End-To-End Text Detection Using Deep Learning

Ahmed S. Ibrahim

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Lynn Abbott, Chair
Jia-Bin Huang
Daniel Stilwell
Bert Huang
Mohamed Hussein

November 27, 2017
Blacksburg, Virginia

Keywords: Text Detection, Deep Learning

End-To-End Text Detection Using Deep Learning

Ahmed S. Ibrahim

(ABSTRACT)

Text detection in the wild is the problem of locating text in images of everyday scenes. It is a challenging problem due to the complexity of everyday scenes. This problem possesses a great importance for many trending applications, such as self-driving cars. Previous research in text detection has been dominated by multi-stage sequential approaches which suffer from many limitations including error propagation from one stage to the next. Another line of work is the use of deep learning techniques. Some of the deep methods used for text detection are box detection models and fully convolutional models. Box detection models suffer from the nature of the annotations, which may be too coarse to provide detailed supervision. Fully convolutional models learn to generate pixel-wise maps that represent the location of text instances in the input image. These models suffer from the inability to create accurate word level annotations without heavy post processing. To overcome these aforementioned problems we propose a novel end-to-end system based on a mix of novel deep learning techniques. The proposed system consists of an attention model, based on a new deep architecture proposed in this dissertation, followed by a deep network based on Faster-RCNN. The attention model produces a high-resolution map that indicates likely locations of text instances. A novel aspect of the system is an early fusion step that merges the attention map directly with the input image prior to word-box prediction. This approach suppresses but does not eliminate contextual information from consideration. Progressively larger models were trained in 3 separate phases. The resulting system has demonstrated an ability to detect text under difficult conditions related to illumination, resolution, and legibility. The system has exceeded the state of the art on the ICDAR 2013 and COCO-Text benchmarks with F-measure values of 0.875 and 0.533, respectively.

End-To-End Text Detection Using Deep Learning

Ahmed S. Ibrahim

(GENERAL AUDIENCE ABSTRACT)

Text detection and recognition in the wild is the problem of locating and reading text in images of everyday scenes. Text detection refers to finding the bounding boxes that describe the location of text areas in an input image, while text recognition describes the problem of generating a transcript out of the detected text areas. Recognition can be viewed as simply Optical Character Recognition (OCR). OCR is an old problem where the developed models are considered mature. Text detection and recognition are challenging problems due to the complexity of everyday scenes, compared to the simpler problem of recognizing text in scanned documents. This problem possesses a great importance to many trending applications that need to locate and read text in the wild, such as self-driving cars. Researchers tend to focus on the text detection problem only due to the maturity of research related to text recognition. Previous research in text detection has been dominated by multi-stage sequential approaches. Those methods suffer from many limitations including, but not limited to, error propagation from the earlier stages to the later stages of the pipeline. Another line of work is the use of deep learning techniques. Deep learning is the state of the art in machine learning. It has demonstrated great success in many domains, including computer vision. Some of the deep methods used for text detection are box detection models and fully convolutional models. Box detection models learn to generate bounding box coordinates for text instances that exist in the input image. Box detection models suffer from the nature of the annotations, which may be too coarse to provide detailed supervision. Fully convolutional models learn to generate pixel-wise maps that represent the location of text instances in the input image. These models suffer from the inability to create accurate word level annotations without heavy post processing. To overcome these aforementioned problems we propose a novel end-to-end system based on a mix of novel deep learning techniques. The proposed system consists of an attention model followed by a network based on Faster-RCNN that has been conditioned to generate word-box predictions. The attention model produces a high-resolution map that indicates likely locations of text instances. A novel aspect of

the system is an early fusion step that merges the attention map directly with the input image prior to word-box prediction. This approach suppresses but does not eliminate contextual information from consideration, and avoids the common problem of discarding small text regions. To facilitate training of the end-to-end system, progressively larger models were trained in 3 separate phases. The resulting system has demonstrated an ability to detect text under difficult conditions related to illumination, resolution, and legibility. The system has exceeded the state of the art on the well-known ICDAR 2013 and COCO-Text benchmarks. For the former case, the system has produced results with an F-measure value of 0.875. For the more challenging COCO-Text dataset, the system has shown a dramatic increase in performance with an F-measure value to 0.533, as compared to previously reported values in the range of 0.33 to 0.37. In order to build a powerful system, we introduced a novel deep learning architecture that achieved impressive performance on standard benchmarks. This architecture has been used as a backbone for the proposed attention model. A description of the proposed end-to-end system, as well as the implementation steps, will be detailed in the following sections.

Dedication

Dedicated to my lovely wife Ola.

Acknowledgments

I do not have enough words to thank Dr. Lynn Abbott for his continuous help and support. I would not be able to complete this work without his help, guidance, and creative ideas. I would also like to thank Dr. Mohamed Hussein for all his time and efforts. The lengthy fruitful discussions with Dr. Mohamed enabled me to do a better research. I would like to thank my committee members, Dr. Jia-Bin Huang, Dr. Daniel Stilwell, and Dr. Bert Huang. They are helpful and supportive. I am honored to have such elite professors serving as my committee members. Also, I like to thank all CESCA members. I thank all of my lab members at CESCA Computer Vision lab, especially Abhijit Sarkar for all the deep learning sessions he organized.

I would like to acknowledge the support I obtained from various entities that enabled me to do my research. I would like to thank Benha University and VTMENA for the financial support. Special thanks to Dr. Sedki Riad for creating this program which brought me to the USA. Also, I would like to thank the Electrical and Computer Engineering department at Virgina Tech for accepting me as a Ph.D. student and providing all the possible help and support. I would like to thank the CESCA lab for giving me the lab space and resources. Special thanks to Virginia Tech Advanced Research Computing (ARC) for providing me access to their powerful clusters that enabled me to run my experiments. ARC also provided me with funding through an assistantship in the late stage of my work on this dissertation. I would also like to thank the Virginia Tech Office of Institutional Research for providing me with three years of funding through assistantship. Also Dr. Peggy Layne, Assistant Provost for Faculty Development, for giving me the opportunity to work on the Elements system and providing me with funding for the summers.

I am blessed to have a lot of loving kind people around me. Without their support, I would not be able to do any of this work. Ola, my dear wife, gave me unconditional love and support. Her love and support are the major reasons I could complete my dissertation. My sweet kids, Salma, Omar, and Aisha, helped me to overcome hard times with their smiles. My parents raised me to be the person I am right now. My brothers did their best to help and support me. My friends in Egypt and USA were always there whenever I needed them. I would like to thank the Virginia Tech Office of Institutional Research (IR) for being my family in the USA. The people in IR are loving and kind to the point I could find a family in them. I could see parents, brothers, and sisters to me in the office of IR. Without the IR family, it would be hard for me to complete this dissertation. Thank you Dennis, Daniel, Raifu, Jacob, Roxanne, Janice, Steve, Becca, Sharon, Erin, Radha, Talia, Kris.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Text Detection

The ability to detect and recognize text in images of everyday scenes will become increasingly important for such applications as robotics and assisted driving. Most previous work involving text has focused on problems related to document analysis, or on optical character recognition (OCR) for text that has already been localized within images (e.g., reading automobile license plates). Unfortunately, text that appears in an unstructured setting can be difficult to locate and process automatically. A high-level diagram of a standard text detection system is shown in figure 1.1.

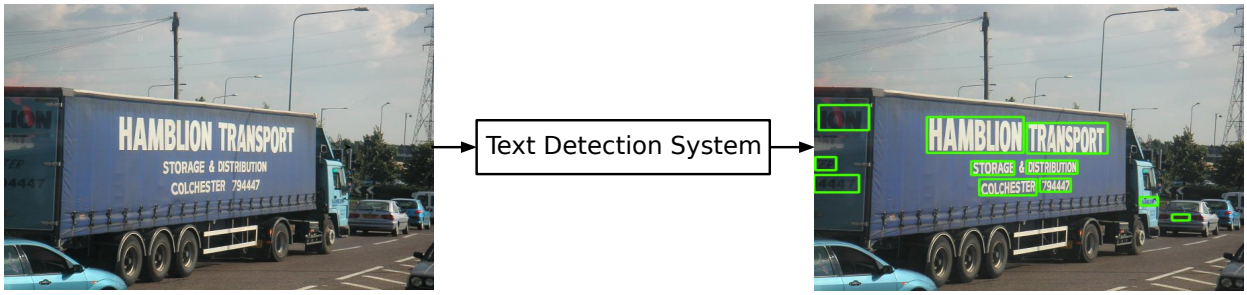


Figure 1.1: A high-level diagram of a standard text detection system. The system takes a standard RGB image and generates bounding boxes that represent the location of the text inside the image.

In spite of recent significant advances in the automated analysis of text, everyday scenes continue to pose many challenges [76]. For example, consider an image sequence taken

from an automobile as it moves along a city street. Text may be visible in such places as traffic signs, billboards, storefronts, and pedestrians' clothing to provide place names, advertisements, and traffic signs. Text will be visible in a rich diversity of sizes, colors, fonts, and orientations. Furthermore, single lines of text may vary in scale due to perspective foreshortening. Unlike most documents, the background may be very complex. Finally, other interference factors such as noise, motion blur, defocus blur, low resolution, nonuniform illumination, and partial occlusion may complicate the analysis.

1.2 Text Detection Pipeline

Text detection is, traditionally, divided into a pipeline of 3 stages as shown in Figure 1.2. The first stage, text localization, generates region proposals, which are typically rectangular sub-images that are likely to contain text. It is expected, however, that this stage will generate a relatively high number of false positives [68]. The next stage, text verification, analyzes each region proposal further in an attempt to remove false positives. The third stage, word/character segmentation, attempts to locate individual words or characters within the surviving region proposals. The output from the text detection pipeline should be sent after that to a recognition module. Text recognition is where OCR-type techniques are applied in an effort to recognize the extracted words and characters from the detection pipeline.

Figure 1.2: The traditional text detection pipeline. The 1st stage identifies candidate text locations in an image, and the 2nd stage removes false positives. The 3rd stage attempts to extract individual words or characters.

The nature of the text detection pipeline leads to accumulation of errors from one stage to

the next. For example, a region skipped during region proposal generation will result in some words that never reach the word segmentation stage. An end-to-end detection system would overcome this problem by generating word bounding boxes from the input image directly. Another limitation of the current traditional pipeline is the lack of any mechanism for using context cues. The early stages try to detect text area and remove all non-text areas from the input image. But those non-text areas include important contextual information that may help in recognizing hard to detect text instances such as occluded text.

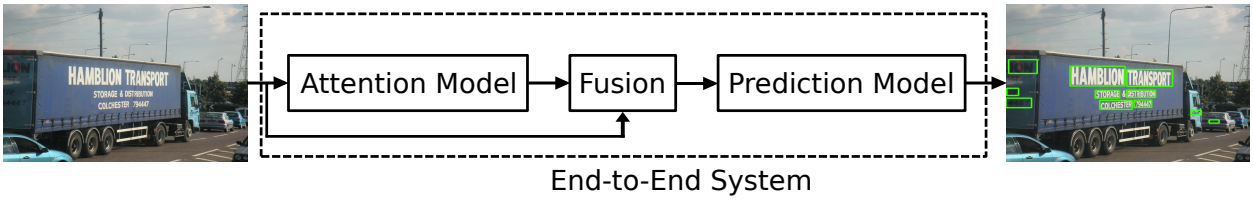


Figure 1.3: A high level diagram of the proposed end-to-end text detection system.

Figure 1.3 contains a sample image from ICDAR 2013 [28], one of the standard text detection benchmarks, along with the bounding boxes generated by the proposed end-to-end system. This image contains some very hard to detect text instances such as the car license plate. It would be the desired behavior of text detection system to use context information in the image to recognize such difficult text instances. The traditional pipeline would propose and crop areas in the its first stage removing important context information. The low-resolution illegible truck license plate marked with the green box in figure 1.1 will not pass the text verification stage. Looking at the image as a whole indicates that this area contains important textual information such as the location of this area with the respect to the truck indicating that it's a license plate. This kind of context information gets discarded in the first stage of the current traditional pipeline. Another line of work is the use of box detection models, such as Faster-RCNN [50], and fully convolutional models, such as the Fully Convolutional Networks for semantic segmentation [41]. Box detection models suffer from the bounding-box annotations, which may be too coarse to provide detailed supervision. Fully convolutional

models suffer from the inability to create accurate word level annotations with a single model.

A high-level diagram of the proposed end-to-end text detection system is shown in figure 1.3. The proposed end-to-end system demonstrated superior performance by detecting extremely hard to detection text instances due to conditions related to lighting, occlusion and legibility. These instances, as will be shown later, would be hard to detect for models that implement a traditional pipeline. Another set of hard to detect text instances are instances of low resolution or illegible text such as the license plate shown in figure 1.1. The implemented system could detect such instances while state-of-the-art models failed to detect them as will be shown later.

1.3 Evaluation Datasets

Many datasets have been created to help with the evaluation of text detection and recognition algorithms. A comprehensive discussion of those datasets can be found in section 2.1.1. For a long time, datasets contained images with mostly legible text instances located near the center of the image. Recently, some of the datasets started to include hard to detect text instances such as small size and occluded text. Those hard to detect text instances introduce a better resemblance of the challenges of text detection and recognition in everyday scenes. COCO-Text, one of the recent datasets that will be discussed in section 2.1.1, introduced some instances with illegible text such as the one in figure 1.4. Although the image is of acceptable resolution, the text instance is not legible. Such text instances that would exist in realistic situations may only be recognized by having cues from the context around the text area.

The introduction of an end-to-end system with the ability to learn context information should yield a better performance of text detection. The proposed end-to-end deep network

Figure 1.4: A sample from COCO-Text. The highlighted text instance can only be detected if the system can look at the whole image and collect context information such as the location of this area with respect to the truck.

is inspired by the recent advances in object detection models. Most of the state of the art deep learning based object detection models would take an image as an input and generate bounding boxes that represent the locations of the various objects inside this image. The proposed system would take an image as an input and generate bounding boxes that represent the collection of all the text instances that occur in this image. Because text is hard to detect, we introduced a novel architecture that fuses text attentional information into the input images to enable better end-to-end text detection.

1.4 Deep Learning

Deep learning is a machine-learning technique that has become increasingly popular in computer vision research. The main difference between classical machine learning (ML) and deep learning is the way that features are extracted. For classical ML techniques such as support vector machines (SVM) [8], feature extraction is performed in advance using techniques crafted by the researchers. Then the training procedure develops weights or rules that map any given feature vector to an output class label. In contrast, the usual deep-learning procedure is to apply signal values as inputs directly to the ML network, without any preliminary efforts at feature extraction. The network takes the input signal (pixel values, in our case), and assigns a class label based on those signal values directly. Because the deep-learning approach must implicitly derive its own features, it requires many more training samples than traditional ML systems.

Several deep-learning packages are available for researchers. The package that we have used to implement various parts of the proposed system is TensorFlow [3], which became very popular recently. TensorFlow is flexible and powerful machine learning package created by Google.

Another package we used to build parts of the system is Keras [7] which is a widely used high-level deep learning package that runs on top of multiple lower-level packages including TensorFlow. Keras is easy to use, flexible and can make use of the powerful features offered by TensorFlow.

1.5 Contributions and Outline

In this dissertation, we have developed new deep learning techniques and architectures that have been applied to the problem of text detection in natural scene images. The contributions included

- Creating a large publicly available dataset for training and evaluating text classifiers. This dataset has been used to build a powerful text classifier. This classifier enabled building an essential part of the proposed system in an efficient way.

- Introducing two novel deep learning architectures, input fast-forwarding (FFNet) and nested auxiliary branches (AuxNet). FFNet, trained using the massive dataset we created, was the backbone of a powerful, lightweight model that boosted the performance of the proposed end-to-end text detection system. We also spawned this lightweight architecture into a more advanced one, AuxNet, which has addressed the well-known and persistent problem of vanishing gradients in ultra-deep models. We built a 4096-layer network, the deepest network to the best of our knowledge, based on this novel architecture.

- Creating a novel text attention model that generates high-resolution heat maps indicating the possible locations of text instances in the input image. This novel attention model has been integrated into the proposed end-to-end text detection system in order

to boost its performance.

- Introducing a novel early fusion technique that modulates the input images with heat maps created by the text attention model. This novel early fusion enabled the detection of very hard to detect text instances.

- Building and training the end-to-end system using a novel multi-phase learning transfer. This end-to-end system introduced a new state of the art performance on ICDAR 2013, one of the widely used text detection benchmarks. A dramatic increase over to the state of the art on COCO-Text, the most challenging text detection benchmark, has also been achieved by the proposed end-to-end system.

The next chapter of this document will give a brief background about topics related to the proposed system including text detection evaluation datasets in section 2.1.1. Then, a discussion of the state-of-the-art text detection models in section 2.1.3. A brief background about deep learning techniques related to the proposed model will be given in 2.2. The research and exploration studies that led to building the proposed end-to-end system will be discussed in the following chapters including the introduction of the multi-phase learning transfer in chapter 3, the introduction of the text attention model in chapter 4, the creation of COCO-Text-Patch in chapter 5, the discussion of FFNet and AuxNet, the the two novel architectural designs, in chapters 6 and chapter 7, respectively. Chapter 8 will introduce the proposed end-to-end system. Chapter 9 will give details about the implementation and the evaluation experiments. Finally, chapter 10 will conclude this dissertation.

Chapter 2

Background

2.1 Text Detection

In this section, details of the most widely used text detection datasets will be given. Also, in this section, the top state-of-the-art text detection models will be analyzed.

2.1.1 Evaluation Datasets

Many datasets have been created to help with various tasks related to text analysis. Example tasks include text detection, numerical digit recognition, character recognition, and word-level text recognition. This section describes several popular datasets that have been used in text detection research.

Images in text-detection datasets can be grouped into two main categories. The first is *focused scene text*, where text is a major emphasis of the image, and most of the text instances are near the center of the image. This category includes the majority of the datasets listed below. The second category is *incidental text*, where text is not the emphasis of the image. This category is much harder to analyze because text can be anywhere in the image, and sometimes in a very small portion of the image.

One of the best-known datasets is MNIST [32], which was released in 1998. The dataset

Figure 2.1: Random samples from the MNIST dataset.

contains samples of the handwritten digits 0 to 9 in monochromatic images of size 32×32 pixels. MNIST contains 60,000 samples in a training set and another 10,000 images in a test set. MNIST is widely used in tutorials because of its simplicity and small size. Some random samples from MNIST are shown in figure 2.1.

The ICDAR 2003 dataset [42] was released as a part of an automated reading competition organized by the International Conference on Document Analysis and Recognition (ICDAR). This dataset contains 258 annotated images to be used for training, and 251 annotated images to be used for validation. The images are in color, showing everyday scenes. Annotations are provided, using rectangular bounding boxes to indicate instances of text. This dataset was used for competitions in 2003 and 2005.

The ICDAR 2013 dataset [28] was created with an emphasis on finding text in natural scene images. ICDAR 2013 is the most widely used dataset for evaluating text detection systems. The dataset contains 229 images for training and 233 images for testing. This dataset was used for automated reading competitions in 2011 and 2013. Sample images from this dataset are shown in figure 2.2.

The ICDAR 2015 robust reading competition introduced the first dataset devoted to inci-

Figure 2.2: Samples from ICDAR'13. From left to right: An input image, the same image with the text areas highlighted, and the ground truth text file.

dental text [29]. This dataset contains 1000 color images for training and 500 images for testing. The competition also provided datasets for born-digital text and text in video.

The MSRA-RD500 dataset [66] was introduced by Yao et al. in 2012. An emphasis of this work was the detection of text at arbitrary orientations in natural images. The dataset contains 500 everyday indoor and outdoor images with English and Chinese text instances. Unlike previous datasets, MSRA-RD500 provides bounding boxes that have been rotated to accommodate instances of rotated text. Some samples from the dataset with rotated bounding boxes are shown in figure 2.3. In other datasets, rotated text instances are simply annotated using larger rectangular boxes that are aligned with the image boundaries. Researchers who consider horizontal text only, or who depend on Latin language characteristics, tend to avoid this dataset.

The largest publicly available non-synthetic dataset to date that supports text detection is

Figure 2.3: Random samples from the MSRA-RD500 with the text areas highlighted.

Figure 2.4: A sample from the COCO-Text dataset with the associated annotations. Image source: https://vision.cornell.edu/se3/coco-text/.

Table 2.1: Counts of text instances in COCO-Text.

	Legible			Illegible		
	Machine-printed	Handwritten	Other	Machine-printed	Handwritten	Other
Training	66479	3687	1528	35968	1905	9103
Validation	30750	1469	753	16722	911	4311
Total	97229	5156	2281	52690	2816	20111

COCO-Text [61]. It was released early in 2016 and was derived from a larger dataset known as COCO (Common Objects in Context) [38]. COCO was developed to support many computer-vision tasks, including image recognition, segmentation, and captioning. COCO contains more than 300,000 images with multiple objects per image. COCO includes more than 2 million instances of 80 object categories. COCO-Text provides bounding rectangles along with a collection of labels for the text instances that appear in COCO images. A sample from COCO-Text with annotations is shown figure 2.4. As shown in the figure, OCR would fail in reading such challenging text instances. COCO-Text contains a total of 63,686 images, with 43,686 training images and 20,000 validation images. In addition to localization information, COCO-Text identifies text instances as machine-printed or handwritten, and legible or illegible. Count of text instances per type is shown in table 2.1. Transcriptions are provided for the legible instances of text. A limitation of COCO-Text is that it does not provide character level labeling, which could be used to extract text sub-images.

Another dataset that was used in the training of the proposed system is SynthText in the Wild [16], which is a dataset of 800,000 training images generated using a synthetic text image generation engine. This engine overlays synthetic text to existing background images in a natural way. It first creates a 3D model of the background image, then, it lays the text instances over the 3D objects. Each image has about ten word instances. SynthText in the Wild provides character level plus word level bounding rectangles along with a collection of labels for the text instances. Samples from SynthText with annotations are shown figure

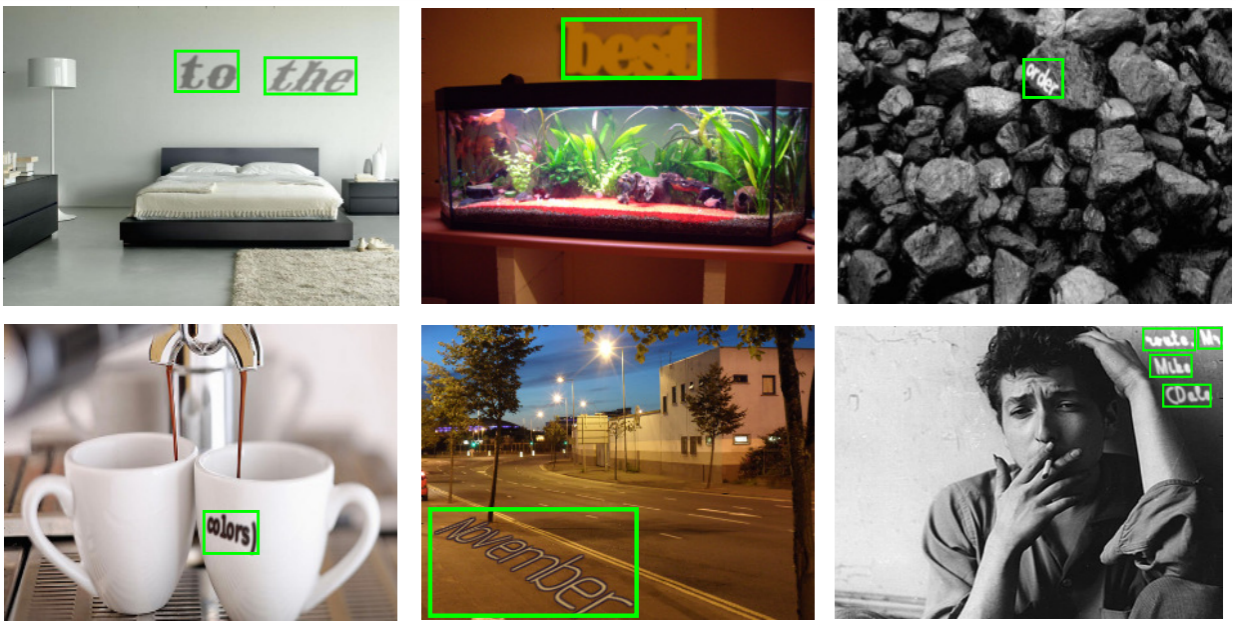


Figure 2.5: Samples from the SynthText dataset with the associated annotations. Figure from [16]

Table 2.2: Most widely used text detection datasets.

Dataset	Year	No. of images
MNIST	198	70K
ICDAR11,13	2011,13	0.5K
MSRA-RD500	2012	0.5K
ICDAR15	2015	1.5K
COCO-Text	2016	63K
SynthText	2016	800K

2.5.

Table 2.2 lists the most widely used text detection datasets ordered by the year in which they have been introduced to the research community. The tables show how the size of datasets is getting bigger over the years. The COCO-Text dataset was a massive jump in 2016, not only in size but also in the complexity of the text instances. While the SynthText dataset is the biggest dataset of all, COCO-Text remains the most significant dataset for being non-synthetic.

Figure 2.6: From left to right: A text instance along with its ground truth and prediction bounding boxes shown in solid green and dotted blue, respectively, the area of overlap between the ground truth and the prediction boxes, and the area of union between the ground truth and the prediction boxes are shown in black.

2.1.2 Evaluation Metrics

The most widely used metrics to evaluate the performance of text detection systems are Precision, Recall, and F-measure. These metrics can be used to evaluate any general object detection system. In this section, a description of those metrics and how they are used to evaluate text detection systems will be given. Before discussing the metrics, we will give a definition of few related terms. Intersection over Union (IoU) is a measure of the overlap between a predicted bounding box and the corresponding ground truth bounding box. As shown in figure 2.6, a text instance along with its ground truth and prediction bounding boxes are shown in solid green and dotted green, respectively. The intersection over union is defined as the ratio between the area of overlap between the ground truth and the prediction boxes, and the area of union between the ground truth and the prediction boxes. A correct prediction, known as a true positive, is a prediction where the value of the IoU ratio is 0.5 or more. A value less than 0.5 indicates that the prediction is so far from the ground truth that it is counted as incorrect detection, known as a false positive. A false negative is a missed instance in which a ground truth bounding box is not associated with any prediction bounding box. Those values, True Positive (TP), False Positive (FP), and False Negative (FN), are used to calculate the Precision, Recall, and F-measure as shown in (2.1), (2.2), (2.3).

$$Precision = \frac{TP}{TP + FP} \quad , \quad Precision \in \mathbb{R} : 0 \leq Precision \leq 1 \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \quad , \quad Recall \in \mathbb{R} : 0 \leq Recall \leq 1 \tag{2.2}$$

$$F\text{-}measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad , \quad F\text{-}measure \in \mathbb{R} : 0 \leq F\text{-}measure \leq 1 \tag{2.3}$$

Most of the datasets provide evaluation scripts or evaluation web servers that can be used to calculate the Precision, Recall, and F-measure values for the test set. Those scripts are open source scripts that follow the same standard procedures of calculating the performance metrics. It is the decision of the dataset creators to set the level of granularity of the annotations. Most of the text detection dataset annotations are set around words. The standard evaluation procedure, described above, would penalize individual character predictions. Although using these metrics encourages creating "word" detection systems, the common name used for such a system in the literature is "text" detection.

2.1.3 State-of-The-Art Models

A variety of techniques and models have been used for text detection. This section will focus only on the most recent promising models that resulted in state-of-the-art results on the standard benchmarks. The reviewed models will be clustered into multi-stage models and end to end models. A summary of the surveyed models can be found in table 2.3.

Tian et al. [59] proposed the Text Flow detection system. This system consists of two steps including character candidate detection handled by cascade boosting and text line extraction solved by a min-cost flow network. To handle the typical error accumulation problem, a flow network model is designed to integrate the three sequential steps into a single process which is solved by a min-cost flow technique. This model could achieve an F-measure value of 0.80

on the ICDAR 2013 dataset.

In [26], Jaderberg et al. combine two detection techniques the Edge Boxes region proposal algorithm [77], and a weak aggregate channel features detector [12]. The authors followed the mainstream in text detection by generating a large number of proposals to achieve a high recall percentage. Then, they used a random forest text/non-text classifier to reduce the number of false positives. For the text recognition, the authors used a Convolutional Neural Network (CNN) with five convolutional layers followed by three fully connected layers. The CNN takes an input text region of size 32×100, and recognize the text in it. This text region is cropped from the original image, then passed the verification step that uses a random forest classifier. This model follows the traditional multistage text detection and recognition pipeline which suffers from errors accumulation and blocking all contextual information. Using these techniques, the authors could achieve an F-measure of only 0.76 on the ICDAR 2013 dataset.

He et al. [19] developed a novel technique called Contrast Enhancement Maximally Stable Extremal Regions (CE-MSERs), which extends the well-known MSERs technique by enhancing intensity contrast between text patterns and background. The authors used CE-MSERs to generate text region proposals with high recall ratio. Then, a novel Text-Attentional Convolutional Neural Network (Text-CNN) has been used as a text classifier to remove false positives. He et al. developed a new learning mechanism to train the Text-CNN with multi-level supervised information. The proposed system in this dissertation uses similar techniques to enhance the training process. Figure 2.7 shows the structure of the Text-CNN with the multi-level supervision denoted by λ_1 and λ_2. The authors discussed how training of the auxiliary tasks enhanced the model performance. This model generated good results on the ICDAR 2013 dataset, with an F-measure of 0.82.

In [74], Zhang et al. trained a Fully Convolutional Network (FCN) model to predict the

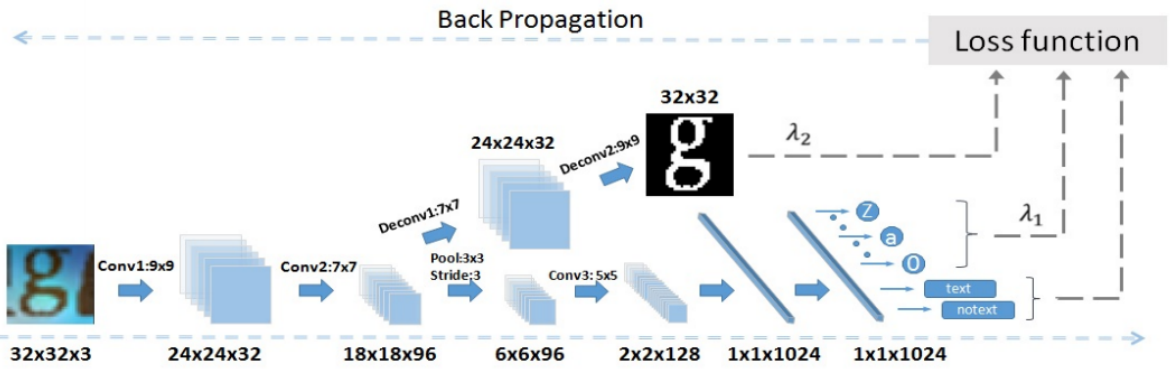


Figure 2.7: Structure of the Text-Attentional Convolutional Neural Network (Text-CNN). Figure from [19].

location of text regions in an input image. Then, another FCN is used to predict the centroid of each character, in order to remove the false proposals. Using FCN enabled this model to detect oriented text instances. This model could achieve an F-score of 0.83 on the ICDAR 2013 benchmark and an F-score of 0.54 on the more challenging ICDAR 2015.

Gupta, Vedaldi, and Zisserman [16] were the first to merge the early two stages of the text detection and recognition pipeline. They proposed a deep network called the Fully-Convolutional Regression Network (FCRN). FCRN efficiently performs text detection, and bounding-box regression at all locations and multiple scales in the input image. The proposed network out performed other methods for text detection , achieving an F-measure of 0.842 on the ICDAR 2013 dataset.

The TextBoxes [36] model is a fast end-to-end model. Their proposed model adds an extra nine layers over the well-known VGG [55] network. Those extra layers are connected to the output layer where bounding boxes are created. The high speed of this model came with an expected side effect of lower accuracy. TextBoxes reported an F-measure of 0.85 on the ICDAR 2013 standard benchmark, and no result has been reported on the more challenging COCO-Text benchmark.

Table 2.3: A summary of the surveyed models.

Model	Published in	F-measure*
Edge Boxes, Random Forests [26]	IJCV 2016	0.76
Text Flow [59]	CVPR 2015	0.80
CE-MSERs, Text-CNN [19]	IEEE TIP 2016	0.82
Two Fully Convolutional Network (FCN) [74]	CVPR 2016	0.83
Fully-Convolutional Regression Network (FCRN) [16]	CVPR 2016	0.84
TextBoxes [36]	AAAI 2017	0.85
Single-Shot Text Detection [18]	ICCV 2017	0.87

*On the ICDAR'13 dataset

He et al. [18] proposed a text attention model as an auxiliary loss that is discarded during inference, built upon complex Inception convolutional features. The Inception module implemented in this model consists of four parallel paths. Those paths contain 1×1, 3×3 and 5×5 convolutional layers in addition to a 3×3 max-pooling layer. This Inception module is inspired by the GoogleNet architecture [57]. The attention model encodes text-specific information using text masks. Second, the authors developed a complex hierarchical Inception module which aggregates multi-scale inception features. An Inception architecture with dilated convolutions, borrowed from the work of [70], is applied to each convolutional layer, enabling the model to capture multi-scale image content. This model, named Single-Shot Text Detection, is the current state of the art on the ICDAR 2013 dataset with F-measure of 0.87. The proposed end-to-end system uses attention information as well, but in a different way that proved to be simpler and more efficient. The single shot text detection model has a complicated architecture as shown in figure 2.8 which illustrates the training phase of this model. During the inference phase, the single shot text detection model will not use text masks. The model will pass the input image through the text attention module to create a 64×64 attention map. This attention map is fused into three different Inception feature maps using pixel-wise dot product.

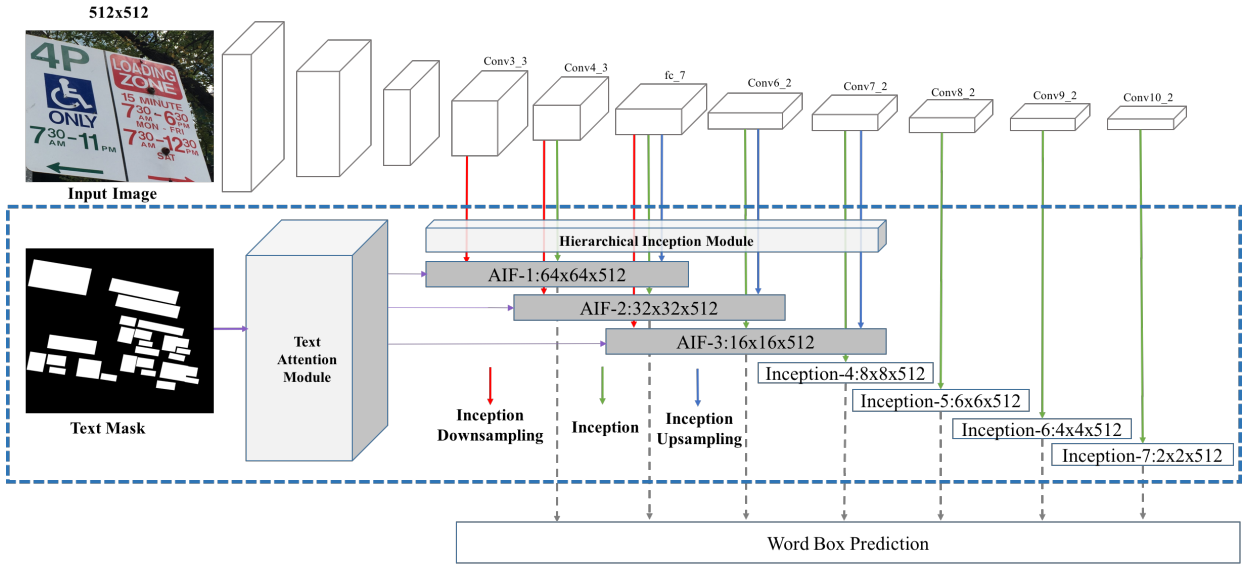


Figure 2.8: The complex architecture of the single-shot text detection model with the multi-layer Inception modules. This figure represent the training phase of the model. Figure from [18].

2.2 Deep Learning

Until recently, most of the machine learning techniques used shallow structures such as Gaussian mixture models (GMMs) [78], hidden Markov models (HMMs)[13], support vector machines (SVMs)[8], logistic regression, kernel regression, and multi-layer perceptron (MLP) [52]. Those shallow machine learning techniques depend heavily on the quality of extracting engineered features from the input signal. Recently, class of machine learning techniques, called deep learning, attracted a huge interest in many domains such as computer vision. Deep learning can be defined as a class of machine learning techniques, where many layers of information processing stages in hierarchical architectures are exploited for unsupervised feature learning, and for pattern analysis [11]. The main difference between deep learning techniques, and shallow structures techniques is the ability of deep learning architecture to learn powerful hierarchical representations of the data. A simple workflow comparison between the traditional machine learning and the deep learning techniques is shown in figure

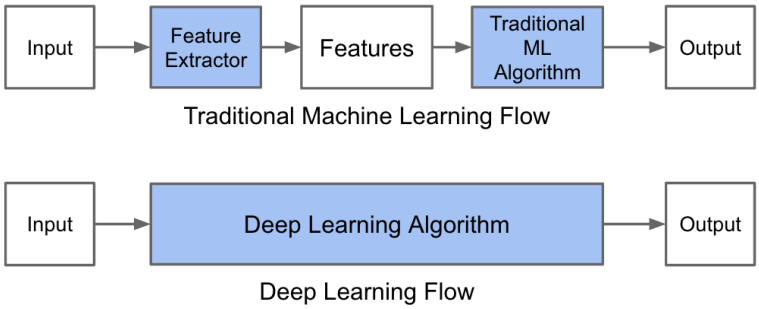


Figure 2.9: Work flow comparison between the traditional machine learning and the deep learning techniques.

2.9

Plenty of deep learning architectures have been introduced recently, such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Fully Convolutional Network (FCN), and many other architectures. In this section, a brief background will be given about the architectures that will be used in building the proposed model.

2.2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) consists typically of several convolutional layers followed by a small amount of fully connected layers. Figure 2.10 is a block diagram for one of the well known CNNs named LeNet [33]. A single convolutional layer can be viewed as a stack of learnable filters that slide over the input image to generate a set of stacked activation maps as shown in figure 2.11.

Looking back at figure 2.10, the first set of feature maps C1 is a stack of six 28×28 activation maps generated by convolving a stack of six 5×5 filters. A single activation map of size 28×28 is generated by convolving a 5×5 filter over a $32 \times 32 \times 3$ input image with stride 1, and with no input padding. There are 28×28 unique positions for a 5×5 filter in a 32×32 input, so the convolution produces a 28×28 activation map, where each element is

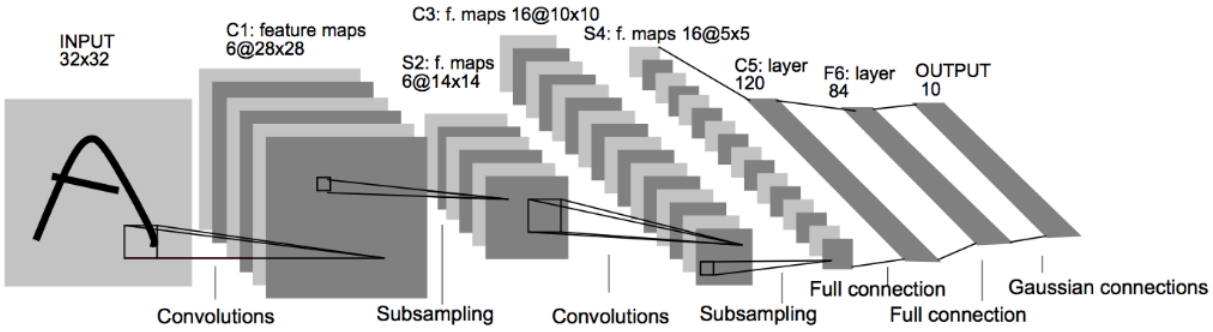


Figure 2.10: Structure of LeNet created by Lecun et al. [33].

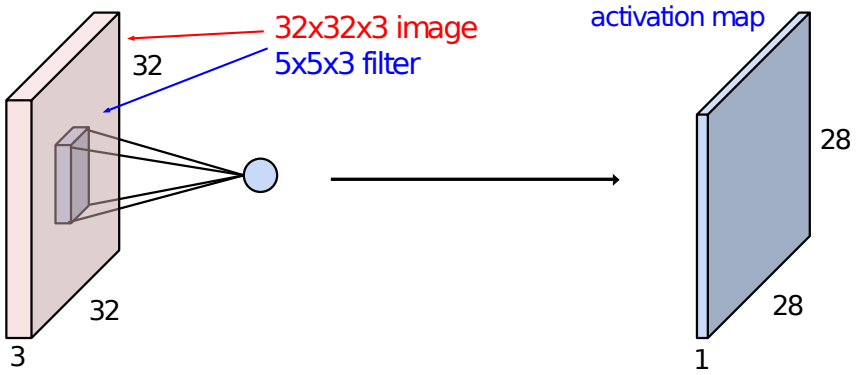


Figure 2.11: Structure of a convolutional layer. A stack of filters will create a stack of activation maps.

the result of a dot product between the filter and the input. The filters are typically small spatially compared to the input image. In the case of LeNet, the input was $32 \times 32 \times 3$, and the filter size is 5×5. The filters always span the full depth of the input array which is 3 in the case of a colored image. A convolutional layer does not contain just one but a set of different filters, each applied in the same way, and independently, resulting in their own activation maps. The activation maps are finally stacked together along depth to produce the output of the layer (e.g., $28 \times 28 \times 6$ array in this case).

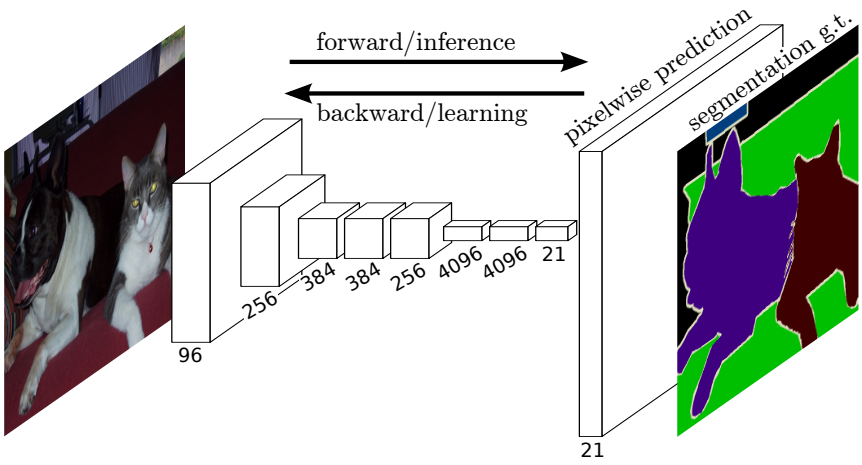


Figure 2.12: The basic architecture of the fully convolutional network for semantic segmentation. Figure from [41]. From left to right: The input image, the model and the segmentation ground truth (g.t.)

2.2.2 Fully Convolutional Networks (FCN)

The basic idea behind a fully convolutional network (FCN) is that all of its layers are convolutional layers. FCNs do not have any of the fully-connected layers at the end, which are typically used for classification. Although FCNs can be used to learn any task, it has been widely used a semantic segmentation model. An FCN has been used in one of the very well-known semantic segmentation models [41]. As shown in figure 2.12, this semantic segmentation model uses convolutional-only layers to perform pixel-wise classification of the input image. Fully convolutional based text detection models, such as the FCRN model [16] introduced in section 2.1.3, suffer from the rough output generated by the model which requires heavy post-processing. The proposed system makes use of such rough activation maps of a lightweight FCN network as an extra aid. The proposed system generates word box predictions without the need for any post-processing other than simple non-maximum suppression.

Chapter 3

Multi-Phase Learning Transfer

In this chapter, an exploration study on how multi-phase learning transfer can enhance the performance of deep models. A novel multi-phase learning transfer technique is used to develop the proposed end-to-end system, as will be shown later in chapter 8. He et al. [19] demonstrated that providing multi-level highly-supervised text information to a deep network during training would yield better performance. In section 2.1.3, we discussed how He et al. used auxiliary tasks to train their text classifier. As shown in figure 2.7, the Text-CNN model has been trained for character recognition and semantic segmentation just to help the main task of text classification.

To assess the impact of learning transfer techniques on the performance of models learning text-related tasks, a small-scale experiment was designed. In this experiment, a model is implemented to identify the handwritten digits in the MNIST dataset [32]. The training is divided into two phases as shown in figure 3.1. In the first phase, ten networks are trained. Each of those networks is trained to recognize only one of the ten digit classes. In the second phase, the trained ten networks are combined into one big network, then an extra Fully Connected (FC) layer is added. During the second phase training, all the network weights are frozen except for the newly added FC layer which is trained to recognize one of the ten digits.

To build this model, a LeNet [33] network has been used as a base model for the single digit networks. To use LeNet for single digit recognition, the last FC layer output had to be

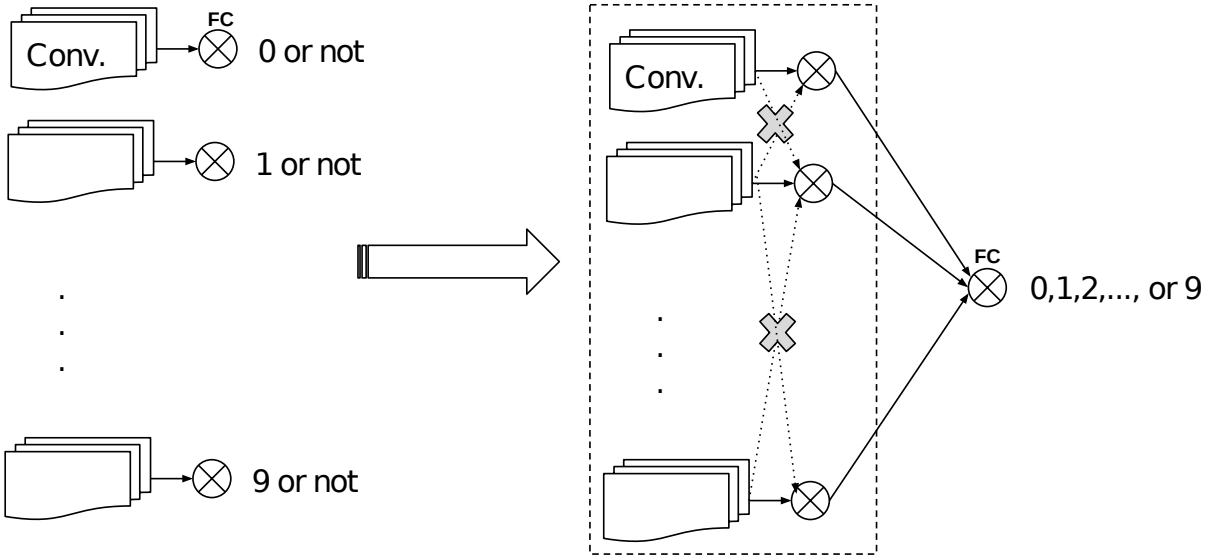


Figure 3.1: To the left: Phase 1, train each network on a single digit. To the right: Phase 2, freeze all the weights of the combined networks, then train only the newly added Fully Connected (FC) layer. The dotted lines with an X mark on them are connections that would exist in a traditional network architecture but do not exist in this implemented network.

modified from ten classes to two classes. The positive class used to train each single digit network is the set of all images for that digit in the training set. The negative class is the set of all images in the training set except for the image of the digit that represents the positive class for this network. Each single digit network has been trained for $10K$ iterations. Then the combined network has been trained for another $10K$ iterations. The model achieved an accuracy of 99.9% on the MNIST test set. This accuracy exceeds the state-of-the-art results on the MNIST dataset as shown in table 3.2.

Note that to build a traditional network of similar size to the combined network, one would simply create a network that is ten times larger than the single digit network. As shown in table 3.1, The traditional network will be ten times bigger due to the fact that the first FC has to connect to all convolutional filters in the layer before it. The connection structure of the proposed model is smaller than the traditional one due to the fact that the first FC layer

Table 3.1: The output shape and weights size for each layer of the proposed network compared to a traditional network of the same size.

	Traditional network		Proposed network	
Layer	Shape	Weights	Shape	Weights
Input	28×28	0	28×28	0
Conv1	$200 \times 24 \times 24$	$5 \times 5 \times 200 = 5K$	$10 \times 20 \times 24 \times 24$	$5 \times 5 \times 10 \times 20 = 5K$
Pool1	$200 \times 12 \times 12$	0	$10 \times 20 \times 12 \times 12$	0
Conv2	$500 \times 8 \times 8$	$5 \times 5 \times 500 = 12.5K$	$10 \times 50 \times 8 \times 8$	$5 \times 5 \times 10 \times 50 = 12.5K$
Pool2	$500 \times 4 \times 4$	0	$10 \times 50 \times 4 \times 4$	0
FC1	5000	$5000 \times 8000 = 40M$	10×500	$10 \times (500 \times 800) = 400K$
FC2	20	$20 \times 5000 = 100K$	10×2	$10 \times (2 \times 500) = 10K$
FC3	10	$10 \times 20 = 200$	10	$10 \times (2 \times 10) = 200$
Total		$40117.7K$		$\mathbf{427.7K}$

Table 3.2: Comparison with existing models on MNIST.

Model	No. of Param.	Error
RCNN-96 [35]	$670K$	0.31%
Maxout [15]	$420K$	0.45%
Proposed	$427.7K$	**0.1**%

is constructed from smaller FC layers of single digit networks as shown in figure 3.1. The figure shows dotted lines with an X mark on them that represent the connections that would exist in a traditional network architecture but do not exist in this implemented network.

This multi-phase method, where a bigger task is divided into smaller tasks, demonstrated a better performance compared to state-of-the-art models on the MNIST benchmark. A novel multi-phase training technique is used in the proposed end-to-end text detection system.

Chapter 4

Text Attention Maps

In this chapter, the details of implementing the text attention map model will be given. Class Activation Maps (CAM) is a related technique that has been researched during the early stages of building the proposed system. CAM will be discussed in this section as well.

4.1 Class Activation Maps (CAM)

Zhou et al. [75] proposed a method termed Class Activation Mapping (CAM). A CAM is heat map that represents the possible location of objects in the input image. In a Convolutional Neural Network (CNN), a CAM can be generated by mapping the categorical knowledge learned by the final layer back to the convolutional layers. This class activation map highlights the informative objects and parts detected by the CNN. As such, the CAM method enables the classification-trained CNN to perform object localization directly, in a single forward path, without training on bounding box annotations. Zhou et al. demonstrated that their weakly supervised CNN methods could achieve reasonably good localization performance, even coming close to some fully supervised CNN methods on the test set of multiple benchmarks. Furthermore, they show that they can use the CAM to find generic visual features useful for both object classification and localization tasks, across various recognition datasets. With the help of CAM technique, the CNN could classify the image into some class and localize the discriminative class-specific image regions in a single forward pass as

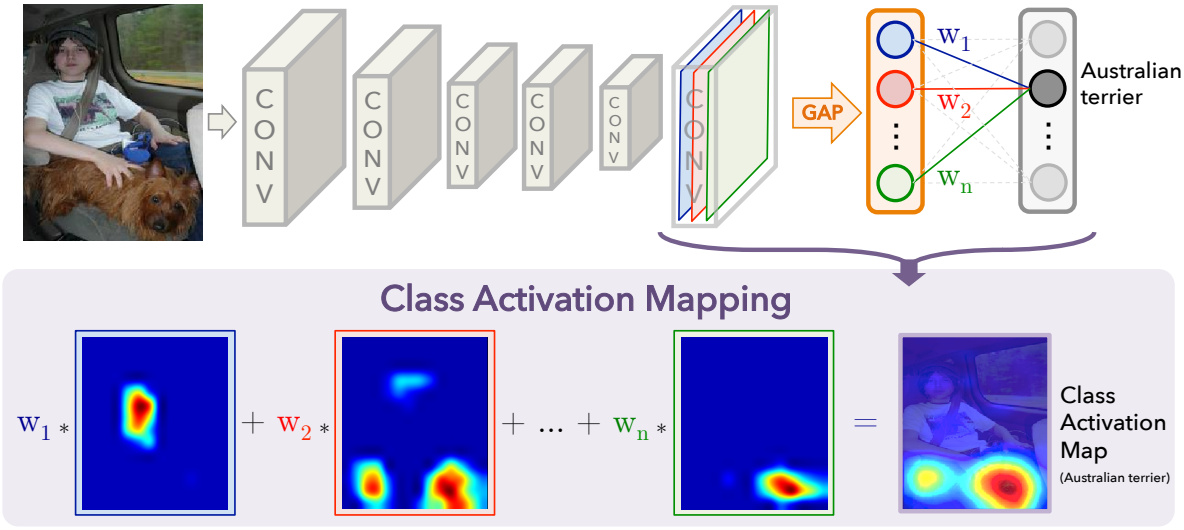


Figure 4.1: Class Activation Mapping [75]: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

shown in figure 4.1. We acquired a similar technique and adapted it to the problem of text detection.

The original model implemented in [75] used several networks built to work on the ImageNet [53] benchmark. The results showed that GoogLeNet-GAP (based on the well known GoogLeNet [57]) achieved the best performance. The output layer of this network was set to 1000 classes to be suitable for the ImageNet benchmark. To adapt this network to text detection, the output layer of the network was set to 2 classes , text and non-text, instead of the original 1000 classes of ImageNet. We call the adapted network, shown in figure 4.2, GoogLeNet-GAP-TXT to distinguish it from the original one in [75].

A dataset is needed to train this model as a text/non-text classifier which will be used to generate the class activate mapping for the "text" class. There is no standard dataset with text/non-text samples that can be used to train such a classifier. To cover this gap, we created a dataset with text samples from COCO-Text, and non-text samples from COCO.

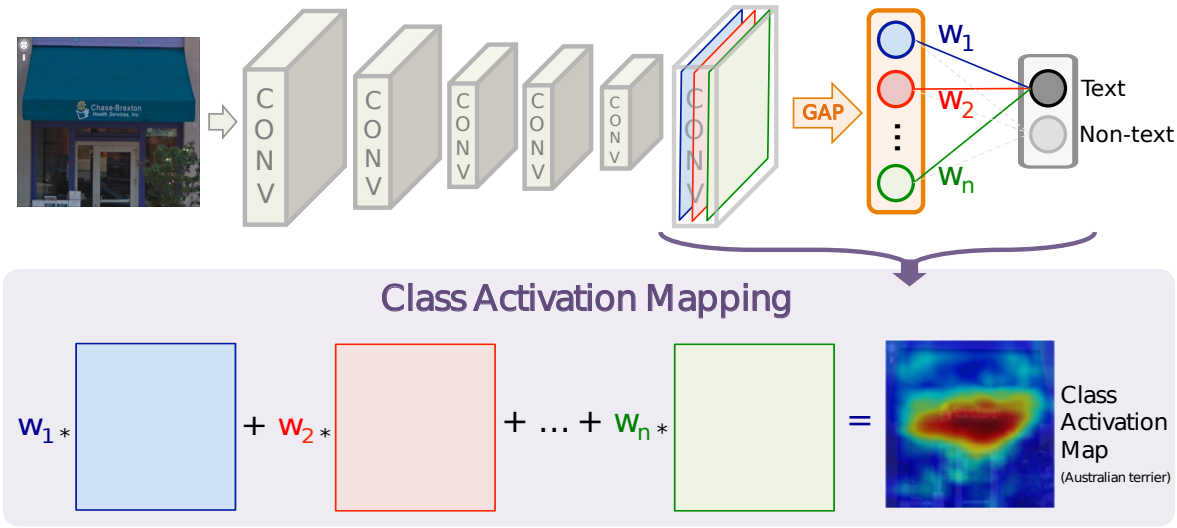


Figure 4.2: GoogLeNet-GAP-TXT: Class Activation Mapping (CAM) after being adapted to text detection with a sample input from the SVT dataset.

Table 4.1: Count of instances in the dataset created to train GoogLeNet-GAP-TXT.

Set	Text	Non-Text	Total
Training	15k	7K	22K
Validation	11K	8K	19K

The count of instances used to create this dataset is shown in table 4.1.

Later, we addressed the lack of a standard dataset for training text classifiers by creating a publicly available text/non-text dataset derived from COCO-Text [61] named COCO-Text-Patch [23]. Extensive discussion of this dataset will be given later in chapter 5.

This model has been trained using the described dataset for 750 thousand iterations with batch size 32 which is equivalent to nearly 1000 epochs. The resulting class activation mappings of the "text" class from this trained model were not promising. A sample test image with the associated class activation mapping is shown in figure 4.3. It is clear in figure 4.3 that the focus is on the bus body while the actual text is located on the front of the bus. This behavior can be explained by the fact that a text instance usually exists in a

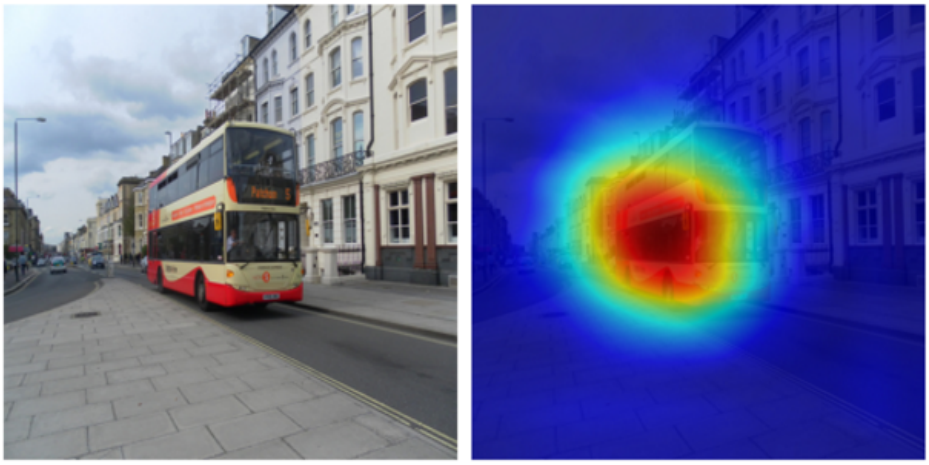


Figure 4.3: A sample image from COCO-Text with the associated class activation mapping generated from GoogLeNet-GAP-TXT.

context which is usually a larger object near or enclosing the text instance. Many examples can be given such as a text instance inside a stop sign, a text instance on top of a store main entrance, or a text instance on a bus as shown in figure 4.3. This behavior gave rise to the need of giving additional supervision to the network. As will be discussed in the next section, this additional supervision can be in the form of a labeling grid that would give a hint to the network about the actual location of the text instance.

4.2 Labeling Grid

The network of GoogLeNet-GAP-TXT has been trained using text/non-text labels as detailed in the previous section. This architecture demonstrated the failure to detect the location of text instances in an input image. In this section, we will describe how adding more supervision about the location of the text will enhance the localization performance of the network. The main idea is to replace the text/non-text labels with a vector of decimal values that represents the location of the text instances in the input image. To explain how

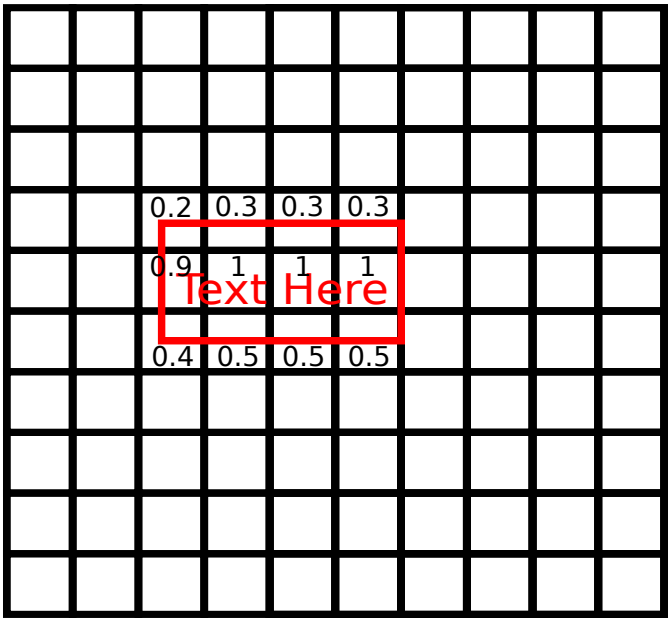


Figure 4.4: An illustration of an imaginary grid imposed on an image that contains a text instance marked with a red square. The gird in this illustration is of size 10×10 which will result in a label vector of size 100. Cells which are totally enclosed inside the text area get a value of 1 while cells that are totally outside the text area get a value of 0. Cells that intersect with the text area get a decimal value depending on the area of intersection.

this vector created, assume that there is a square grid imposed over the image associated with this vector. This grid will be of size $C \times C$ where C is the width of this grid in cells as shown in figure 4.4. A decimal value between 0 and 1 will be assigned to each cell according to criteria that will be explained shortly. For any input image, its label vector of size C^2 is simply the flattened version of this grid of decimal values.

The formula used to calculated the value of each cell is

$$L_{i,j} = \frac{T \cap C_{i,j}}{C_{i,j}} \tag{4.1}$$

where $L_{i,j}$ is the value of the label vector corresponding to the cell $C_{i,j}$ where $i, j \in 1, 2, .., C$ are the location indexes, C is the width of the grid in cells, and T is the set of pixels that

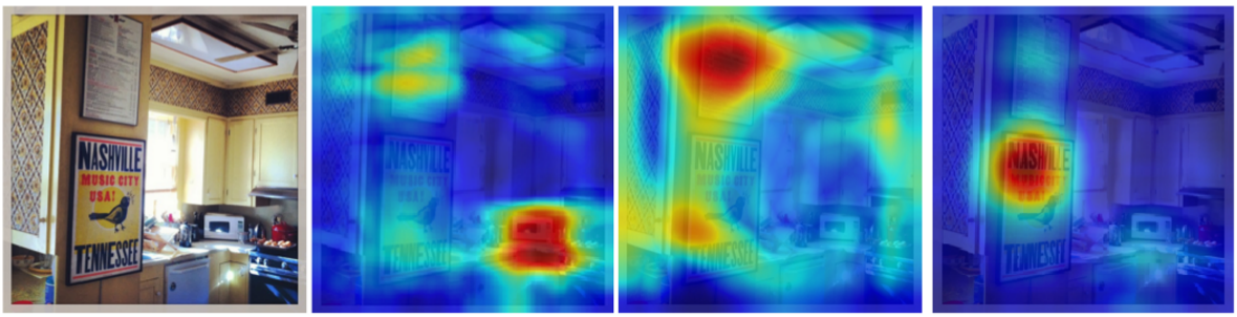


Figure 4.5: From left to right: A sample image from COCO-Text, the CAM generated from GoogLeNet-GAP-TXT without a labeling grid, the CAM generated from GoogLeNet-GAP-TXT with a 10×10 labeling grid, the CAM generated from GoogLeNet-GAP-TXT with a 6×6 labeling grid.

represent the text area.

The width of the grid C remains a hyper parameter that needs to be optimized. Three experiments have been conducted to investigate the effect of the parameter C on the model performance. The three experiments are for grid sizes 10×10, 8×8, and 6×6. The experiment for $C = 8$ generated class activation maps that are very similar to the ones generated with $C = 10$. Results from the original model without a labeling grid, with labeling grid of size 10×10, and of size 6×6 are shown in figure 4.5. The results show how the labeling grid enhanced the performance of the model substantially. The original model without the labeling grid activated over a microwave while the modified model activated over the actual text areas in the image.

The results in figure 4.5 also show how the choice of the parameter C value affects the behavior of the model. The network activated over wider areas of the image in the case of $C = 10$ compared to the tight activation in case of $C = 6$. The choice of this parameter value is directly related to the localization recall and precision. A higher value for C will result in a higher recall, but lower precision. While lowering the value of C would enhance the precision, but will reduce the recall which would result in missing some text instances

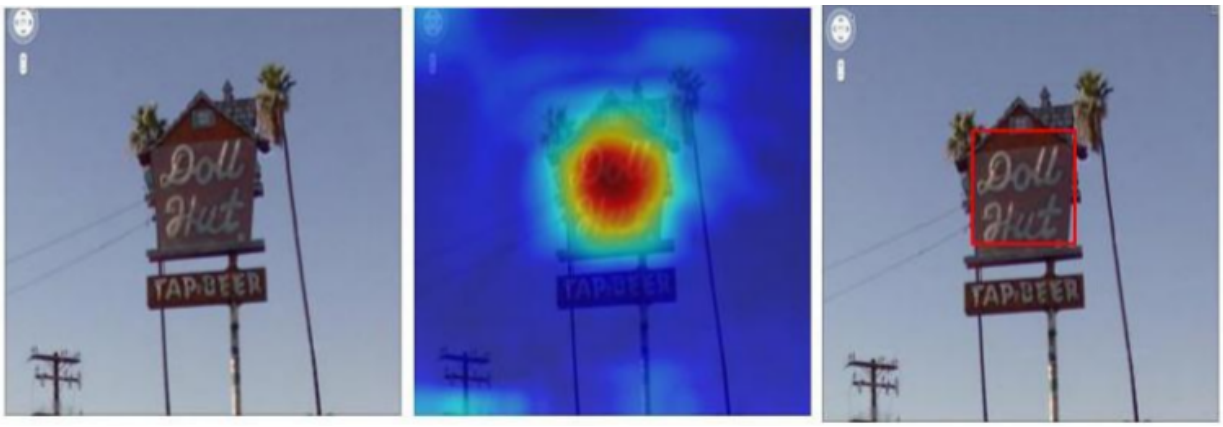


Figure 4.6: From left to right: A sample from SVT, the Class Activation Map (CAM) generated with a labeling grid of size 6×6, and the detected text instance after applying a threshold on the CAM. This grid resolution enhanced the precision compared to a grid size of 10, but resulted in missing one text instance.

as shown in figure 4.6. A solution to this problem will be discussed in the next section.

4.3 Dual-Stage CAM

The first stage of the traditional text detection and recognition pipeline is the region proposal stage. Usually, the hyperparameters of the algorithms used in this stage are set such that it generates a high recall. Then, the second stage, which is the text verification, filters out false positives to increase the precision of the model. More details about this pipeline had been discussed in section 1.2. Inspired by this traditional technique, we split the Class Activation Map (CAM) generation into two stages. In this model, the input image is fed to a GoogLeNet-GAP-TXT with labeling grid of size 10×10 to generate a CAM with a high recall. This CAM will be used to crop out the area in the input image that corresponds to high activation as shown in figure 4.7. This cropped area will be fed into the second stage which is an identical replica of the first stage. Experiments have demonstrated that the

second stage will generate CAM with better precision as shown in figure 4.8.

This research about CAMs can be extended further to enhance the general object detection model proposed in [75]. But the results on the text detection problem gave rise to the need for a different attention technique for two reasons. Firstly, the class activation map generation process is complex and will be hard to integrate into a bigger end-to-end system. Secondly, there was no easy way to reduce the size of such a huge model. Using such a big network in the proposed end-to-end system will introduce a performance bottleneck. In the next section, a novel, powerful and light-weight attention model, that became part of the proposed end-to-end system, will be discussed.

4.4 Text Attention Maps

The attention model, an essential part of the proposed text detect system, produces high-resolution attention maps, often represented as "heat maps," that indicate salient parts of an input image. Attention maps are similar to CAMs except for how they are generated. CAMs are generated using a complex process, explained in the previous sections, while attention maps are generated as the direct output from a semantic segmentation network which can be integrated easily into end-to-end systems. The use of attention maps has been shown to enhance performance in various tasks such as image captioning [64], visual question answering [65], weakly-supervised object localization [48], and classification [44]. Such maps have been used in various ways to provide visual guidance to deep models. Of particular interest here, attention maps have been used successfully in state-of-the-art text detection models [18].

The system presented here uses attention maps in a novel arrangement that is both simpler and more effective than those used in previous text-detection systems. The fusion step

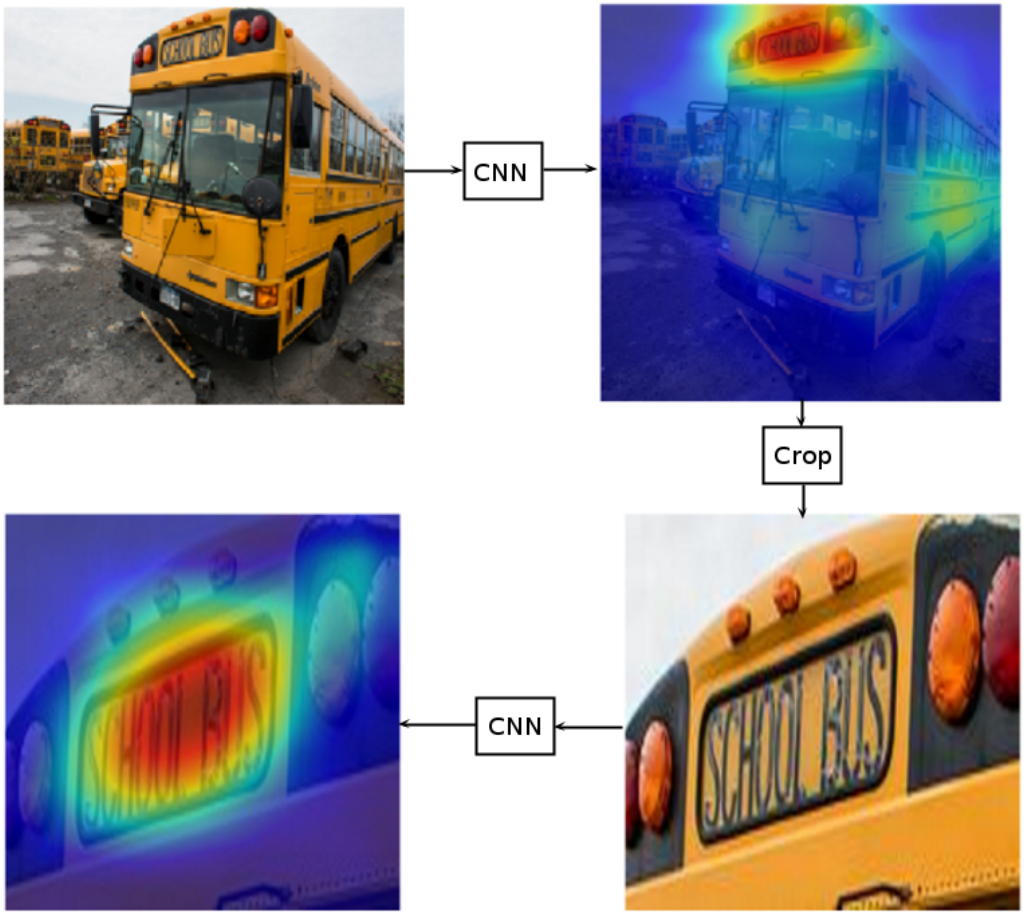


Figure 4.7: Multistage Class Activation Mapping: A sample image from COCO-Text is fed into the network. The class activation map generated from the first stage is used to crop out the candidate text area. Then, the class activation map generated from feeding the cropped image to the second stage.

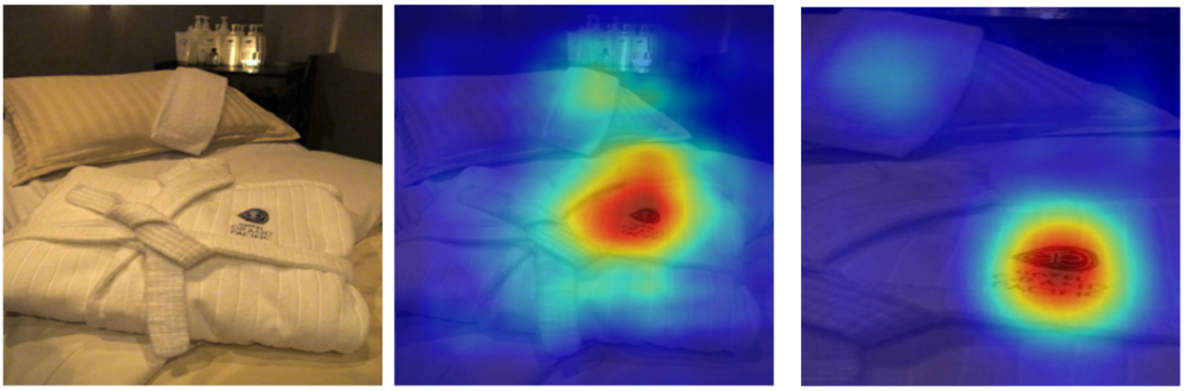


Figure 4.8: From left to right: A sample image from COCO-Text, the class activation map generated from the first stage, and the class activation map generated from the second stage.

produces a modified version of the image, called Attention Modulated Image (AMI), through direct pixel-by-pixel multiplication of the input image with its associated attention map. As indicated in figure 4.9, The resulting AMI highlights likely text positions while attenuating the background, and it serves as the direct input to the prediction model. Details about the proposed system will be given in chapter 8. To our knowledge, the system presented here is the first text-detection system to provide high-resolution maps directly as input to a prediction model.

There are two important benefits to this "early fusion" approach in contrast to the late fusion used in in the state-of-the-art text detection model[18]. First, the prediction model is guided to focus its early feature extraction layers on text-only regions. This emphasis enables such layers to learn features that are more relevant to text, and eventually leads to better detection of difficult text instances. Second, the usage of a full-resolution attention map, in contrast to the down-sampled attention maps used in [18], enables fine-grain text feature learning. It is important to note that early fusion of the attention map does not cause contextual information to be abandoned. In fact, visual context is indeed fully utilized in generating the attention maps themselves. We believe that such decoupling between context

consideration and fine-grain textual feature learning plays an important role in the success of our end-to-end system. Furthermore, the approach has allowed us to stage the learning process by focusing on one task at a time, hence training a powerful model in less time than otherwise required.

The training of the text attention model can be divided into two phases as shown in figure 4.9. The first phase is training the text classifier. We created a 12-layers model based on the FFNet architecture [24] that has been proven to achieve good performance using small models. The details of the FFNet architecture will be discussed in the next chapter. We modified the FFNet architecture by removing the fully connected layers and one of the convolutional layers. Removing the fully connected layers was necessary to create a fully convolutional network in order to serve our model. The classifier has been trained for 70 epochs and achieved an error rate of 10.4% on the validation set of COCO-Text-Patch. The architectural modification did not result in much loss of accuracy compared to the original FFNet model which achieved a slightly better error rate, 9%.

The second phase of the training used the SynthText in the wild dataset to train a semantic segmentation fully convolutional network. The dataset provides bounding boxes annotations which are not suitable for training a semantic segmentation model. We created text masks corresponding to the provided bounding boxes. These text masks will serve as the targets for the semantic segmentation network. Training a semantic segmentation network initialized with random weights will require creating a bigger network, which is not desirable, and requires much more data than what is available. Initializing, then training, this network with the weights of convolutional layers that have been pre-trained on Imagenet resulted in falling in a local minimum. We could identify that easily from the behavior of the network after the loss curve plateau. The model generates the same output for any supplied input. This is a known behavior for semantic segmentation networks that fail to learn. On the

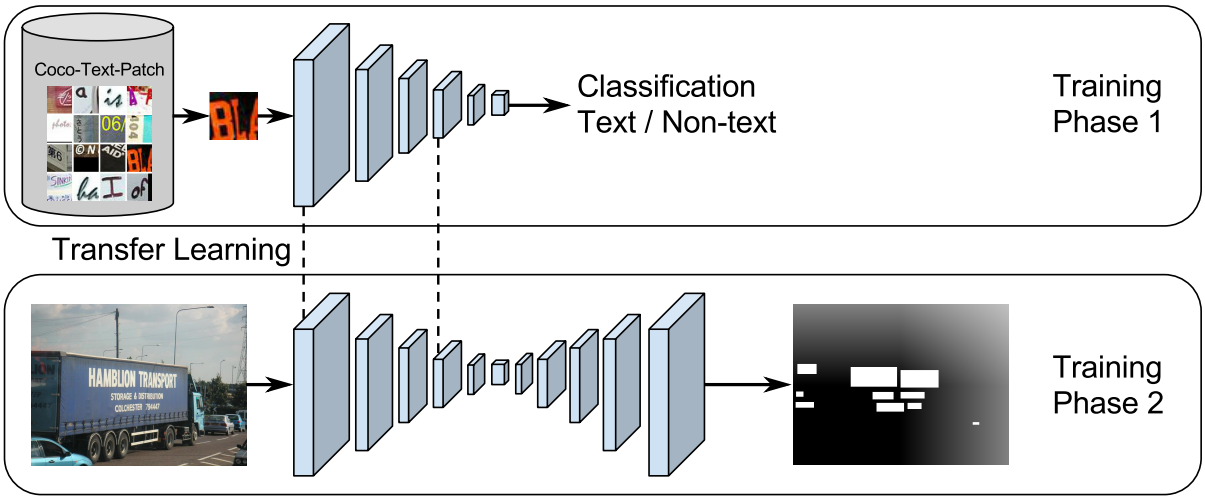


Figure 4.9: The proposed text attention map model with the learning transfer illustrated.

other hand, when initializing, then training, the network with the weights of convolutional layers that have been pre-trained on COCO-Text-Patch. The network could learn the task and create output maps that are relevant to the location of the text instances inside the supplied input images.

Chapter 5

COCO-Text-Patch

5.1 Introduction

This chapter describes how we created a dataset containing small images of text from everyday scenes. This work was published in [23]. purpose of the dataset is to support the development of a powerful text classifier network to be used in the text attention map model that is discussed in chapter 4. This new dataset, known as COCO-Text-Patch, contains approximately 354,000 small images that are each labeled as "text" or "non-text". This dataset also addresses the problem of text verification, which is an essential stage in the traditional multi-stage text detection pipeline. The primary role of the text-verification stage is to analyze tentative text regions from the text-localization stage and remove false positives [68].

As shown in the examples of Figure 5.1, each small image in COCO-Text-Patch is of size 32×32 pixels, which is very well suited for many deep-learning implementations. In order to evaluate the utility of this dataset, it has been used to train two deep convolution neural networks to distinguish text from non-text. One network is inspired by the GoogLeNet architecture, and the second one is based on CaffeNet. Accuracy levels of 90.2% and 90.9% were obtained using the two networks, respectively. All of the images, source code, and deep-learning trained models described in this chapter are publicly available [1].

[1] https://aicentral.github.io/coco-text-patch/

Figure 5.1: Left: A sample image from COCO-Text [61], with all text instances labeled as "legible" shown below it. Right: Several small images that are provided in the new dataset COCO-Text-Patch, which is introduced in this chapter. Each small patch shown at the right is a 32×32 sample that contains text. The dataset also provides non-text (background) patches, as needed for training.

Figure 5.2: Sample text patches from COCO-Text-Patch. The text patches represent a wide range of visual textures, colors, font types, and character orientations. In order to emphasize textural cues over character shapes, no attempt was made to capture individual characters or words.

5.2 Text and Non-text Patch Extraction

The procedure for extracting small text patches was relatively straightforward. For every legible text box that has been indicated by COCO-Text, including both machine-printed and handwritten cases, our system extracted non-overlapping sub-images of size 32×32 directly from the original COCO images. The patch dimensions were chosen largely because this size is convenient for some of the popular deep network architectures. Most of the deep network architectures that are designed for small image datasets such as MNIST, CIFAR10, and CIFAR100 expect input images to be of size 32×32. The other widely used image size used in training deep networks is 256×256. This size is suitable for representing complex scenes with multiple instances of objects, which is not the case for text patches. A few examples of the resulting COCO-Text-Patch images are shown in Figure 5.2.

Similarly, small non-text patches were extracted from portions of COCO images that are outside the text boxes indicated by COCO-Text. Text, by its nature, implies significant variations in visual texture. It was therefore important for COCO-Text-Patch to provide non-text examples that contain substantial levels of texture. For this reason, a texture-based measure was employed during the balancing step, as described in the next section. Python was used to implement all extraction and balancing algorithms.

Table 5.1: Number of patches in the final COCO-Text-Patch dataset.

	Text	Non-text	Total
Training	112044	130041	242085
Validation	52085	59977	112062
Total	164129	190018	354147

5.3 Dataset Balancing

Because text represents a relatively small proportion of image area within COCO-Text images, many more non-text patches than text patches were detected initially using the extraction strategy described in the previous section. In fact, as indicated in Figure 5.3, the number of patches extracted from legible machine-printed text represented less than 10% of the patches that were initially extracted. When text patches were also extracted from legible handwritten text, the proportion of text patches rose to about 22%.

In order to support ML approaches, particularly deep-learning, it was decided to provide further balance to COCO-Text-Patch by removing some of the non-text cases. Random sampling was considered briefly. However, the importance of texture led us to implement a fast texture-based approach. In our implementation, this analysis was accomplished by applying Prewitt [49] filters to a grayscale version of each non-text patch. The resulting gradient magnitudes were binarized, to indicate the presence of intensity edges. If the number of edge pixels exceeded an empirically selected threshold, then the patch was retained as a non-text sample. This edge-count threshold was adjusted so that a split of approximately 50:50 for text:non-text was achieved. As shown in the figure, the actual final ratio in the COCO-Text-Patch dataset was close to 46:54. Actual image quantities are shown in Table 5.1.

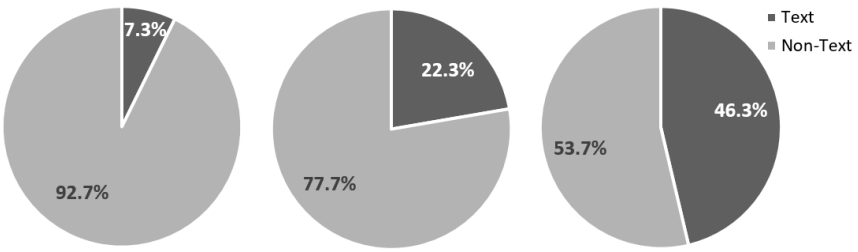


Figure 5.3: The ratio between text and non-text instances during development of COCO-Text-Patch. Left: the initial proportion of legible machine-printed text patches to non-text patches was approximately 7% to 93%. Center: increased proportion after inclusion of legible handwritten text. Right: final well-balanced proportion, after texture-based filtering of non-text patches.

5.4 Evaluation

The utility of the new COCO-Text-Patch dataset was evaluated using two convolutional neural networks. Both CaffeNet [1] and GoogLeNet [57] were trained using Caffe [27], and the resulting models will be made available to the research community. All training and testing has been done on the Virginia Tech NewRiver HPC [2].

Both CaffeNet and GoogLeNet were created to be used with images of size 256×256, and each was designed to learn 1000 classes. We modified both network architectures to be able to learn from the smaller size 32×32 images in COCO-Text-Patch as well as being able to learn 2 classes instead of 1000. The modified CaffeNet and GoogLeNet will be referred to as CaffeNet-TXT and GoogLeNet-TXT, respectively.

A close examination of the layers of CaffeNet show that the network tries to perform dimension reduction in the early layers. This can be inferred from the parameters of the CONV1 and POOL1 layers. The stride of CONV1 is set to 4, which was originally chosen to reduce the input size by a factor of four from 256×256 to 64×64. Then the POOL1 layer with stride 3 would cause more reduction. Those rapid reductions are not suitable for a input size of 32×32, so those layers have been modified such that the CONV1 and POOL1 layers

both have a stride of 1. Similar stride reduction has been performed on the GoogLeNet architecture as well.

The CaffeNet and GoogLeNet architectures can both be viewed simply as a set of convolutional layers followed by a set of fully connected layers, which is then followed by a softmax layer that will generate a one-hot class label output. Both networks are typically set to have 1000 outputs in the last fully connected layer, corresponding to 1000 classes. The last fully connected layer of both networks has been modified to have 2 outputs corresponding to the classes text and non-text.

5.5 Experiments and Results

We conducted 4 experiments using the COCO-Text-Patch dataset. Two experiments used the CaffeNet-TXT architecture. The first experiment used a preliminary dataset that was balanced using random sampling, and the second experiment used the final dataset that was balanced using texture analysis. The other two experiments used GoogLeNet-TXT, with the same two datasets.

The best average accuracy for the dataset that was balanced using random sampling was 90.1% (using GoogLeNet), while the best average accuracy for the dataset that was balanced using texture analysis was 90.9% (using CaffeNet). The results are summarized in Table 5.2.

The lower accuracy that was obtained using random sampling may be due to the lower proportion of non-text training samples having significant levels of texture. If a substantial number of low-texture training samples are used, then the final system may be biased in a way that favors low texture in order to receive the non-text label. This bias is reduced somewhat for the case that texture-based selection was used for balancing. This massive

Table 5.2: Evaluation results for the new COCO-Text-Patch dataset. Two methods of balancing were performed: textural analysis (left) and random selection (right), with the former yielding better accuracy values. Two architectures were considered: CaffeNet and GoogLeNet. The lower part of the table contains a confusion matrix for each of the four experiments.

Balancing	Texture Analysis				Random Sampling			
Network	CaffeNet		GoogLeNet		CaffeNet		GoogLeNet	
Accuracy	90.9%		90.2%		85.2%		90.1%	
	Text	Non-text	Text	Non-text	Text	Non-text	Text	Non-text
Text	0.868	0.057	0.838	0.042	0.856	0.054	0.729	0.022
Non-text	0.132	0.943	0.162	0.958	0.144	0.946	0.271	0.978

dataset has been used in training a powerful text classifier that became an integrated part of the proposed end-to-end system.

Chapter 6

Input Fast Forwarding: FFNet

This chapter will present a new concept, called *input fast-forwarding*, which results in improved performance for deep-learning systems. This was published in [24]. The approach utilizes parallel data paths that provide two advantages over previous approaches. One advantage is the explicit merging of higher-level representations of data with lower-level representations. A second advantage is a substantial reduction to the effects of the vanishing gradients problem. A modified version of this model will be used in the end-to-end system.

The main idea is to incorporate a parallel path that sends representations of input values forward to deeper network layers. This scheme is substantially different from "deep supervision," in which the loss layer is re-introduced to earlier layers. The parallel path provided by fast-forwarding enhances the training process in two ways. First, it enables the individual layers to combine higher-level information (from the standard processing path) with lower-level information (from the fast-forward path). Second, this new architecture reduces the problem of vanishing gradients substantially because the fast-forwarding path provides a shorter route for gradient backpropagation. In order to evaluate the utility of the proposed technique, a Fast-Forward Network (FFNet), with 20 convolutional layers along with parallel fast-forward paths, has been created and tested. The chapter presents empirical results that demonstrate improved learning capacity of FFNet due to fast-forwarding, as compared to GoogLeNet (with deep supervision) and CaffeNet, which are 4× and 18× larger in size, respectively. All of the source code and deep learning models described in this chapter will

be made available to the entire research community[1].

6.1 Introduction

Developments in deep learning have led to networks that have grown from 5 layers in LeNet [33], introduced in 1998, to 152 layers in the latest version of ResNet [17]. One consequence of deeper and deeper networks is the problem of vanishing gradients during training. This problem occurs as error values, which depend on the computed gradient values, are propagated backward through the network to update the weights at each layer. With each additional layer, a smaller fraction of the error gradient is available to guide the adjustment of network weights. As a result, the weights in early layers are updated very slowly; hence, the performance of the entire training process is degraded.

Many models have been proposed to overcome the vanishing-gradient problem. One approach is to provide alternative paths for signals to travel, as compared to traditional layer-to-layer pathways. An example of this approach is the Deeply-Supervised Network (DSN) [34], where a companion objective function is added to each hidden layer in the network, providing gradient values directly to the hidden layers. DSN uses Support Vector Machines (SVM) [8] in its companion objective function, which means that end-to-end training of the network is not supported. Another example is relaxed deep supervision [40], where an improvement over a holistic edge detection model [63] is made by providing relaxed versions of the target edge map to the earlier layers of the network. This approach provides a version of the gradient directly to the early layers. However, relaxed deep supervision is suitable only for problems where relaxed versions of the labels can be created, such as for maps of intensity edges. GoogLeNet [57] is another model that uses a mechanism to address the problem

[1] https://github.com/aicentral/FFNet

of vanishing gradients. The most similar model to FFNet is ResNet [17]. ResNet utilized "shortcut connections" to help deeper layers learn a residual mapping of the input. The main motivation for using the shortcut connection was the counterintuitive performance degradation of deeper networks as compared to shallower ones. A deeper network that performs at least as good as a given shallower one can be constructed simply by adding identity layers to the shallower network. The effectiveness of using shortcut connections in combating the performance degradation phenomenon was empirically demonstrated.

The novel approach that is proposed here provides parallel signal paths that carry simple representations of the input to deeper layers through what we call a fast-forwarding branch. This approach allows for a novel integration of "shallower information" with "deeper information" by the network. During training the fast-forwarding branch provides an effective means for back-propagating errors so that the vanishing-gradient problem is reduced.

To demonstrate the efficacy of the model, we created a 20 layer network with fast-forwarding branches, which we call FFNet. To study the effect of the fast-forwarding concept, the network layers are made of simple convolutional layers followed by fully connected layers with no additional complexities. The results that we have obtained using the the relatively small and simple FFNet model have been surprisingly good, especially when compared with the performance of bigger and more complex models.

6.2 Proposed Model: FFNet

The new FFNet model consists of convolutional units that are organized into a sequence of stages. Within each stage, as illustrated in figure 6.1, computations are performed in 2 parallel paths. The left branch in the figure represents a standard convolutional path, whereas the right branch represents an extra parallel data path. It is this parallel, "fast-

forwarding", path that delivers the improved performance of the network.

The input to the stage, $S1$, arrives from the previous layer, and the output to the next layer is shown as $S2$. The standard (deep) branch consists of three consecutive $3 \times 3 \times 64$ convolutional layers. Each layer is followed by an in-place Rectified Linear Unit (ReLU). The last layer of the deep branch is padded with zeros, for reasons that are described below.

Let the input $S1$ be of size $N \times N \times C$. The value of C is the number of channels, which is typically 128 except for the first stage where $C = 3$ to match the input data. Refer to a stage's deep convolutional layers as $S2C1$, $S2C2$, and $S2C3$, as shown in the figure. The deep branch's output $S2C3$ can be represented as follows, where $CONV$ is the convolutional operation, s is the stride, and p is the padding:

$$S2C3 = CONV_{3\times3,s=1,p=1}(CONV_{3\times3,s=1,p=0}(CONV_{3\times3,s=1,p=0}(S1))) \tag{6.1}$$

The size of $S2C3$ will be $(N-2) \times (N-2)$.

The fast-forwarding branch consists of a single $5 \times 5 \times 64$ convolutional layer followed by a ReLU. This branch takes $S1$ as input, and generates the output $B2C1$ that can be represented as follows:

$$B2C1 = CONV_{5\times5,s=1,p=0}(S1) \tag{6.2}$$

No padding is used for the fast-forwarding branch, so that the resulting output size is also $(N-2) \times (N-2)$. This branch will provide a "shallower" representation of the input $S1$ to the next stage.

The outputs of the deep branch and of the fast-forwarding branch are concatenated to create the single stage output $S2$. The size of $S2$ will be $(N-2) \times (N-2) \times 128$. Because the last layer of the deep branch is padded with zeros, both branches provide data of the same

size to the output. The major difference between ResNet [17] and FFNet is the residual representation used by ResNet versus the merged high-level and low-level representations of FFNet achieved by concatenating activations from the shallow branches and the deep branches. It may seem that the concatenated features will be summed when passed through the next convolutional layer, but this is a weighted summation where the weights are learned by the model. This weighted summation comes in contrast to the non-weighted summation used in ResNet. The weighted summation used in FFNet allows for an extra degree of freedom compared to ResNet.

To evaluate the fast-forwarding concept, we built a Fast-Forwarding Network (FFNet) that consists of 6 consecutive fast-forwarding stages followed by two fully connected layers plus an output layer, as shown in figure 6.2. The 6 fast-forwarding stages consist of a total of 18 convolutional layers, each of size $3 \times 3 \times 64$. The first layer of the two fully-connected layers consists of 400 nodes, while the second layer consists of 100 nodes.

6.3 Evaluation

To evaluate the performance of the proposed model, a number of experiments were conducted that compare FFNet to AlexNet, CaffeNet, and GoogLeNet. The publicly available datasets CIFAR-10 [30] and COCO-Text-Patch [23] were used in the evaluation, as described previously. FFNet was implemented using Caffe [27]. Standard 10-crop augmentation was applied to the datasets. All the training and testing were performed on a GPU with batch size 32. The training was stopped after 150,000 iterations as the validation accuracy and loss started to plateau.

A summary of results is provided in table 6.1. Despite its relatively small size, the performance of the proposed FFNet model exceeded the performance of CaffeNet and GoogleNet

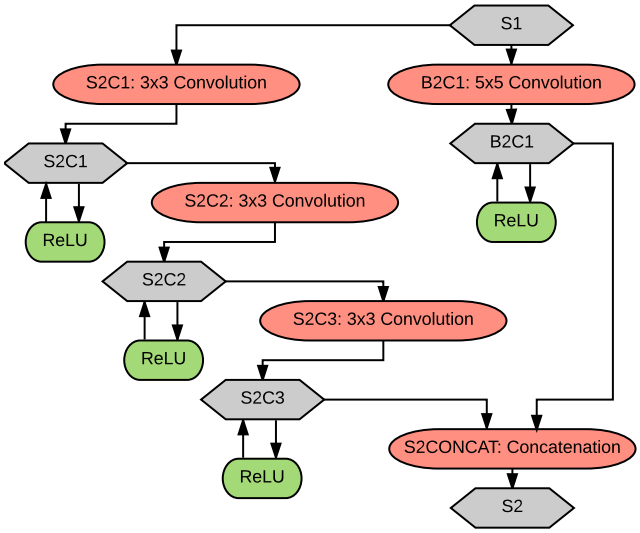


Figure 6.1: A single fast-forwarding stage. Node $S1$ represents the input, and $S2$ is the output. The left pathway contains common convolutional blocks. At the right is the fast-forward path.

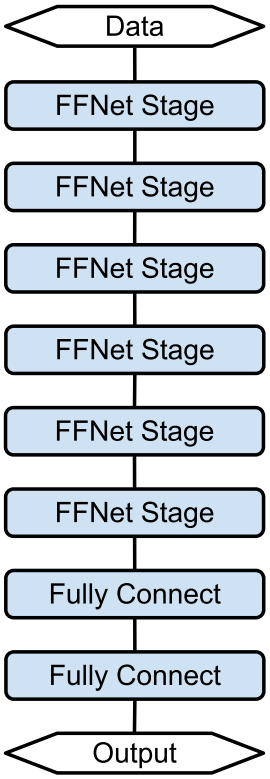


Figure 6.2: Proposed FFNet model. Because of fast-forwarding, this relatively small network has yielded empirical results that are better than much larger deep networks.

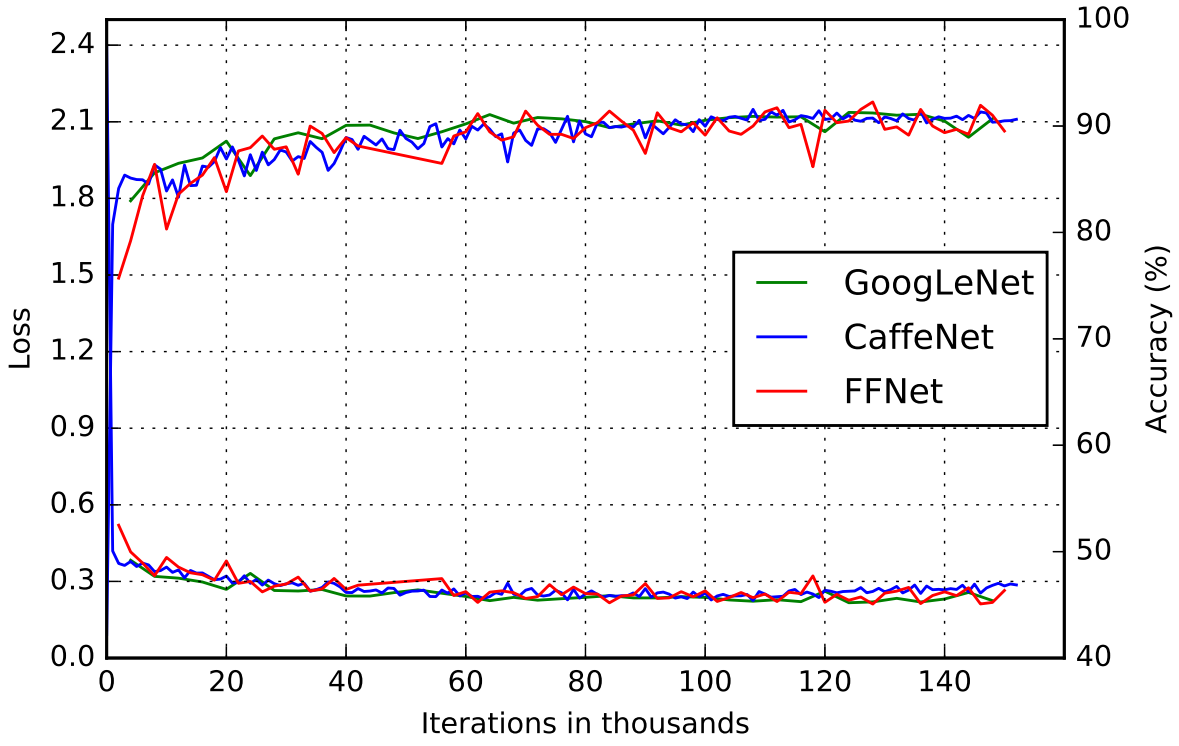


Figure 6.3: COCO-Text-Patch validation accuracy and loss for the proposed FFNet model (red), CaffeNet (blue), and GoogLeNet (green).

in these experiments. The accuracy and validation loss graphs shown in figure 6.3 demonstrate how the proposed model converges with the same speed as CaffeNet and GoogLeNet. These trends provide evidence of the effectiveness of the fast-forwarding approach in fighting the vanishing-gradient problem.

A modified version of this model will be used in the proposed end-to-end system as a part of the text attention model. Being able to learn complex tasks with minimal number of parameters enabled the creation of lightweight, yet, powerful text attention model.

Table 6.1: Performance comparison of the FFNet model with several common alternatives. Although FFNet is much smaller than the other models, its error rate was lower than the others (with one exception), using publicly available test sets.

Model				*Error Rate* (%)	
Description	Layers	Size (MB)	Time*(ms)	CIFAR-10	CTP**
AlexNet with dropout [31]	8	181.3	-	15.6	-
AlexNet with stoch. pooling [71]	8	181.3	-	15.3	-
AlexNet with channel-out [62]	8	181.3	-	**13.2**	-
GoogLeNet [23]	22	41.2	9.4	-	9.9
AlexNet [31], CaffeNet [23]	8	181.3	5	18.0	9.1
FFNet (the proposed model)	20	**10.8**	**2.8**	13.6	**9.0**

* Average forward path time per image on a K80 GPU
** CTP: COCO-Text-Patch dataset [23]

Chapter 7

Nested Auxiliary Branches: AuxNet

This chapter introduces a new architectural framework, built on top of the FFNet architecture, that has been shown to enhance the performance of deep networks. The central concept is to augment a pipeline of convolutional layers with a hierarchy of shallow, parallel data paths. These auxiliary paths branch and merge at regular intervals, and they enhance the training process in two primary ways. First, they enable the individual layers to combine higher-level information (from the standard processing branch) with finer details (from the auxiliary branches). Second, this new architecture substantially reduces the problem of vanishing gradients, because the parallel paths provide shorter routes for gradient back-propagation. Several experiments have been conducted to analyze improvements related to vanishing gradients by adding auxiliary branches.

In order to evaluate the utility of this architectural innovation, four models (collectively called AuxNet) have been created and tested. The first pair of networks contain 64 and 128 layers, with hierarchies of 3 and 4 nested auxiliary paths, respectively. The last two models consist of 1024 and 4096 layers, with correspondingly larger hierarchies of auxiliary paths, and they have been built to demonstrate the feasibility of training very deep networks using the AuxNet concept. To our knowledge, this last model represents the deepest convolutional network to date for which training has been performed effectively. In the empirical results presented here, the AuxNet architecture has exceeded state-of-the-art performance on several standard benchmarks (MNIST, CIFAR-10, and CIFAR-100). The section also compares and

contrasts the new AuxNet framework with other deep models, including Deep Residual Networks (ResNet) and Densely Connected Networks (DenseNet).

7.1 Introduction

The number of layers in deep networks has increased dramatically in recent years. LeNet [33], introduced in 1998, contained 5 layers, whereas ResNet [17] has grown to 1202 layers in its latest version. It is well known, however, that training is not a straightforward task with deeper models. This fact has led to many non-sequential designs that can be trained more effectively than a standard architecture. A primary reason is the problem of vanishing error gradients, which result from repeated application of the chain rule during the backpropagation procedure. Figure 7.1 provides a simple illustration of the problem using two networks that are identical except for the number of layers. Although both networks achieved approximately the same level of performance in the end, the deeper network required many more epochs of training for this particular task. More important than slower convergence, it is possible that a deeper network may completely fail to learn a given task, as we show later in the chapter. Not only a deeper network may exhibit a slower convergence than a shallower one, as we show later in the chapter, a deeper network may completely fail to learn a given task. This observation suggests the existence of a *maximum trainable depth* for a given network, beyond which the network may not learn at all, for a given training algorithm. According to He et al. [17], deeper models are harder to train as shown in figure 7.1. In this section, we will demonstrate how the vanishing gradient is the main problem in training such deep networks. A novel architecture will be proposed to overcome this problem.

This section introduces an architectural framework, called AuxNet, that improves the performance of deep convolutional networks. The fundamental idea is to incorporate additional

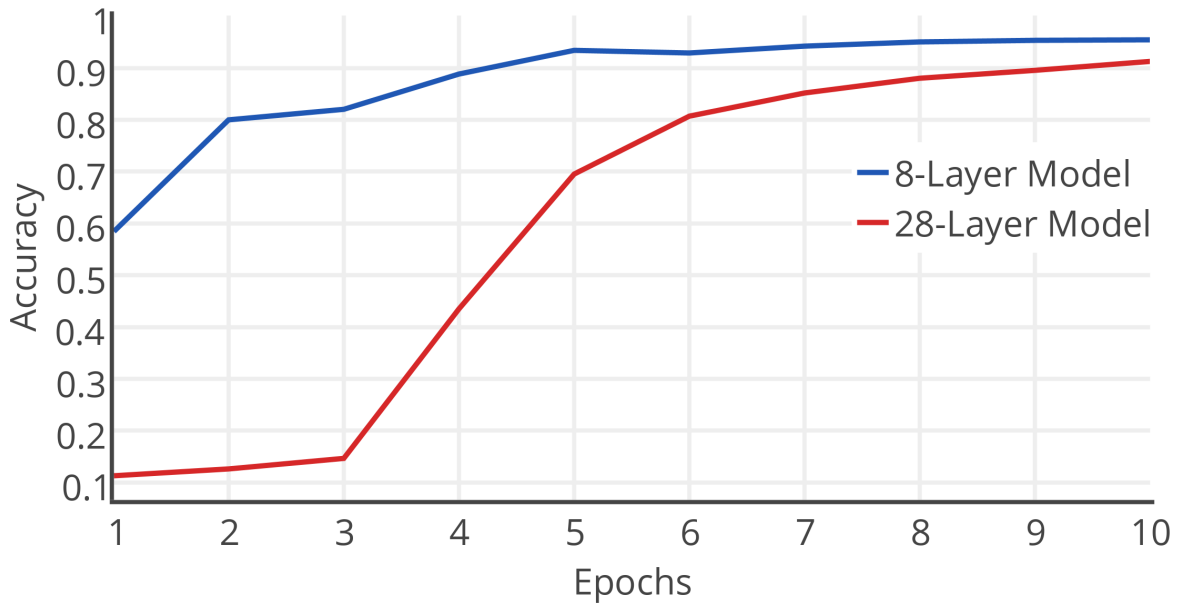


Figure 7.1: Comparison of learning rates for two standard convolutional networks that are identical except for the number of layers. Using the MNIST dataset [33] as a benchmark, the deeper (28-layer) network required many more epochs of training to achieve the same accuracy as the shallower (8-layer) network.

("auxiliary") shallow paths in parallel with the standard processing pipeline. Each "auxiliary" path contains a single convolutional layer. These parallel paths branch and merge with the standard pipeline at regular intervals, and they improve training performance in two principal ways. First, the design allows coarse and fine details to be combined repeatedly throughout the pipeline. Second, the problem of vanishing gradients is addressed through the auxiliary paths, which provide shorter routes for backpropagation of error. In contrast to other architectures in the literature, AuxNet guarantees that the shortest available path for gradient backpropagation grows only logarithmically with the number of layers. This allows for training extremely deep networks without making the shortest available path for gradient backpropagation exceed the maximum trainable depth of the network, and hence eliminating the vanishing gradient problem in such networks.

Figure 7.2 motivates the discussion further, as it compares two 32-layer networks during the first 25 epochs of training. (The CIFAR-100 dataset [30] was used for this comparison.) The AuxNet architecture, being introduced here, shows steady improvement in performance. A standard network, on the other hand, does not improve during the same amount of training time.

This chapter provides additional insights, both empirical and theoretical, into problems associated with the training of very deep networks. Several researchers have emphasized the importance of deeper networks. These include the work by the Visual Geometry Group (VGG) [55], as well as top-down modulation as introduced in [54]. Another relevant model is the Deeply-Supervised Network (DSN) [34], for which a companion objective function is added to each hidden layer in the network, providing gradient values directly to the hidden layers. DSN uses Support Vector Machines (SVM) [8] to provide those values, however. This approach does not allow for end-to-end training of the network. Another example is relaxed deep supervision [40], in which an improvement over a holistic edge detection model

[63] is made by providing relaxed versions of the target edge map to the earlier layers of the network. This approach provides a version of the gradient directly to the early layers. However, relaxed deep supervision is suitable only for problems where relaxed versions of the labels can be created, such as for maps of intensity edges.

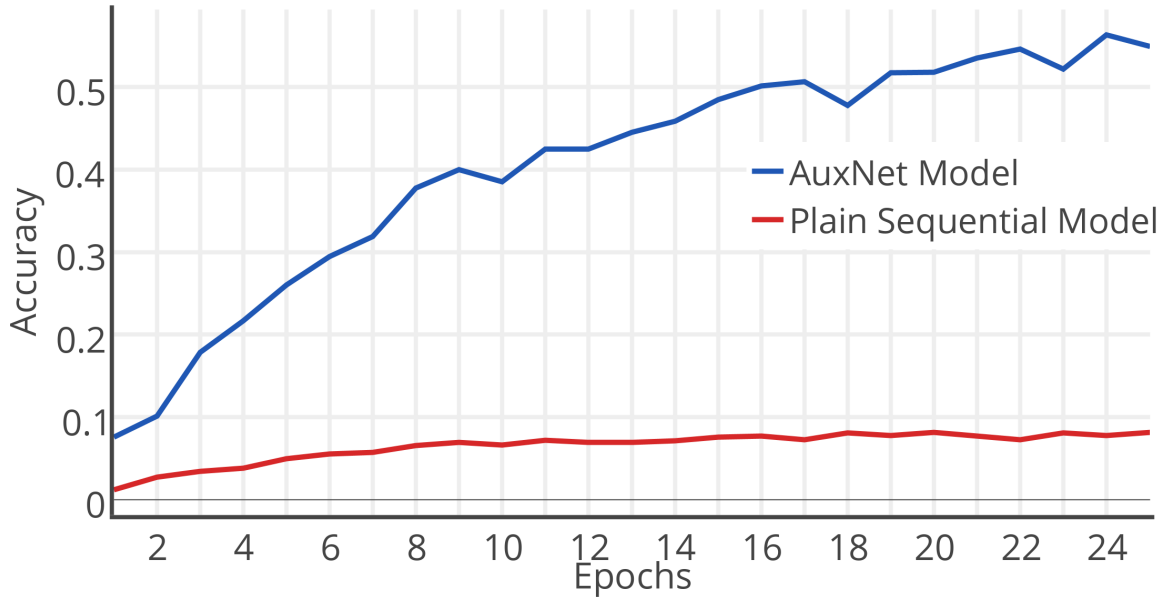


Figure 7.2: Comparison of learning rates for networks with and without the auxiliary parallel branches that are proposed here. Both networks are 32 layers in depth. The proposed AuxNet architecture exhibited steady improvement in accuracy, whereas training was ineffective for the standard network.

The AuxNet design was inspired by the building blocks of the FFNet model that has been discussed in chapter 6. A more detailed comparison between AuxNet and FFNet will be given in section 7.3. The closest model to FFNet and to the proposed AuxNet architecture is ResNet [17], in which "shortcut connections" are provided to skip one or more layers. The shortcuts help deeper layers learn a residual mapping of the input. We propose a model where nested branches of convolutional layers are used instead of the ResNet-style shortcuts. This chapter will show that the proposed nested auxiliary shallow branches are more effective

at combating the vanishing gradient problem. More details about ResNet stages will be given in section 7.2.3.

7.2 Related Work

7.2.1 GoogleNet and DSN

One approach to reduce the effect of vanishing gradients is to incorporate additional classifiers into the convolutional-network model. The purpose of these additional classifiers is to provide gradient information directly to earlier network layers. A well-known example is GoogLeNet [57], which was the winner of the ILSVRC 2014 competition [10] with a top-5 test error rate of 6.6%. The network consists of 22 layers with a relatively complex design called "inception." The inception module, which is used to implement the stages of GoogLeNet, consists of parallel paths of convolutional layers of different sizes concatenated together. The number of filters in the convolutional layers inside the inception modules ranges from 16 to 384. In addition to using the inception design, GoogLeNet uses three auxiliary classifiers connected to the intermediate layers during training. These auxiliary classifiers are used to combat the vanishing gradient problem. The auxiliary classifiers are smaller convolutional networks attached to earlier layers of the main network. A similar technique has been used by the Deeply Supervised Networks (DSN) [34], but, with SVM [8] as the auxiliary classifier.

7.2.2 Highway Networks

One of the early models that introduced the concept of skip connections is the Highway Networks [56]. This model provided a means to train end-to-end networks deeper than 100 layers. Deep networks could be trained through the use of bypassing paths along with gating

units. The authors claimed that the bypassing paths are the key factor that enabled the training of these deep models. This model was inspired by the long short-term memory (LSTM) architecture [14]. LSTM is a very successful recurrent neural model that uses a similar gating technique to enhance the model performance. The Highway Networks model uses a non-standard gating technique while the proposed model in this chapter uses only standard convolutional layers to build very deep networks.

7.2.3 ResNet

The 152-layer version of ResNet [17] was the winner of ILSVRC 2015 competition [10]. ResNet utilized "shortcut connections" to help deeper layers learn a residual mapping of the input. The main motivation for using the shortcut connection was the counterintuitive performance degradation of deeper networks as compared to shallower ones. A deeper network that performs at least as good as a given shallower one can be constructed simply by adding identity layers to the shallower network. The effectiveness of using shortcut connections in combating the performance degradation phenomenon was empirically demonstrated. Nevertheless, the argument behind using such connections did not provide an explanation to the phenomenon. Shortcut connections in ResNet were primarily introduced to learn residual functions, while batch normalization [25] was deployed to overcome the vanishing gradient problem. In this chapter, we take the opposite stance. In particular, we empirically show that the vanishing gradient problem may still occur even with batch normalization. Moreover, we show how the vanishing gradient problem can be effectively addressed by relying only on auxiliary branch connections, without using batch normalization.

7.2.4 Deep Networks with Stochastic Depth

This work [20] is simply an extra advanced regularization technique that can be used with the ResNet [17] model. The main idea is to randomly drop all the convolutional layers of one or more of the ResNet stages. When the convolutional layers of a ResNet stage are dropped during training, the skip connection is kept to maintain the network connectivity. They achieved impressive results on some of the standard benchmarks. The proposed model achieved comparable performance without using such advanced regularization technique.

7.2.5 Densely Connected Convolutional Networks

The DenseNet model [21] is an extreme example of combining lower level and higher level features. In a DenseNet block, each layer is connected to all the previous layers in the same block. Although the experiments demonstrated excellent classification performance on standard benchmarks, the model does not actually provide a solution to the vanishing gradient problem because it is mainly a set of stacked blocks. That was clear in their reported results with the deepest model having only 250 layers compared to 1024 and 4096 networks built using the proposed architecture. We will demonstrate later that in a stacked blocks models, the path available for the gradient backpropagation grows linearly with the number of layers. We propose a new architecture in which the shortest available path for gradient backpropagation grows only logarithmically with the number of layers in the network. Another issue with DenseNet is its size. Stacking features from all previous layers in the same block resulted in a huge model size (nearly 25M parameters). The proposed architecture obtained comparable results with networks that are less than half the size of DenseNet.

7.2.6 Residual Networks of Residual Networks

The Residual Networks of Residual Networks model (RoR) [72] is a parallel work that was not published till recently. Both AuxNet and RoR utilize an unusual hierarchy of nested skip connections. However, the 2 architectures are based on very different hypotheses. For AuxNet, the skip connections explicitly combine lower level and higher level representations (concatenation of activation maps), whereas RoR learns residuals of residuals (summation of activation maps). Our AuxNet model is based on the FFNet architecture [24], whereas RoR is based on the ResNet model [17]. We provided deeper analysis of the reason that nested skip connections help fight the vanishing gradient problem in very deep models. We provide equations for the shortest available path for backpropagation. We provided experimental proof that our AuxNet model can be used to fight the vanishing gradient problem in a very deep network: 4096 layers, which to our knowledge is the deepest model that has been trained and reported using standard datasets (CIFAR-10 and CIFAR-100). In contrast, the RoR paper presents no models that are deeper than the original ResNet. Furthermore, it seems strange that RoR best result on CIFAR-10 was for a wide, shallow model of 58 layers. The vanishing gradient problem is not a major issue for such shallow models. The RoR paper has therefore not demonstrated success for the vanishing gradient problem, as we have done with AuxNet. Also, AuxNet exceeds the performance of RoR when considering the RoR models trained using the same number of epochs used for training AuxNet. RoR achieved higher performance only using extended lengthy training.

7.2.7 FFNet Versus ResNet

The Fast-Forwarding (FFNet) model [24], discussed in chapter 6, consists of convolutional units that are organized into a sequence of stages. Within each stage, as illustrated in Figure

6.1, computations are performed in two parallel paths. One of these paths is a standard chain of multiple convolutional layers, whereas the other path is a single convolutional layer (the "fast-forwarding branch"). The outputs of the deep branch and of the auxiliary shallow branch are concatenated to create the single output of the stage.

Notice that a single ResNet stage can be formulated as $\mathcal{H}(x) := \mathcal{F}(x) + x$, where $\mathcal{F}(x)$ is the combination of the convolutional layers. Similarly, a single stage of the FFNet model can be represented as $\mathcal{H}(x) = \mathcal{F}_{\mathcal{D}}(x)||\mathcal{F}_{\mathcal{S}}(x)$, where $\mathcal{F}_{\mathcal{D}}(x)$ is the combination of the convolutional layers in the deep branch, $\mathcal{F}_{\mathcal{S}}(x)$ is the convolutional layer in the auxiliary shallow branch, and $||$ is the concatenation operation. In both models, the gradient flowing from the output to the early layers of the network would be less likely to vanish because of the skip-like branches, which provide shorter paths for the backpropagation process. However, a major difference between ResNet and FFNet is the residual representation used by ResNet versus the merged high-level and low-level representations achieved by concatenating activations from the shallow branches and the deep branches of FFNet. In this chapter, we will show that the auxiliary shallow branches in FFNet are more effective (and indeed more general) than the shortcut connections used in ResNet. The aspect of generality can be shown using a constructive argument: by setting the weights of the auxiliary shallow branch to perform identity mapping and setting the weights right after the concatenation layer to behave as if the two concatenated segments are added first, we can convert an auxiliary shallow branch into a shortcut connection. For this reason, we base our proposed model, AuxNet, on the FFNet model.

Although in both ResNet and FFNet, the vanishing gradient problem is subdued by providing a shorter path for gradient backpropagation, the two models are not scalable for very deep networks. As we demonstrate below for both models, the shortest path for gradient backpropagation grows linearly with the network's depth. In this chapter, we propose a new

architecture in which the shortest available path for gradient backpropagation grows only logarithmically with the network's depth.

7.3 Proposed Model: AuxNet

The proposed AuxNet model consists of a nested hierarchy of FFNet modules. Each AuxNet module consists of smaller AuxNet modules. The deepest level consists of FFNet modules, as shown in Figure 7.4, which is constructed of two parallel branches. The deep branch of the deepest level consists of 4 convolutional 3-padded convolutional layers, while the shallow branch consists of a single 3-padded convolutional layer. Max pooling as well as doubling the number of filters in each convolutional layer is done after each stage at the top level of the nesting hierarchy. To be able to merge the shallow branch and the deep branch at the top level of the nesting hierarchy, the activation size of both branches must match. To match the activation size of those branches, a number of max-pooling layers have been added to the shallow branch.

The following analysis will show how the proposed AuxNet architecture has much shorter backpropagation paths than the FFNet model. Let the total number of stages in a plain sequential model without any auxiliary branches be N, and let the number of layers in each stage be n. Then the total number of layers in sequence is $D = N \times n$, That will make the total number of layers D, which which is the path that the gradient should take from the output layer back to the first convolutional layer. On the other hand, for an FFNet model with shallow auxiliary branches, the first layer will have $n - 1$ layers on top of it in the first stage, then, $N - 1$ auxiliary shallow branches, in the $N - 1$ stages, on top of it, as marked red in Figure 7.3 The gradient can go through the shallow branches which will reduce the length of the shortest available path in the top $N - 1$ stages from $(\frac{D}{n} - 1) \times n$ in a plain

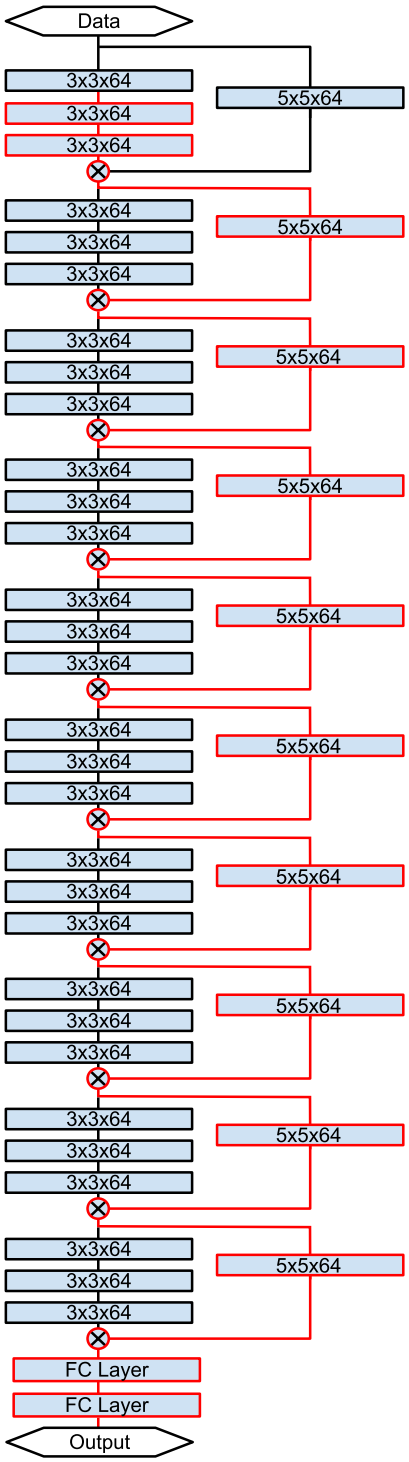


Figure 7.3: The FFNet model, extended to 32 layers. (The original model contained 20 layers [24].) The shortest available path is marked red.

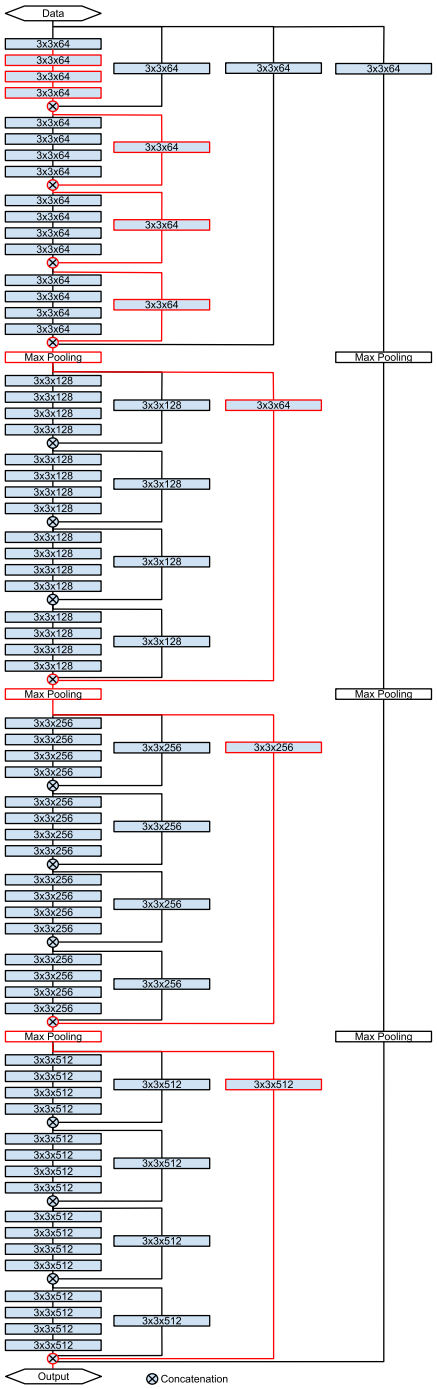


Figure 7.4: A nested AuxNet architecture with 4 convolutional layers at the deepest level, plus 3 levels of nested shallow branches. The total number of layers in the deepest path is $4^3 = 64$ layers. The shortest available path is highlighted in red.

network to only $\frac{D}{n} - 1$ in the FFNet model. That will make the shortest available path that the gradient can take be $(\frac{D}{n} - 1) + (n - 1)$. While this is a significant improvement over plain networks, the shortest available path for gradient backprropagation still grows linearly with the network's depth (D), assuming n is constant (with a much smaller constant though). This linear growth would prohibit training very deep networks without falling into the vanishing gradient problem.

To calculate the shortest available backpropagation path between the output and the first layer in AuxNet, let the number of layers in the deep branch of the deepest level be n and the levels of nesting be L. As shown in Figure 7.4, the total number of layers D can be calculated as $D = n^L$, and the number of stages would be $N = n^{L-1}$. The first layer of this model will have $n - 1$ convolutional layers on top of it in the deepest level, then, $n - 1$ layers in each of the next level of the hierarchy until the top level. This can be written as $(n-1) \times L$ which is equivalent to $(n - 1) \times (\log_n D))$. Therefore, the nested architecture reduced the shortest available path for gradient backpropagation from the output layer to the first layer from $\mathcal{O}(D)$ in both the plain model and the FFNet model (with a much smaller constant factor in the latter) to $\mathcal{O}(\log D)$ in the AuxNet model, assuming n is a constant.

To demonstrate the ability of building extremely deep models using the nested AuxNet architecture, 1024 and 4096 layers models have been built and trained, as will be shown in Section 7.4. As far as we know, the 4096 layers AuxNet model is the deepest model that could be trained to date. This model would have a shortest available path of $(4-1) \times (\log_4 4096)) = 18$, while it would have a shortest available path of 1003 with the FFNet model, and 4096 with a plain sequential model. A comparison between the shortest available path growth rates in AuxNet vs FFNet is shown in Figure 7.5, which demonstrates the superiority of AuxNet over FFNet in terms of the ability to train very deep networks. Figure 7.6 implies that an extremely deep AuxNet model of 1,000,000 layers could be trained if bigger and

more powerful GPUs become available in the future.

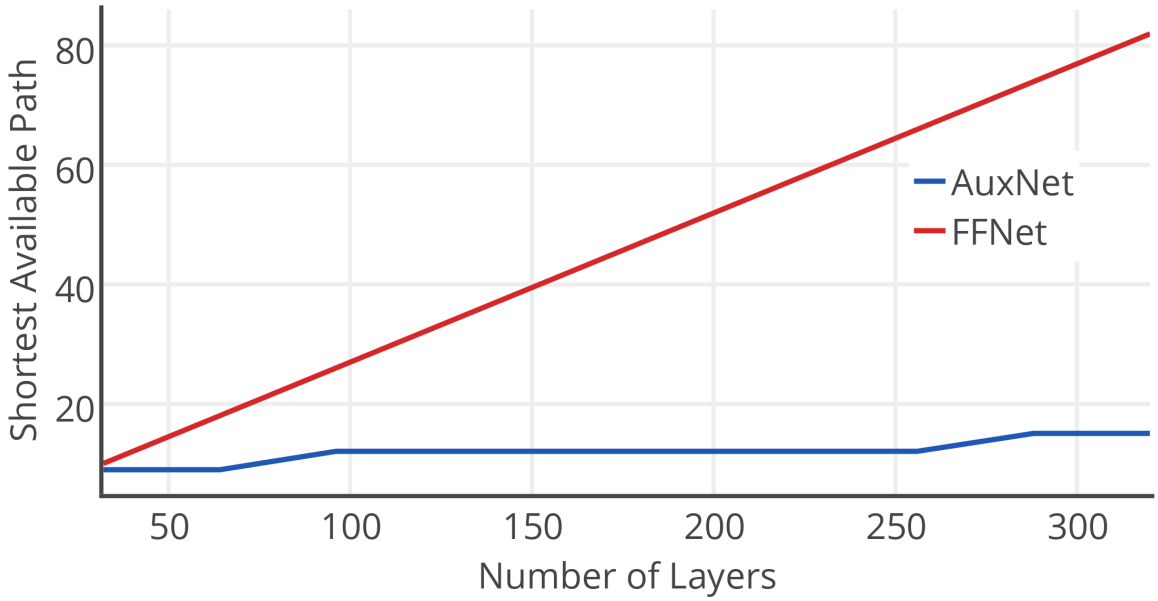


Figure 7.5: Shortest available path for backpropagation between the output and the first convolutional layer in the AuxNet and FFNet models. The number of layers per stage is set to $n = 4$.

7.4 Evaluation

Several experiments have been conducted with the purpose of analyzing and evaluating the proposed models. Four experiments have been designed to analyze the vanishing gradient problem and the efficiency of the proposed model in overcoming it. Another set of experiments were conducted to compare the performance of AuxNet with other deep models. All the models have been built using the Keras package [7] over TensorFlow [3]. TensorFlow is written with a Python API over a C/C++ engine, using GPU optimization libraries such as CuDNN [6].

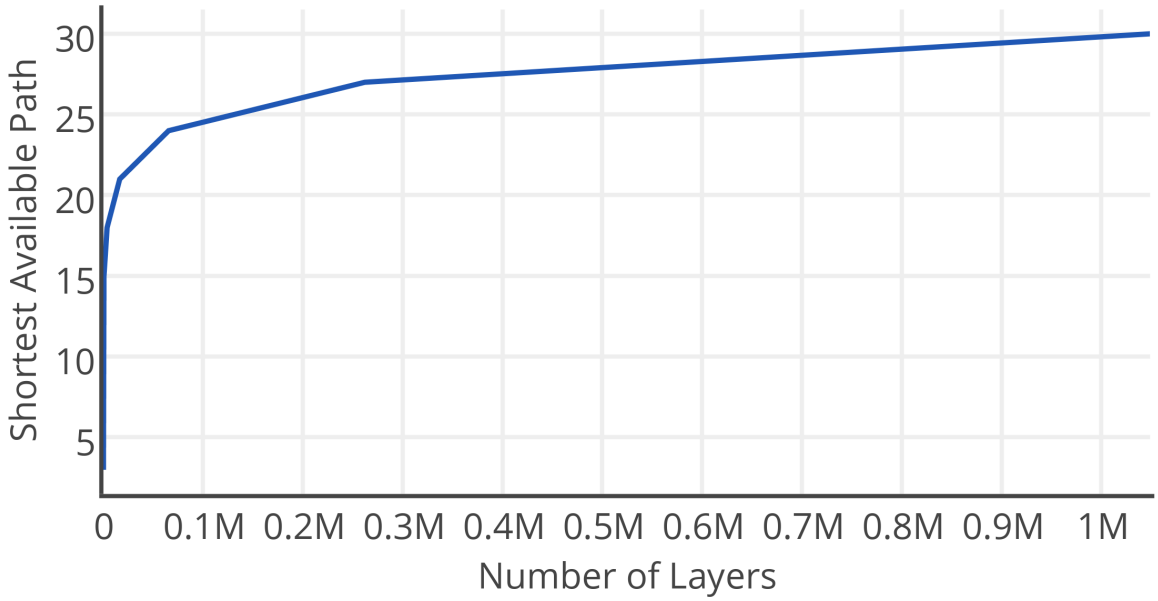


Figure 7.6: Shortest available path for backpropagation between the output and the first convolutional layer in the AuxNet models. The number of layers per stage n is set to 4. Although the graph implies the ability to train a million-layer model, this is not currently feasible with any available GPU.

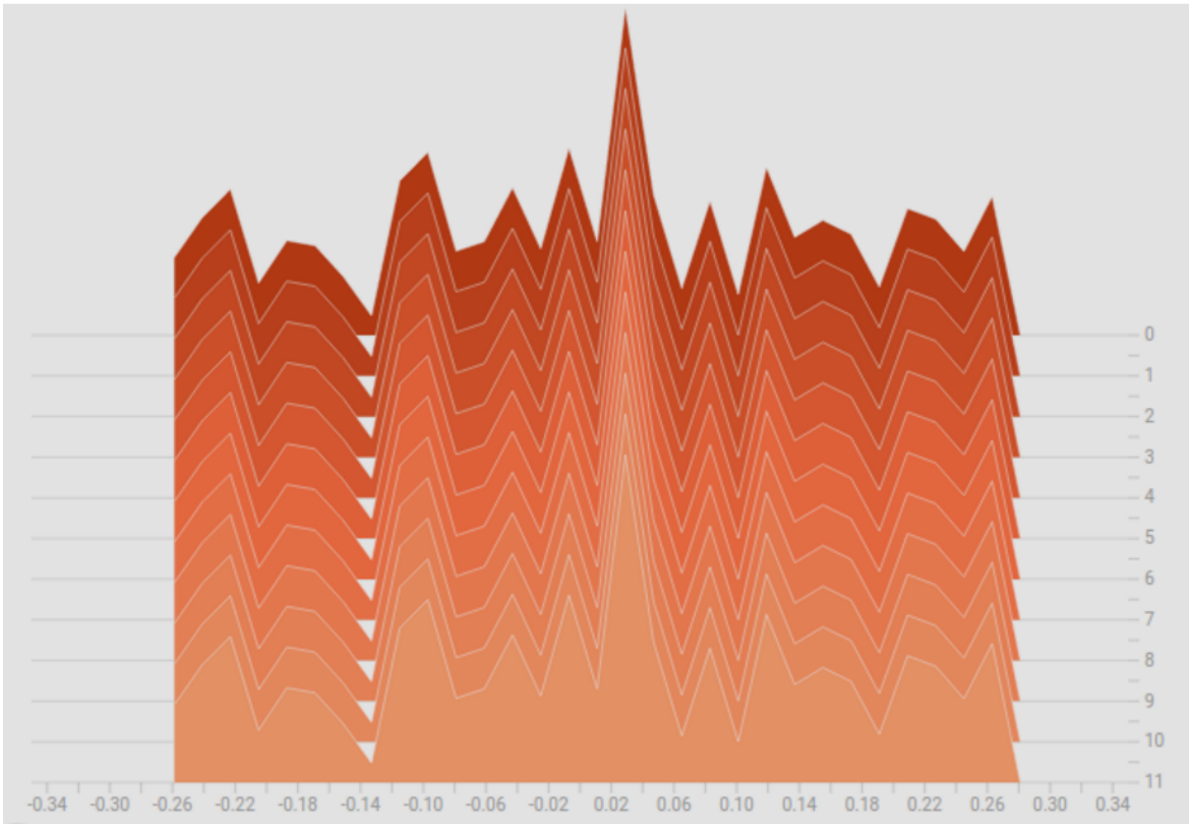


Figure 7.7: Histogram of weights, as generated by the Tensor Board tool, in the first convolutional layer of a plain 32 layers model trained for 12 epochs, marked on the chart as 0 to 11, using the MNIST dataset. Notice how the distribution of the weights remained constant over the course of 12 epochs

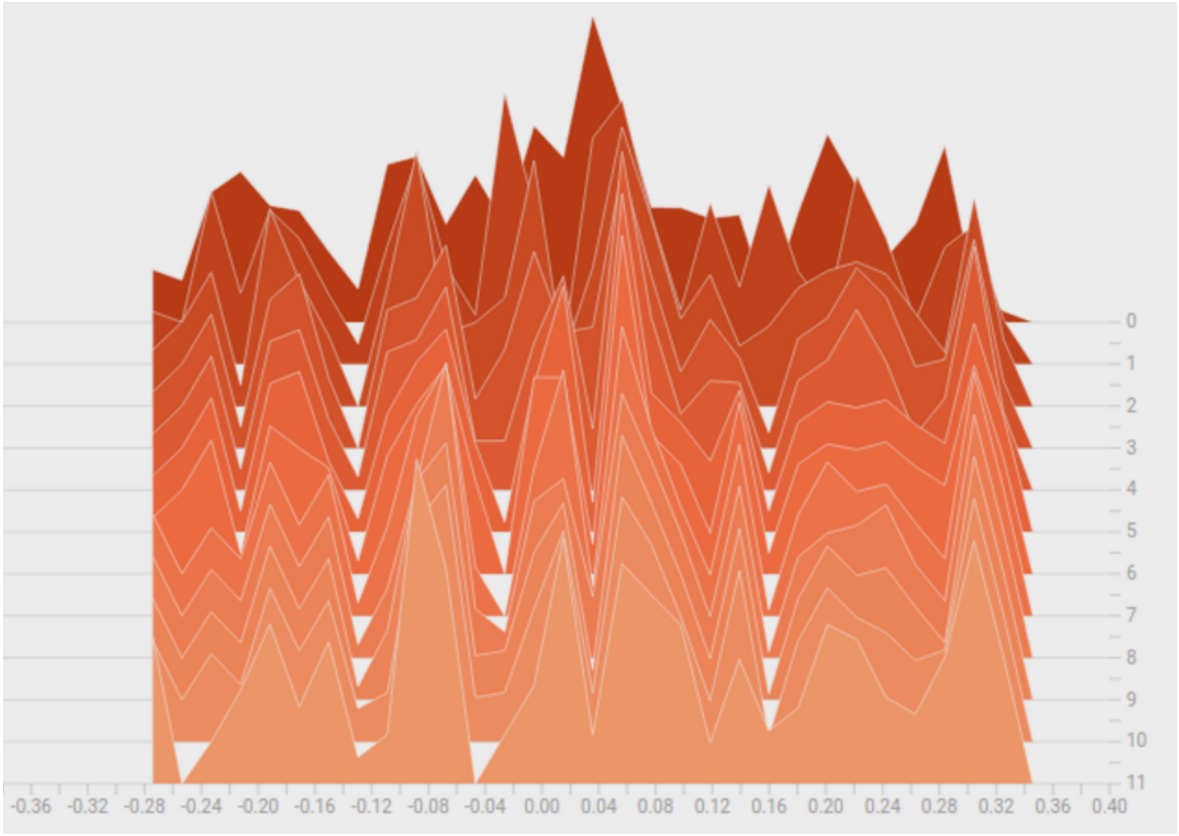


Figure 7.8: Histogram of weights, as generated by the Tensor Board tool, in the first convolutional layer of a 32 layers AuxNet model trained for 12 epochs, marked on the chart as 0 to 11, using the MNIST dataset. Notice how the weights are being updated from epoch to epoch

Four experiments have been conducted using the MNIST [33] dataset. The first one, is training a 32 layers plain sequential model with 30 convolutional layers plus a global average pooling layer and a fully connected layer. The purpose of this experiment is to monitor the weights of the early layers of the network and see if the gradient can back propagate through such deep network without vanishing. As shown in Figure 7.7, the weights of the first convolutional layer of this model remained constant over the course of 12 epochs. This lack of change in this early layer confirms the hypothesis of the vanishing gradient in deep networks.

The second experiment is identical to the first one except for the addition of the proposed auxiliary shallow branches to the network. In contrast to the first model, this updated model could learn the task. Figure 7.8 shows clearly how the weights are being updated from epoch to epoch. We also inspected the histogram of weights of several other layers from various deep branches and they all showed changes in the weights from epoch to epoch. The change in the weights indicates that the gradient could back-propagate through the newly added auxiliary shallow branches without vanishing.

The third experiment involved training 50 plain networks using both the MNIST and CIFAR-10 datasets. The purpose of this experiment was to investigate the effect of the network depth on the learning behavior of such simple task as MNIST. The experiment started by training a single hidden layers model for 10 epochs, once with batch normalization and once without batch normalization, then save the validation accuracy. The next step was to increase the number of layers by one, re-initialize the weights of the network, and re-train from scratch for 10 epochs. The number of layers has been increased one layer at a time till the last training session where the number of hidden layers was 50. As shown in Figure 7.9 and 7.10, networks with 20 layers or more started to suffer from performance degradation. Also, networks with 30 layers or more stopped learning completely. Batch normalization did not help much with

solving the vanishing gradient problem, which is against the observations reported in the analysis of the ResNet model [17]. Figure 7.1 demonstrates how the learning process is much slower in a 28-layer model compared to an 8-layer one. This can be understood by how the gradient in shallower networks can easily reach the early layers, which enables a faster learning rate compared to deeper networks, where the nearly vanished gradient slows down the learning process, and could even block the learning completely in the deeper models.

The fourth experiment involved a harder task which is the CIFAR-100 [30]. Two models have been trained for 25 epochs. the first is the FFNet model and the second is the same network with the auxiliary shallow branches removed. Figure 7.2 demonstrates how the removal of the auxiliary shallow branches disabled the learning process, leaving the model with accuracy close to the random chance.

The next set of experiments measures the performance of the proposed model on other datasets. A summary of the results of these experiments comparing them to other deep models are shown in Table 7.1. The experiments used a learning rate scheduler that reduced the learning rate by a factor of $\sqrt{0.1}$ of the validation accuracy plateau for 4 epochs. The number of epochs is different from one experiment to another, as training was stopped automatically if the validation accuracy did not increase for 10 epochs.

This set of experiments can be divided into two groups. the first group consists of two FFNet models with 20 layers and 32 layers, respectively. The second group consists of four AuxNet models with 64, 128, 1024, 4096 layers, respectively. To our knowledge, the 4096-layer model is the deepest convolutional network to date for which training has been performed effectively.

Some additional experiments have been conducted to compare the usage of the concatenation operation versus the summation operation to merge the activations from the deep branch

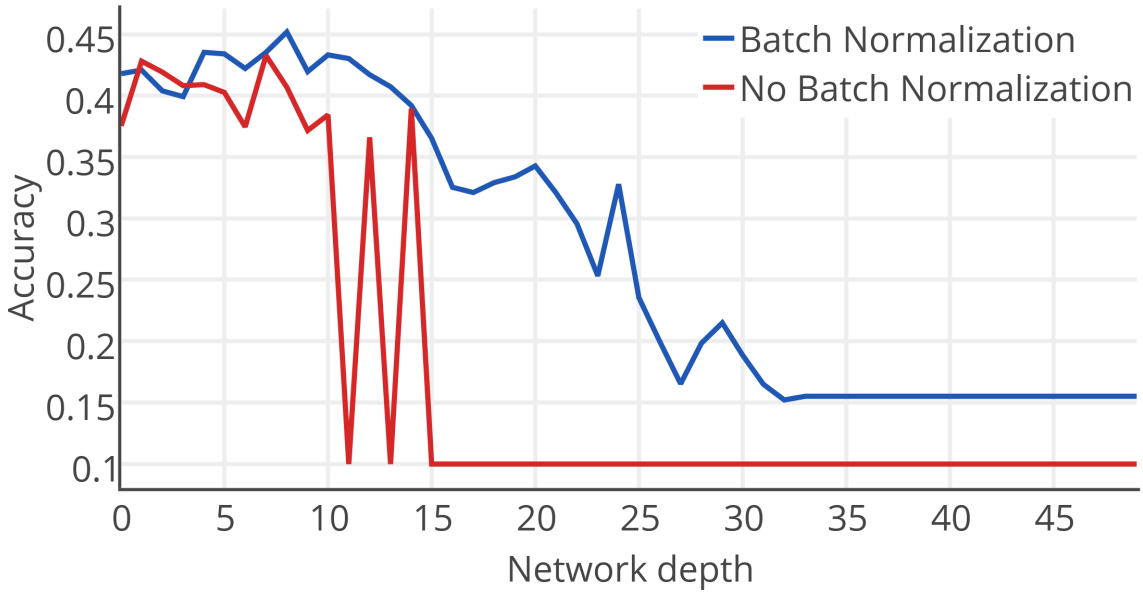


Figure 7.9: Validation accuracy vs. number of layers of a plain network trained for 19 epochs with and without batch normalization on the CIFAR10 dataset.

with the activation from the auxiliary branch in the FFNet model. The results showed degraded performance in case of the ResNet-like summation operation compared to the concatenation operation used in the proposed model. Also, note how the AuxNet models of different depth reached better error rates compared to ResNet models of comparable depth, such as AuxNet-1024 when compared to ResNet-1202. Although AuxNet-1024 contains about 8 million parameters, it is performing better than ResNet-1202 which has over 19 million parameters. Note how the deeper models perform better in complex tasks such as CIFAR100, while they overfit when trained on easier tasks due to the lack of heavy regularization techniques. Adding heavy regularization will create better performance for the deeper models. We decided not to use heavy regularization in order to prove the effectiveness of using the nested auxiliary branches alone in fighting the vanishing gradient problem.

The AuxNet architecture is suitable for training ultra deep models. But we decided to use

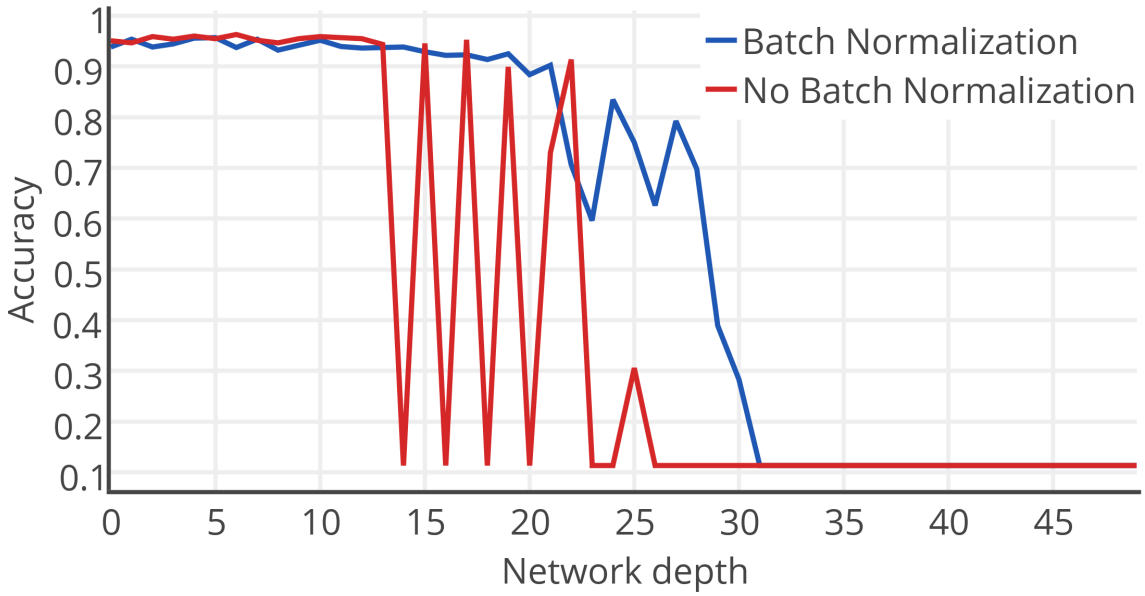


Figure 7.10: Validation accuracy vs. number of layers of a plain network trained for 12 epochs with and without batch normalization on the MNIST dataset.

FFNet in the proposed end-to-end text detection model for its small size. Adding nested auxiliary branches for a shallow model will only increase its size without enhancing the performance much.

Table 7.1: Performance comparison for the proposed AuxNet model, with hierarchical nesting, against several alternative networks. The error rate for AuxNet was better than the state of the art, for three publicly available datasets.

Model	*Error Rate* (%)			
Description	MNIST[1]	CIFAR-10[2]	CIFAR-100[2]	SVHN[3]
AlexNet with Stoch. Pooling [71]	0.47	15.3	42.51	-
AlexNet with Channel-Out [62]	-	13.2	36.59	-
Highway Network [56]	0.45	7.72	32.39	-
Deeply Supervised Network [4]	0.39	7.97	34.57	1.92
Network in Network [37]	0.47	8.80	35.65	2.35
FitNet [51]	0.51	8.39	35.04	2.38
FitNet with LSUV [43]	0.38	6.06	27.66	-
All-CNN [51]	-	7.25	33.71	-
ResNet-34 [17]	-	7.51	-	-
ResNet-110 [17]	-	6.43	27.67	1.75
ResNet-1202 [17]	-	7.93	-	-
RoR-3-WRN58-4+SD [72]	-	-	-	1.59
RoR-110 (after 164 epochs) [72]	-	5.71	26.16	-
FFNet: 20 Layers [24]	1.45	8.60	35.65	3.58
Sequential: 32 Layers	90.10	88.90	99.10	90.80
FFNet: 32 Layers	1.30	8.20	33.10	3.17
AuxNet: 64 Layers	**0.10**	**5.50**	27.60	**1.54**
AuxNet: 128 Layers	0.31	7.50	**25.60**	3.03
AuxNet: 1024 Layers	0.24	5.86	25.76	2.11
AuxNet: 4096 Layers	0.53	7.42	26.13	3.61

[1]LeCun et al. [33] [2]Krizhevsky and Hinton [30] [3]Netzer et al. [45]

Chapter 8

The End-to-End System

This chapter provides the details of the proposed end-to-end system. A higher level diagram of the proposed system is shown in figure 8.1. The proposed system consists primarily of two models as shown in figure 8.2. The first model is the attention map fusion where a text attention map is created using a fully convolutional network (FCN) based on the FFNet architecture. The text attention map is then fused with the input image. The output of this fusion process goes to the next model. The second model of the proposed system is a Faster-RCNN based text prediction model conditioned to generate word prediction boxes. The Faster-RCNN conditioning includes doubling the extracted feature spatial resolution in addition to adjusting the number of object types to only one type, text. This increase in the spatial resolution was necessary to enable the detection of small text instances. Details about the different models of the proposed end-to-end system will be given in the following sections including the text attention model, the fusion mechanism and the prediction model.

Transfer learning has been used extensively in the development of the new system. During the first phase of training, a network based on FFNet, that was discussed in chapter 6, was trained from scratch using a rich text classification dataset. Transfer learning was then used to initialize the second phase of training, in which a preliminary version of the attention model, that was discussed in chapter 4, was created. Transfer learning was applied again to initialize the third phase of training, in which the attention model was enhanced along with training of the prediction model which has been pre-trained, separately, on one of the

text detection datasets. One result of this strategy is an attention model that is lightweight but effective, as it incorporates strong text-related feature extractors. Empirically, we found that the transfer learning strategy used here was more effective than transfer learning from models previously trained on ImageNet [53]. Apparently, text instances in natural scene images are not closely correlated with ImageNet's more general object categories.

8.1 Text Attention Maps

Creating the text attention maps can be viewed as a semantic segmentation task where the input is RGB images and the target is binary masks that represent the text areas inside those images. Semantic segmentation is inherently difficult task that requires plenty of training data and huge models. One of the most popular semantic segmentation models is the one created by Long et al. [41]. This model contains nearly $134M$ parameters. Adding such a huge model to the proposed system will introduce a performance bottleneck. To make the text attention map generation lightweight, we used a clever learning transfer scheme to enable the use of a small model in performing such difficult task. Using this scheme, we could train a model that is nearly 5% of the model size in [41].

As shown in Figure 8.2, a classification model has been trained on the COCO-Text-Patch[23] dataset. This training created convolutional layers that are tuned to extract text-related features. Transferring this knowledge to the semantic segmentation model enabled the training of a very small model that performed the required task. We created a control experiment to check the necessity of the pre-training on COCO-Text-Patch. In this control experiment, a model pre-trained on the well-known Imagenet dataset [53] was used instead of the model pre-trained on COCO-Text-Patch. Although models pre-trained on the Imagenet dataset are known to be strong feature extractors, this control experiment showed that using a model

pre-trained on COCO-Text-Patch yielded a better performance.

8.2 Attention Map Fusion

The final phase of building the proposed end-to-end text detection model is attaching the text attention module by fusing the attention map into the input image as shown in Figure 8.3. Let the input image be $I \in \mathbb{R}^{WH}$, Where W and H are the width and height of the input image, respectively. Then, $AM \in \mathbb{R}^{WH}$, the attention map generated by the attention module, $\mathcal{F}$, would be

$$AM = \mathcal{F}(I) \tag{8.1}$$

The output from the attention module is fused into the input image to create a new enhanced representation, $\hat{I}$, such as

$$\hat{I} = AM \odot I \tag{8.2}$$

where $\odot$ is the element-wise dot product operation. The system output is a collection of bounding boxes expressed as coordinates. Those coordinates are normalized with respect to the width and the height of the input image. Each bounding box is expressed by the coordinates of its top-left corner, $x_1, y_1 \in \mathbb{R}$, and the coordinates of its right-bottom corner, $x_2, y_2 \in \mathbb{R}$.The collection of word box predictions, the expected output from the system, can be expressed as

$$P = \{p_i \in \mathbb{R}^4 | i = 1, 2, \ldots, n\} \tag{8.3}$$

where n is the number of predicted bounding boxes which is variable from image to image. Let the Faster-RCNN detection module be $\mathcal{D}$. Then, from (8.1) and (8.2), the whole system can be expressed as

$$P = \mathcal{D}(\mathcal{F}(I) \odot I) \tag{8.4}$$

In the next chapter, a detailed description of the steps implemented toward training and evaluating the proposed end-to-end system will be given. Those steps include how multi-phase learning transfer has been used to build a powerful attention model.

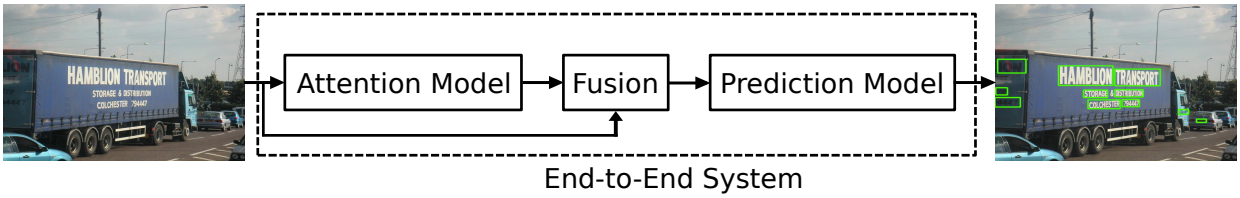


Figure 8.1: A high level diagram of the proposed end-to-end text detection system.

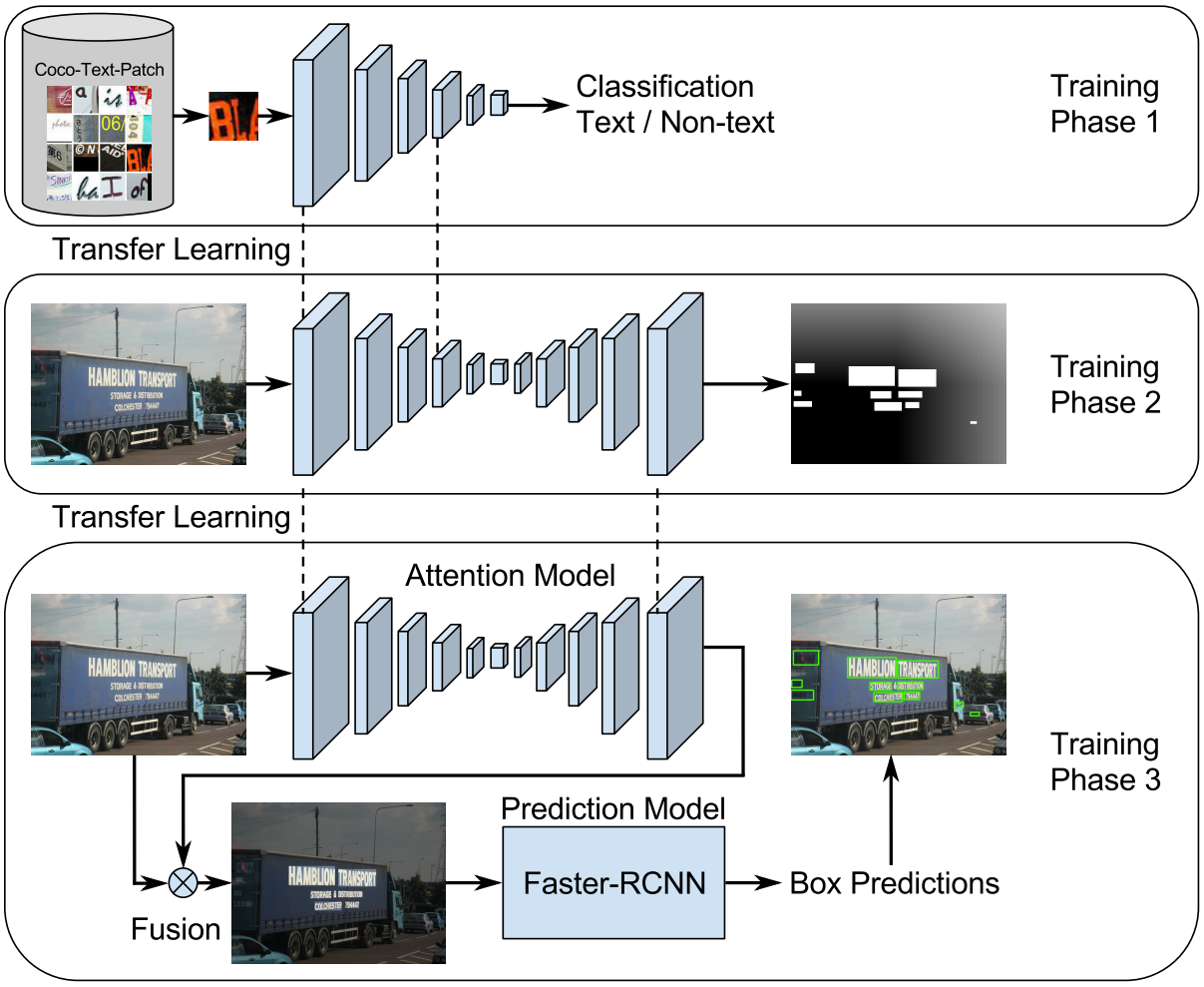


Figure 8.2: The proposed end-to-end system, shown at the bottom, consists of an attention model, a fusion step, and a prediction model. Transfer learning is used to progressively train larger models over three training phases.

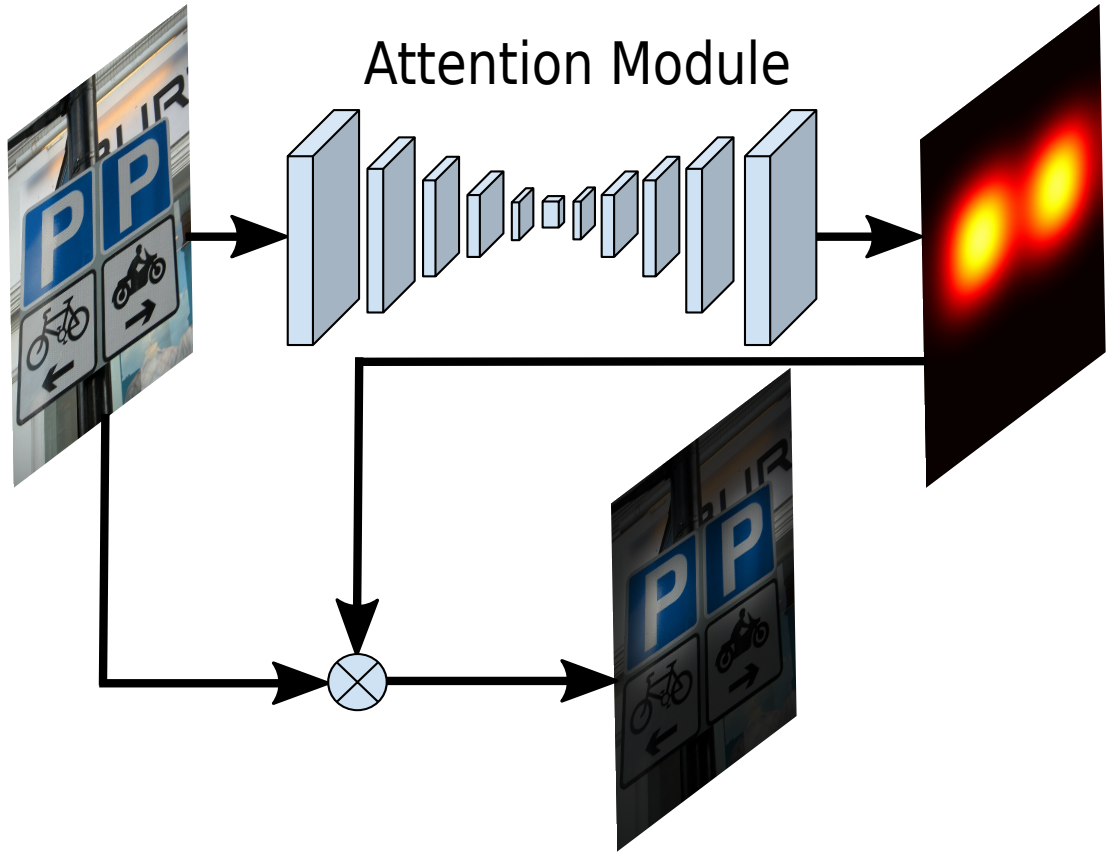


Figure 8.3: The fusion process: The input image is passed through the attention module to create an attention map. This attention map is fused into the input image.

Chapter 9

Evaluation and Experimental Results

9.1 The Prediction Model Evaluation

9.1.1 Detection Models

A study on the speed/accuracy trade-offs of modern object detectors [22] showed that Faster-RCNN [50] is the model with the highest detection accuracy among all the investigated models. The investigated models included also the Single Shot Multi-Box Detector (SSD) [39] and the Region-based Fully Convolutional Networks (RFCN) [9] model. This study by Huang et al. [22] evaluated the models using standard object detection benchmark datasets. Because text detection is a different task, we decided to make an exploration study to evaluate the performance of those three object detection models against ICDAR 2013, the standard text detection benchmark.

SSD performs object detection in standard RGB images using a single deep neural network. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. The SSD model is simple relative to methods that require object proposals, because it

completely eliminates proposal generation and subsequent pixel or feature re-sampling stage and encapsulates all computation in a single network. But, this simplification comes with a sacrifice of performance.

The Region-based Fully Convolutional Networks (RFCN) method which is very similar to Faster R-CNN, but instead of cropping features from the same layer where region proposals are predicted, crops are taken from the last layer of features prior to prediction. Pushing cropping to the last layer reduces the amount of per-region computation that must be done. Another difference between RFCN and Faster-RCNN is the position-sensitive cropping mechanism that is used in the RFCN model instead of the more standard ROI pooling operations used in the Faster-RCNN model.

The proposed model uses Faster-RCNN as its detection module. Faster-RCNN introduced a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and object class scores at each position. Box proposals are used to crop features from the same intermediate convolutional feature. Those cropped features are passed to two fully connected layers in order to create a class and an adjusted class-specific bounding box for each proposal.

9.1.2 Exploration Study

To reach the decision of using the Faster-RCNN as the text prediction model in the proposed end-to-end system, we conducted an empirical study to evaluate the top performing detection models. According to [22], those models are SSD [39], RFCN [9], and Faster-RCNN [50]. Three experiments have been conducted to evaluate the performance of these models when used as text detectors. In these three experiments, standard SSD, RFCN, Faster-RCNN

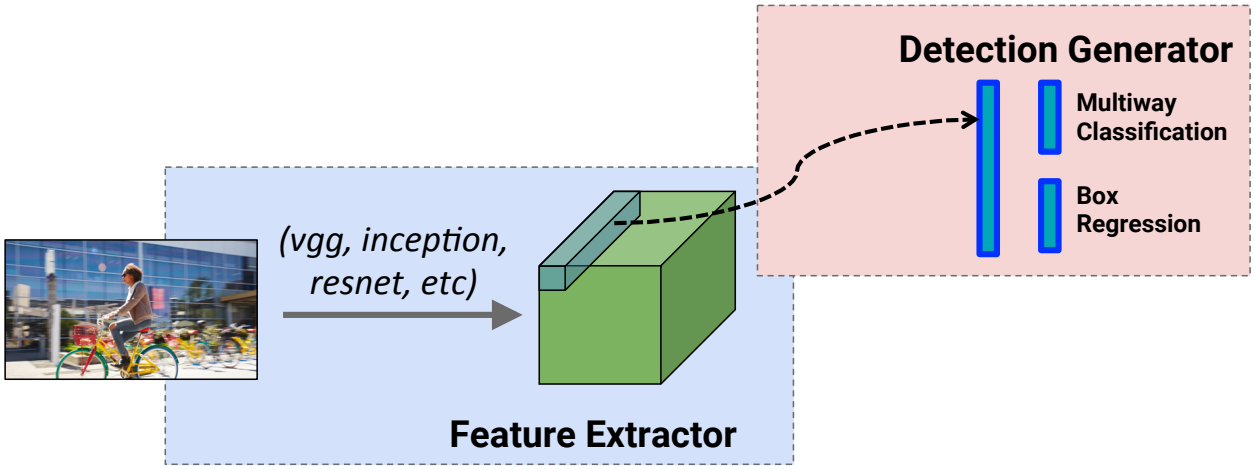


Figure 9.1: High level diagrams of SSD. Figure from [22].

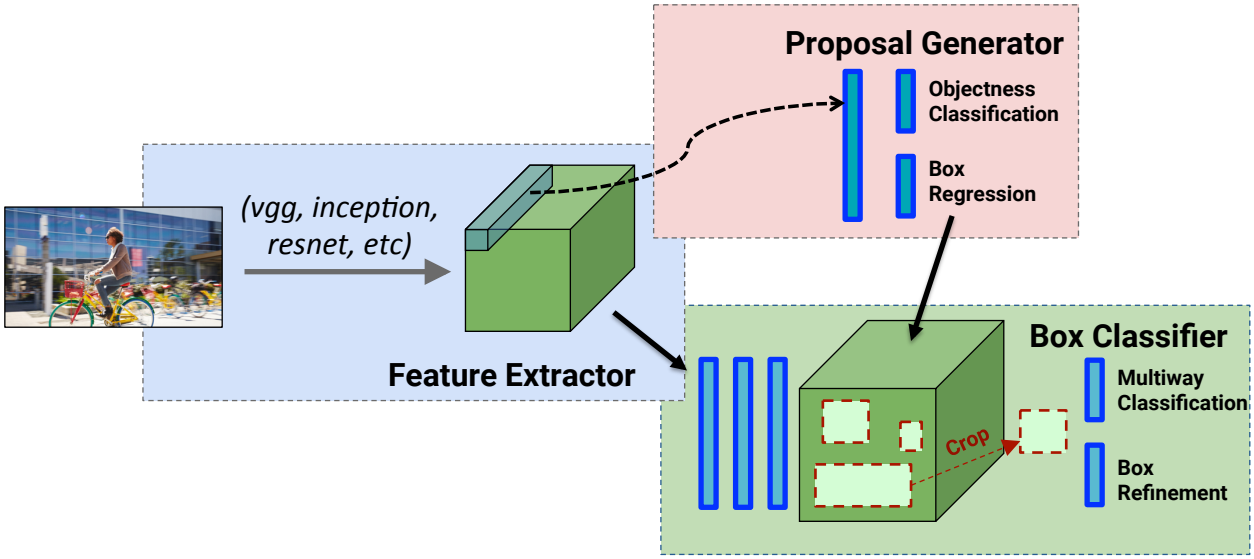


Figure 9.2: High level diagrams of RFCN. Figure from [22].

networks have been trained using the SynthText in the wild dataset and evaluated using the ICDAR 2013 dataset. The empirical results showed that the Faster-RCNN is the best detection model. These results are consistent with the results that came from the experiments on general objects detections in [22]. In the final experiment, the Faster-RCNN network has been plugged into the end-to-end system. The results of this exploration study have been

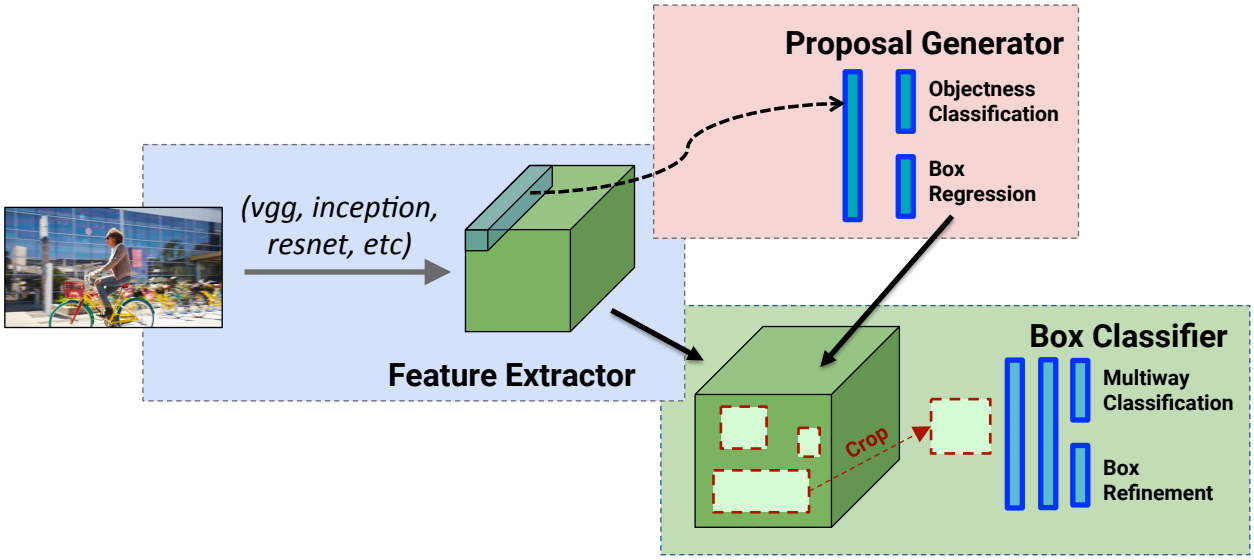


Figure 9.3: High level diagrams of Faster-RCNN. Figure from [22].

summarized in table 9.1.

Table 9.1: Comparisons of the explored models evaluated on the ICDAR 2013. The results are reported in the terms of Recall (R), Precision (P) and F-measure (F)

Model	R	P	F
SSD detector	0.54	0.67	0.61
RFCN detector	0.68	0.74	0.71
Faster-RCNN detector	**0.69**	**0.75**	**0.72**

9.2 The Attention Model Evaluation

To evaluate the effect of the attention model on the end-to-end system, three experiments were conducted. In the first experiment, the attention model is removed from the end-to-end system. The second experiment is the proposed end-to-end system without any modifications where the attention map is fused into the input image using pixel-wise dot

product. In the third experiment, the ground truth masks are used instead of the attention maps. Fusing the ground truth masks into the input images will completely remove the background in contrast to just attenuating the background in the second experiment. The third experiment is designed to compare the effect of background removal against background attenuation. Figure 9.4 shows example outputs from the attention model for each of the three experiments. In these three experiments, the three systems have been trained using the SynthText in the wild dataset and evaluated using the ICDAR 2013 dataset. The results, shown in table 9.2, demonstrate how the attention model used in the proposed end-to-end system performs much better than the system that did not use any attention information or the system that removed the background completely.

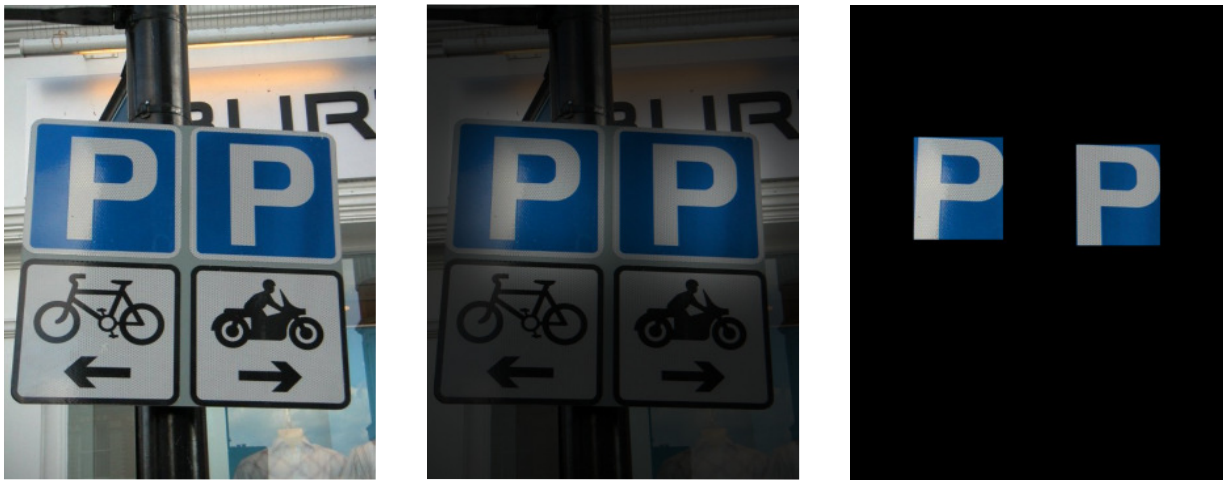


Figure 9.4: From left to right: An example image without any attention fusion, after fusing with the text attention map, and after fusing with ground truth. Those three images are example inputs to the prediction model for each of the three experiments.

Table 9.2: Comparisons of the proposed system using different attention techniques evaluated on the ICDAR 2013. The results are reported in the terms of Recall (R), Precision (P) and F-measure (F).

System	R	P	F
Without attention (plain input)	0.69	0.75	0.72
With extreme attention (background removal)	0.67	0.69	0.68
With attention fusion (the proposed technique)	**0.83**	**0.87**	**0.85**

9.3 The End-to-End System Evaluation

Training the proposed model has been done over three phases as illustrated in figure 8.2. For each phase, a different task with different dataset has been used. TensorFlow [3] has been used to create the end-to-end system. We used the Virginia Tech clusters Huckleberry and Newriver to run these experiments. The details of creating and evaluating the proposed system will be discussed in this section.

The first phase is training the text classifier. We created a 12-layer model based on the FFNet architecture [24] that was shown to achieve good performance with small models. We modified the FFNet architecture by removing the fully connected layers and one of the convolutional layers. Removing the fully connected layers was necessary to create a fully convolutional network in order to serve our system. The classifier has been trained for 70 epochs and achieved an error rate of 10.4% on the validation set of COCO-Text-Patch. The architectural modification did not result in much loss of accuracy compared to the original FFNet model, which achieved a slightly better error rate, 9%.

The second phase of the training used the SynthText in the wild dataset to train a semantic segmentation fully convolutional network. The dataset provides bounding box annotations which are not suitable for training a semantic segmentation model. We created text masks corresponding to the provided bounding boxes to serve as the targets for this network. Train-

ing a semantic segmentation network initialized with random weights will require creating a bigger network, which is not desirable, and requires much more data than what is available. Initializing this network with the weights of convolutional layers that have been pre-trained on ImageNet resulted in failure to learn. We could identify that easily from the behavior of the network after the loss curve plateau. The model generated the same output for any supplied input. This is a known behavior for semantic segmentation networks that fail to learn. On the other hand, when initializing the network with the weights of convolutional layers that have been pre-trained on COCO-Text-Patch, the network was able to learn the task and create output maps that are relevant to the location of the text instances inside the supplied input images.

The third phase of the training used the trained semantic segmentation network that has been created in the second phase as the text attention map model as shown in figure 8.2. For this phase, we used a Faster-RCNN model [50] that has been pre-trained on the COCO Object detection dataset [38]. We made a few changes to the Faster-RCNN model to condition it to the text detection task. First, we used the Inception-V4 model [58] model as the feature extractor for the Faster-RCNN. Second, the number of classes that Faster-RCNN detects has been changed to only one, text. Finally, we changed the spatial resolution of the extracted features from 16 to 8 as suggested by [22] in order to capture finer details that will enable the detection of smaller text instances, as demonstrated by the output samples from the COCO-Text test set shown in figures 9.5, 9.6 and 9.7. The samples demonstrate how the system could detect extremely hard to detect instances in addition to text instances of different types and different scripts. More samples are given in appendices A and B. Table 9.3 compares the performance of the proposed system against the state-of-the-art models on the ICDAR 2013 benchmark. The outstanding performance of the proposed system on the challenging COCO-Text dataset is shown in table 9.4. If bigger dataset with rotated annotations became

available, the system can be modified easily to detect rotated text instances by modifying the shape of the output layer to generate an extra value that represents the box rotation angle.

Table 9.3: Comparisons of the state-of-the-art results on the ICDAR 2013. The results are reported in the terms of Recall (R), Precision (P) and F-measure (F).

Model	R	P	F
SSD [39]	0.60	0.80	0.68
Yin [69]	0.66	0.88	0.76
Neumann [47]	0.71	0.82	0.76
Neumann [46]	0.72	0.82	0.77
FASText [5]	0.69	0.84	0.77
Zhang [73]	0.74	0.88	0.80
TextFlow [59]	0.76	0.85	0.80
Text-CNN [19]	0.73	0.93	0.82
CTPN [60]	0.73	0.93	0.82
FCRN [16]	0.76	**0.94**	0.84
TextBoxes [36]	0.82	0.88	0.85
TAM+HIM [18]	**0.86**	0.88	0.87
The proposed model	0.854	0.896	**0.875**

Three datasets have been used to train and tune the end-to-end system. The main dataset used for training was the SynthText in the Wild dataset because of its large size compared to the other datasets. After training on SynthText in the Wild and reaching a point where the performance is not being enhanced by extra training time, a mix of the training sets of COCO-Text and ICDAR 2013 datasets has been used to tune the system. This tuning was necessary to give the the end-to-end system a feel of how real text instances look like after being only trained on a synthetic dataset.

During the process of building the proposed end-to-end system, a few lessons have been learned that can be used by other researchers to enhance their systems. One of the important lessons is using transfer learning from models that have been trained on relevant tasks.

Table 9.4: Comparisons of the state-of-the-art results using the COCO-text dataset.

Method	Recall	Precision	F-measure
Baseline C [61]	0.05	0.19	0.07
Baseline B [61]	0.11	**0.90**	0.19
Baseline A [61]	0.23	0.84	0.36
Yao [67]	0.27	0.43	0.33
He [18]	0.31	0.46	0.37
The proposed model	**0.51**	0.65	**0.53**

We found that the transfer learning from a model previously trained on COCO-Text-Patch was more effective than transfer learning from models previously trained on ImageNet. Apparently, text instances in natural scene images are not closely correlated with ImageNet's more general object categories. Another lesson is how batch normalization can be used to reduce the effect of the vanishing gradient problem, but it does not eliminate it. Empirically, we found that very deep models fail to train even if batch normalization is used. The techniques used to develop the proposed end-to-end system can be adopted easily by other domains. For instance, the architecture of the end-to-end text detection system, including the text attention model, can be used to build better general object detection systems. The experiments conducted in this dissertation show how adding attention information enhanced the performance of the proposed text detection system. The multi-phase transfer learning can be used in enhancing the attention model in a variety of systems such as image captioning [64], visual question answering [65], and weakly-supervised object localization [48]. The multi-phase transfer learning can also be used to develop better semantic segmentation models.

Figure 9.5: Output samples from the COCO-Text test set. The samples demonstrate how the system could detect very difficult instances for being fuzzy, occluded or illegible.

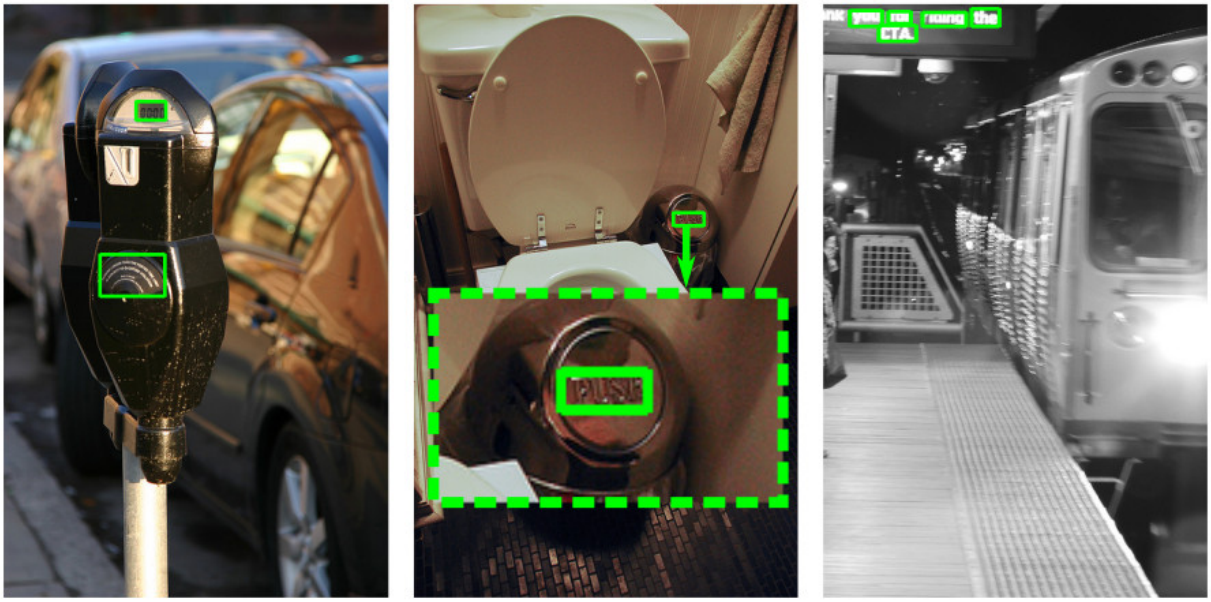


Figure 9.6: Output samples from the COCO-Text test set. The samples demonstrate how the system could detect different types of text instances such as LED, curved, 3D and billboard text.

Figure 9.7: Output samples from the COCO-Text test set. The samples demonstrate how the system could detect different scripts such as Latin, Arabic, Hindi and Japanese.

Chapter 10

Conclusion

In this dissertation, we proposed a novel end-to-end text detection system based on a collection of novel deep learning techniques. The dissertation included analysis of how using an end-to-end system that utilizes these novel deep learning techniques would yield a better performance on the challenging problem of text detection in natural scene images. Those novel deep learning techniques can also be adopted by other domain to enhance the performance of relevant models. A description of the proposed system, as well as the novel techniques used to develop this system, have been detailed.

We introduced two new deep learning architectures, input fast-forwarding (FFNet) and nested auxiliary branches (AuxNet). FFNet was trained using the COCO-Text-Patch dataset we created, and made available to the research community with the purpose of being a standard dataset that can be used to train, and evaluate text classifiers. FFNet was the backbone of a powerful, lightweight attention model that boosted the performance of the proposed end-to-end text detection system. We also spawned this lightweight architecture into a more advanced one, AuxNet, which has addressed the well-known and persistent problem of vanishing gradients in ultra-deep models. We built a 4096-layer network, the deepest network to the best of our knowledge, based on this novel architecture.

The implemented end-to-end system included the text attention model, the attention map fusion, and the prediction model. In addition to the proposed end-to-end system implementation, some exploration studies have been performed in order to decide on the optimal

choices of the end-to-end system. Those exploration studies included the development of the Class Activation Map (CAM) model, and evaluating the performance of the different object detection models when used for text detection.

The dissertation also discussed how multi-phase learning transfer has been used in the proposed model. Another exploration study has been performed where a bigger task is divided into smaller tasks, demonstrated a better performance in small experiment designed for this purpose. The conducted experiment on the MNIST dataset demonstrated how a small network can outperform state of the art networks of comparable size.

The end-to-end system has been evaluated using the ICDAR 2013 and the COCO-Text standard benchmarks. The system demonstrated superior performance, especially, in detecting difficult text instances such as low resolution, low contrast, and illegible text instances. The implemented end-to-end system exceeded the state-of-the-art on the ICDAR 2013 standard benchmark with an F-measure value of 0.875 and exceeded the state-of-the-art on COCO-Text by large margin with an F-measure value of 0.53.

Although the results on COCO-Text are impressive compared to the previous state of the art, the numbers show how text detection is far from being a solved problem. The recall value demonstrates how the system failed to detect nearly half of the text instances in the dataset. Also, the precision value needs to be improved a lot before considering text detection systems mature. The proposed end-to-end text detection system can be enhanced in many ways to achieve better performance. Building a new object detection model with text detection in mind would make a big difference in the performance of text detection systems. Most of the currently available object detection models do not take into consideration the nature of text in terms of size and location distribution inside the input image. Another way is to use temporal information. Current text detection systems take individual images as input, discarding any available temporal information. Building bigger datasets will certainly enable

the training of better deep models.

Bibliography

[1] BVLC reference CaffeNet model. https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet. Accessed: 06-2016.

[2] Virginia Tech New River super computer. http://www.arc.vt.edu/computing/newriver/. Accessed: 2016-06-06.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 343–351, 2016.

[5] Michal Busta, Lukas Neumann, and Jiri Matas. Fastext: Efficient unconstrained scene text detector. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 1206–1214, 2015.

[6] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[7] François Chollet. Keras (2015). *URL http://keras.io*, 2017.

[8] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems (NIPS)*, pages 379–387, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[11] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, 2014.

[12] Piotr Dollar, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transaction Pattern Analysis Machine Intelligence*, 36(8): 1532–1545, August 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2300479. URL http://dx.doi.org/10.1109/TPAMI.2014.2300479.

[13] Sean R Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3): 361–365, 1996.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

[15] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of The International Conference on Machine Learning (ICML)*, volume 28, pages 1319–1327, 2013.

[16] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single shot text detector with regional attention. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[19] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional neural network for scene text detection. *IEEE Transactions on Image Processing (TIP)*, 25 (6):2529–2541, June 2016. ISSN 1057-7149. doi: 10.1109/TIP.2016.2547588. URL http://dx.doi.org/10.1109/TIP.2016.2547588.

[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV*, pages 646–661. Springer, 2016.

[21] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Ahmed Ibrahim, Lynn Abbott, and Mohamed Hussein. An image dataset of text patches in everyday scenes. In *Proc. 12th International Symposium on Visual Computing (ISVC)*, Dec 2016.

[24] Ahmed Ibrahim, A. Lynn Abbott, and Mohamed E. Hussein. *Input Fast-Forwarding for Better Deep Learning*, pages 363–370. Springer International Publishing, Cham, 2017. ISBN 978-3-319-59876-5. doi: 10.1007/978-3-319-59876-5_40. URL https://doi.org/10.1007/978-3-319-59876-5_40.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[26] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision (IJCV)*, 116(1):1–20, January 2016. ISSN 0920-5691. doi: 10.1007/s11263-015-0823-z. URL http://dx.doi.org/10.1007/s11263-015-0823-z.

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[28] D. Karatzas, S. R. Mestre, J. Mas, F. Nourbakhsh, and P. P. Roy. ICDAR 2011 robust reading competition - challenge 1: Reading text in born-digital images (web and email). In *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1485–1490, Sept 2011. doi: 10.1109/ICDAR.2011.295.

[29] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, Aug 2015. doi: 10.1109/ICDAR.2015.7333942.

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

[33] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, E Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

[34] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 2, pages 562–570, 2015.

[35] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015.

[36] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *The AAAI Conference on Artificial Intelligence*, pages 4161–4167, 2017.

[37] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2014.

[38] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick.

Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision (ECCV)*, pages 21–37. Springer, 2016.

[40] Yu Liu and Michael S. Lew. Learning relaxed deep supervision for better edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 231–240, 2016.

[41] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[42] Simon M. Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, Robert Young, Kazuki Ashida, Hiroki Nagai, Masayuki Okamoto, Hiroaki Yamamoto, Hidetoshi Miyao, JunMin Zhu, WuWen Ou, Christian Wolf, Jean-Michel Jolion, Leon Todoran, Marcel Worring, and Xiaofan Lin. ICDAR 2003 robust reading competitions: entries, results, and future directions. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(2):105–122, 2005. ISSN 1433-2825. doi: 10.1007/s10032-004-0134-3. URL http://dx.doi.org/10.1007/s10032-004-0134-3.

[43] Dmytro Mishkin and Jiri Matas. All you need is a good init. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2016.

[44] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2204–2212, 2014.

[45] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[46] Lukáš Neumann and Jiří Matas. Efficient scene text localization and recognition with local character refinement. In *13th IEEE International Conference on Document Analysis and Recognition (ICDAR)*, pages 746–750, 2015.

[47] Lukáš Neumann and Jiří Matas. Real-time lexicon-free scene text localization and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38 (9):1872–1885, 2016.

[48] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 685–694, 2015.

[49] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the International Conference on Advances in neural information processing systems(NIPS)*, pages 91–99, 2015.

[51] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2015.

[52] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[54] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2015.

[56] Rupesh K. Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2377–2385, 2015.

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

[58] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *The Associ-*

ation for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI), pages 4278–4284, 2017.

[59] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 4651–4659, 2015.

[60] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 56–72, 2016.

[61] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. In *arXiv preprint arXiv:1601.07140*, 2016. URL http://vision.cornell.edu/se3/wp-content/uploads/2016/01/1601.07140v1.pdf.

[62] Qi Wang and Joseph JaJa. From maxout to channel-out: Encoding information on sparse pathways. In *International Conference on Artificial Neural Networks*, pages 273–280. Springer, 2014.

[63] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[64] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.

[65] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention

networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21–29, 2016.

[66] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1083–1090, June 2012. doi: 10.1109/CVPR.2012.6247787.

[67] Cong Yao, Xiang Bai, Nong Sang, Xinyu Zhou, Shuchang Zhou, and Zhimin Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016.

[68] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, July 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2366765.

[69] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(5):970–983, 2014.

[70] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[71] Matthew Zeiler and Robert Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2013.

[72] Ke Zhang, Miao Sun, Xu Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[73] Zheng Zhang, Wei Shen, Cong Yao, and Xiang Bai. Symmetry-based text line detection in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2558–2567, 2015.

[74] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[75] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[76] Yingying Zhu, Cong Yao, and Xiang Bai. Scene text detection and recognition: Recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, 2016.

[77] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Proceedings of the International Conference on European Conference on Computer Vision (ECCV)*, 2014.

[78] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 28–31. IEEE, 2004.

Appendices

Appendix A

Examples from COCO-Text

This Appendix contains output examples from the test set of the COCO-Text dataset. These examples were generated using the proposed system. Because examples generated by other state of the art models are not publicly available, we could not make a side-by-side comparison to other models. Because the ground truth for the COCO-Text test set is not publicly available, color-coding of the bounding boxes is not possible. The green boxes indicate text regions that have been detected by the proposed system, and no further interpretation is available. The examples have been chosen to demonstrate the power of the implemented system. Most of the examples contain hard to detect instances due to conditions related to lighting, occlusion or legibility. The samples also have been chosen to include text instances of different scripts such as Latin, Arabic, Hindi, Chinese, Japanese and Korean. In addition to that, some of the failure cases are shown.

Example images containing stop signs.

Example images containing street signs.

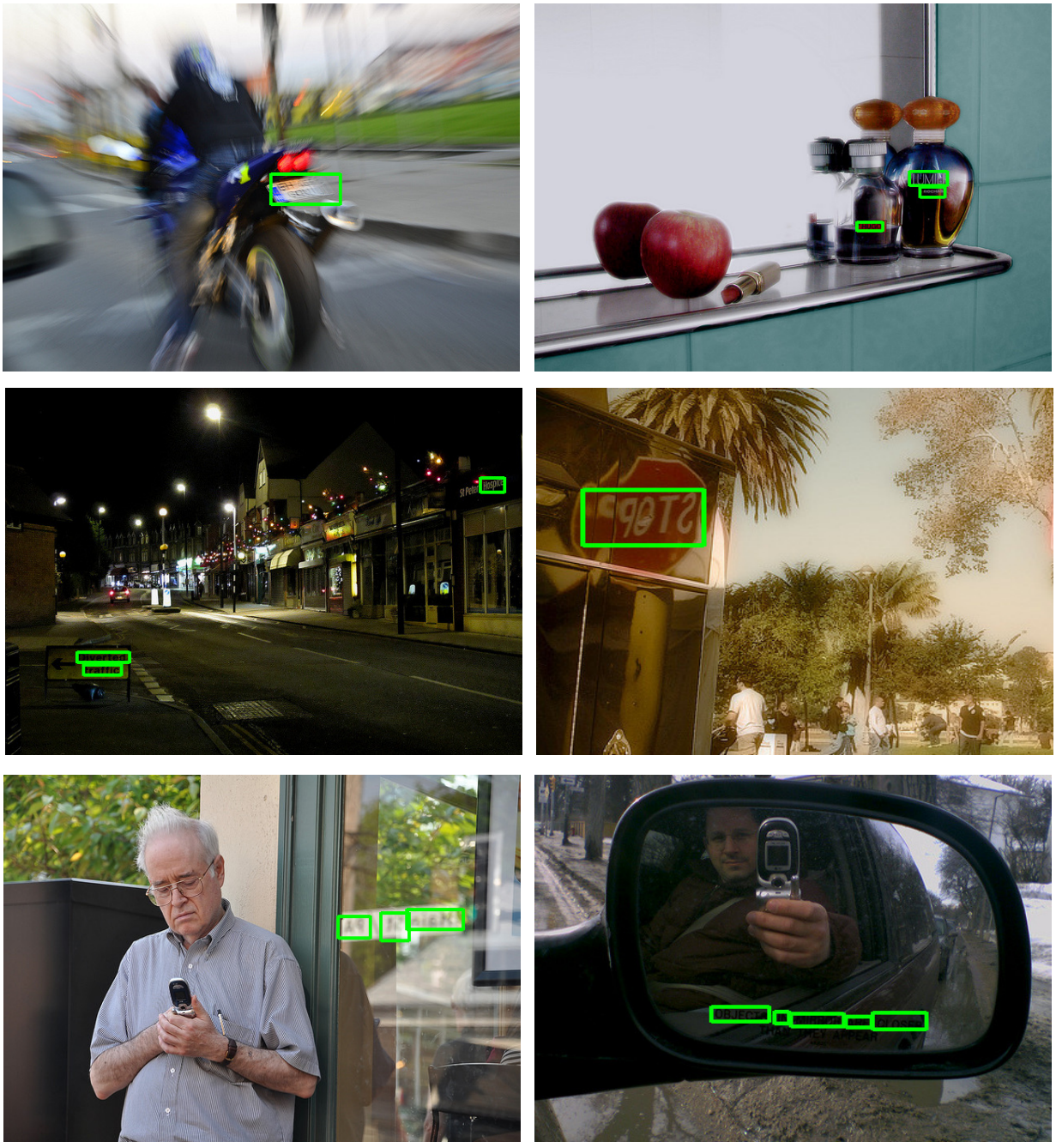
Example images containing street scenes.

Example images containing Japanese/Chinese/Korean script.

Example images containing other scripts such as Hindi, Thai and Arabic.

Example images containing unusual fonts or upside down text.

Example images containing blurred, low-light or reflections.



Example images with incorrect detections.

Appendix B

Examples from the ICDAR 2013 Dataset

This appendix contains output examples from the ICDAR 2013 test set. These examples were generated using the proposed system. The examples have been chosen to demonstrate the power of the proposed system in comparison to the previous state of the art model[1]. Most of the examples contain hard to detect instances due to conditions related to lighting, occlusion or legibility. The bounding boxes are color coded. The coding is such that the green, red, and yellow boxes represent true positives, false positives and detections covering multiple ground truth instances, respectively. Some of the images contains gray areas. These gray areas indicate a true positive on a "do not care" instance. "Do not care" instances are text instances in the ground truth of the ICDAR 2013 dataset but are not used in the evaluation process.

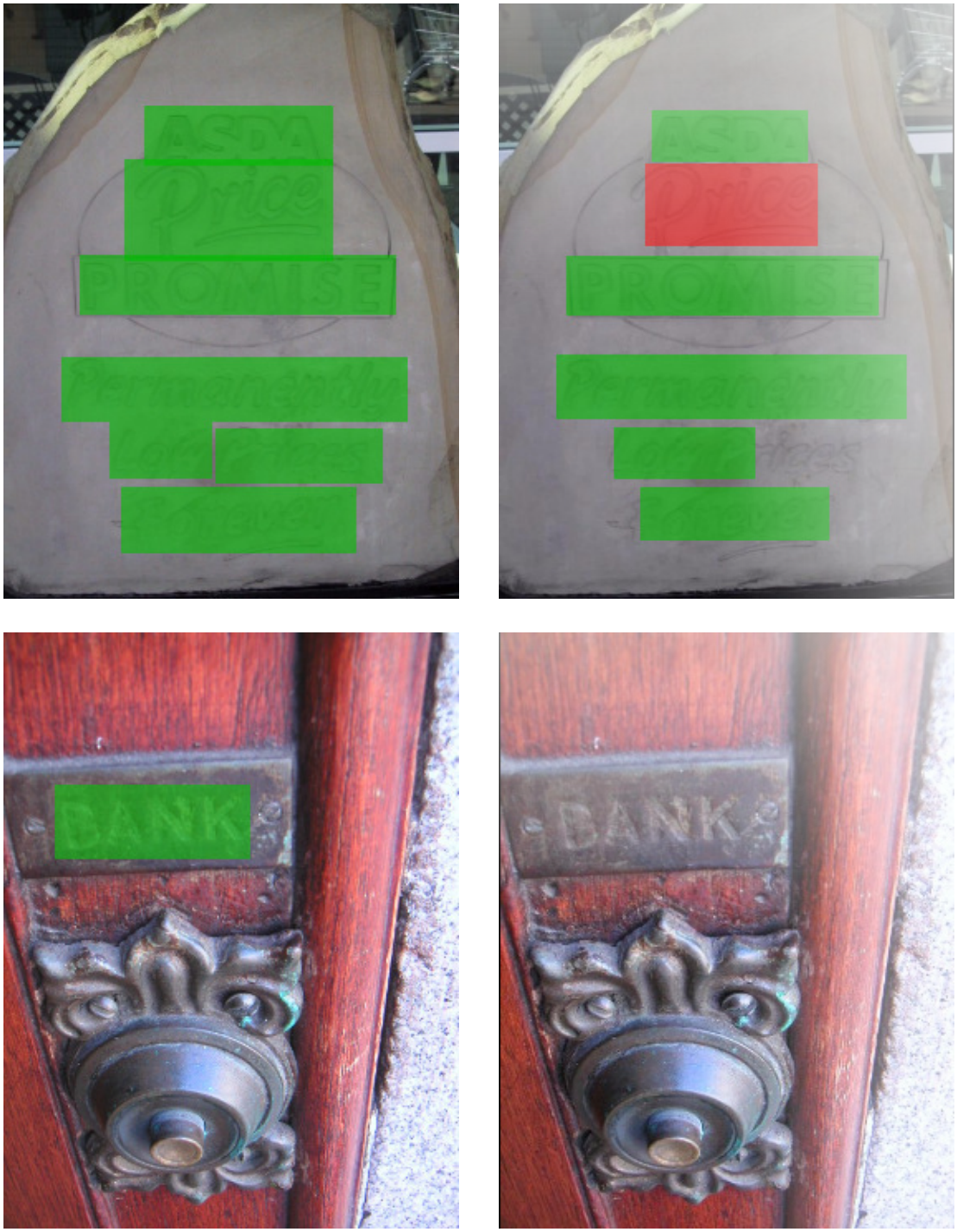
[1]He, P., Huang, W., He, T., Zhu, Q., Qiao, Y., & Li, X. (2017). Single Shot Text Detector With Regional Attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 3047-3055.

Examples from outdoor scenes.

Our system. He et al.

Examples with engraved text.

Our system. He et al.



Examples with low-resolution or occluded text.

Our system.

He et al.

Examples with unusual fonts.

Our system. He et al.

The gray area in the lower right image indicates a true positive on a "do not care" instance. "Do not care" instances are text instances in the ground truth of the ICDAR 2013 dataset

but are not used in the evaluation process. Because the dataset is not perfectly labeled, only the “R” is labeled in the ground truth separately from the whole occluded text. As a result, our detection of the complete occluded text is marked false positive. Meanwhile, the incomplete detection by He et al. is characterized as a correct detection.