input                output
  0 conv     32       3 x 3/ 2    416 x 416 x   3 ->  208 x 208 x  32 0.075 BF
  1 conv     64       3 x 3/ 2    208 x 208 x  32 ->  104 x 104 x  64 0.399 BF
  2 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
  3 route  2                                 1/2 ->  104 x 104 x  32
  4 conv     32       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  32 0.199 BF
  5 conv     32       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  32 0.199 BF
  6 route  5 4                               ->  104 x 104 x  64
  7 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
  8 route  2 7                               ->  104 x 104 x 128
  9 max               2x 2/ 2    104 x 104 x 128 ->   52 x  52 x 128 0.001 BF
 10 conv    128       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
 11 route  10                                1/2 ->   52 x  52 x  64
 12 conv     64       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x  64 0.199 BF
 13 conv     64       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x  64 0.199 BF
 14 route  13 12                            ->   52 x  52 x 128
 15 conv    128       1 x 1/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
 16 route  10 15                            ->   52 x  52 x 256
 17 max              2x 2/ 2     52 x  52 x 256 ->   26 x  26 x 256 0.001 BF
 18 conv    256       3 x 3/ 1     26 x  26 x 256 ->   26 x  26 x 256 0.797 BF
 19 route  18                                1/2 ->   26 x  26 x 128
 20 conv    128       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 128 0.199 BF
 21 conv    128       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 128 0.199 BF
 22 route  21 20                            ->   26 x  26 x 256
 23 conv    256       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 256 0.089 BF
 24 route  18 23                            ->   26 x  26 x 512
 25 max              2x 2/ 2     26 x  26 x 512 ->   13 x  13 x 512 0.000 BF
 26 conv    512       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.797 BF
 27 conv    256       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
 28 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
 29 conv     27       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x  27 0.005 BF
 30 yolo
yolo] params: iou loss: ciou (4), iou_norm: 0.07, cls_norm: 1.00, scale_x_y: 1.05
ms_kind: greedynms (1), beta = 0.600000
 31 route  27                               ->   13 x  13 x 256
 32 conv    128       1 x 1/ 1     13 x  13 x 256 ->   13 x  13 x 128 0.011 BF
 33 upsample                 2x    13 x  13 x 128 ->   26 x  26 x 128
 34 route  33 23                            ->   26 x  26 x 384
 35 conv    256       3 x 3/ 1     26 x  26 x 384 ->   26 x  26 x 256 1.196 BF
 36 conv     27       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x  27 0.009 BF
 37 yolo
```

Table 7.1: Architecture of Yolov4 Tiny

By using this architecture of Yolov4 tiny, we were able to develop a successful model that detects the text in the signboard by drawing a boundary box and further classifying it to different types of classes (Language).

Fig 7.1: Detecting a Signboard with Multiple Languages.

The algorithm is able to detect multiple languages present in the signboard and also detect separate words among a long sentence and classify with its distinct language.

Fig 7.2: Detecting a Signboard and the Language at a Metro Station

Fig 7.3: Detection of Signboard consisting numerous Strings of the same Language in a Signboard.

An another algorithm that perform this task is EasyOCR. This is a pretrained OCR tool used to detect the boundary box of the language and as well as detect the text present in native Indian Languages as well(Hindi).

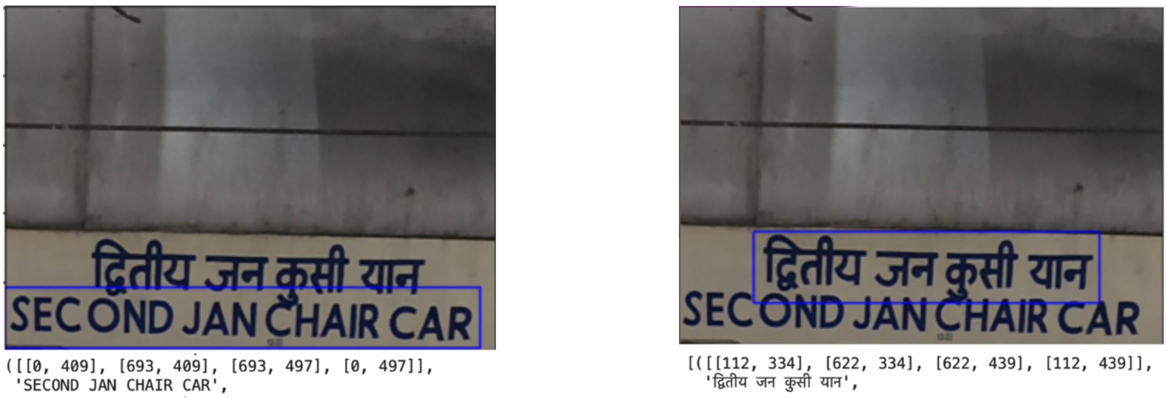

Fig7.4: Detection of Text using Boundary Box and outputting the Text Present in the Image. This Optical Character Recognition tool recognises the text and provides accurate Hindi text present in the image as well.

We also made use of the EAST (Efficient and Accurate Scene Text Detector) algorithm to perform text detection. The EAST algorithm makes boundary boxes surrounding the text regions. Once the boundary boxes are made, text recognition is done using the Optical Character Recognition (OCR) Tool called Pytesseract.
Here are some of the results we obtained by using the EAST algorithm and Pytesseract.

Test case 1:

Fig7.5: Input image - A signboard which has good text clarity

Fig 7.6: Output image – Boundary boxes are drawn around every piece of text region and the text within is also recognised with full precision

Test case 2:

Fig 7.7: Input image – A signboard picture which is photographed from a distance

Fig 7.8: Output image – Text detection is done successfully and most of the text is recognised

Test case 3:

Fig 7.9: Input image – A picture in which the signboard is surrounded by a lot of things in the background like road, trees, etc. In this image, the background occupies more area than the signboard.

Fig 7.10 Output image – Text detection is fully completed successfully and some parts of the text gets recognised successfully

Transliteration Output:

```
Input: bhatnagar
Predicted translation: भटनगर

Input: chennai
Predicted translation: चननई

Input: shivaani
Predicted translation: शिवाणी
```

Fig 7.11 Attaining the desired output after transliterating the given Input from English to Hindi

8. SUMMARY

The project involves three different phases to be accomplished. While the first being detecting the text by drawing the boundary box over the text location and classifying it with respect to its language. To successfully accomplish this we made use of Yolov4 tiny algorithm and EAST. The second phase involves the recognition of text/string the image. To retrieve this we made use of PyTesseract and EasyOCR tool to extract text involving multiple languages. The final phase involves transliterating of the text after obtain the string. To accomplish this final task we made use of encoder decoder model with attention to successfully transliterate the text in English to Hindi (The highly spoken language in India). In the end the model efficaciously transliterate with an accuracy of 90.55%.

9. REFERENCES:

[1] Long, S., He, X., & Ya, C. (2018). Scene text detection and recognition: The deep learning era. arXiv preprint arXiv:1811.04256.

[2] Bhunia, A. K., Kumar, G., Roy, P. P., Balasubramanian, R., & Pal, U. (2018). Text recognition in scene image and video frame using Color Channel selection. Multimedia Tools and Applications, 77(7), 8551-8578.

[3] Busta, M., Neumann, L., & Matas, J. (2017). Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2204-2212).

[4] Deselaers, T., Hasan, S., Bender, O., & Ney, H. (2009, March). A deep learning approach to machine transliteration. In Proceedings of the Fourth Workshop on Statistical Machine Translation (pp. 233-241). Association for Computational Linguistics.

[5]Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi

[6] EAST: An Efficient and Accurate Scene Text Detector. Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang Megvii Technology Inc., Beijing, China

[7] Neumann, L. and J. Matas, A method for text localization and recognition in real-world images. In Proceedings of the 10th Asian Conference on Computer Vision - Volume Part III, ACCV'10. Springer-Verlag, Berlin, Heidelberg, 2011. ISBN 978-3-642-19317-0.

[8] Rosca, M., & Breuel, T. (2016). Sequence-to-sequence neural network models for transliteration. *arXiv preprint arXiv:1610.09565*.

[9] Bartz, C., Yang, H., & Meinel, C. (2018, April). SEE: towards semi-supervised end-to-end scene text recognition. In Thirty-Second AAAI Conference on Artificial Intelligence.

[10] Tian, Z., Huang, W., He, T., He, P., & Qiao, Y. (2016, October). Detecting text in natural image with connectionist text proposal network. In European conference on computer vision (pp. 56-72). Springer, Cham.

[11] Canedo-Rodríguez, A., Kim, S., Kim, J.H. and Blanco-Fernández, Y., 2009, March. English to Spanish translation of signboard images from mobile phone camera. In IEEE Southeastcon 2009 (pp. 356-361). IEEE.

[12] Kore, A.B. and Mehta, D.R., 2014. TEXT AND AUDIO TRANSLATION OF TEXT FROM SIGNBOARD IMAGES-REVIEW. International Journal of Advances in Engineering & Technology, 7(2), p.500.

[13] Panhwar, M.A., Memon, K.A., Abro, A., Zhongliang, D., Khuhro, S.A. and Memon, S., 2019, July. Signboard detection and text recognition using artificial neural networks. In 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC) (pp. 16-19). IEEE.

[14] Arafat, S.Y., Ashraf, N., Iqbal, M.J., Ahmad, I., Khan, S. and Rodrigues, J.J., 2021. Urdu signboard detection and recognition using deep learning. Multimedia Tools and Applications, pp.1-23.

[15] Lin, S., Li, W. and Wei, Q., 2020, December. Research on Detection Algorithm of Utility Pole Signboard Based on Faster R-CNN. In 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) (Vol. 9, pp. 1841-1844). IEEE.

10. POSTER

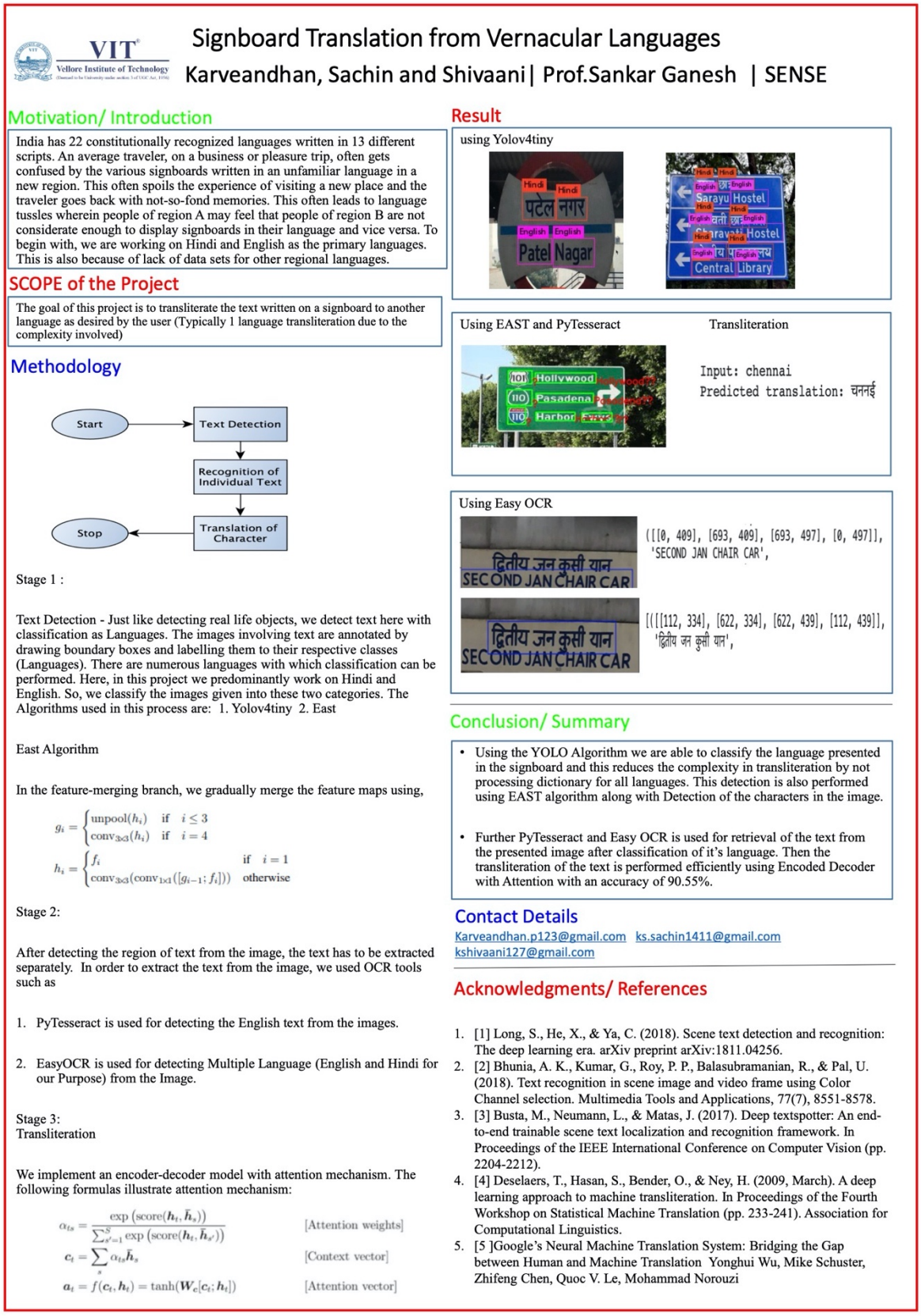

Fig 10.1: Poster of our project