

# Translation / Transliteration of Vernacular Languages from Signboards

Project ID:21G378383

Review - III

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

- 1 Objective
- 2 Summary of Literature Review
- 3 Architectural Design for Proposed System
- 4 Dataset Specifications
- 5 Methodology / Algorithms / Techniques to be used

# Table of Contents II

6 Partial Implementation

7 Experimental Results

8 References

# 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.

- 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

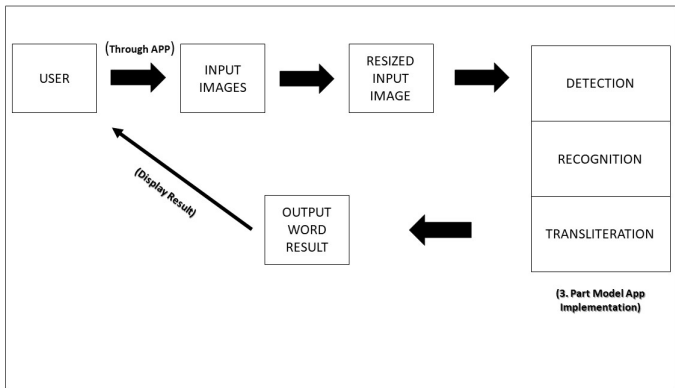


Figure: Functioning of App

- 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

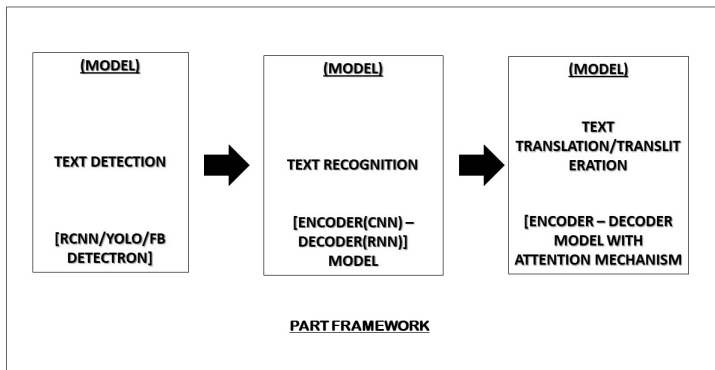


Figure: 3-part model

# 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

```
[[{"x1": 320, "y1": 10, "x2": 480, "y2": 30, "text": "एकारक"}, {"x1": 320, "y1": 40, "x2": 480, "y2": 60, "text": "रमानाथ"}, {"x1": 320, "y1": 70, "x2": 480, "y2": 90, "text": "हृदय"}]]
```

(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.

```
1 <?xml version="1.0" encoding="UTF-8"?><TransliterationCorpus CorpusID = "NEWS2012-Training-EnHi-13937" SourceLang = "English" TargetLang = "Hindi" CorpusType = "Training" CorpusSize = "13937" CorpusFormat = "UTF8">
2 <Name ID="1">
3 <SourceName>RAASAVIHAAREE</SourceName>
4 <TargetName ID="1">रासविहारी</TargetName>
5 </Name>
6 <Name ID="2">
7 <SourceName>DEOGAN ROAD</SourceName>
8 <TargetName ID="1">देवगन रोड</TargetName>
9 </Name>
10 <Name ID="3">
11 <SourceName>SHATRUMARDAN</SourceName>
12 <TargetName ID="1">शत्रुमर्दन</TargetName>
13 </Name>
14 <Name ID="4">
15 <SourceName>MAHIJUBA</SourceName>
16 <TargetName ID="1">महिजुबा</TargetName>
17 </Name>
18 <Name ID="5">
19 <SourceName>SABINE</SourceName>
20 <TargetName ID="1">सेबिन</TargetName>
21 </Name>
22 <Name ID="6">
23 <SourceName>BILL COSBY</SourceName>
24 <TargetName ID="1">बिल कंसबी</TargetName>
25 </Name>
26 <Name ID="7">
27 <SourceName>RISHTA KAGAZ KA</SourceName>
28 <TargetName ID="1">रिश्ता कागज़ का</TargetName>
29 </Name>
30 <Name ID="8">
```

Figure: Transliteration Set



- 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

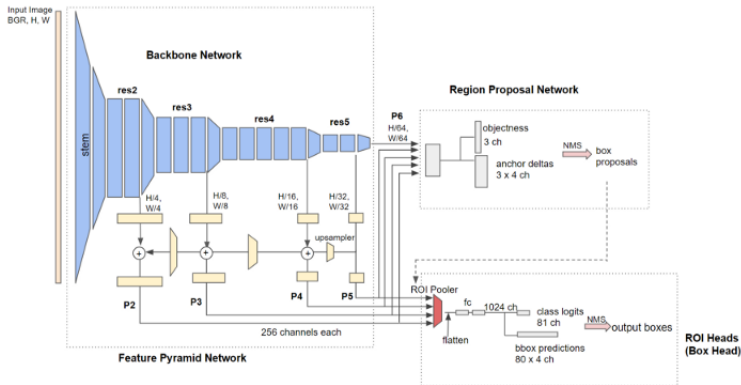


Figure: Faster R-CNN Object Detection Model

- 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.

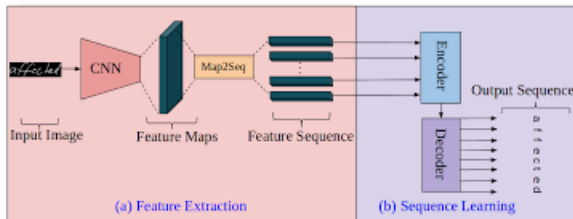


Figure: CNN-LSTM Encoder-Decoder Architecture

# Methodology / Algorithms / Techniques to be used IV

- The transliteration task will be 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.

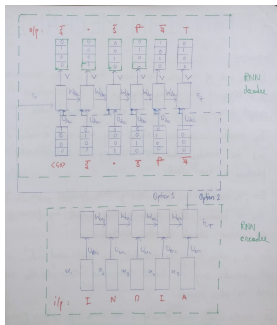


Figure: LSTM-LSTM Encoder-Decoder Architecture

- 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 Tensorflow Lite.

# Partial Implementation I

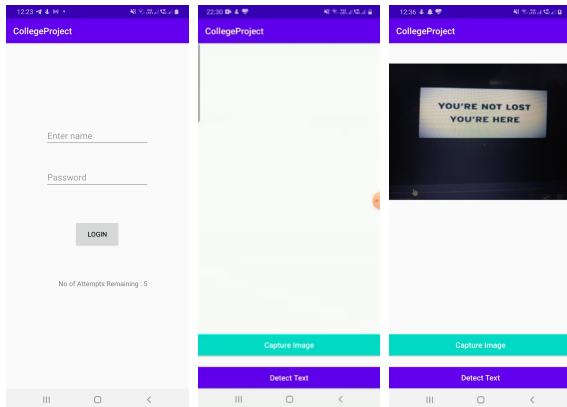


Figure: Skeleton App

# Partial Implementation II

```
def __init__(self, num_chars, rnn_hidden_size=256, dropout=0.1):  
  
    super(CRNN, self).__init__()  
    self.num_chars = num_chars  
    self.rnn_hidden_size = rnn_hidden_size  
    self.dropout = dropout  
  
    # CNN Part 1  
    resnet_modules = list(resnet.children())[:3]  
    self.cnn_p1 = nn.Sequential(*resnet_modules)  
  
    # CNN Part 2  
    self.cnn_p2 = nn.Sequential(  
        nn.Conv2d(256, 256, kernel_size=(3,6), stride=1, padding=1),  
        nn.BatchNorm2d(256),  
        nn.ReLU(inplace=True)  
    )  
    self.linear1 = nn.Linear(1024, 256)  
  
    # RNN  
    self.rnn1 = nn.GRU(input_size=rnn_hidden_size,  
                       hidden_size=rnn_hidden_size,  
                       bidirectional=True,  
                       batch_first=True)  
    self.rnn2 = nn.GRU(input_size=rnn_hidden_size,  
                       hidden_size=rnn_hidden_size,  
                       bidirectional=True,  
                       batch_first=True)  
    self.linear2 = nn.Linear(self.rnn_hidden_size*2, num_chars)
```

Figure: Text Recognition Model

# Partial Implementation III

```
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 IV

```
# 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[1].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

```

cfg.MODEL ROI_READS.NUM_CLASSES = 3 # No. of classes = [HINDI, ENGLISH, OTHER]
cfg.TEST.EVAL_PERIOD = 500
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()

```

[08/01 12:28:15 d2.utils.events]:	eta: 0:25:53	iter: 39	total_loss: 1.522	loss_cls: 0.602	loss_box_reg: 0.758	loss_rpn_cls: 0.055	loss_rpn_loc: 0.056	time: 1.0614	data_time
[08/01 12:28:16 d2.utils.events]:	eta: 0:25:06	iter: 59	total_loss: 1.295	loss_cls: 0.395	loss_box_reg: 0.712	loss_rpn_cls: 0.044	loss_rpn_loc: 0.062	time: 1.0534	data_time
[08/01 12:28:17 d2.utils.events]:	eta: 0:24:48	iter: 79	total_loss: 1.110	loss_cls: 0.366	loss_box_reg: 0.632	loss_rpn_cls: 0.026	loss_rpn_loc: 0.051	time: 1.0476	data_time
[08/01 12:29:18 d2.utils.events]:	eta: 0:24:31	iter: 99	total_loss: 1.007	loss_cls: 0.351	loss_box_reg: 0.478	loss_rpn_cls: 0.041	loss_rpn_loc: 0.072	time: 1.0517	data_time
[08/01 12:29:39 d2.utils.events]:	eta: 0:24:14	iter: 119	total_loss: 0.804	loss_cls: 0.251	loss_box_reg: 0.400	loss_rpn_cls: 0.029	loss_rpn_loc: 0.059	time: 1.0510	data_time
[08/01 12:30:00 d2.utils.events]:	eta: 0:23:48	iter: 139	total_loss: 0.738	loss_cls: 0.230	loss_box_reg: 0.397	loss_rpn_cls: 0.030	loss_rpn_loc: 0.053	time: 1.0483	data_time
[08/01 12:30:21 d2.utils.events]:	eta: 0:23:28	iter: 159	total_loss: 0.694	loss_cls: 0.229	loss_box_reg: 0.383	loss_rpn_cls: 0.027	loss_rpn_loc: 0.050	time: 1.0494	data_time
[08/01 12:30:41 d2.utils.events]:	eta: 0:23:03	iter: 179	total_loss: 0.688	loss_cls: 0.224	loss_box_reg: 0.351	loss_rpn_cls: 0.024	loss_rpn_loc: 0.059	time: 1.0466	data_time
[08/01 12:31:03 d2.utils.events]:	eta: 0:22:45	iter: 199	total_loss: 0.758	loss_cls: 0.247	loss_box_reg: 0.388	loss_rpn_cls: 0.022	loss_rpn_loc: 0.046	time: 1.0475	data_time
[08/01 12:31:23 d2.utils.events]:	eta: 0:22:21	iter: 219	total_loss: 0.631	loss_cls: 0.221	loss_box_reg: 0.350	loss_rpn_cls: 0.015	loss_rpn_loc: 0.043	time: 1.0467	data_time
[08/01 12:31:44 d2.utils.events]:	eta: 0:22:00	iter: 239	total_loss: 0.654	loss_cls: 0.208	loss_box_reg: 0.346	loss_rpn_cls: 0.017	loss_rpn_loc: 0.044	time: 1.0454	data_time
[08/01 12:32:05 d2.utils.events]:	eta: 0:21:38	iter: 259	total_loss: 0.581	loss_cls: 0.184	loss_box_reg: 0.340	loss_rpn_cls: 0.017	loss_rpn_loc: 0.044	time: 1.0444	data_time
[08/01 12:32:25 d2.utils.events]:	eta: 0:21:20	iter: 279	total_loss: 0.613	loss_cls: 0.184	loss_box_reg: 0.355	loss_rpn_cls: 0.016	loss_rpn_loc: 0.041	time: 1.0439	data_time
[08/01 12:32:46 d2.utils.events]:	eta: 0:20:56	iter: 299	total_loss: 0.680	loss_cls: 0.217	loss_box_reg: 0.359	loss_rpn_cls: 0.025	loss_rpn_loc: 0.057	time: 1.0440	data_time
[08/01 12:33:08 d2.utils.events]:	eta: 0:20:39	iter: 319	total_loss: 0.599	loss_cls: 0.197	loss_box_reg: 0.363	loss_rpn_cls: 0.015	loss_rpn_loc: 0.043	time: 1.0470	data_time
[08/01 12:33:29 d2.utils.events]:	eta: 0:20:18	iter: 339	total_loss: 0.553	loss_cls: 0.172	loss_box_reg: 0.324	loss_rpn_cls: 0.010	loss_rpn_loc: 0.035	time: 1.0468	data_time
[08/01 12:33:50 d2.utils.events]:	eta: 0:19:57	iter: 359	total_loss: 0.543	loss_cls: 0.150	loss_box_reg: 0.309	loss_rpn_cls: 0.010	loss_rpn_loc: 0.050	time: 1.0467	data_time
[08/01 12:34:12 d2.utils.events]:	eta: 0:19:38	iter: 379	total_loss: 0.569	loss_cls: 0.163	loss_box_reg: 0.295	loss_rpn_cls: 0.024	loss_rpn_loc: 0.037	time: 1.0480	data_time
[08/01 12:34:31 d2.utils.events]:	eta: 0:19:14	iter: 399	total_loss: 0.620	loss_cls: 0.212	loss_box_reg: 0.326	loss_rpn_cls: 0.016	loss_rpn_loc: 0.036	time: 1.0448	data_time
[08/01 12:34:53 d2.utils.events]:	eta: 0:18:54	iter: 419	total_loss: 0.574	loss_cls: 0.167	loss_box_reg: 0.323	loss_rpn_cls: 0.017	loss_rpn_loc: 0.043	time: 1.0460	data_time
[08/01 12:35:14 d2.utils.events]:	eta: 0:18:34	iter: 439	total_loss: 0.561	loss_cls: 0.160	loss_box_reg: 0.301	loss_rpn_cls: 0.013	loss_rpn_loc: 0.040	time: 1.0468	data_time
[08/01 12:35:34 d2.utils.events]:	eta: 0:18:10	iter: 459	total_loss: 0.543	loss_cls: 0.171	loss_box_reg: 0.332	loss_rpn_cls: 0.010	loss_rpn_loc: 0.034	time: 1.0440	data_time
[08/01 12:35:54 d2.utils.events]:	eta: 0:17:48	iter: 479	total_loss: 0.532	loss_cls: 0.172	loss_box_reg: 0.329	loss_rpn_cls: 0.009	loss_rpn_loc: 0.038	time: 1.0438	data_time

Figure: Text Detection Training



# Expected Outcomes II

```
D [07/28 12:59:29 d2.data.comma]: Serializing 27 elements to byte tensors and concatenating them all ...
[07/28 12:59:29 d2.data.comma]: Serialized dataset takes 0.018 s
[07/28 12:59:29 d2.evaluation.evaluator]: Start inference on 27 images
[07/28 12:59:33 d2.evaluation.evaluator]: Inference done 11/27: 0.1363 s / img, ETA=0:00:05
[07/28 12:59:39 d2.evaluation.evaluator]: Inference done 27/27: 0.1266 s / img, ETA=0:00:00
[07/28 12:59:39 d2.evaluation.evaluator]: Total inference time: 0:00:07.27531 (0.33667 s / img per device, on 1 devices)
[07/28 12:59:39 d2.evaluation.evaluator]: Total inference pure compute time: 0:00:07 (0.324569 s / img per device, on 1 devices)
[07/28 12:59:39 d2.evaluation.coco_evaluation]: Preparing results for COCO format ...
[07/28 12:59:39 d2.evaluation.coco_evaluation]: Saving results to /output/coco_instances_results.json
[07/28 12:59:39 d2.evaluation.coco_evaluation]: Evaluating predictions ...
Loading and preparing results...
DONE (t=0.08s)
creating index...
index created!
Running per image evaluation...
Evaluating annotation type 'bbox'
DONE (t=0.07s).
Accumulating evaluation results...
DONE (t=0.09s).
Average Precision (AP) @ [100=0.50=0.95] area= all | maxDets=100 | = 0.486
Average Precision (AP) @ [100=0.50] area= all | maxDets=100 | = 0.794
Average Precision (AP) @ [100=0.75] area= all | maxDets=100 | = 0.588
Average Precision (AP) @ [100=0.50=0.95] area= small | maxDets=100 | = 0.416
Average Precision (AP) @ [100=0.50=0.95] area= medium | maxDets=100 | = 0.541
Average Precision (AP) @ [100=0.50=0.95] area= large | maxDets=100 | = 0.454
Average Recall (AR) @ [100=0.50=0.95] area= all | maxDets= 1 | = 0.195
Average Recall (AR) @ [100=0.50=0.95] area= all | maxDets= 10 | = 0.516
Average Recall (AR) @ [100=0.50=0.95] area= all | maxDets=100 | = 0.518
Average Recall (AR) @ [100=0.50=0.95] area= small | maxDets=100 | = 0.408
Average Recall (AR) @ [100=0.50=0.95] area= medium | maxDets=100 | = 0.610
Average Recall (AR) @ [100=0.50=0.95] area= large | maxDets=100 | = 0.519
[07/28 12:59:39 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
| AP | AP50 | AP75 | APs | APm | APl |
|-----|-----|-----|-----|-----|-----|
| 48.434 | 79.410 | 58.818 | 41.596 | 54.075 | 45.268 |
[07/28 12:59:39 d2.evaluation.coco_evaluation]: Per-category box AP:
| category | AP | category | AP | | |
|---|---|---|---|---|---|
| NOCNC | 57.467 | ENGLISH | 45.207 | OTHER | 43.108 |
OrderedDict([('NOCNC', 57.467), ('ENGLISH', 45.207), ('OTHER', 43.108)])
['AP', 48.434, 36895031696,
'AP-ENGLISH', 45.207, 717037128,
```

Figure: Text Detection Testing

# Expected Outcomes III

```
ax1.plot(epoch_losses)
ax1.set_xlabel("Epochs")
ax1.set_ylabel("Loss")

ax2.plot(iteration_losses)
ax2.set_xlabel("Iterations")
ax2.set_ylabel("Loss")

plt.show()
```

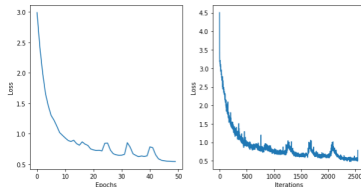


Figure: Text Recognition Training and Testing

# Expected Outcomes IV

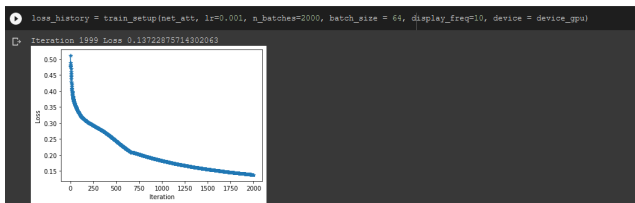


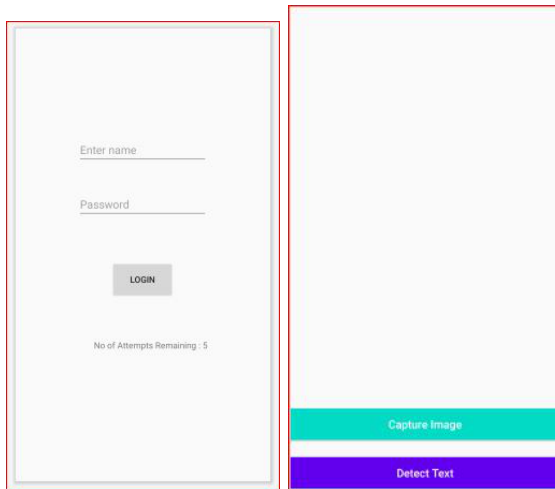
Figure: Text Transliteration Training

```
# Accuracy of the Encoder Decoder model, with and without attention
accuracy = calc_accuracy(net) * 100
accuracy_attn = calc_accuracy(net_att) * 100
print('Accuracy w/o attention ', accuracy)
print('Accuracy with attention', accuracy_attn)
```

Accuracy w/o attention 69.25025668775679  
Accuracy with attention 69.93080114330115

Figure: Text Transliteration Testing

# Expected Outcomes V



Enter name

Password

LOGIN





No of Attempts Remaining : 5

Capture Image

Detect Text

Figure: App Result

# 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. Likhitar, 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 challenges.

*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](http://www.ETTelecom.com).

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

Thank You