ENERGY-EFFICIENT VIDEO TEXT-SPOTTING

A Thesis

by

RAHUL SRIDHAR

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Zhangyang Wang
Committee Members,	Nima Kalantari
	Chao Tian
Head of Department,	Scott Schaefer

May 2021

Major Subject: Computer Science

ABSTRACT

There is a lot of research done to increase the accuracy and reduce the latency of deep learning algorithms. But, there is very little research done to reduce the energy consumption of the deep learning models. For applications that require deploying the deep learning models on the edge devices that have low compute resources, it is important that these algorithms are energy-efficient. Efficient Video Text Spotting is the field that deals with developing deep learning models to be deployed on edge devices to detect, localize, and recognize text appearing in the frames of the videos. Previous methods followed a four-step pipeline: text detection in every frame, text recognition for the localized text region in every frame, tracking text streams, and post-processing. The two main problems with the above approach are high computational cost and low performance. This thesis focuses on the text spotting model design for an Efficient Video Text Spotting System. In this thesis, model design experiments are carried out keeping efficiency in mind. Two different real-time text spotting models were experimented i.e. ABCNet and FOTS. For ABCNet different backbones, normalization schemes, and feature pyramid variations are experimented with to attain the best accuracy and energy tradeoff. For the FOTS model, the two-step text spotting and two-stage text spotting model design are experimented. The influence of various factors such as bounding box to character count ratio, character count, blur level, bounding box count, bounding box area are experimented. From the experiments, it was observed that the two-step text spotting model design method performed better for all resolutions. Further, it was observed that the recognition performance improves with a higher bounding box to character count ratio and lower character count. The energy measurement of the two-step FOTS text spotting model on Raspberry Pi is also presented.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to Professor Zhangyang Wang, my research supervisors, for his patient guidance, enthusiastic encouragement and useful critiques of this work. I would also like to thank Professor Nima Kalantari and Professor Chao Tian for their advice and being the committee members.

My grateful thanks are also extended to the group of Visual Informatics Group @ University of Texas at Austin lab directed by Professor Zhangyang Wang for their insightful suggestions.

I would also like to thank all the faculty members of Department of Computer Science & Engineering for their help in offering me the resources and advice throughout my masters program.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Zhangyang Wang [advisor] and Professor Nima Kalantari of the Department of Computer Science & Engineering and Professor Chao Tian of the Department of Electronics & Communication Engineering.

The model used for deployment on Raspberry Pi was provided by Yunhe Xue of the Department of Electrical & Computer Engineering.

Funding Source

There are no outside funding contributions to acknowledge related to the research and compilation of this document

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iii

TABLE OF CONTENTS iv

LIST OF FIGURES vi

LIST OF TABLES vii

1. INTRODUCTION 1

2. LITERATURE REVIEW 3

2.1 Scene Text Spotting in Images and Videos 3
2.1.1 Scene Text Spotting in Images 3
2.1.1.1 Two-Step Text Spotting Pipeline 3
2.1.1.2 Two-Stage Pipeline 3
2.1.1.3 One-Stage Pipeline 5
2.1.2 Scene Text Spotting in Videos 5
2.2 ABCNet 7
2.3 FOTS 8

3. PROPOSED METHODOLOGY 10

3.1 Overview of the entire system 10
3.1.1 Text Spotting Model 10
3.1.1.1 Two-step Crop+Resize based Text Spotting Models . . 10
3.1.1.2 Two-stage Aligned RoIPool based Text Spotting Models 11
3.2 Auto Labeling: Vision-Aided Annotation for Video Text Spotting 12

4. EXPERIMENTS AND DATASETS 15

4.1 Dataset 15
4.2 Evaluation Metric 16

4.3 Energy Measurement Tool 16
4.4 ABCNet Experiments 17
4.5 FOT++ Experiments 26
4.6 Model Deployment Results 29
4.7 FLOPs Calculation and Speed Test 29

5. CONCLUSION 31

REFERENCES 33

LIST OF FIGURES

FIGURE Page

2.1 Quadratic Bezier Curve Construction 7
2.2 ABCNet Model Architecture 8
3.1 Two-step FOT++ Model Architecture 11
3.2 Two-stage FOT++ Model Architecture 12
4.1 Proof of Concept Experiments for Energy Measurement tool 17

LIST OF TABLES

TABLE Page

4.1 Effect of Cropping Text Region and Upscaling Images 19

4.2 Experimental Setup for different ABCNet Models 21

4.3 Experimental Results for different ABCNet Models 22

4.4 Effect of Case-Sensitive Labelling 23

4.5 Effect of Finetuning with Case-sensitive Dataset with Repo Model 23

4.6 Effect of Different Scale of Feature Pyramid Network used for Detection and Recognition 24

4.7 Effect of Different Cropped Images used for training 26

4.8 EAST + CRNN (Image) results on Case-Sensitive Sample Dataset Test Set 28

4.9 EAST + BezierAlign + CRNN (Feature Map) results on Case-Sensitive Sample Dataset Test Set 28

4.10 CRNN+Attn (Image) results on Case-Sensitive Sample Dataset Test Set . 29

4.11 CRNN+Attn (Feature Map) results on Case-Sensitive Sample Dataset Test Set 29

4.12 Performance, Latency, and Energy Measurement of two-step FOT++ model on Raspberry Pi 30

4.13 FLOPS calculation of ABCNet for different resolution 30

4.14 Speed Test (in ms) for different components of FOTS and ABCNet 30

1. INTRODUCTION

The text has been a very important method to convey information. Recognizing the text in natural environments has numerous applications such as robot navigation, building assistive technology for disabled people, and translation of board signs for tourists. The task of detecting and recognizing the text appearing in an image or video is called text spotting. The two main components of the text spotting pipeline are the detector and recognizer. The task of the detector is to detect the text regions in the image and the task of the recognizer is to recognize the text in the localized text regions.

Most previous works in video text spotting follow a four-step process which includes detecting the text regions, recognizing the localized text regions, tracking the same text region temporally for the same scene frames, post-processing to generate final results. This method has huge computational complexity make it unreal for resource-constrained edge devices and lower performance due to recognizing every text region especially those frames containing heavy artifacts such as high motion blur. Hence a select and recognize strategy is preferred for video text spotting similar to FREE [1]. Most of the research in the field of text spotting for images focuses on improving the performance by increasing the model complexity making it unfit to deploy on resource-constrained edge devices. The Efficient Video Text Spotting system used in this work consists of a two-stage selector and the text spotting model. The two-stage selector task is to reject non-text frames temporally and to perform spatial aware cropping for the frames that contain text. The text spotting model used is the two-step text spotting model consisting of EAST [2] text detector and the CRNN [3] recognizer. The recognizer makes predictions on the input image. To deploy the model on Raspberry Pi quantization and compression are done. This thesis focuses on the efficient text spotting model design for the system.

Two real-time models used are ABCNet [4] and FOT++. For ABCNet different components of the model are experimented with an aim to have good energy to accuracy trade-off. Below are the set of experiments carried out for ABCNet - Effect of Cropping Text Regions and Upscaling during Inference - Experiment Results for different components of ABCNet model in terms of normalization scheme, backbone, training resolution - Effect of Case-Sensitive Labelling - Effect of Finetuning with Case-sensitive Dataset with Repo model - Effect of Different Scale of Feature Pyramid Network used for Detection and Recognition - Effect of Different Cropped Images used for training

The FOT++ experiments are carried out for two-step text spotting and two-stage text spotting methods. The first set of experiments involve using the EAST bounding box predictions for recognition. The second set of experiments use a ground truth bounding box instead of an EAST detector. This is done to remove the influence of the detector to understand the performance of both methods. The results indicate that the two-step text spotting method performs better than the two-stage text spotting method for all input resolutions. Experiments are also conducted to understand the effect of parameters such as bounding box area, bounding box count, blur level, bounding box area to character count ratio, and character count on the recognition results. Further energy measurement results of the FOTS text spotting model deployed on the Raspberry Pi. The proof of concept for the energy measurement tool is also done.

2. LITERATURE REVIEW

2.1 Scene Text Spotting in Images and Videos

In this literature review, only deep learning-based methods are discussed.

2.1.1 Scene Text Spotting in Images

The scene text spotting models developed consist of detection and recognition both of which are trained end-to-end. The deep learning-based scene text spotting models can be categorized as Two-Step Pipelines, Two-Stage Pipelines, and One-Stage Pipeline [5].

2.1.1.1 Two-Step Text Spotting Pipeline

The two-step text spotting model consists of a detector to predict bounding boxes which are cropped from the input image and fed to the recognizer. Jaderberg et al. [6] use a combination of Edgeboxes [7] and weak aggregate channel features for proposal generation. The generated proposals are filtered, refined, and fed to the recognizer. The recognizer makes a classification of the proposal into one of the words in a pre-defined dictionary. Liao et al. [8] uses the TextBoxes [8] inspired by SSD [9] for text detection and CRNN [3] for text recognition. The detection score used for training the detector uses information from the recognizer to remove false-positive detection to improve the detection performance.

2.1.1.2 Two-Stage Pipeline

Liao et al. [10] propose an end-to-end deep learning model with detector and recognizer trained in a semi-supervised manner where the ground truth only contains the text labels associated with the images. No ground truth bounding box is used for training. The detection part uses a spatial transformer consisting of first finding the parameters to apply a spatial transformation, then creating the sampling grid, and finally applying bi-

linear interpolation to generate spatially transformed input image region containing text. This is then fed to the recognizer. FOTS [11], Deep TextSpotter [12], and text spotter by He et al. [13] have similar architecture. FOTS [11] uses EAST [2] detector. The fixed height varying width features are extracted from the feature map of the detector using the detector bounding box predictions. This is then fed to a CTC [14] based recognizer consisting of convolutional, LSTM [15], and FC layers. Deep TextSpotter [12] uses YOLOv2 [16] and Region Proposal Network for the text detection. The YOLOv2 by removing the fully connected layer is a fully convolutional network with the output dimension of $W/32 \times H/32 \times 1024$ where W and H are the width and height of the source image. The region proposal network takes in $W/32 \times H/32 \times 1024$ as input tensor and returns outputs a tensor of dimension $W/32 \times H/32 \times 6k$ where k is the number of anchor boxes for every point in the final feature map and 6 is the number of predicted parameter for one anchor box which include 2-position, dimensions, and the rotation angle of the bounding box. The predicted bounding boxes are then converted into fixed height tensor and bilinear sampling is used for mapping from the feature map to the input of the text recognizer. For text recognition, a fully convolutional network is used which takes W*H*C as input and returns W/4*A as output where A is the number of alphabets. The model then uses CTC to predict word labels for the predicted bounding box. He et al. [13] use PVANet [17] as the text detector and the text recognition branch consists of LSTM [15] and attention [18]. It uses bilinear sampling to map the arbitrary shaped quadrilateral text detection predictions region to fixed-size features which are then fed to the recognizer. MaskTextSpotter [19] is a modification of Mask-RCNN [20]. The Resnet [21] is used for feature extraction and FPN [22] is used to make the model robust to scale. The region proposal network (RPN) [23] is then used to propose candidate text regions. The candidate text regions of arbitrary shape are then converted to a fixed size using RoIAlign. This is then passed to two branches. The first branch has the horizontal box classification and box regression similar

to the Fast R-CNN. The second is the mask branch which takes fixed size features from the RoIAlign as input, applies convolutional layers, and outputs word-level segmentation and character level segmentation masks. A post-processing step is carried out to recognize the words present in the image along with their word-level and character-level masks. Qin et. al. [24] textspotter is again a modification of the MaskTextSpotter. The detection module is similar to Mask R-CNN [20] to generate the bounding box and mask for each text region in the image. The recognition module takes one-eight and one-fourth of the input dimension from the feature extractor as input and fuses the features. This is then fed to the RoIMasking module which extracts fixed size features but at the same time multiplies with the segmentation mask predicted by the detection module. This allows the decoder to focus on the text region to make the final predictions. The decoder consists of LSTM and attention.

2.1.1.3 One-Stage Pipeline

Xing et al. [25] is the first one-stage text spotting model wherein detection and recognition are done parallelly. The model consists of two branches. The first branch is the detection branch which predicts the bounding box surrounding each text in the image. The second branch is the character branch which performs detection and recognition at the character level. The model is trained end-to-end using the word-level and character-level bounding boxes and labels.

The two real-time text spotting models used for the model design for the FOT++ text spotting model are FOTS and ABCNet [4]. Below is an explanation of the two models.

2.1.2 Scene Text Spotting in Videos

Nguyen et al. [26] was the first method to come up with an end-to-end solution for the video text spotting problem. The entire system consists of character detection per frame, temporal smoothing, word detection, temporal smoothing, and Linker to get the final pre-

dictions. The temporal smoothing is done to remove the false positives and reduce the word detection search space. A linker is used to combine the same text across frames and linearly interpolate the text location in frames that might have been missed due to image artifacts. Merino-Gracia et al. [27] developed an real-time end-to-end video text spotting method for large text in outdoor environments. The method uses light-weight real-time text detection and text aggregation modules. A tracking module is proposed to keep track of the same text region across frames. It uses Tesseract OCR [28] for text recognition. The recognition is performed parallelly with the text tracking system. From the available text tracker, the recognizer selects the tracker based on recognition availability for a text instance, a recognition confidence score for the same text instance if exists, and time since the last recognition prediction. The same tracker might be used by the recognizer and the result with the highest confidence is kept. Wang et al. [29] used an end-to-end text spotting model where the detector is similar to Fast R-CNN with feature extractor, RPN, RoI Pool. The detector predicts 8 coordinate bounding boxes to handle the text of arbitrary orientation horizontal text. The detected text regions are fed to the recognizer consisting of the LSTM [15] to obtain the final recognition results. It uses a tracking-by-detection strategy to combine text regions across frames in order to obtain the tracking trajectory. The tracking trajectory is used to improve the detection and recognition results. The same text region missed for certain frames due to artifacts could be added by linearly interpolating the detections using the trajectory for the text instances across the frames. To improve the recognition result the text prediction label which appears the most number of times in the text trajectory across frames is considered as the final prediction result for all the frames in the text trajectory. FREE [1] proposes a video text spotting framework that can be trained end-to-end. It follows the strategy of select and recognizes instead of applying a recognizer module for all the frames. The method consists of two parts, a spatial-temporal video text detector, and the text recommender. The text detector module uses TextPercep-

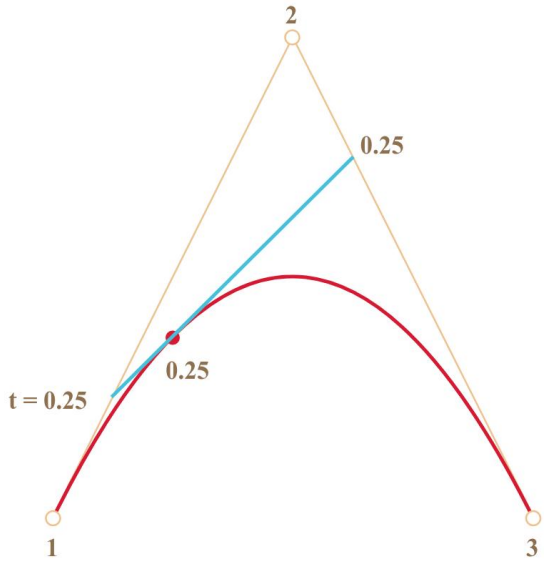

Figure 2.1: Quadratic Bezier Curve Construction

tron [30] for text detection for every frame. The detection results are improved by using the temporal information with the help of a spatial-temporal aggregation strategy. The text recommender consists of three parts. The first is the text tracking branch to group the same text instance across frames. The second branch is the text quality branch which assigns a numeric score in $[0, 1]$ where 1 means high-quality frame without artifact and 0 means the low-quality frames containing motion blur and other artifacts. The third branch is the text recognition module which is an LSTM based attention decoder. The text tracker groups the same text instance across frames, the text quality scoring branch assigns a score for each frame, the text recognition then makes recognition prediction for the text instance with the highest quality from the group of all the instances of the same text.

2.2 ABCNet

The Adaptive Bezier Curve Network (ABCNet) model uses the concept of Bezier curves. Bezier curve is generated using control points. Figure 2.1 shows the quadratic bezier curve generation using three control points. In the figure 1, 2, 3 are the control

Figure 2.2: ABCNet Model Architecture

points of the bezier curve. To obtain one point on the bezier curve first two points on the line segment 12 and 23 at a distance t are obtained. These two points are then used to draw the blue segment. A point at a distance of t on the blue line segment originating from the line segment 12 is one point on the bezier curve. This is repeated for all the values of t in the interval [0,1] to obtain the bezier curve. The ABCNet model converts arbitrary-shaped scene text detection into eight control point bounding box regression problem. It uses FCOS [31] based detector which is a single-shot anchor free detector that makes predictions in a per-pixel fashion. Bezier Align is used to extract fixed-size features. It samples equally spaced points along the row and the column wherein the columns are orthogonal to the two bezier curves and the points are bilinearly interpolated. The fixed size features extracted using Bezier Align are fed to the CRNN text recognizer. The text recognizer consists of three convolution blocks, followed by CRNN and Attention. Figure 2.2 shows the ABCNet model architecture. FOTS model can be seen as a special case of ABCNet.

2.3 FOTS

The Fast Oriented Text Spotting (FOTS) [11] is an end-to-end real-time oriented text spotting model. It uses EAST as the text spotter. The EAST detector is similar to U-Net in that it has a downsampling path followed by the upsampling path where the features

are merged. The output feature map is one-fourth of the resolution of the original image. Lastly, one convolution is applied to the feature map obtained from the upsampling path. This gives six channels wherein each pixel of each channel represents the probability of text, top, left, right, and bottom, and the orientation of the bounding box. Finally, NMS is applied to the positive samples. The bounding box predictions from the EAST [2] detector is fed to the RoIRotate to obtain features of fixed height and varying width length to maintain the aspect ratio. The RoIRotate first applies Affine transformation and bilinear interpolation to obtain fixed height features. This is then fed to the text recognizer. The recognizer consists of convolution and max pool along the height axis followed by LSTM and CTC.

3. PROPOSED METHODOLOGY

3.1 Overview of the entire system

The entire video text spotting pipeline consists of two main parts i.e. the two-stage selector and the FOT++ text spotting model. The first stage in the two-stage selector is designed to reject non-text frames and low-quality text frames. The task of the second stage to perform spatial aware cropping of the text regions in the text frames. The cropped text region is then fed to the FOT++ text spotting model to detect and recognize the text present in the frames. To deploy the model on Raspberry Pi quantization and pruning are done. My work was focused on the text spotting model design. The remainder of the thesis will focus on ABCNet [4] and FOT++ model design and various aspects of the text spotting model design would be discussed.

3.1.1 Text Spotting Model

The text spotting model consists of two main parts. The detector module and the recognizer module. Both of these modules are connected with an Aligned RoIPool based component to extract fixed size features from detection predictions and feed them to the recognizer. The two major methods to design a deep learning text spotting model can be classified into two-step Crop+Resize text spotting model and two-stage Aligned RoIPool text spotting model. Since our problem requires real-time predictions subsequent sections only describe real-time deep learning models.

3.1.1.1 Two-step Crop+Resize based Text Spotting Models

In this class of text spotting models, the detector and the recognizer are trained separately. During inference, the predicted bounding box from the text detector is used to extract fixed size features directly from the image using Bezier Align. We call this model

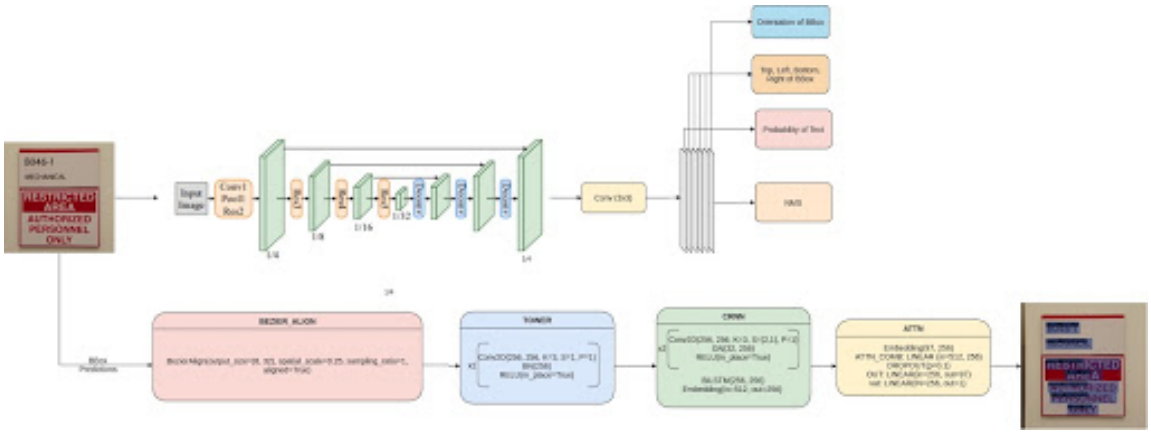

Figure 3.1: Two-step FOT++ Model Architecture

to be two-step FOT++ as it is an extension of the FOTS [11] model. The text detector used is EAST and the text recognizer used is CRNN. The advantage of this method is that it gives flexibility in changing the input dimension fed to the detector and the recognizer. Since text detection is an easier problem when compared with recognition as it only needs to identify and localize text regions, a lower resolution input image is sufficient to achieve good performance. Text recognition results depend on the content of the text so it is important to feed high-resolution image to the recognizer. With a two-step text spotter, it is possible to feed low-resolution images to the detector and original resolution images to the recognizer. This helps to reduce energy consumption while maintaining performance. Figure 3.1 shows the model architecture of the two-step FOT++ text spotting model.

3.1.1.2 Two-stage Aligned RoIPool based Text Spotting Models

In this class of text spotting models, the detector and the recognizer are trained in an end-to-end fashion. During inference, the predicted bounding box from the text detector is used to extract fixed size features from the detector feature map and fed to the recognizer. For our experiments, two different text spotting models are used i.e. ABCNet and two-stage FOT++. The ABCNet model uses FCOS as the text detector and the text recognizer

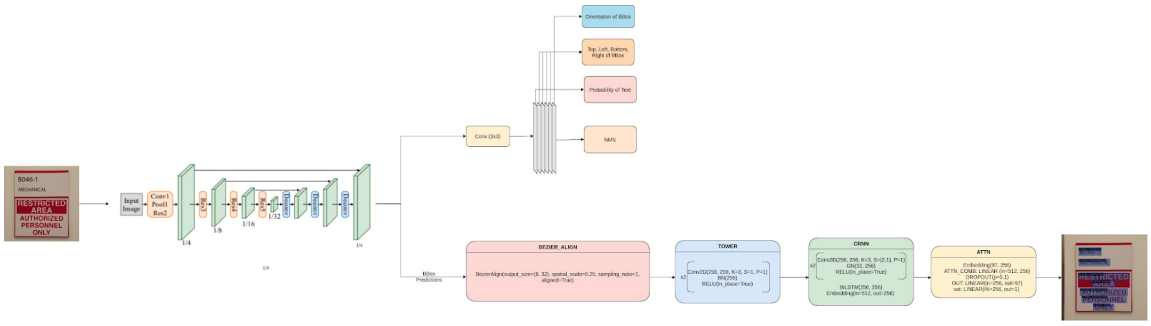

Figure 3.2: Two-stage FOT++ Model Architecture

is CRNN with attention. Bezier Align is used to extract fixed-size feature representation. During training, the text detector is fed the input image and the predicted bounding box along with the ground truth bounding box is used for loss calculation. For the recognizer, the ground truth bounding box is used to extract fixed-size features from the feature map of the detector and fed to the recognizer. The two-stage FOT++ uses EAST as the text detector. The text recognizer and the fixed size feature representation extractor used are the same as the ABCNet. The detector is first trained and the weights are frozen. Then for training the recognizer the grounding bounding box is used to extract fixed size features from the feature map of the detector. Figure 3.2 shows the model architecture of the two-stage FOT++ text spotting model.

3.2 Auto Labeling: Vision-Aided Annotation for Video Text Spotting

This section introduces the Auto Labeling algorithm to label the LPCVC [32] Challenge dataset. Each frame in a video is annotated by the bounding box location of the text and the text present in the bounding box. For a video the scene does not change drastically for adjacent frames, resulting in many frames with similar scenes. It is possible to automatically annotate one group of frames by manually annotating one of the frames. We use feature matching and homography to transfer the annotations from one frame to another. Since the UAV can move horizontally or vertically the homography assumption of

the frames being on the same planar surface breaks. This could lead to errors in obtained annotations. Hence, after obtaining the annotations using the auto-labeling algorithm we manually verify each annotation and make the required corrections. Algorithm 1 shows the above-described algorithm.

Algorithm 1: Auto-Labeling

$I_s \leftarrow V[1]$
$b_s \leftarrow \text{Annotate}(I_s)$
$N \leftarrow \text{Size}(V)$
for $i \leftarrow 2$ **to** N **do**
    $I_t \leftarrow V[i]$
    $k_1, d_1 \leftarrow \text{SIFT}(I_s)$
    $k_2, d_2 \leftarrow \text{SIFT}(I_t)$
    $m \leftarrow \text{BFMatcher}(d_1, d_2)$
    $m' \leftarrow \text{LoweRatioTest}(m)$
    $p_s \leftarrow \text{FilterKeyPts}(m', k_1)$
    $p_t \leftarrow \text{FilterKeyPts}(m', k_2)$
    $M \leftarrow \text{FindHomography}(p_s, p_t)$
    $b_t \leftarrow \text{PerspectiveTransform}(b_s, M)$
    $I_s \leftarrow I_t$
    $b_s \leftarrow b_t$

The psuedo code shown in Algorithm 1 uses feature matching and Perspective Transformation to find the annotation b_t of the destination frame I_t given the annotation b_s of the source frame I_s. First the keypoints k_1, k_2 and the descriptors d_1, d_2 for the both the frames are found using SIFT [33]. Second, the two best matches of every descriptor of source frame with respect to every descriptor of the destination frame. Third, Lowe's Ratio test [33] is applied to get best keypoints p_s, p_t for which the first and second best match are good. Fourth, homography matrix M is found using RANSAC [34] technique using the good features from source and destination frame. Fifth, the annotation for the destination

frame b_t is obtained by doing Perspective transformation for the source frame annotations b_s using the homography matrix M. Finally, the source frame and annotations are updated to the current destination frame and annotations.

4. EXPERIMENTS AND DATASETS

4.1 Dataset

Five word-level datasets are used for training and evaluating the performance of the text spotting model utilizing the cubic bezier curve in Bezier Align. Two synthetic datasets are generated using VGG Synthetic method [35]. The first one (S1) consists mostly of straight text (94723 images). The second one (S2) mostly consisting of curved text (54327 images). The three real-world datasets used are 7k ICDAR-MLT (MLT17) [36], TotalText (TT) [37] and LPCVC challenge dataset. The subset of the ICDAR-MLT 2017 dataset consists of English words is used which has 7000 images. The training set of the TotalText consisting of 1255 images. The LPCVC challenge dataset consists of images collected extracted from five videos. The videos are captured by UAVs flying indoors on the corridors capturing posters and board signs containing text. The text consists of oriented horizontal text where the orientation angle is very small. The text is not arbitrarily shaped. The text was annotated using the Semi-Supervised algorithm described in Section 3.2. Frames from four videos were used for training and one video was reserved for testing. This dataset is called the LPCVC challenge Dataset (SD). The LPCVC challenge dataset consists of 5755 train images and 1187 test images. The dataset consists of images of resolution 3840×2160, 1920×1080, and 1280×720. The LPCVC challenge Dataset has case-insensitive text labels i.e. all the text appearing in the image are stored as lowercase alphabets in the text label. To add the uppercase and lowercase information to the ground truth another version of the LPCVC challenge dataset is developed which consists of Case-Sensitive text labels. Two more datasets are used for training the Linear ABCNet i.e. ICDAR MLT 2019 (MLT19) [36] consisting of 10000 images and the Synthtext in the Wild dataset (ST) [35] consisting of 858750 images. S1, S2, TT, MLT17, SD have 16 coordinate bezier anno-

tations to denote the bounding box of the text. The MLT19 and ST dataset consists of 8 coordinate bezier annotations to denote the bounding box of the text.

4.2 Evaluation Metric

The evaluation metric used to measure the performance of the algorithm was the Edit Distance. To calculate the Edit Distance first the ground truth bounding box with the maximum IoU with the predicted bounding box is paired. The Edit Distance between the predicted word and the ground truth text for the paired bounding box is then calculated. The final Edit Distance is then obtained by dividing the total Edit Distance for the entire test dataset and the number of text instances in the ground truth test set. To get a better understanding of the different parameters such as blur level, bounding box area, bounding box count, bounding box area to character count ratio, and character count in a word on the model predictions the Edit Distance with respect to each of these parameters are calculated.

4.3 Energy Measurement Tool

To measure energy consumption MakerHawk UM34C USB 3.0 Multimeter Bluetooth USB Voltmeter Ammeter was used. The USB power meter was connected to the power source. The USB power meter was also connected to the power supply of the Raspberry Pi. In this setup, the Raspberry Pi drew power from the power source through the power meter which was measured by the power meter. The power meter were connected with the Desktop through Bluetooth and energy measurements of the Raspberry Pi measured by the power meter were read using [38] and stored locally. On raspberry pi, the timestamp for model inference was noted. The energy consumption of the deep learning algorithm was then calculated by measuring the energy consumption within the timestamp during which model inference was performed. Figure 4.1 shows the experimental results of the proof of concept experiments carried out to verify the correctness of the energy measurement tool.

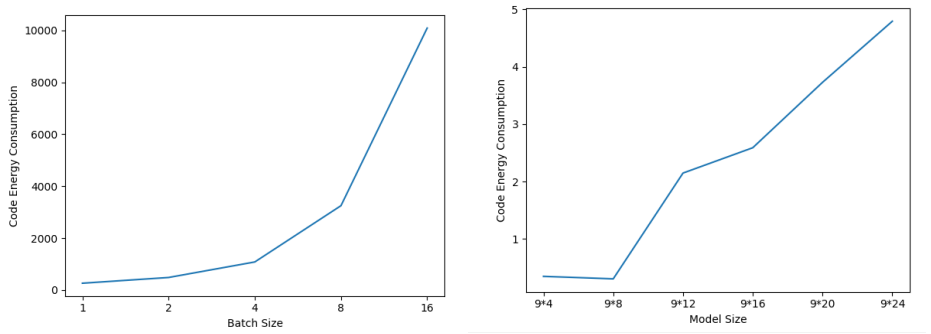

Figure 4.1: Proof of Concept Experiments for Energy Measurement tool

A dummy randomly initialized model consisting of nine layers of convolutions followed by a fully connected layer was used. The batch size indicates the number of images fed as input to the model during inference and was increased by a factor of two. The model size was varied by changing the number of filters in each layer by a constant factor. From the below figure it can be seen that as the batch size and the model size increases the energy consumption increases during inference. This indicates that the energy measurement works correctly.

4.4 ABCNet Experiments

This section provides results by experimenting with different components of the ABCNet model. Two different backbones are experimented i.e. Mobilenetv2 [39] and Resnet. Two different normalization schemes are experimented i.e. group normalization and batch normalization. Two different types of Bezier Align are experimented i.e. Cubic and Linear Bezier Align. The cubic Bezier Align consists of eight control points and the linear Bezier Align consists of four control points. For all the experiments there are two stages of training. The first one is pretraining which runs for 260000 iterations and starts at a learning rate of 0.01 which is reduced by a factor of 10 at 160000 and 220000 iterations.

The second stage of the training is called finetuning which runs for 40000 iterations with early stopping. It has a learning rate of 0.001. The two types of resolution used for training and testing are Type A and Type B of multi-scale setting. In both types, there are four parameters i.e. minimum and maximum dimension size for training and testing. In Type A the minimum size of both the dimensions for training is selected at random from the set (640, 672, 704, 736, 800, 832, 864, 896). The maximum size of both the dimensions is set to 1600. The minimum and maximum dimension limits for inference are 1000 and 1824. In Type B the minimum size of training is selected at random from the set (224, 232, 240, 248, 256, 264, 272, 280, 288). The maximum size of training for both dimensions is set to be 400. The minimum and maximum dimensions for inference are 250 and 456.

Algorithm 2 shows the algorithm used to generate cropped text regions from the image. The algorithm groups nearby texts which are less than 50 pixels apart and crops the text region from (xmin-50, ymin-50) to (xmax+50, ymax+50) where xmin, ymin, xmax, ymax, are obtained from the minimum and maximum coordinate of the text regions. Another parameter is upscaling the image. The inference image when not resized the height and the width is converted to the nearest multiple of 32. Table 4.1 shows the result of cropping text region in the image and upscaling the image during inference. Original means the input image is used as such for testing. Cropped means the test dataset created by using Algorithm 2 used to measure the performance of the algorithm. Resized means the image is resized during inference such that the minimum dimension of the image is greater than 1000 and the maximum dimension of the image is lesser than 1824. Repo ABCNet means the official GitHub repository [40] finetuned model was used. After resizing the image to within the interval both the dimensions of the image are converted such that both the dimensions are multiples of 32. The results show that by performing inference on the cropped image the model has better recognition results. From the results, it can be inferred that cropping the text region in the image increases the Signal-to-noise (SNR) ratio of the

image and hence increases the recognition results of the algorithm. This was the motivation to use a spatially aware cropping mechanism in the two-stage selector for the cropping text region in the image. Upscaling the input image also improves the recognition results.

Algorithm 2: Algorithm to generate Cropped Text Regions in Image

```
BBoxes ← GT BBoxes of Image
while BBoxes not empty do
    xmin, ymin, xmax, ymax ← BBoxes[0]
    xmin, ymin ← xmin − 50, ymin − 50
    xmax, ymax ← xmax + 50, ymax + 50
    while ith BBox intersects do
        Update(xmin, ymin, xmax, ymax)
        BBoxes.remove(i)
    Save image[xmin : xmax, ymin : ymax]
```

Table 4.1: Effect of Cropping Text Region and Upscaling Images

Model	Inference Image	Inference Res	IoU	IoP	IoG	EditDistance
Repo ABCNet	Original	Not Resized	81.96	92.44	86.79	0.24
Repo ABCNet	Original	Resized	81.85	91.47	87.48	0.26
Repo ABCNet	Cropped	Not Resized	81.76	94.31	85.60	0.21
Repo ABCNet	Cropped	Resized	83.34	94.69	87.48	0.13

Table 4.3 and 4.2 shows the experimental results for different combinations in the ABCNet model. Column P represents whether pretraining was done or not. It has two values Y meaning pretraining was done, Y* means GitHub provided a pre-trained model was used. The third column is the dataset used for pretraining. Column 4 indicates whether finetuning was done. Column 5 indicates the dataset on which finetuning was done. Column 6 indicates the normalization scheme used. The two types of normalization schemes

tested are batch normalization [41] and group normalization [42]. Column 7 indicates the backbone used. The two types of backbone used for experimentation are Resnet-50 and Mobilenet-v2. Column 8 indicates the batch size used for training. The batch size used was 4. Column 9 indicates the device used for inference. There are two options one is GPU and the other is CPU. Column 11 indicates the resolution used for training and testing during the pretraining and fine-tuning stage. Column 12, 13, 14 are IoU, IoP, and IoG which indicate the detection performance of the algorithm. Column 15 indicates the Edit Distance which measures the recognition performance of the algorithm. Column 16 indicates the Model Size in Megabyte. It is approximately 295MB when Resnet-50 is used as the backbone and 115 MB when Mobilenet-v2 is used as backbone. Column 17 indicates the average latency per prediction in seconds. From the table, it can be observed that by changing the backbone from Resnet to Mobilenet the recognition performance decreases but the model becomes more efficient as it has a smaller model size and lower latency for prediction. For the two normalization schemes tried it can be seen that the group normalization better performance with a lower edit distance than the batch normalization but with batch normalization the CPU latency significantly decreases. Another advantage of using batch normalization is that it is quantization friendly. The third observation is that the model has better recognition results when it is finetuned on the sample dataset with a smaller learning rate after it has been pretrained. The next observation is that the Cubic Bezier curve has better accuracy when compared with Linear Bezier Curve. This is expected because the cubic bezier curve uses has eight control points whereas the linear bezier curve has only four control points. But the expected latency trend is that the linear bezier curve model would have lower latency when compared with the cubic bezier curve. But the results show that the cubic bezier curve has lower latency than the linear bezier curve. The reason behind this could be that the original repository is built for cubic bezier curve and hence is optimized for the same. Only making the necessary changes to convert

the cubic bezier curve to the linear bezier curve would not work.

Table 4.2: Experimental Setup for different ABCNet Models

ID	P	Pretrain Dataset	F	FD	Norm	BB	BS	D	Bezier Curve	Res(PF)
1	Y*	MLT17, S1, S2, TT	N	N/A	GN	R50	4	G	Cubic	A-
2	Y*	MLT17, S1, S2, TT	Y	SD	GN	R50	4	G	Cubic	AA
3	Y	MLT, S1, S2, TT, SD	N	N/A	GN	R50	4	G	Cubic	A-
4	Y	MLT, S1, S2, TT, SD	Y	SD	GN	R50	4	G	Cubic	AA
5	Y	SD	N	N/A	GN	R50	4	G	Cubic	A-
6	Y	MLT, S1, S2, TT, SD	N	N/A	GN	Mv2	4	G	Cubic	A-
7A	Y	MLT, S1, S2, TT, SD	Y	SD	GN	Mv2	4	G	Cubic	AA
7B	Y	MLT, S1, S2, TT, SD	Y	SD	GN	Mv2	4	C	Cubic	AA
8	Y	MLT, S1, S2, TT, SD	N	N/A	GN	Mv2	4	G	Linear	A-
9	Y	MLT, S1, S2, TT, SD	Y	SD	GN	Mv2	4	G	Linear	AA
10	Y	MLT, S1, S2, TT, SD	Y	SD	GN	R50	4	G	Cubic	AB
11	Y	MLT, S1, S2, TT, SD	Y	SD	GN	Mv2	4	G	Cubic	AB
12	Y	MLT, S1, S2, TT, SD	Y	SD	GN	R50	4	C	Cubic	AB
13	Y	MLT, S1, S2, TT, SD	Y	SD	GN	Mv2	4	C	Cubic	AB
14	Y	MLT, S1, S2, TT, SD	Y	SD	GN	R50	4	C	Cubic	BB
15	Y	MLT, S1, S2, TT, SD	Y	SD	GN	R50	4	C	Cubic	BB
16	Y	MLT, S1, S2, TT, SD	F	N/A	GN	Mv2	4	C	Cubic	B-
17	Y	MLT, S1, S2, TT, SD	N	SD	GN	Mv2	4	C	Cubic	BB
18	Y	MLT, S1, S2, TT, SD	Y	SD	BND	Mv2	4	C	Cubic	AA
19	Y	MLT, S1, S2, TT, SD	Y	SD	BN	Mv2	4	C	Cubic	AA

Table 4.4 shows the effect of case-sensitive labels. Both the models are trained with Type A resolution. The models are first pretrained and then finetuned. During pretraining and finetuning the sample dataset images remain the same for both cases. The results in the first row indicate the result when all the ground-truth text are in lowercase. The second row indicates the results when the ground-truth text is case-sensitive. It can be seen from Table 4.4 that the model performs better when case-sensitive labels are used.

Table 4.5 shows the effect of finetuning the original repository pretrained and finetuned model with the case sensitive sample dataset. The repository pretrained model is trained

Table 4.3: Experimental Results for different ABCNet Models

ID	IoU	IoP	IoG	Edit Distance	Model Size (MB)	Latency (s)
1	72.24	94.20	74.58	0.42	295	
2	77.58	83.24	89.32	0.81	295	
3	77.31	81.62	89.62	0.70	295	
4	77.68	82.51	90.74	0.54	295	0.117
5	76.40	82.74	88.51	1.37	295	
6	68.00	78.00	83.00	1.98	115	
7A	75.75	79.12	92.34	0.64	115	0.096
7B	76.28	80.51	91.08	0.66	115	2.836
8	66.77	90.98	70.63	1.48	115	
9	74.78	78.48	92.05	1.44	115	0.102
10	66.03	74.35	83.91	3.33	295	0.107
11	64.5	72.33	85.26	4.28	115	0.094
12	65.28	73.63	84.03	3.37	295	1.040
13	64.53	72.33	85.27	4.28	115	0.863
14	67.30	73.80	86.92	3.41	295	1.000
15	68.36	74.79	86.39	2.76	295	0.838
16	61.48	68.19	86.53	4.28	115	0.704
17	65.72	71.96	86.49	3.39	115	0.964
18	74.11	79.70	86.93	0.87	115	1.313
19	74.03	80.30	86.09	1.03	115	1.330

on S1, S2, MLT2017, TT. The repository model is finetuned on the TT dataset. The first two rows indicate the results of not finetuning and finetuning the pretrained original repository model. The finetuning using a case-sensitive sample dataset is done for $5k$ iterations. The results show that the pre-trained model has poor performance regardless of whether it is later finetuned on the case-sensitive sample dataset. The last two rows show the results of not finetuning and finetuning using a case-sensitive sample dataset on the original repository finetuned model. From the results, it can be seen that the finetuned model when further finetuned on the case-sensitive sample dataset has better recognition

Table 4.4: Effect of Case-Sensitive Labelling

Sample Dataset Label	IoU	IoP	IoG	EditDistance
Case-Insensitive Lower	79.04	88.35	86.61	2.31
Case-Sensitive	78.87	86.86	88.52	2.15

results when compared with the finetuned model used as such for testing. This result is as expected.

Table 4.5: Effect of Finetuning with Case-sensitive Dataset with Repo Model

Model	Backbone	Pretrain	Finetune1	Finetune2	IoU	IoP	IoG	EditDistance
Cubic ABCNet	R50	Syn1,Syn2,MLT2017, TotalText			65.97	99.30	66.37	5.17
Cubic ABCNet	R50	Syn1,Syn2,MLT2017, TotalText	CSD(5k)		88.26	92.70	95.13	5.35
Cubic ABCNet	R50	Syn1,Syn2,MLT2017, TotalText	TT		85.34	93.96	90.81	0.26
Cubic ABCNet	R50	Syn1,Syn2,MLT2017, TotalText	TT	CSD(5k)	88.40	93.05	94.97	0.14

The feature pyramid network extracts features from the feature extractor at half (p2), one fourth (p3), one-eight (p4), one-sixteenth (p5), one-thirty-two (p6), and one-sixty-fourth (p7) the original resolution. For ABCNet, the default setting uses p3, p4, p5, p6, p7 for extracting features for detection. The features are then converted into fixed-size using RoIAlign. The recognizer uses the features at p2, p3, p4 for recognition. The extracted features are converted into fixed-size features using Bezier Align. Table 4.6 shows the experiment results of using different scales of Feature Pyramid Network for detection and recognition. The main motive behind the experiments is to find the minimum FPN scales extracted features required for detection and recognition such that the designed model is energy efficient. From the table, it can be seen that using only one scale is not enough for detection. A combination of p3 and p4 or p4 and p5 has moderate performance. The performance is better for p3, p4. Using a combination of p3,p4,p5 has close to the same

results as the default settings. This suggests that for the detector p6 and p7 features can be discarded for detection predictions to make a tradeoff for efficiency over accuracy. In the recognition branch, the default setting uses p2, p3, p4 for recognition prediction. From the results, it can be seen that p2 alone is enough to get good recognition results. This suggests that the recognizer requires a feature map of larger size with more information to make good recognition results and using one scale is sufficient. For the recognizer using only p2 is a good tradeoff for energy efficiency over accuracy.

Table 4.6: Effect of Different Scale of Feature Pyramid Network used for Detection and Recognition

Model	Detection	Recognition	IoU	IoP	IoG	EditDistance
Cubic ABCNet*	p3,p4,p5,p6,p7	p2,p3,p4	85.34	93.96	90.81	0.26
Cubic ABCNet	p3	p2,p3,p4	84.49	91.43	92.29	3.67
Cubic ABCNet	p4	p2,p3,p4	86.13	95.97	89.67	2.84
Cubic ABCNet	p5	p2,p3,p4	85.47	96.68	88.50	3.97
Cubic ABCNet	p3,p4	p2,p3,p4	85.17	93.53	91.01	1.48
Cubic ABCNet	p4,p5	p2,p3,p4	86.26	96.03	89.80	1.78
Cubic ABCNet	p3,p4,p5,p6,p7	p2,p3	85.34	93.96	90.81	0.26
Cubic ABCNet	p3,p4,p5	p2	85.37	93.88	90.93	0.43
Cubic ABCNet	p3,p4,p5	p3	85.37	93.88	90.93	1.37
Cubic ABCNet	p3,p4,p5	p4	85.37	93.88	90.93	5.67
Cubic ABCNet	p3,p4,p5	p2,p3	85.37	93.87	90.93	0.43
Cubic ABCNet	p3,p4,p5	p3,p4	85.37	93.88	90.93	1.37
Cubic ABCNet	p3,p4,p5	p2,p3,p4	85.37	93.86	90.93	0.43

Table 4.7 shows the effect of different cropped images used for training. The model used is the ABCNet model with the Bezier Align consisting of Cubic Bezier Curve with eight points. The backbone used is Mobilenetv2. The normalization scheme used is batch normalization. The column 2 value C denotes the cropped image finetune dataset obtained by Algorithm 3. The algorithm uses images resized 960×540. Then the center point of the bounding box is calculated and the x and y coordinate of the center of the bounding box

is shifted based on cropXChange and cropYChange. Next, the cropped image is resized to all the values in the anchor list [640, 672, 704, 736, 768, 800, 832, 864, 896]. A total of 45 different combinations for the same text region is created. Column 3 Default means Type A setting is for training. No means the input resolution is resized to the nearest multiple of 32. Column 4 indicates the dataset used for inference. It could be O which denotes the original test dataset or C meaning the cropped image dataset generated by Algorithm 3. Column 5 indicates the number of iterations for which finetuning is done. The goal behind the experiments is to attain good model performance on cropped images. Row 2 indicates the model directly tested on cropped images. Row 3 indicates the model fine-tuned on cropped images dataset with default multi-scale training strategy and tested on cropped images. Row 4 indicates the model fine-tuned on the cropped dataset without any multi-scale training strategy and tested on cropped images. Row 4 performs best as expected. The model is trained and tested on the same resolution. Row 3 involves using the same dataset but trained on a resolution that is two times of test image resolution. This model performs worse as it misses detecting many texts. Row 2 has slightly poor performance.

Algorithm 3: Algorithm to generate Cropped Text Regions in Image

1 cropXChange = [0.25, 0.5, 0.5, 0.5, 0.75]; cropYChange = [0.5, 0.25, 0.5, 0.75, 0.5]
2 anchorList = [640, 672, 704, 736, 768, 800, 832, 864, 896]
3 Downsample image to 960x540
4 For all the annotations of an image - xmin, ymin, xmax, ymax is calculated. This is used to calculate the center point x,y of the rectangle with start point (xmin, ymin) and end point (xmax, ymax).
5 45 different combinations from anchorlist, cropXChange, cropYChange are checked for validity. If it is valid the image is cropped.

Table 4.7: Effect of Different Cropped Images used for training

Model	Finetune	MS Strategy	Test	Inf Res	Iter	IoU	IoP	IoG	EditDistance
Cubic ABCNet+Mv2+BN		Default	O	Orig	5k	86.65	94.06	92.01	0.61
Cubic ABCNet+Mv2+BN		Default	C	N/A	5k	84.76	87.60	96.98	1.59
Cubic ABCNet+Mv2+BN	C	Default	C	N/A	20k	81.00	84.09	96.45	4.57
Cubic ABCNet+Mv2+BN	C	No	C	N/A	25k	85.07	93.39	91.09	1.07
Cubic ABCNet+Mv2+BN		Default	O	N/A	5k	86.67	94.67	91.45	0.55
Cubic ABCNet+Mv2+BN	C	No	O	N/A	25k	82.03	97.01	84.30	0.54

4.5 FOT++ Experiments

Two different types of FOT++ methods were experimented. The first one is where the recognizer makes predictions by extracting predicted bounding box regions from the text detector on the input image. This is called the two-step text spotting method. This is because first detection is done, then recognition is done on the input image. The second one is where the recognizer makes predictions by extracting the predicted bounding box from the text detector on the detector feature map. This is called the two-stage text spotting method because it is analogous to the Faster R-CNN object detector. To measure the performance of the algorithms Edit Distance is used. First, the FOT++ experiment is carried out between the two-step and the two-stage text spotting method. In both methods, the EAST text detector is first trained. To train the EAST detector the Case-Sensitive LPCVC Challenge Dataset and the Synthetic dataset are used. For training the recognizer of the two-step text spotting method the ground truth bounding box is used. The ground truth bounding box is fed to the BezierAlign to extract fixed size features from the input image. This is then fed to the recognizer which consists of CRNN and Attention to make word-level predictions for each ground truth bounding box. The model is trained using the CTC loss. In the two-stage text spotting method, the only difference is that the Bezier Align extracts fixed size features from the feature map using the ground truth bounding box. During inference, the two-step text spotting method uses the EAST bounding box

predictions which are then used to extract fixed size features from the input image with the help of BezierAlign. The extracted features are fed to the recognizer to predict the word label associated with the predicted bounding box. For two-stage text spotting inference similar to training the only difference is that EAST bounding box predictions are extracted from the feature map using the Bezier Align. Table 4.8 and Table 4.9 are the prediction results of the two-step text spotting and two-stage text spotting method. Both the methods were trained on Syntext1, Syntext2, MLT7K, TotalText, and Case Sensitive Sample Dataset. The Case Sensitive LPCVC Challenge Dataset test set was used to measure the performance of the models. Table 4.8 and Table 4.9 indicates that for any given input resolution of the image the two-step text spotting method performs better than the two-stage text spotting method for our use case. This result is significant because the two-stage text spotting method also called the Aligned RoIPool method which is the method followed in the literature for text spotting model design performs inferior when compared with the two-step text spotting method. From Table 4.8 and Table 4.9 it can be seen that the model performs better for larger bounding box area to char count ratio and lower character count. The bounding box area to char count ratio is the area of one character under the assumption that all the characters occupy an equal area in the bounding box. The bounding box to character count ratio is the area of one character in the image. The results indicate that the larger the area of one character in the image the better is the recognition result i.e. lesser the Edit distance. The second parameter is the character count. The results indicate that the lesser the number of characters in the word, the better is the recognition results. The third metric is the Blur level. It comes as a surprise that the recognition result gets better as the blur level in the image increases. To measure the blur level the metric calculates the blur for the entire image. This might not be indicative of the blur level in the local text regions. The number of the bounding box in the image does not show any trend in the results. This indicates that the recognition results are not dependent on the number

of bounding boxes in the input image. The bounding box area parameter shows an unexpected trend i.e. as the bounding box area increases the recognition result becomes poorer. This is not expected. This suggests that the bounding box area is not a good metric to understand the recognition result performance. This is because the text bounding box area generally has a fixed height with varying width. This might result in a higher bounding box area for a smaller height of the bounding box with a large width due to text containing many characters. So, the bounding box area might not be a good metric to understand the recognition results.

Table 4.8: EAST + CRNN (Image) results on Case-Sensitive Sample Dataset Test Set

Res	BBox Area			BBox Count			Blur Level			BBox Area/Char Count			Char Count			Total
	<=1024	<=9216	<=9216	<=5	<=10	>10	L	M	H	<=20	<=60	>60	<=4	<=8	>8	
2240	0.68	1.00	3.28	1.36	2.38	1.36	2.46	1.84	0.79	Inf	Inf	1.83	0.87	1.97	4.00	1.83
1200	1.14	1.43	4.61	1.46	2.49	1.42	2.62	1.91	0.93	Inf	5.25	1.93	1.08	1.86	4.2	1.93
600	1.80	2.14	7.38	1.92	2.58	1.24	3.02	1.92	1.05	6.86	2.12	2.00	1.31	2.08	3.79	2.04
300	3.13	4.20	Inf	3.52	3.64	2.39	4.09	3.07	2.94	3.69	3.60	2.92	2.21	3.56	5.32	3.26

Table 4.9: EAST + BezierAlign + CRNN (Feature Map) results on Case-Sensitive Sample Dataset Test Set

Res	BBox Area			BBox Count			Blur Level			BBox Area/Char Count			Char Count			Total
	<=1024	<=9216	<=9216	<=5	<=10	>10	L	M	H	<=20	<=60	>60	<=4	<=8	>8	
2240	1.22	1.36	3.15	1.58	2.50	1.55	2.31	2.09	0.98	Inf	Inf	2.00	1.07	2.26	3.86	2.00
1200	1.72	2.50	3.72	2.42	2.86	2.40	2.75	2.73	1.79	Inf	4.56	2.62	1.83	2.98	3.95	2.62
600	3.45	3.38	7	3.39	3.66	3.28	3.90	3.43	3.10	6.84	2.84	3.48	2.38	4.21	4.92	3.48
300	5.15	6.67	Inf	5.95	5.36	4.49	5.64	5.24	4.70	4.19	5.19	5.97	3.28	6.12	8.61	5.25

In both the two-stage text spotting and two-step text spotting method described earlier the EAST detector predicted bounding box is used by the recognizer. The main question is to understand the influence of the recognizer results on the feature map and on the image. So, to remove the influence of the text detector experiments were conducted by using the ground truth bounding boxes. The ground truth bounding box region extracted from the image is the CRNN+Attn (Image) results shown in Table 4.10. The ground truth bounding box region extracted from the feature map is the CRNN+Attn (Feature Map)

results shown in Table 4.11. Even after removing the effect of the detector, it can be seen that the recognizer performs better when the input features are extracted from the input image. Here again, the recognition results are better as the character count in the word decreases and when the ratio of bounding box area to character count increases.

Table 4.10: CRNN+Attn (Image) results on Case-Sensitive Sample Dataset Test Set

Res	BBox Area			BBox Count			Blur Level			BBox Area/Char Count			Char Count			Total
	<=1024	<=9216	<=9216	<=5	<=10	>10	L	M	H	<=20	<=60	>60	<=4	<=8	>8	
2240	0.43	0.36	1.05	0.44	0.89	0.33	0.97	0.57	0.37	Inf	Inf	0.62	0.23	0.45	1.94	0.62
1200	0.55	0.45	1.30	0.47	0.85	0.32	0.93	0.56	0.35	Inf	2.69	0.60	0.22	0.46	1.83	0.60
600	0.47	0.92	3.43	0.62	1.00	0.30	1.08	0.66	0.31	5.76	0.49	0.68	0.28	0.54	2.09	0.70
300	1.69	1.83	Inf	1.77	1.84	1.40	2.06	1.64	1.37	2.97	1.64	1.20	1.12	1.72	3.09	1.69

Table 4.11: CRNN+Attn (Feature Map) results on Case-Sensitive Sample Dataset Test Set

Res	BBox Area			BBox Count			Blur Level			BBox Area/Char Count			Char Count			Total
	<=1024	<=9216	<=9216	<=5	<=10	>10	L	M	H	<=20	<=60	>60	<=4	<=8	>8	
2240	0.98	0.45	0.51	0.42	0.65	0.31	0.58	0.51	0.30	Inf	Inf	0.49	0.35	0.45	0.94	0.49
1200	1.48	1.56	1.10	1.27	1.73	1.22	1.57	1.53	1.00	Inf	4.44	1.47	1.17	1.51	2.16	1.47
600	3.16	2.17	2.05	2.51	2.86	2.75	3.00	2.65	2.84	5.82	2.36	2.73	2.02	3.18	3.74	2.74
300	4.73	5.88	Inf	5.38	4.88	4.21	5.05	4.79	4.50	3.99	4.79	5.28	3.26	5.48	7.46	4.80

4.6 Model Deployment Results

Table 4.12 shows the model deployment results of two-step FOT++ text spotting model on the Raspberry Pi. The model used for deployment was work done by another person in the team. My work was focused on creating an energy measurement tool to measure the energy consumption of the algorithm. The FOT++ model deployed is slightly different from the FOT++ explained earlier. The recognizer only contains the CRNN and the attention is removed. The Bezier Align is not used. Instead, Affine Transformation is used. The model is pruned and quantized to improve energy efficiency and reduce latency.

4.7 FLOPs Calculation and Speed Test

Table 4.13 shows the FLOPS calculation of the ABCNet model for different input resolution. It indicates the theoretical amount of multiply-add operations required during

Table 4.12: Performance, Latency, and Energy Measurement of two-step FOT++ model on Raspberry Pi

Model	FOTS++
IoU	72.21
IoP	76.24
IoG	93.94
Edit Distance	1.39
Latency (s)	12.90
Avg Energy (J)	31.77

Table 4.13: FLOPS calculation of ABCNet for different resolution

Model	Resolution	Ops	Params
ABCNet	1280*720	137.33G	28.21M
ABCNet	1920*1080	304.06G	28.21M

model inference. The fourth column in the table indicates the number of trainable parameters. From the table, it can be seen that as the input resolution increases the flops count increases. Table 4.14 shows the speed test of the ABCNet and the FOTS text spotting model. From the table, it can be inferred that the lower inference time of the FOTS model is due to the lightweight EAST detector.

Table 4.14: Speed Test (in ms) for different components of FOTS and ABCNet

	ABCNet	FOTS
Detector	83	7
Bezier Align	1.1	2.4
Recognizer	18	8
Post Processing	0.9	
Total Time	100	70

5. CONCLUSION

In the thesis, experiments are conducted for efficient design of the ABCNet and FOT++ text spotting model. Different backbones, normalization schemes, FPN features for detection and recognition are experimented with for ABCNet. Two different backbones were experimented i.e. Resnet-50 and Mobilenetv2. From the results, it can be seen that the Resnet-50 model has better performance but the Mobilenetv2 model has less than half model size and lower inference time with a minor performance hit. The same is observed while using batch normalization when compared with group normalization. Group Normalization has better performance than batch normalization but batch normalization has half the inference time on CPU for a small performance hit. Also, batch normalization is quantization-friendly. The default ABCNet model uses p3,p4,p5,p6,p7 for detection and p2,p3,p4 for recognition. From the experiments it can be concluded that using p3, p4, p5 for detection, and p2 for recognition has the best tradeoff for efficiency over accuracy. Another result is that using Case-sensitive labeling improves the recognition performance. Two different methods i.e. two-step Crop+Resize and two-stage text spotting methods are experimented with for the FOT++ model. From the experimental results, it can be seen that the two-step text spotting method performs better than the two-stage text spotting method for all input resolutions. Further to better understand the influence of different parameters on the recognition results the effect of bounding box count, bounding box area, character count, bounding box to character count ratio, and the blur level is measured. From the experiments, it can be concluded that the greater bounding box to character count ratio better is the recognition results. This means that the greater the area of a character better is the recognition results. Another conclusion that can be made is that the lesser the character count in the text better is the recognition results. One surprising result is that as

the blur level increases the models' recognition results get better. This is contradictory to what is expected. The reason for this could be because of the metric used for measuring the blur level of the image. There could be cases where the text region is blurry but the overall image is clear so the blur metric might indicate that the image is clear. A better blur quantification metric could be used as well. The energy measurement of the FOT++ two-step text spotting model deployment results on Raspberry Pi is done using an energy measurement tool. Proof of concept for the energy measurement tool is also shown. Finally, the flops count of ABCNet at different input resolutions and the speed test of FOTS and ABCNet are also shown.

REFERENCES

[1] Z. Cheng, J. Lu, B. Zou, L. Qiao, Y. Xu, S. Pu, Y. Niu, F. Wu, and S. Zhou, "FREE: A Fast and Robust End-to-End Video Text Spotter," *IEEE Transactions on Image Processing*, vol. 30, pp. 822–837, 2021.

[2] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An Efficient and Accurate Scene Text Detector," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2651, 2017.

[3] 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," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[4] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Real-Time Scene Text Spotting With Adaptive Bezier-Curve Network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9806–9815, 2020.

[5] S. Long, X. He, and C. Yao, "Scene Text Detection and Recognition: The Deep Learning Era," *International Journal of Computer Vision*, vol. 129, pp. 161–184, 2020.

[6] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading Text in the Wild with Convolutional Neural Networks," vol. 116, p. 1–20, Jan. 2016.

[7] C. L. Zitnick and P. Dollár, "Edge Boxes: Locating Object Proposals from Edges," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 391–405, Springer International Publishing, 2014.

[8] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A Fast Text Detector with a Single Deep Neural Network," in *AAAI*, 2017.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.

[10] C. Bartz, H. Yang, and C. Meinel, "SEE: Towards Semi-Supervised End-to-End Scene Text Recognition," in *AAAI*, 2018.

[11] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast Oriented Text Spotting with a Unified Network," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5676–5685, 2018.

[12] M. Bušta, L. Neumann, and J. Matas, "Deep TextSpotter: An End-to-End Trainable Scene Text Localization and Recognition Framework," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2231, 2017.

[13] T. He, Z. Tian, W. Huang, C. Shen, Y. Qiao, and C. Sun, "An End-to-End TextSpotter with Explicit Alignment and Attention," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5020–5029, 2018.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), p. 369–376, Association for Computing Machinery, 2006.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[17] K.-H. Kim, Y. Cheon, S. Hong, B.-S. Roh, and M. Park, "PVANET: Deep but Lightweight Neural Networks for Real-time Object Detection," *ArXiv*, vol. abs/1608.08021, 2016.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[19] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai, "Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 532–548, 2021.

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis*

and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[24] S. Qin, A. Bissaco, M. Raptis, Y. Fujii, and Y. Xiao, "Towards Unconstrained End-to-End Text Spotting," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4703–4713, 2019.

[25] L. Xing, Z. Tian, W. Huang, and M. Scott, "Convolutional Character Networks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9125–9135, 2019.

[26] P. X. Nguyen, K. Wang, and S. Belongie, "Video text detection and recognition: Dataset and benchmark," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 776–783, 2014.

[27] C. Merino-Gracia, "Real-time text tracking in natural scenes," *IET Computer Vision*, vol. 8, pp. 670–681(11), December 2014.

[28] A. Kay, "Tesseract: An Open-Source Optical Character Recognition Engine," *Linux J.*, vol. 2007, p. 2, July 2007.

[29] X. Wang, Y. Jiang, S. Yang, X. Zhu, W. Li, P. Fu, H. Wang, and Z. Luo, "End-to-End Scene Text Recognition in Videos Based on Multi Frame Tracking," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 1255–1260, 2017.

[30] L. Qiao, S. Tang, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "Text Perceptron: Towards End-to-End Arbitrary-Shaped Text Spotting," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 11899–11907, AAAI Press, 2020.

[31] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully Convolutional One-Stage Object Detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9626–9635, 2019.

[32] "Low-Power Computer Vision Challenge 2020 CVPR Workshop, UAV video," https://lpcv.ai/2020CVPR/video-track.

[33] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," vol. 60, p. 91–110, Nov. 2004.

[34] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, p. 381–395, June 1981.

[35] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic Data for Text Localisation in Natural Images," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2315–2324, 2016.

[36] N. Nayef, Y. Patel, M. Busta, P. N. Chowdhury, D. Karatzas, W. Khlif, J. Matas, U. Pal, J.-C. Burie, C.-l. Liu, and J.-M. Ogier, "ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition — RRC-MLT-2019," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1582–1587, 2019.

[37] C. K. Ch'ng and C. S. Chan, "Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 935–942, 2017.

[38] "Web GUI for RuiDeng USB testers (UM34C, UM24C, UM25C, TC66C)," https://github.com/kolinger/rd-usb.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[40] "ABCNet Official GitHub Repository," https://github.com/aim-uofa/AdelaiDet.

[41] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[42] Y. Wu and K. He, "Group Normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.