Translation / Transliteration of Vernacular Languages from Signboards

Project ID:21G378383

Review - II

Group Members

RA1711003030378 Vishnu Teja Chikkala

RA1711003030383 Rajpreet Srivastav

Supervised By:

Mr. Sunil Kumar

Assistant Professor

Department of Computer Science & Engineering

Faculty of Engineering & Technology

SRM Institute of Science & Technology

Table of Contents I

Table of Contents II

Objective I

- India has 22 constitutionally recognized languages written in 13 different scripts. An average traveler, when travelling to a new region, might often get confused with signboards written in an unfamiliar language. It is also impossible to have every signboard in every city / town / village written in 22 different languages, as there will not be enough room to accommodate more than 2 – 3 scripts.
- The objective of this project is to develop a simple and easy-to-use Android mobile app which provides a two-click, picture-to-text, translation / transliteration service for Indian vernacular languages, using deep neural networks and natural language processing models trained for text detection, recognition and translation tasks on collected and freely-available datasets.

Objective II

- For the scope of this project, we will design a system which works for names (such as road names, city names, shop names, organization names, etc.) which typically are not longer than 4-5 words, and support translation for 1 or 2 languages. Further scope for the project involves including support for more languages and building models to support translation / transliteration of longer pieces of text.

Summary of Literature Review I

- Indian community faces a "Digital Divide" due to dominance of English as mode of communication in higher education, judiciary, corporate sector and Public administration at Central level whereas the government in states work in their respective regional languages [7]
- India has 22 scheduled languages. While 99 % of the population speak one of these scheduled languages in various dialects (which number in the thousands) [1], according to Census 2011, the total percentage of English speakers is at 10 %, and that too is skewed towards the urban population. [10] Hence, there lies a need for developing NLP architectures for facilitating flow of digital content and information in and between local, national and international levels.

Summary of Literature Review II

- The above also means that a large percentage of the literate population is either monolingual or bilingual, and across 22 languages, an accessible, easy-to-use and intuitively developed system is required, which enables intercommunication.
- While traditionally NLP has been approached with statistical methods such as Hidden Markov Machines (HMM), Support vector machine(SVM), Conditional Random Field(CRF), Naive Bayes(NB), etc, which take a large amount of tagged/annotated data (corpus) to statistically analyze and learn the language characteristics [3], the research into deep learning or 'connectionist approach' [3] with the use of trained artificial neural networks (ANNs), has gained impetus due to (i) the simplicity of the solution in rapidly prototyping and establishing practically effective systems (ii) the lower cost of annotation of the training data [8], and the fact that they attempt to

Summary of Literature Review III

more closely emulate the learning process of biological brains, among other reasons. [3], [9], [4]

- Particularly, the collection of a uniform corpus and standard datasets for training models remains a challenge across all regional languages. The large number of morphological variations across Indic languages also contributes to this issue.[6], [12]
- Sharma et al., 2017 concluded that almost all existing Indian language machine transliteration systems are based on statistical and hybrid approach [11]
- Kulkarni et al., achieved upto 98% training and 94% testing accuracy on the IndicNLP library for Marathi, using an CNN-LSTM based model architecture. [5]

Summary of Literature Review IV

- Most of the Indian population accesses digital content through smartphones. In 2019, the number of smartphone users in the country passed 500 million [2], and is estimated to increase to 850 million by 2022 [13]. This, hence, also makes smartphones and smartphone apps in particular an ideal platform on which to launch NLP applications for the wider population, and directly help facilitate flow of information past language barriers.

Architectural Design I

- The basic functioning of the app is as follows:
  - User captures a photo of the signboard
  - The image is resized so as to be suitable as input to the model
  - The model takes the image as input. The text within the message is detected, extracted and transcribed to target language.
  - The text output (or error message, in case of failure to generate output within threshold confidence), is displayed on screen.

Architectural Design II

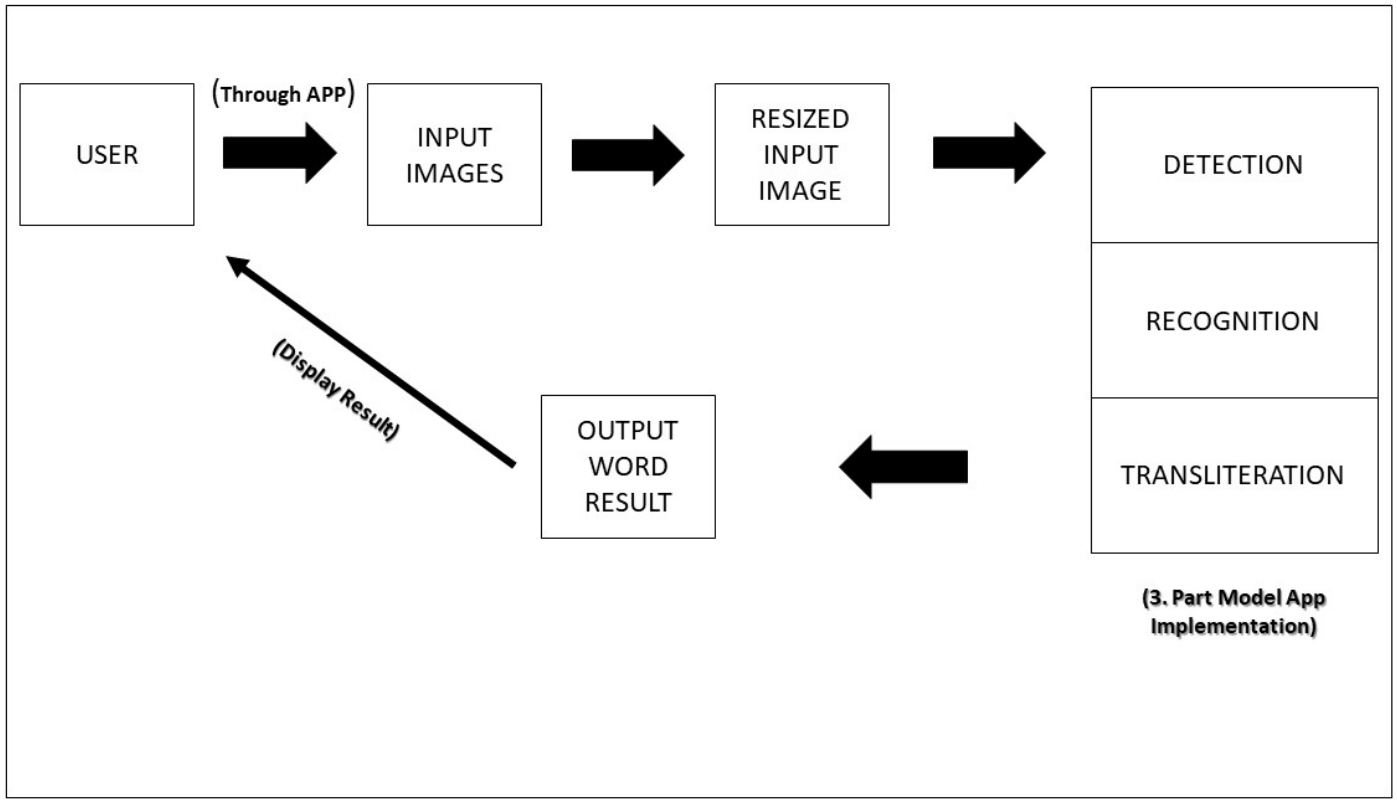


Figure: Functioning of App

Architectural Design III

- The artificial neural network (ANN) behind the core functioning of the app is made up of 3 models performing consecutive tasks. That is, the output of a preceding model will be fed as input to the succeeding model, and thus they act as one model unit.

Architectural Design IV

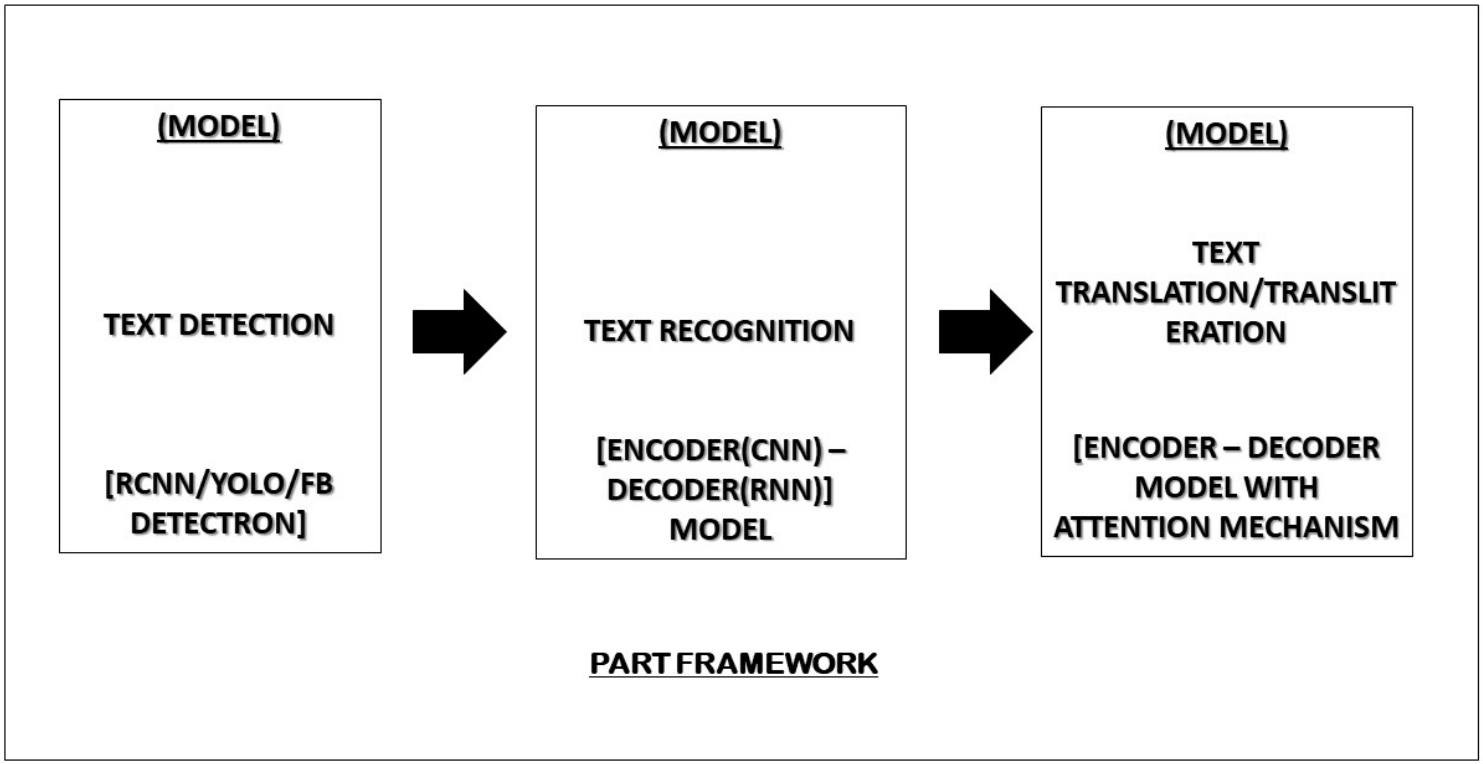
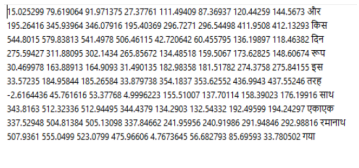


Figure: 3-part model

Dataset Specification I

- The training dataset for the text detection and text recognition tasks consists of a large number of images containing scene text, synthetically generated. Each of the images has a corresponding annotation, listing the bounding boxes and text script present in the images.

(a) Training Set Image


15.025299 79.619064 91.971375 27.37761 111.49409 87.36937 120.44259 144.5673 और
195.26416 345.93964 346.07916 195.40369 296.7271 296.54498 411.9508 412.13293 किस
544.8015 579.83813 541.4978 506.46115 42.720642 60.455795 136.19897 118.46382 दिन
275.59427 311.88095 302.1434 265.85672 134.48518 159.5067 173.62825 148.60674 रूप
30.469978 163.88913 164.9093 31.490135 182.98358 181.51782 274.3758 275.84155 इस
33.57235 184.95844 185.26584 33.879738 354.1837 353.62552 436.9943 437.55246 तरह
-2.6164436 45.761616 53.37768 4.9996223 155.51007 137.70114 158.39023 176.19916 साथ
343.8163 512.32336 512.94495 344.4379 134.2903 132.54332 192.49599 194.24297 एकाएक
337.52948 504.81384 505.13098 337.84662 241.95956 240.91986 291.94846 292.98816 रमानाथ
507.9361 555.0499 523.0799 475.96606 4.7673645 56.682793 85.69593 33.780502 गया

(b) Set Annotation

Dataset Specification II

- The test dataset for text detection and text recognition tasks consists of actual images containing natural scene text.

Figure: Test Set Image

Dataset Specification III

- The train and test datasets are both in form of xml files containing serialized pairs of source language script and corresponding target language script.

```
<?xml version="1.0" encoding="UTF-8"?><TransliterationCorpus CorpusID = "NEWS2012-Training-EnHi-13937" SourceLang = "
English" TargetLang = "Hindi" CorpusType = "Training" CorpusSize = "13937" CorpusFormat = "UTF8">
<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">
```

Figure: Transliteration Set

Methodology / Algorithms / Techniques to be used I

- The text detection task will be carried out by a Faster Region-based Convolutional Network (Faster R-CNN) with Feature Pyramid Network (FPN; for bounding box tightening) from the FAIR Detectron2 kit, which has been trained for object detection on the COCO dataset. We will fine-tune this model to the task at hand by training and validating further on the above scene text database.

Methodology / Algorithms / Techniques to be used II

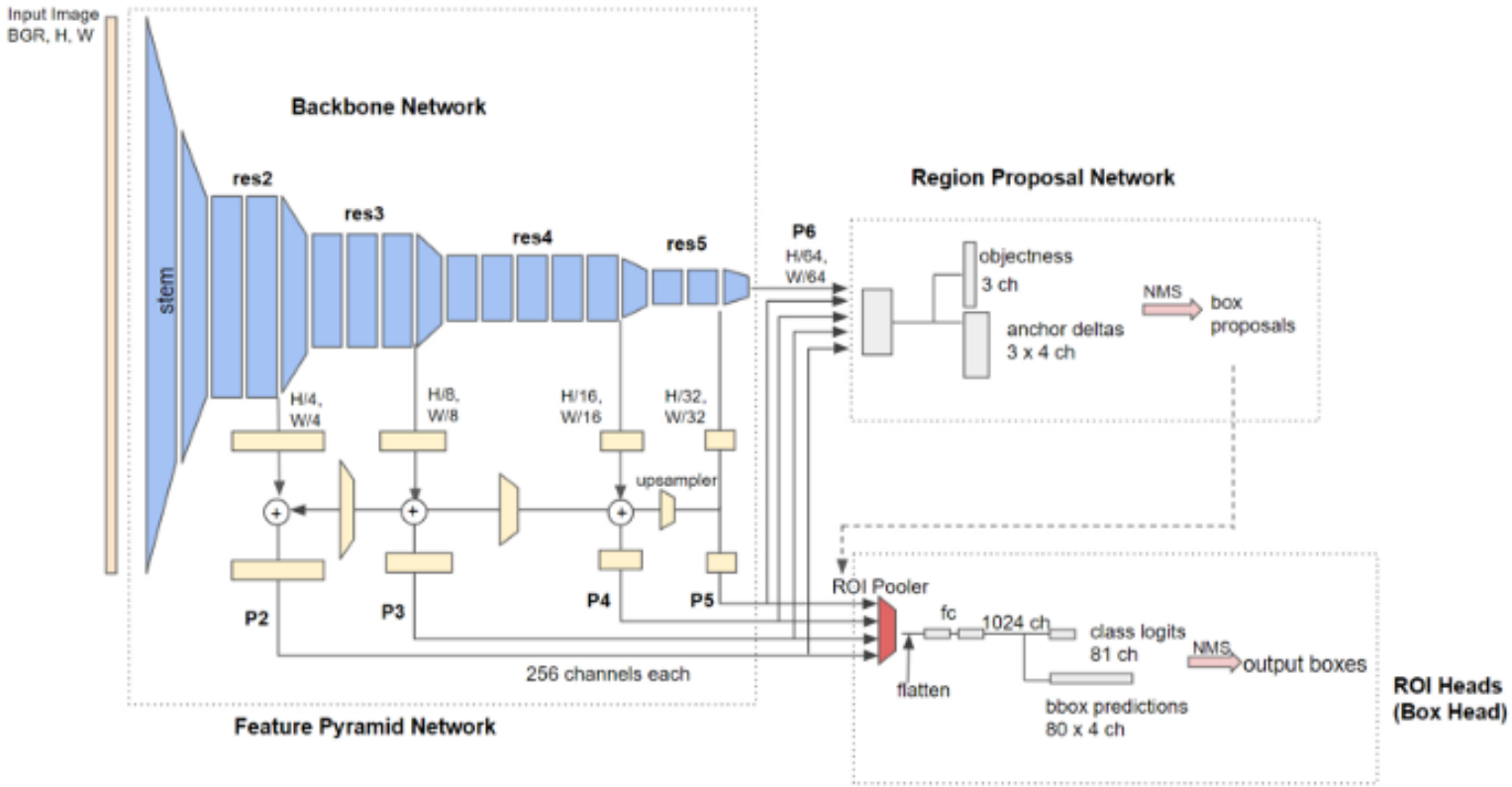


Figure: Faster R-CNN Object Detection Model

Methodology / Algorithms / Techniques to be used III

- The text recognition task will be carried out by an encoder-decoder model setup which takes the cropped bounding box of text as input. The encoder is a Convolutional Neural Network (CNN) while the decoder is a Long Short-Term Memory model (LSTM). Connectionist Temporal Classification (CTC) loss will be used to eliminate duplicate recognition of the same letter by adjacent CNN features.

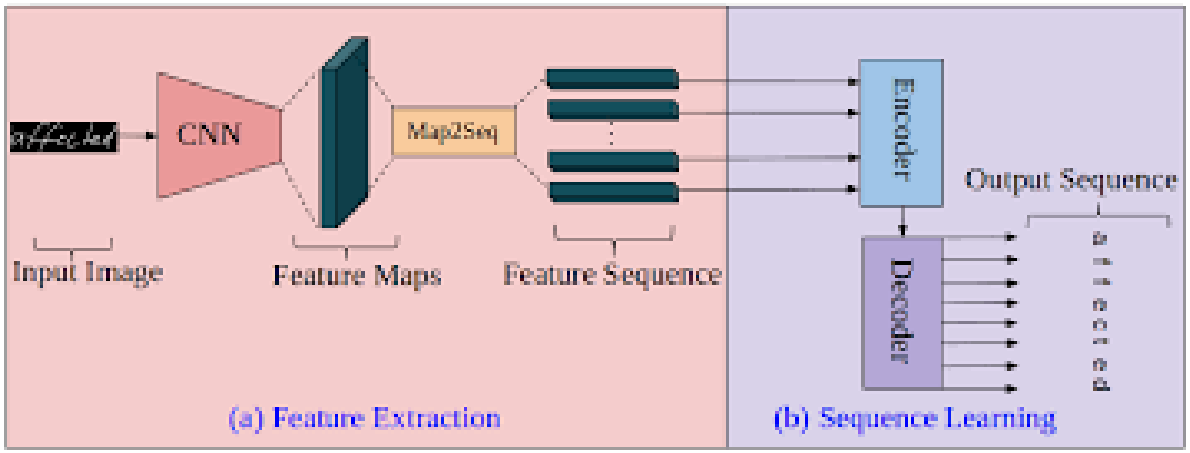


Figure: CNN-LSTM Encoder-Decoder Architecture

Methodology / Algorithms / Techniques to be used IV

- The transliteration task will carried by a LSTM - LSTM encoder-decoder model with attention mechanism, which takes source language script as input and generates target language script as output.

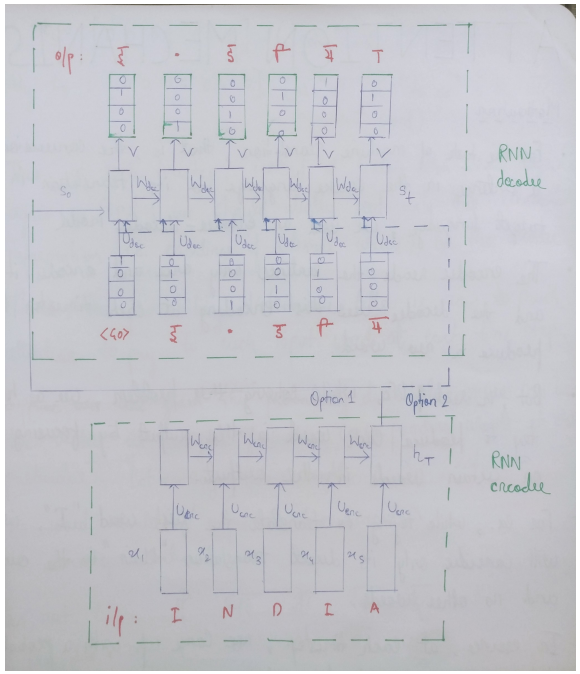


Figure: LSTM-LSTM Encoder-Decoder Architecture

Methodology / Algorithms / Techniques to be used V

- The app will make use of Google Firebase API to provide UI functionality and user services.
- The model will be mounted on and integrated with the app with the help of Tenserflow Lite.

Partial Implementation I

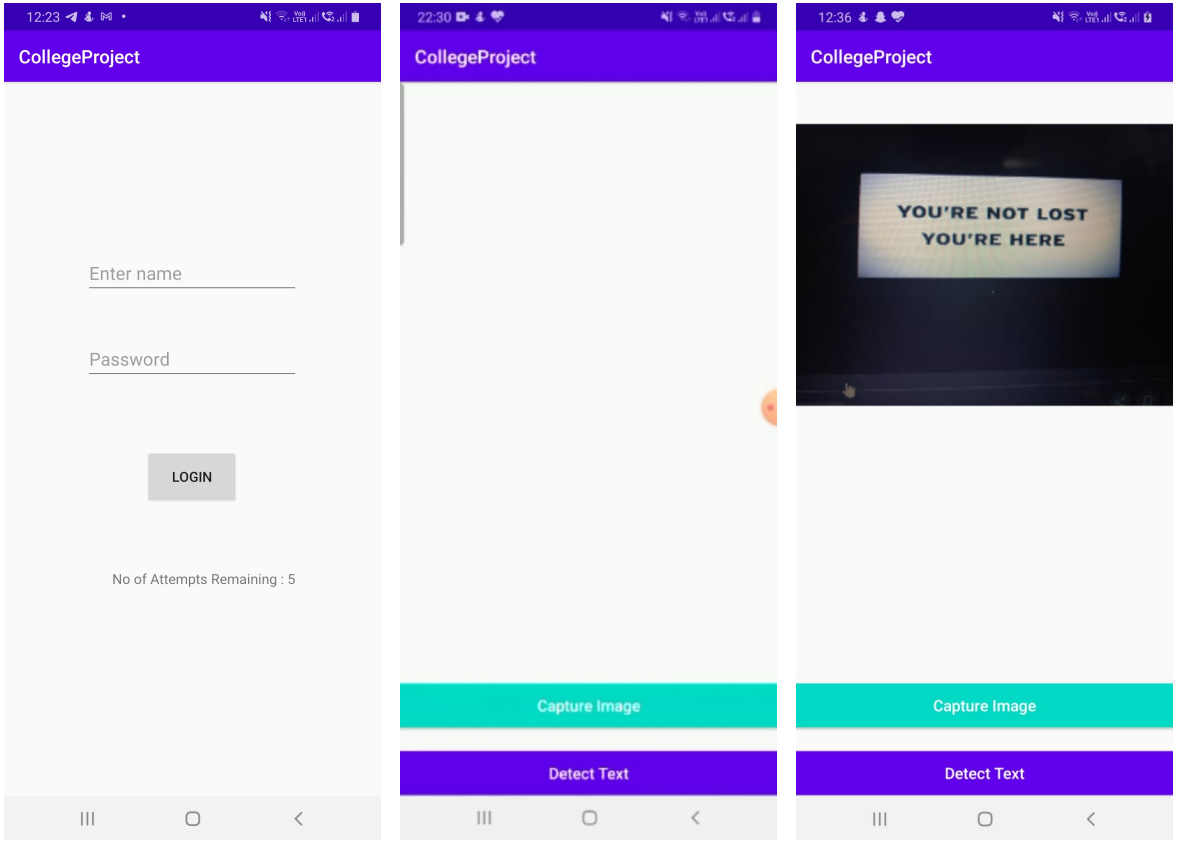


Figure: Skeleton App

Partial Implementation II

```
MAX_OUTPUT_CHARS = 30
class Transliteration_EncoderDecoder(nn.Module):

    # 'input_size' is the size of the English vocabulary
    # 'output_size' is the size of the Hindi vocabulary
    # 'hidden_size' is a hyper-parameter. More the number of hidden layers in network, greater will be the training accuracy, but also more data will be re
    # 'verbose' enables testing of the model through the execution of it's forward pass
    def __init__(self, input_size, hidden_size, output_size, verbose=False):
        super(Transliteration_EncoderDecoder, self).__init__()

        # Initializes parameters as internal variables for reuse
        self.hidden_size = hidden_size
        self.output_size = output_size

        # Defines the GRU cell for encoder and decoder model
        self.encoder_rnn_cell = nn.GRU(input_size, hidden_size)     # Encoder model takes letter by letter input and outputs to hidden layer
        self.decoder_rnn_cell = nn.GRU(output_size, hidden_size)    # The input to Decoder model is the output of the previous cell

        self.h2o = nn.Linear(hidden_size, output_size)              # In the decoder, converts hidden state to the output ('Hindi' representation)
        self.softmax = nn.LogSoftmax(dim=2)                         # Softmax layer as this is a classification problem

        self.verbose = verbose

    # Forward pass for the encoder-decoder model
    # The input parameters are the actual input word, the max no. of character in the output word, the device used by the model (CPU or GPU)
    # We can also choose to pass the ground truth (true output) to enable 'teacher forcing'
    def forward(self, input, max_output_chars = MAX_OUTPUT_CHARS, device = 'cpu', ground_truth = None):

        # encoder
        # While the internal classification happens character by character in the model, by passing input as a vector of characters, we get a vectorized ou
        out, hidden = self.encoder_rnn_cell(input)
```

Figure: Text Transliteration Model

Partial Implementation III

```
            # Hidden state and Output state from previous cell is passed to GRU Cell as input
            # The output is the next hidden state and the next Output state
            out, decoder_state = self.decoder_rnn_cell(decoder_input, decoder_state)

            # Printing the shape of the intermediate output from GRU Cell
            if self.verbose:
                print('Decoder intermediate output', out.shape)

            out = self.h2o(decoder_state)      # The decoder hidden state is passed through linear layer for classification of character
            out = self.softmax(out)            # Softmax layer is applied to produce a probability distribution
            outputs.append(out.view(1, -1))    # Output is flattened into a 1D layer

            # Printing the shape of this output form
            if self.verbose:
                print('Decoder output', out.shape)
                self.verbose = False

        # Now, instead of passing the softmax output as is as input to the next GRU Cell iteration, we convert it into a one-hot encoded vector (with the
        # This is because of the method of 'Teacher Forcing' which is described below
        # Thus, regardless of whether the output of current timestep is from the cell or the ground truth, the input to next timestep will be in one-hot
        max_idx = torch.argmax(out, 2, keepdim=True)
        if not ground_truth is None:
            max_idx = ground_truth[i].reshape(1, 1, 1)
        one_hot = torch.FloatTensor(out.shape).to(device)
        one_hot.zero_()
        one_hot.scatter_(2, max_idx, 1)

        # We do not want gradients to flow through this output-to-input path (only through hidden states), and thus we disable training for this path
        decoder_input = one_hot.detach()

    return outputs
```

Figure: Text Transliteration Training

Expected Outcomes I

- Achieve greater than 80 % accuracy on both train and validation / test datasets, across all models, after training, with optimal bias-variance tradeoff and set of tuned hyperparameters.
- Propose general scalable and upgradable framework to similarly train model on other Indian languages / scripts as well as on longer pieces of text.
- Optimize the performance of the app w.r.t. model, to deliver seamless user experience.
- Design app to be intuitive, easy-to-use and non-dependent on the internet.

References I

- More than 19,500 mother tongues spoken in india: Census, Jul 2018.
- Smartphone users in india crossed 500 million in 2019, states report, Jan 2020.
- N. P. Desai and V. K. Dabhi.
  Taxonomic survey of hindi language nlp systems.
  arXiv preprint arXiv:2102.00214, 2021.
- T. Deselaers, S. Hasan, O. Bender, and H. Ney.
  A deep learning approach to machine transliteration.
  In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 233–241, 2009.

References II

A. Kulkarni, M. Mandhane, M. Likhitkar, G. Kshirsagar, J. Jagdale, and R. Joshi.
Experimental evaluation of deep learning models for marathi text classification.
arXiv preprint arXiv:2101.04899, 2021.

A. Kunchukuttan, D. Kakwani, S. Golla, A. Bhattacharyya, M. M. Khapra, P. Kumar, et al.
Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages.
arXiv preprint arXiv:2005.00085, 2020.

C. Kurian and K. Kannan Balakrishnan.
Natural language processing in india prospects and challanges.
In *Proceedings of the International Conference on "Recent Trends in Computational Science*, 2008.

References III


J. Philip, V. P. Namboodiri, and C. Jawahar.
A baseline neural machine translation system for indian languages.
arXiv preprint arXiv:1907.12437, 2019.


M. Rosca and T. Breuel.
Sequence-to-sequence neural network models for transliteration.
arXiv preprint arXiv:1610.09565, 2016.


R. S.
In india, who speaks in english, and where?, May 2019.


A. Sharma and D. Rattan.
Machine transliteration for indian languages: A review.
International Journal of Advanced Research in Computer Science, 8(8), 2017.

References IV

N. Singh.
Nlp for indian languages.
2020.

www.ETTelecom.com.
India to have 820 million smartphone users by 2022 - et telecom, Jul 2020.

Thank You