Recognizing Text in Ubiquitous Scenes

Deepak Mishra
Professor, Department of Avionics
IIST Thiruvananthapuram, India
deepak.mishra@iist.ac.in

Sri Aditya Deevi
Student, Department of Avionics
IIST Thiruvananthapuram, India
SC18B080

Asish kumar Mishra
Student, Department of Avionics
IIST Thiruvananthapuram, India
SC18B074

Kothadiya Princekumar Balkrushna
Student, Department of Avionics
IIST Thiruvananthapuram, India
SC18B078

Aditya Amrit
Student, Department of Avionics
IIST Thiruvananthapuram, India
SC18B087

***Abstract*—Text is one of the most effective forms of communication among human beings. Recognizing text *automatically* & *efficiently* in everyday scenes is an invaluable tool in many applications like Document Analysis, Autonomous Vehicles, Augmented Reality etc. In this project, we try to design, explore and experiment with various deep learning architectures for both Regular & Irregular Text Recognition. We also try to provide an analytical understanding of the reason for their good performance on various images from different perspectives. These analyses basically "unblackbox" the peculiar but interesting aspects of the models, and also lead us to some insightful possible extensions of this project.**

***Index Terms*—Scene Text Recognition, Computer Vision, Document Analysis, Deep Learning, Spatial Transformation Networks**

I. Introduction

Human beings have had the zeal to automate various tasks in their daily lives and build remarkable things since time immemorial. Recognizing text in images of scenes everywhere around us, to enrich its information content, is one such task. The recent advances in the field of Deep Learning and Computer Vision have opened up new avenues of research and significantly improved performance for *Scene Text Recognition*.

Scene Text recognition has basically two *variants* of information intertwined : Visual and Semantic. Especially in natural scenes, the aforementioned typical aspect is what makes this task challenging and many researchers have devised several intuitive solutions over the years.

The various techniques developed can be broadly classified to solve two sub-tasks of increasing difficulty level namely: *Regular* and *Irregular* Text Recognition. Nowadays, it is becoming popular to do text localisation to detect ROI and use an OCR engine(like Tesseract) to determine the text in a natural scene[1].

There are two types of recent methods for irregular text detection: Spatial Transformation Network(STN)[2] and Fully Convolutional Network(FCN)[3] by [4]. Convolutional Neural Networks lack of ability to be spatially invariant to the input data in a computationally and parameter efficient manner. The STN explicitly allows the spatial manipulation of data within the network, giving neural networks the ability to actively spatially transform feature maps, conditional on the feature map itself, resulting in models which learns invariance to translation, scale, rotation and more generic warping[5].

Zhou et al developed an end-to-end trainable arbitrary orientation network (AON)[6], solving the problem of FCN[4] and STN[2] to directly capture the deep features of irregular texts, combined with an attention-based decoder to generate character sequences using word-level annotations for training.

In one of the other approaches, a *Line-fitting transformation* is used to correct perspective and curvature distortions, where the line passing through the middle of the text and by iterative method is fit with a polynomial equation to and an effort is made to make that line horizontal along with the text[7].

In this work, we try to explore and experiment with various deep learning models for both Regular & Irregular Text Recognition and try to provide a analytical understanding of the reason for their good performance on various images from various vantage points. This report is organized into the following sections :

- → Section II is a basic overview of all the elemental "blocks" used at different stages in various models.
- → The various details related to the block level model design, experimental setup and other implementation details are provided as a part of Section III.
- → The results obtained from various experiments conducted by us on various popular datasets are summarized and compared with state-of-the-art techniques in Section IV.
- → In Section V, we provide a detailed analysis from various perspectives about the performance of different models that we experimented on.
- → In Section VI, we conclude our report and briefly review of some avenues of further research.

II. Our Methodology

Our approach is designed based on the *typical* top level block diagram shown in FIG. 1. *Visual Features* are first

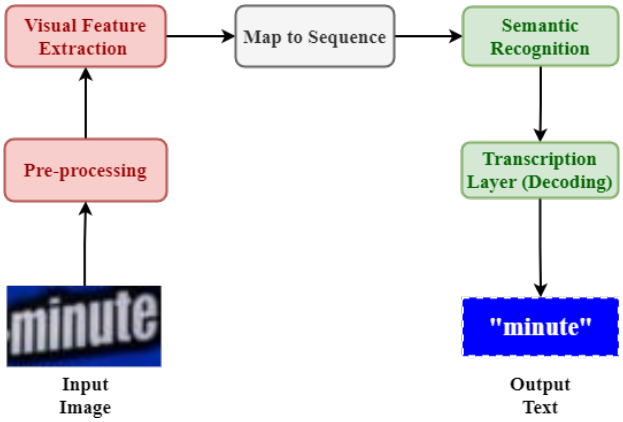

FIG. 1: High Level Block Diagram

extracted from the input image, in an automated fashion by the *Visual Feature Extraction* stage. These features are then mapped into sequence of feature of vectors which serve as the input for the *Semantic Feature Extraction* stage, that basically generates a probability distribution over the *output classes*, based on the semantic content *inferred*. The job of the *Transcription Layer* is to convert this distribution into a *predicted label sequence* using a suitable *decoding* strategy. Note that, in some of our models we also pass the images through an auxillary *Pre-Correction* Stage, in order to get its rectified version by *learnt* spatial transformations, that generally prove to be helpful for subsequent stages.

A. Visual Feature Extraction Stage

Visual Features are generally extracted in a *effective* manner by CNN's, due to their added benefits of *translational invariance*,*weight sharing* and *sparse connectivity*.

Operations	Architectural Parameters
2D Convolution	$\#\text{maps} : 512, \text{k} : 2 \times 2,\ \text{s} : 1, \text{p} : 0$
2D MaxPooling	Window: $1 \times 2,\ \text{s} : 2$
2D BatchNorm	–
2D Convolution	$\#\text{maps} : 512, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
2D BatchNorm	–
2D Convolution	$\#\text{maps} : 512, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
2D MaxPooling	Window: $1 \times 2,\ \text{s} : 2$
2D Convolution	$\#\text{maps} : 256, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
2D Convolution	$\#\text{maps} : 256, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
2D MaxPooling	Window: $2 \times 2,\ \text{s} : 2$
2D Convolution	$\#\text{maps} : 128, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
2D MaxPooling	Window: $2 \times 2,\ \text{s} : 2$
2D Convolution	$\#\text{maps} : 64, \text{k} : 3 \times 3,\ \text{s} : 1, \text{p} : 1$
Input Image	$W \times 32$ gray-scale image

TABLE I: Architecture of Basic CNN[8]

We experiment with two different blocks for this stage, namely *Basic CNN*[8] and *Modified ResNet18*, whose architectural parameters are displayed in TABLE I and TABLE II respectively.

Note that, the *Modified ResNet18*[1] possesses the following advantages, also reflected in performance, due to the use of residual skip-links[9] :

(i) Mitigates the *Vanishing Gradients* problem in deep netowrks.

(ii) Instead of say H(x), initial mapping, the network learns to a fit $F(x) := H(x) - x$ (turns out to be a simpler mapping to learn most of the times) which gives $H(x) := F(x) + x$.

(iii) Identity mappings through skip links avoid excessive non-linear *destructive* morphing of the input and help preserve the necessary features.

Layer Name	Output Size	Modified ResNet18
conv1	$112 \times 112 \times 64$	$7 \times 7, 64$, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Bridge conv6	$16 \times 1 \times 512$	$[4 \times 4, 512] \times 1$

TABLE II: Architecture of Modified ResNet18

B. Semantic Recognition Stage

The Semantic Recognition Stage predicts a conditional label distribution y_t for each frame x_t in the feature sequence vectors $\mathbf{x} = x_1, \ldots, x_T$. Generally, RNN's because of their following features, are used in this stage :

- Capture contextual information in a sequence which leads to good generalization as language is generally context dependent.
- Capable of operating on sequences of arbitrary lengths, traversing from <SOS> to <EOS> tags.

In this work, for our models we use a *Bidirectional, two layered stacked LSTM (Long Short Term Memory)* network as shown in the Fig. 2. The intuition behind choosing this network is summarized in the following distinctive advantages:

(i) Long-Term Dependencies are efficiently captured[2]

[1] inspired by the deeper versions of the ResNet[9] used for ImageNet Object Classfication

[2] This is made possible by the use of Selective Read, Write, Forget operations using multiple gates (input, output, forget gates resp.)

(ii) Solves the *Vanishing Gradients* problem in deep networks2[0]

(iii) The use of a Bidirectional network is beneficial due to the presence of *typical* useful context from sides in scene text images.

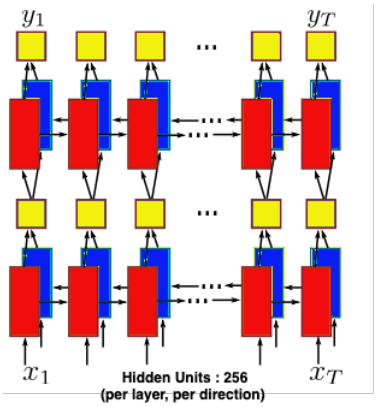

FIG. 2: Structure of BiLSTM[8]

C. Map to Sequence Block

The output of the *Visual Feature Recognition* stage is mapped into a sequence of feature vectors as mentioned to be given as an input to the next stage, by passing it through the *Map to Sequence Block*. Each vector corresponds is a descriptor of a rectangular receptive field portion of the image as shown in FIG. 3. Note that, during weight updation the

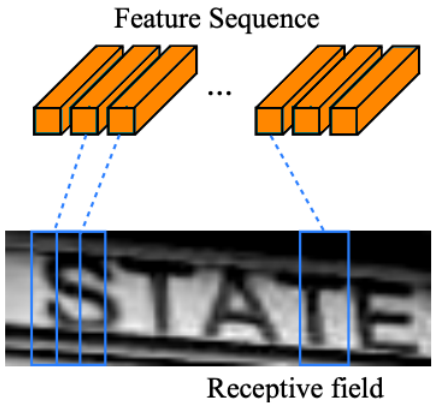

FIG. 3: Illustrative Representation of a Receptive Field[8]

error differentials are backpropagated using BPTT[3] algorithm, and then "sequences" of error differentials are concatenated to maps for using BackProp. to train lower layers i.e., basically this *Map to Sequence* operation is *inverted*.

D. Transcription Layer

In order to convert the sequence of probability distributions $\{y_t | t = 1, 2, \cdots, T\}$ predicted by the *Semantic Recognition* stage and also, for getting a *alignment-free* loss function for the End2End training of the network we utilize the *Connectionist Temporal Classification* Approach. Note, that each pdf y_t is

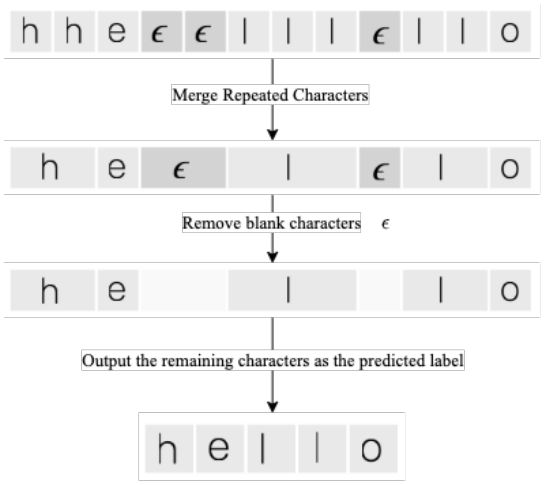

FIG. 4: Procedure for CTC Decoding

over all the Output classes (labels) and a "blank" character introduced by CTC. Now, a probability for the ground truth label $\mathbf{G}$ conditioned on the pdf sequence $\{y_t\}$ is calculated by the marginalizing over a set of valid alignments[4] $\mathcal{A}_{valid}$ as follows :

$$p(\mathbf{G} \mid \{y_t\}) = \sum_{A \in \mathcal{A}_{valid}} \left[\prod_{t=1}^{T} p_t\left(a_t \mid \{y_t\}\right) \right]$$

where $p_t\left(a_t \mid \{y_t\}\right)$ is the probability of having a particular element (character) a_t at time t according to a given valid alignment A.

The Loss Function $\mathcal{L}$ is then defined as:

$$\mathcal{L} = -\sum_{\mathbf{X}_i, \mathbf{G}_i \in \mathcal{X}} \log p\left(\mathbf{G}_i \mid \{y_t\}_i\right)$$

where $\mathbf{X_i}$ is the i^{th} training image and $\mathbf{G}_i$ is the i^{th} training ground truth label.

During the inference stage, we consider a *Lexicon-free* Transcription by considering the predicted label as the CTC Decoded version (See FIG. 4) of the alignment obtained by taking the $argmax$ element (character) of each pdf in $\{y_t\}$.

E. Pre-Correction Stage

In *Visual Feature Extraction* stage, one of the reasons for choosing a CNN was its translational invariance property[5]. Now, in order to try to make the model insenstive to rotations and also, be able to perform scaling, cropping and possibly other non-rigid deformations[5], we introduce an auxillary block called *Spatial Transformer Network (STN)*[5][2] before *Visual Feature Extraction*, in some of the models considered for experimentation (on irregular text images).

[3]BackPropagation Through Time

[4]Alignments are mapped onto labels using the CTC Decoding Procedure shown in FIG. 4

[5]Facilitated by the use of MaxPool Layers

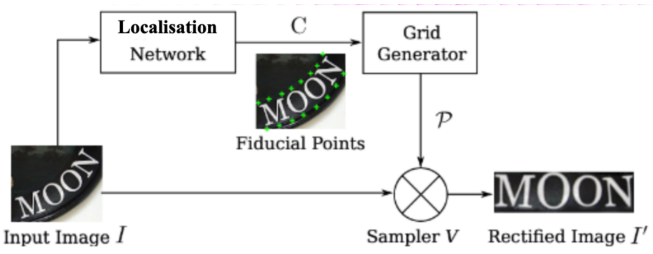

FIG. 5: Block Diagram for the Spatial Transformation Network (STN)[2]

A brief description of the various sub-blocks of the *STN* is shown below[2]:

- *Localisation Network*– Here, basically we perform a regression for locating a set C of k "*fiducial points*" on an image[6] using a simple CNN with *Conv2D*, MaxPool2D and fully connected layers[7]. The output fully-connected layer uses $tanh(.)$ activation and outputs $2k$ values (reshaped to k co-ordinates).
- *Grid Generator*– The Grid Generator basically creates a sampling grid, for mapping points of a standard normalized co-ordinates set C' of the rectified image to a region defined by C on the input image, by using the *Thin Plate Spline (TPS)* transformation[2].

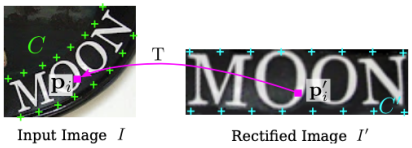

FIG. 6: An illustration of *TPS* [2]

- *Sampler*– Finally, the *Sampler* block takes in the sampling grid generated and uses *bilinear interpolation* to find the pixel values of the rectified image. Note that the sampler is *differentiable*, allowing End2End training using BackPropagation.

III. Experimental Setup

A. Datasets

We used Synth90k dataset for training purpose and other popular datasets such as IIIT5k, SVTP, ICDAR13, ICDAR15, CUTE80 etc. for testing purpose.

Synth90k is Synthetically generated dataset contain total 9 million images covering 90k English words and ground truth of all the images. From 9 million, 7.2 million images are for training and others for validation and testing. This dataset contains mostly regular images. we trained models for around 0.5 million images from this due to time and resources constraint.[10]

IIIT5k is developed from google image search by IIIT. It contains total 5000 almost regular images.[11]

SVT and SVTP contains images which are harvested from the google street view. SVT is the original dataset which contains 647 uncropped images. SVTP is the improved version of that which contains 645 cropped images with ground truth. the original images are mostly irregular. we used improved version SVTP for testing purpose.[12]

ICDAR13 is the dataset from *Robust reading competition* which contains around 1095 regular images.[13]

ICDAR15 is the next version of ICDAR13 where most of the images are from ICDAR13. It contains total 2077 images which includes regular as well as irregular. It also includes the real-life factors such as Occlusions, Motion blur, Noise etc. in the images.[14]

CUTE80 is also known as curved dataset which contains 288 high resolution curved text images. Initially, it used for text detection but later people are using for text recognition also.[14]

Born-digital images is generated from web and emails such as headings, advertisement etc. It is also from *Robust reading competition* contains 541 low-resolution almost regular images.[13]

English character dataset is generated by using different font styles which contains total 62 batches with 1016 images per batch. this 62 batches are from Uppercase letters(26), Lowercase latters(26) and Numerical from 0 to 9 (10). Most of the images are regular with different font styles.

B. Implementation details

Pytorch framework was used for training, testing and validation purpose. We used NVIDIA RTX GeForce 2060 GPU for training and validation while Google Colab's Tesla GPUs for testing on various other datasets.

PyTorch's Lightning Data Collator and Data Transformer was used for preprocessing of images such as *RGB to greyscale conversion*, transforming, resizing and normalizing wherever required in the model. The Train-Validation Split considered is 80% and 20 % respectively[8]. For the purpose of training, we use the CTC Loss (See Section II-D) and optimizer considered is the *Adam* Optimizer. Note that, checkpointing of models is done, whenever there is a decrease in the validation loss after every epoch.

C. Model design

Our whole work are basically classified into 2 stages : (1) Regular Text Recognition and (2) Irregular Text Recognition

1) Regular Text Recognition: The first model which we trained for this stage is known as **Basic CNN + BiLSTM** as shown in the figure 7. It is taking image as an input and passing through the *pre-processing layer* followed by main block basic CNN+BiLSTM and then Transcription layer which is consist of CTC loss function and CTC decoding. Note that, we designed all the blocks with parameters as mentioned in the previous sections.

[6]A normalized co-ordinate system is used with the image centered at origin
[7]specific details can be found in the code implementation

[8]Note that, batching of images is considered (32 images/batch for smaller models and 64 images/batch for larger models)

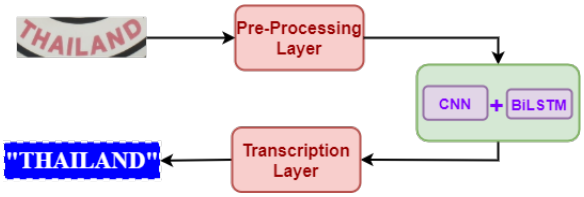

FIG. 7: CNN + BiLSTM Model for Regular Text Recognition

The second model is known as **ResBiL** which is consist of ResNet+BiLSTM. The model is same as figure 7 with change of main block to ResNet+BiLSTM.

As a Transfer Learning Experiment, we also trained a model *FrozenResNet+BiLSTM* by freezing the ResNet block weights obtained from a pretrained ResNet model trained on ImageNet Dataset for image classification. The results were poor as expected because the features extraction is *task-specific* and features extracted for image classification could be completely different from those extracted for scene text recognition.

2) Irregular Text Recognition: Here, we just added an extra **STN** part for correcting the orientation of the images after pre-processing block in both the previous models as discussed in the previous sections. So, The first model will be **STN + CNN + BiLSTM** as shown in the figure 8.

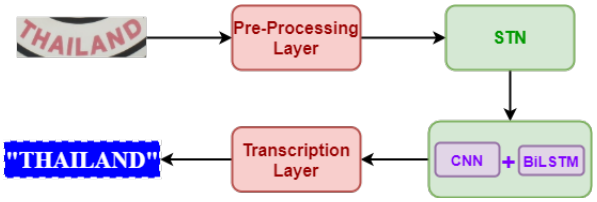

FIG. 8: STN + CNN Model for irregular Text Recognition

The second model is known as **STN+ResBiL** which is again same as figure 8 with change of main block as ResNet+BiLSTM.

D. Model Evaluation matrix

We used character accuracy (CA) and Word accuracy (WA) as the evaluation parameters for testing of models on various datasets. they can be calculated as per the equation 1 and 2 respectively.

$$CA(pred, truth) = \left(1 - \frac{Lev(pred, truth)}{\text{mod}truth}\right) \times 100\% \quad (1)$$

$$WA(pred, truth) = (pred == truth) \times 100\% \quad (2)$$

where, $Lev((pred, truth))$ is the Levenshtein Distance which is the min. of operations to be performed to convert one string to another one such as substitution, insertion and deletion.

For batch containing more than one image, CA and WA can be calculated as average of all images' CA and WA as per the above equations.

IV. Results

In this section, we are presenting the results of our proposed method. Since we have just used $\sim$10% of training dataset(Synth90k) and other works might have adopted the complete training sets (or might have used some special data augmentation schemes to increase the number of training samples), therefore it is impossible to make an absolutely fair comparison between different works. However, this analysis is very useful for evaluating the development of state-of-the-art methods in this field and establishing future directions.

Table III shows the comparison of results between some of the state-of-the-art methods and our proposed approaches over standard benchmark datasets, we can observe that there is a considerable differece in accuracy of our proposed methods and other works because of lesser size of training dataset. Table IV shows the comparison of results between our different proposed approaches over over the datasets mentioned in Section III-A. We can observe that STN+RseNet+BiLSTM model is giving best results among the proposed models.

	IIIT5k	ICDAR13	ICDAR15
Liu et al. [15]	85.2	91.1	**86.7**
Zhang et al. [16]	–	89.4	68.1
Luo et al. [17]	91.2	92.4	68.8
Du et al. [18]	88.6	93.8	79.9
Zhang et al. [19]	**94.7**	**94.2**	81.8
Chen et al. [20]	93.6	92.9	71.5
Yang et al.[21]	88.5	90	68.8
Basic CNN	65.55	73.24	36.18
ResNet+BiLSTM	67.77	70.51	39.92
STN+Basic CNN+BiLSTM	67.41	73.42	39.84
STN+ResNet+BiLSTM	66.81	71.16	41.17

TABLE III: Comparison of results among some of the state-of-the-art methods and our approaches

V. Analysis

In this section, we have tried to analyse and understand the models better based on the results obtained from the four models. This section includes comparison and analysis of the training parameters against epochs, some interesting and unique observations about the STN network and finally a comparison of word length versus accuracy.

A. Training Curves

Loss Comparison: All the models were trained for 6 epochs. The average training loss and the corresponding average validation loss found after all the epochs has been plotted on FIG. 9. From these loss plot, it is clear that there is a substantially good decrease in the loss between the first and second epochs for all the models. Out of these four models, *"STN+ResNet+BiLSTM"* model turned out to be the best model with lowest training and validation loss.

We can also observe a sharp decrease in the validation loss between the fifth and sixth epoch, which we suspect is due to insufficient number of epochs for that model. The high initial loss of the *"STN+Basic CNN+BiLSTM"* model maybe the due to the initial weights taken randomly for the model. Also,

Model		Synth90k	IIIT5k	SVTP	ICDAR13	ICDAR15	Born-Digital Images	CUTE80
Basic CNN	CA	91.86	81.44	47.13	85.05	60.45	85.29	45.28
	WA	78.74	65.55	26.51	73.24	36.18	72.1	39.37
ResNet+BiLSTM	CA	91.74	**84.29**	52.31	84	64.81	**86.7**	46.73
	WA	77.11	**67.77**	30.69	70.51	39.92	**72.14**	40.28
STN+Basic CNN+BiLSTM	CA	91.79	83.78	52.59	**85.13**	**66.49**	86.38	63.32
	WA	78.41	67.41	32.4	**73.42**	39.84	71.26	38.19
STN+ResNet+BiLSTM	CA	**92.45**	83.48	**52.59**	83.59	65.91	85.9	**65.57**
	WA	**79.92**	66.81	**34.73**	71.16	**41.17**	71.39	**40.28**

TABLE IV: Comparison of results between our approaches

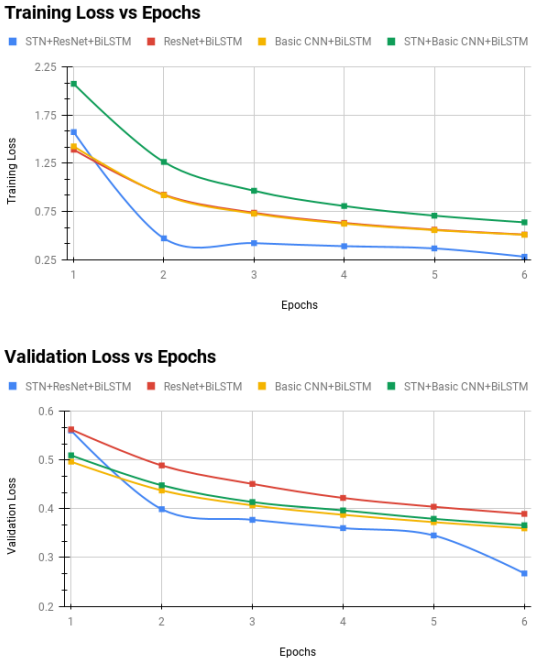

FIG. 9: Training and Validation Loss Plot for various models

based on the observation from the FIG. 9 plot, we are sure that the model hasn't reached over-fitting yet, since both the loss are still decreasing.

Character Accuracy Comparison: An opposite trend is also followed by the Character Accuracy plot. A high learning rate for characters at the initial epochs is shown by all the models. The *"STN+ResNet+BiLSTM"* model is also the best model from this trend as expected. Also, we see a similar increase in Character Accuracy towards the fifth and sixth epoch like the Loss trend. Apart from the *"STN+ResNet+BiLSTM"* model, other models show almost constant Character Accuracy after the second epoch.

Word Accuracy Comparison: The Word Accuracy trend(See FIG. 10) is similar to the Character Accuracy trend. Here, we can observe a high increase in Word Accuracy shown by all the models between the first and the second epoch but, the word accuracy value is still average. The *"STN+ResNet+BiLSTM"* model is again the best model from this trend as expected. But, we can see an significant increase in both the Training Word Accuracy and the Validation Word Accuracy during the last epochs for all the models, which suggests that the models are first trying to learn character recognition and then trying to learn character sequencing, which is also an interesting finding for our models behaviour.

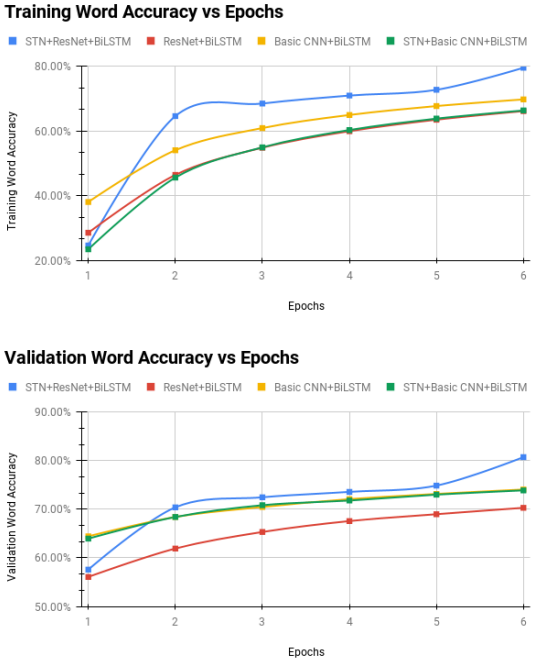

FIG. 10: Training and Validation Word Accuracy Plot for various models

B. Output Characteristics of STN

This subsection captures some of the unique and interesting features of the STN Transformation. Since the STN network is end-end trainable, we don't have any control on what transformation it tries to make for a better learning. In FIG. 11, we can see that the STN tries to transform all the images so that there is a specific orientation for the text parts. We suspect that the reason for such a transformation is the mean orientation of the text inside the training images used.

In FIG. 12, we can see how the STN tries to distort the irrelevant texts inside an image and focus on only the relevant part. This helps the upcoming neural networks to predict text in a better way by ignoring the background clutters. The reason for this distortion is the TPS-transformation used inside the

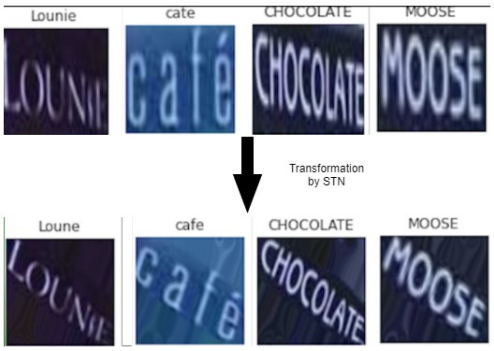

FIG. 11: Change of orientation by STN

STN network which distorts the surrounding keeping only the relevant information in focus.

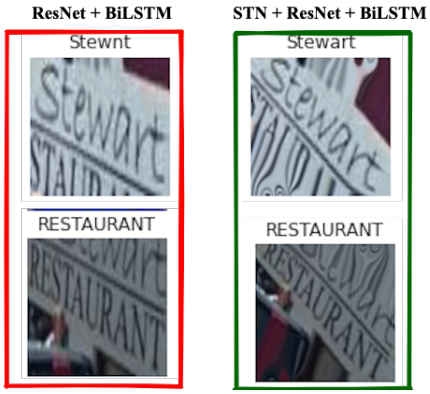

FIG. 12: Distortion of irrelevant information by STN

C. Accuracy against Word Length

Based on the word accuracy versus length comparison for models without STN(See FIG. 13) and for models with STN (See FIG. 14), there is high accuracy for an average word length for almost all the model. The *"Basic CNN+BiLSTM"* model has shown good performance over wider range and the reason could be the varying length of the images. Also, the square size pre-processing image in *"ResNet+BiLSTM"* model could be a reason for poor performance over wider range.

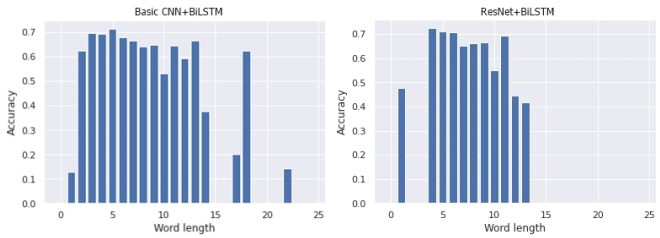

FIG. 13: Word Accuracy versus Length for non-STN Models

Addition of STN has led to increase in the word accuracy beyond 70% for some of the word lengths as seen in FIG. 14. Also, from our observation in we see that *"STN+Basic CNN+BiLSTM model"* is length selective, giving very good results for some lengths and very poor for some other lengths. For the *"STN+ResNet+BiLSTM"* model, again square processing of images could be the reason for restriction in accuracy performance over wider word length ranges.

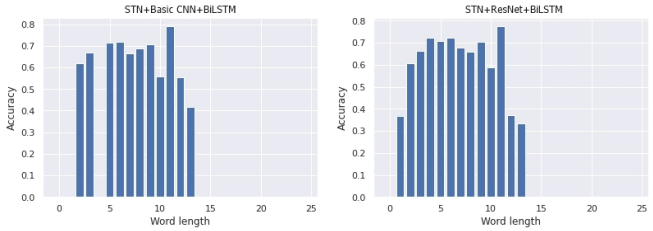

FIG. 14: Word Accuracy versus Length for STN models

VI. Conclusion and Future Work

In this report, we have presented four end-to-end models which can be broadly be classified into *"CNN+RNN"* with or without STN. Out of the four models discussed the *"STN+ResNet+BiLSTM"* model turned out to be the best among all our models. The peculiar characteristics of our models makes it more interesting and opens an area for future work. The immediate future works can be training model with larger data set containing lots of variety and orientation of images since our model was trained on around 7% images of the total Synth90K data. Combination of transformer along with STN can be another direction for future work(Also suggested by [22]). Addition of a small-sized lexicon to this model just to give a bias about some day-to-day English words may make the model very powerful as speculated by the authors.

Acknowledgment

We are indebted to Dr. Deepak Mishra, Professor and Head, Department of Avionics, Indian Institute of Space Science and Technology for his wonderful guidance, teaching and providing us with an opportunity without which this piece of work couldn't have been successful.

Bibliography

[1] E. Zacharias, M. Teuchler, and B. Bernier, "Image processing based scene-text detection and recognition with tesseract," *arXiv preprint arXiv:2004.08079*, 2020.

[2] B. Shi, X. Wang, P. Lv, C. Yao, and X. Bai, "Robust scene text recognition with automatic rectification," *CoRR*, vol. abs/1603.03915, 2016. [Online]. Available: http://arxiv.org/abs/1603.03915

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[4] X. Yang, D. He, Z. Zhou, D. Kifer, and C. L. Giles, "Learning to read irregular text with attention mechanisms." in *IJCAI*, vol. 1, no. 2, 2017, p. 3.

[5] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015. [Online]. Available: http://arxiv.org/abs/1506.02025

[6] Z. Cheng, X. Liu, F. Bai, Y. Niu, S. Pu, and S. Zhou, "Arbitrarily-oriented text recognition," *CoRR*, vol. abs/1711.04226, 2017. [Online]. Available: http://arxiv.org/abs/1711.04226

[7] F. Zhan and S. Lu, "Esir: End-to-end scene text recognition via iterative image rectification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2059–2068.

[8] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *CoRR*, vol. abs/1507.05717, 2015. [Online]. Available: http://arxiv.org/abs/1507.05717

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[10] "Synth90k dataset," Available at https://www.robots.ox.ac.uk/~vgg/data/text.

[11] "Iiit5k dataset," Available at https://cvit.iiit.ac.in/research/projects/cvit-projects/the-iiit-5k-word-dataset.

[12] "The street view text dataset," Available at http://vision.ucsd.edu/~kai/svt/.

[13] R. R. Competition, "Robust reading competition," Available at https://rrc.cvc.uab.es/.

[14] M. Yang, "Test datasets," Available at https://github.com/bgshih/aster/releases/.

[15] W. Liu, C. Chen, and K. K. Wong, "SAFE: scale aware feature encoder for scene text recognition," *CoRR*, vol. abs/1901.05770, 2019. [Online]. Available: http://arxiv.org/abs/1901.05770

[16] Y. Zhang, S. Nie, W. Liu, X. Xu, D. Zhang, and H. T. Shen, "Sequence-to-sequence domain adaptation network for robust text image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] C. Luo, L. Jin, and Z. Sun, "A multi-object rectified attention network for scene text recognition," *CoRR*, vol. abs/1901.03003, 2019. [Online]. Available: http://arxiv.org/abs/1901.03003

[18] X. Du, T. Ma, Y. Zheng, H. Ye, X. Wu, and L. He, "Scene text recognition with temporal convolutional encoder," *CoRR*, vol. abs/1911.01051, 2019. [Online]. Available: http://arxiv.org/abs/1911.01051

[19] H. Zhang, Q. Yao, M. Yang, Y. Xu, and X. Bai, "Efficient backbone search for scene text recognition," *CoRR*, vol. abs/2003.06567, 2020. [Online]. Available: https://arxiv.org/abs/2003.06567

[20] X. Chen, T. Wang, Y. Zhu, L. Jin, and C. Luo, "Adaptive embedding gate for attention-based scene text recognition," *CoRR*, vol. abs/1908.09475, 2019. [Online]. Available: http://arxiv.org/abs/1908.09475

[21] M. Yang, Y. Guan, M. Liao, X. He, K. Bian, S. Bai, C. Yao, and X. Bai, "Symmetry-constrained rectification network for scene text recognition," *CoRR*, vol. abs/1908.01957, 2019. [Online]. Available: http://arxiv.org/abs/1908.01957

[22] M. Bleeker and M. de Rijke, "Bidirectional scene text recognition with a single decoder," *CoRR*, vol. abs/1912.03656, 2019. [Online]. Available: http://arxiv.org/abs/1912.03656

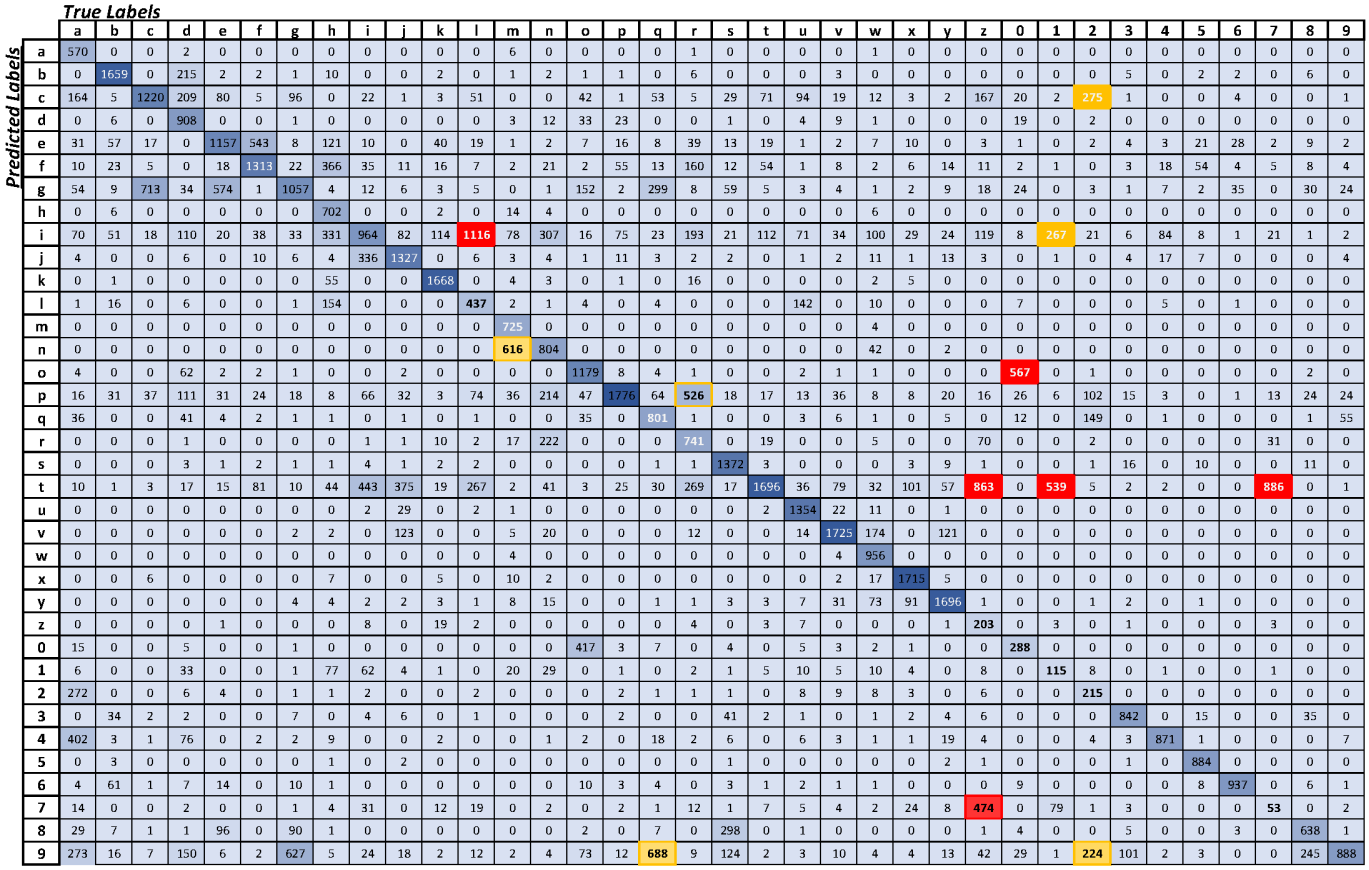

FIG. 1: Character Confusion Matrix

APPENDIX A
CHARACTER CONFUSION ANALYSIS

In this section, we have tried to analyse and understand how our models predicts image containing only single character. As we have trained our model for detecting sequential text, the receptive fields used in the CNN is causing overlapping of the sequences and prediction error and making it to predict more than one character for some of the cases.

For this analysis we tested the English character Dataset on *"STN+ResNet+BiLSTM"* model and generated a confusion matrix for one character predictions.

As sometimes the model is predicting more than one character, the highest length of predicted word for one character set is 5. And M is the most wrongly predicted character in terms of extra word length.

For single character prediction, with almost comparable accuracy with other datasets the diagonal of the matrix is almost perfectly aligned. As expected the model is also committing human errors frequently like predicting 'zero' as 'O' or predicting 'I' as 'l' or predicting 1 as I. The model is also getting confused between some lowercase and uppercase letters like (z,Z) or (y,Y) or (x,X) or (w,W) or (v,V) etc.

The two most predicted characters are T and I respectively. Probably because the model is giving a high priority to the upper horizontal bar for predicting T. So, most of the letters having only one horizontal bar at top(like I,J,7 not E.F) are predicted as T. It is predicting I for most of the cases where there is a presence of vertical bar. If the characters don't have a vertical bar(like 1,l,T H etc).

There are some case where the model is predicting a class more than or almost equal to what it is predicting for it's own true class (Ignoring cases). Some examples are tabulated below:

	True class	Predicted class
Greater than true class	l	I
	z, 7, l	T (t)
	0(Zero)	o
	7	z
Comparable to true class	R (r)	p
	m	n
	q	9
	z	7

TABLE I: Predictions for some selected class