TEXT SPOTTING AND RECOGNITION USING DEEP LEARNING

A PROJECT REPORT

Submitted By

NACHIAPPAN N.	**312214104059**
GOWTHAM S.	**312214104035**
KHARTHIK KUMAR C.R.	**312214104304**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

ANNA UNIVERSITY :: CHENNAI - 600025

April 2018

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled **"Text Spotting and Text Recognition using Deep Learning"** is the *bonafide* work of "**NACHIAPPAN N. (312214104059)**, **GOWTHAM S. (312214104035)**, and **KHARTHIK KUMAR C.R. (312214104304)**" who carried out the project work under my supervision.

Dr. Chitra Babu	**Ms. S. Manisha**
Head of the Department	**Supervisor**
Professor,	Assistant Professor,
Department of CSE,	Department of CSE,
SSN College of Engineering,	SSN College of Engineering,
Kalavakkam - 603 110	Kalavakkam - 603 110

Place:
Date:

Submitted for the examination held on. . . . .

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

We thank GOD, the almighty for giving us strength and knowledge to do this project.

We would like to thank and deep sense of gratitude to our guide **Ms. S. MANISHA**, Assistant Professor, Department of Computer Science and Engineering, for her valuable advice and suggestions as well as her continued guidance, patience and support that helped us to shape and refine our work.

My sincere thanks to **Dr. CHITRA BABU**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and we would like to thank our project Coordinator **Dr. S. SHEERAZUDDIN**, Professor, Department of Computer Science and Engineering for his valuable suggestions throughout this first phase of project.

We express our deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. We also express our appreciation to our **Dr. S. SALIVAHANAN**, Principal, for all the help he has rendered during this course of study.

We would like to extend my sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of our project work. Finally, we would like to thank our parents and friends for their patience, cooperation and moral support throughout our lives.

NACHIAPPAN N. **GOWTHAM S.** **KHARTHIK KUMAR C.R.**

ABSTRACT

The goal of this work is text spotting in natural images.This is divided into two sequential tasks: detecting words regions in the image, and recognizing the words within these regions. A Convolutional Neural Network (CNN) with a novel architecture that enables efficient feature sharing (by using a number of layers in common) for text detection, character case-sensitive and insensitive classification, and bigram classification is developed. A number of technical changes is made over the traditional CNN architectures, including no downsampling for a per-pixel sliding window, and multi-mode learning with a mixture of linear models. A method of automated data mining of Flickr, that generates word and character level annotations is developed. These components are used together to form an end-to-end,text spotting system. The text-spotting system is evaluated on two standard benchmarks, the ICDAR Robust Reading data set and the Street View Text (SVT) data set.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	- Convolutional Neural Networks
SR	- Super-Resolution
CAT	- Computer Aided Tomography

CHAPTER 1

Introduction

Extracting textual information from natural images is a challenging problem with many practical applications. We attack the problem from a different angle.For low-level data representation, we use an unsupervised feature learning algorithm that can automatically extract features from the given data. We integrate the learned features into a large, discriminatively trained Convolutional Neural Network (CNN) and recognize text from images.

Text spotting in natural images is usually divided into two tasks: text detection, and word recognition. Text detection involves generating candidate bounding boxes that are likely to contain lines of text, while word recognition takes each candidate bounding box, and attempts to recognize the text depicted within it, or potentially reject the bounding box as a false positive detection.

1.1 Motivation

Text recognition has quite a lot of relevant application for information retrieval such as content-based image retrieval. A neural network is a powerful data modelling tool that is able to capture and represent complex input/output relationships. Motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform ”intelligent” tasks similar to those performed by the human brain.

We use a Convolutional Neural Network (CNN) [1] and generate a per-pixel text/no-text saliency map, a case-sensitive and case-insensitive character saliency map, and a bigram saliency map. The text saliency map drives the proposal of word bounding boxes, while the character and bigram saliency maps assist in recognizing the word within each bounding box through a combination of soft costs.

1.2 Deep Learning

Learning multiple levels of representations of the underlying distribution of the data to be modelled. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning models are loosely related to information processing and communication patterns in a biological nervous system, such as neural coding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

Deep learning is a class of machine learning algorithms that:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts

1.2.1 Deep Learning is Hierarchical Feature Learning

Benefit of deep learning models is their ability to perform automatic feature extraction from raw data, also called feature learning. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features. Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.

Deep learning is a kind of learning where the representation you form have several levels of abstraction, rather than a direct input to output.

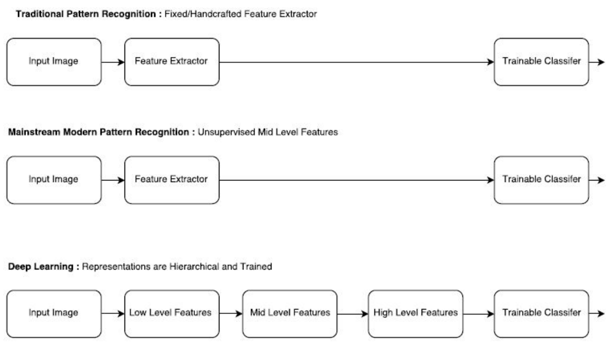

Traditional Pattern Recognition : Fixed/Handcrafted Feature Extractor
Input Image
Feature Extractor
Trainable Classifer
Mainstream Modern Pattern Recognition : Unsupervised Mid Level Features
Input Image
Feature Extractor
Trainable Classifer
Deep Learning : Representations are Hierarchical and Trained
Input Image
Low Level Features
Mid Level Features
High Level Features
Trainable Classifer

FIGURE 1.1: Deep Learning flow diagram

1.2.2 Deep Learning as Scalable Learning Across Domains

Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are images of pixel data, documents of text data or files of audio data. The network architecture that excels at object recognition in image data called the Convolutional Neural Network (CNN). This technique is seeing great success because like multi-layer perceptron feed-forward neural networks, the technique scales with data and model size and can be trained with back-propagation.

Deep learning is a pipeline of modules all of which are trainable. Deep because it has multiple stages in the process of recognizing an object and all of those stages are part of the training.

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

1.2.3 Concepts

The assumption underlying distributed representations is that observed data are generated by the interactions of layered factors.

Deep learning adds the assumption that these layers of factors correspond to levels of abstraction or composition. Varying numbers of layers and layer sizes can provide different degrees of abstraction.

Deep learning exploits this idea of hierarchical explanatory factors where higher level, more abstract concepts are learned from the lower level ones.

Deep learning architectures are often constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks.

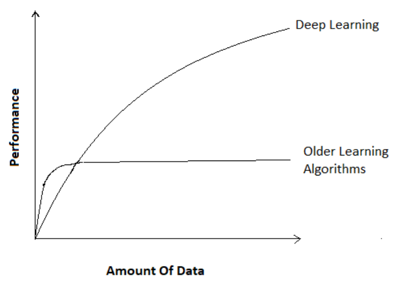


FIGURE 1.2: Performance of Deep Learning

1.2.4 Why Deep Leaning?

- Automatically extracts the low and high-level features for classification
- Good classification performance in face and character recognition
- Time consuming hand-crafted feature design is eliminated

1.3 Text Spotting

Text spotting in natural images is usually divided into two tasks: text detection, and word recognition. Text detection involves generating candidate bounding boxes that are likely to contain lines of text, while word recognition takes each candidate bounding box, and attempts to recognize the text depicted within it, or potentially reject the bounding box as a false positive detection.

Text, as the physical incarnation of language, is one of the basic tools for preserving and communicating information. Much of the modern world is designed to be interpreted through the use of labels and other textual cues, and so text finds itself scattered throughout many images and videos. Through the use of text spotting, an important part of the semantic content of visual media can be decoded and used, for example, for understanding, annotating, and retrieving the billions of consumer photos produced every day.

We use a Convolutional Neural Network (CNN) and generate a per-pixel text/no-text saliency map, a case-sensitive and case-insensitive character saliency map, and a bigram saliency map. The text saliency map drives the proposal of word bounding boxes, while the character and bigram saliency maps assist in recognizing the word within each bounding box through a combination of soft costs.

CHAPTER 2

Data mining and Contributions

We describe a method for automatically mining suitable photo sharing websites to acquire word and character level annotated data. This annotation is used to provide additional training data for the CNN.

2.1 Word Mining

Photo sharing websites such as Flickr [20] contain a large range of scenes, including those containing text. In particular, the ”Typography and Lettering” group on Flickr [21] contains mainly photos or graphics containing text.

As the text depicted in the scenes are the focus of the images, the user given titles of the images often include the text in the scene. Capitalizing on this weakly supervised information, we develop a system to find title text within the image,automatically generating word and character level bounding box annotations.Using a weak baseline text-spotting system based on the Stroke Width Transform (SWT), we generate candidate word detections.

2.2 Character Annotation

In addition to mining data from Flickr, we also use the word recognition system described to automatically generate character bounding box annotations for datasets which only have word level bounding box annotations. For each cropped word, we perform the optimalfitting of the groundtruth text to the character map.This places inter-character breakpoints with implied character centers, which can be used as rough character bounding boxes. We do this for the SVT and Oxford Cornmarket datasets, allowing us to train and test on an extra 22,000 cropped characters from those datasets.

2.3 Contributions

First, we introduce a method to share features which allows us to extend our character classifiers to other tasks such as character detection and bigram classification at a very small extra cost: we first generate a single rich feature set, by training a strongly supervised character classifier, and then use the intermediate hidden layers as features for the text detection, character case-sensitive and insensitive classification, and bigram classification. This procedure makes best use of the available training data: plentiful for character/non-character but less so for the other tasks. It is reminiscent of the Caffe idea, but here it is not necessary to have external sources of training data.

A second key novelty in the context of text detection is to leverage the convolutional structure of the CNN to process the entire image in one go instead of running CNN classifiers on each cropped character proposal.This allows us to generate efficiently, in a single pass, all the features required to detect word bounding boxes, and that we use for recognizing words from a fixed lexicon using the Viterbi algorithm. We also make a technical contribution in showing that our

CNN architecture using maxout as the non-linear activation function has superior performance to the more standard rectified linear unit.

Our third contribution is a method for automatically mining and annotating data. Since CNNs can have many millions of trainable parameters, we require a large corpus of training data to minimize over-fitting, and mining is useful to cheaply extend available data. Our mining method crawls images from the Internet to automatically generate word level and character level bounding box annotations, and a separate method is used to automatically generate character level bounding box annotations when only word level bounding box annotations are supplied.

CHAPTER 3

Literature Survey

Max Jaderberg in his seminal paper ”Deep Features for Text Spotting” [1] describes a method for text spotting in natural images. This is divided into two sequential tasks: detecting words regions in the image, and recognizing the words within these regions.He develops a Convolutional Neural Network (CNN) classifier that can be used for both tasks. The CNN has a novel architecture that enables efficient feature sharing (by using a number of layers in common) for text detection, character case-sensitive and insensitive classification, and bigram classification.

Chen,Tsai,Schroth,Chen propose a novel text detection algorithm in there paper ”Robust text detection in natural images”[12], which employs an edge-enhanced Maximally Stable Extremal Regions as basic letter candidates. These candidates are then filtered using geometric and stroke width information to exclude non-text objects. Letters are paired to identify text lines, which are subsequently separated into words. They evaluate their system using the ICDAR competition dataset and our mobile document database. The experimental results demonstrate the excellent performance of the proposed method.

In ”End-to-end text recognition with convolutional neural networks.” [4] paper Wang,Wu and Coates take a different route and combine the representational power of large, multi-layer neural networks together with recent developments in unsupervised feature learning, which allows them to use a common framework to train highly-accurate text detector and character recognizer modules. Then, using only simple off-the-shelf methods, they integrate these two modules into a full end-to-end, lexicon-driven, scene text recognition system that achieves state-of-the-art performance on standard benchmarks, namely Street View Text and ICDAR 2003.

The end-to-end text recognition problem can be decomposed into three natural sub-problems: text detection, character recognition and word recognition. The first problem is to identify text locations in a natural image. The second problem requires identifying characters in cropped image patches; this is a classification problem with high confusion due to upper-case/lower-case and letter/number confusion. The third problem is a sequencing problem, given an image of a word, to output the most likely obtained word for this paper will be provided with the final version, corresponding to that image. These problems clearly overlap as character recognition is a sub-problem of word recognition, which itself is a sub-problem of text detection.

The end-to-end problem presents technical challenges on multiple levels. On the character level, the main challenge is to achieve high recognition accuracy. On the word level, the word recognizer needs to be accurate, fast, and scalable with lexicon size. On the end-to-end level, the system needs to balance precision, recall, complexity and speed. In our work, we aim for a highly accurate character recognizer, a fast, accurate and scalable word recognizer and for the end-to-end system, we aim for fast performance at query time, and high F-measure.

Detecting text in natural images, as opposed to scans of printed pages, faxes and business cards, is an important step for a number of Computer Vision applications, such as computerized aid for visually impaired, automatic geo-coding of businesses, and robotic navigation in urban environments.

Retrieving texts in both indoor and outdoor environments provides contextual clues for a wide variety of vision tasks. Moreover, it has been shown that the performance of image retrieval algorithms depends critically on the performance of their text detection modules. For example, two book covers of similar design but with different text, prove to be virtually indistinguishable without detecting and OCRing the text.

Neumann describe efficient method for text localization and recognition in real-world images[11]. The method exploits higher-order properties of text such as word text lines. We demonstrate that the grouping stage plays a key role in the text localization performance and that a robust and precise grouping stage is able to compensate errors of the character detector. The method includes a novel selector of Maximally Stable Extremal Regions (MSER) which exploits region topology. Experimental validation shows that 95.7 percentage of characters in the ICDAR dataset are detected using the novel selector of MSERs with a low sensitivity threshold. The proposed method was evaluated on the standard ICDAR 2003 dataset where it achieved state-of-the-art results in both text localization and recognition.

In ”Super Resolution Reconstruction of an Image” [16] paper,it presents a generalization of restoration theory for the problem of Super-Resolution Reconstruction (SRR) of an image. In the SRR problem, a set of low quality images is given, and a single improved quality image which fuses their information is required. We present a model for this problem, and show how the classic restoration theory tools - ML, MAP and POCS - can be applied as a solution. A hybrid algorithm which joins the POCS and the ML benefits is suggested. Image Resolution depends on the physical characteristics of the sensor:the optics and the density and spatial response of the detector elements.Increasing the resolution by sensor modification may not be an available option.An increase in the sampling rate could ,however, be achieved by obtaining more samples of the scene from a sequence displaced pictures.An estimate of the sensor's spatial response helps obtain a sharper image. An iteration algorithm to increase image resolution ,together with a method for image registrarion with subpixel accuracy is preseneted in the paper "Image Resolution by Image Registration".Examples are shown for low-resolution gray-level and color images,with an increase in resolution is clearly observed after only a few iterations. The same method can also be used for debluring a single blurred image.

Optical character recognition (also optical character reader, OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image.It is widely used as a form of information entry from printed paper data records [18], whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitising printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Decomposing the text-spotting problem into text detection and text recognition was first proposed by [15]. Authors have subsequently focused solely on text detection [2,7], or text recognition,or on combining both in end-to-end systems [11,13,17].Text detection methods are either based on connected components (CCs) [1,11,12,13,17] or sliding windows [3,5].Connected component methods segment pixels into characters, then group these into words. For example, Epshtein et al. take characters as CCs of the stroke width transform [1], while Neumann and Matas [11,17] use Extremal Regions, or more recently oriented strokes [13], as CCs representing characters. Sliding window methods approach text spotting as a standard task of object detection. For example, Wang et al. [5] use a random ferns sliding window classifier to find characters in an image, grouping them using a pictorial structures model for a fixed lexicon. Wang Wu et al. [5] build on the fixed lexicon problem by using CNNs with unsupervised pre-training as in [20].

Alsharif et al. [4] and Bissacco et al. [10], also use CNNs for character classification -both methods over-segment a word bounding box and find an approximate solution to the optimal word recognition result, in [21] using beam search and in [4] using a Hidden Markov Model. The works by Mishra et al. [13] Novikova et al. focus purely on text recognition - assuming a perfect text detector has produced cropped images of words. In [11], Neumann combines both visual and lexicon consistency into a single probabilistic model.

S.No	Name	Year	Journal	Inference	Future Works
1	Robust Text Detection in natural images with edge-enhanced maximally stable extremal regions	2015	IEEE	Developed using edge-enhanced MSER	Efficiently combined with visual search system
2	End to End Text Recognition using Convolutional Neural Networks	2016	IEEE	Demonstrate the feasibility of using large, multilayer convolutional neural networks	Continue improving the performance of the text detection system and aim for better-cropped bounding boxes
3	Deep Features for Text Spotting	2015	European Conference on Computer Vision	CNN has a novel architecture that enables efficient feature sharing for text detection	Implementing sliding window detection as a byproduct of the CNN

FIGURE 3.1: Survey on Text Spotting

S.No	Name	Year	Journal	Inference	Future Works
4	Character Recognition System: Performance Comparison of Neural Networks and Genetic Algorithm	2015	1st International Conference on Computer & Information Engineering	Developed a recognition system for character using BPN and GA and shows the various parameters comparison	Improve the performance of Neural Networks
5	Optical Recognition Technique Algorithms	2016	Theoretical and Applied Information Technology	Develop a neural network based method for accurate optical character recognition	Improve accurracy by using chararcter segmentation for pre-processing
6	Wordfence: Text detection in natural images with border awareness	2017	IEEE	The concept of WordFence, which detects border areas surrounding each individual word	Achieve competitive performance on ICDAR11 and ICDAR13 without utilizing any heuristics or knowledge based approaches.

FIGURE 3.2: Survey on Character Recognition

CHAPTER 4

Problem Definition and Architectural Design

4.1 Problem statement

1. To segment image to individual characters and learn features using Convolutional Neural Networks (CNN).
2. To detect and recognize text after classification.

4.2 Architectural Design of Proposed System

The proposed system has the following steps:

1. Preprocessing
2. Feature Learning
3. Training and Implementation
4. Text Spotting End-to-End Pipeline
5. Performance Evaluation
6. Super-Resolution
7. Text Recognition

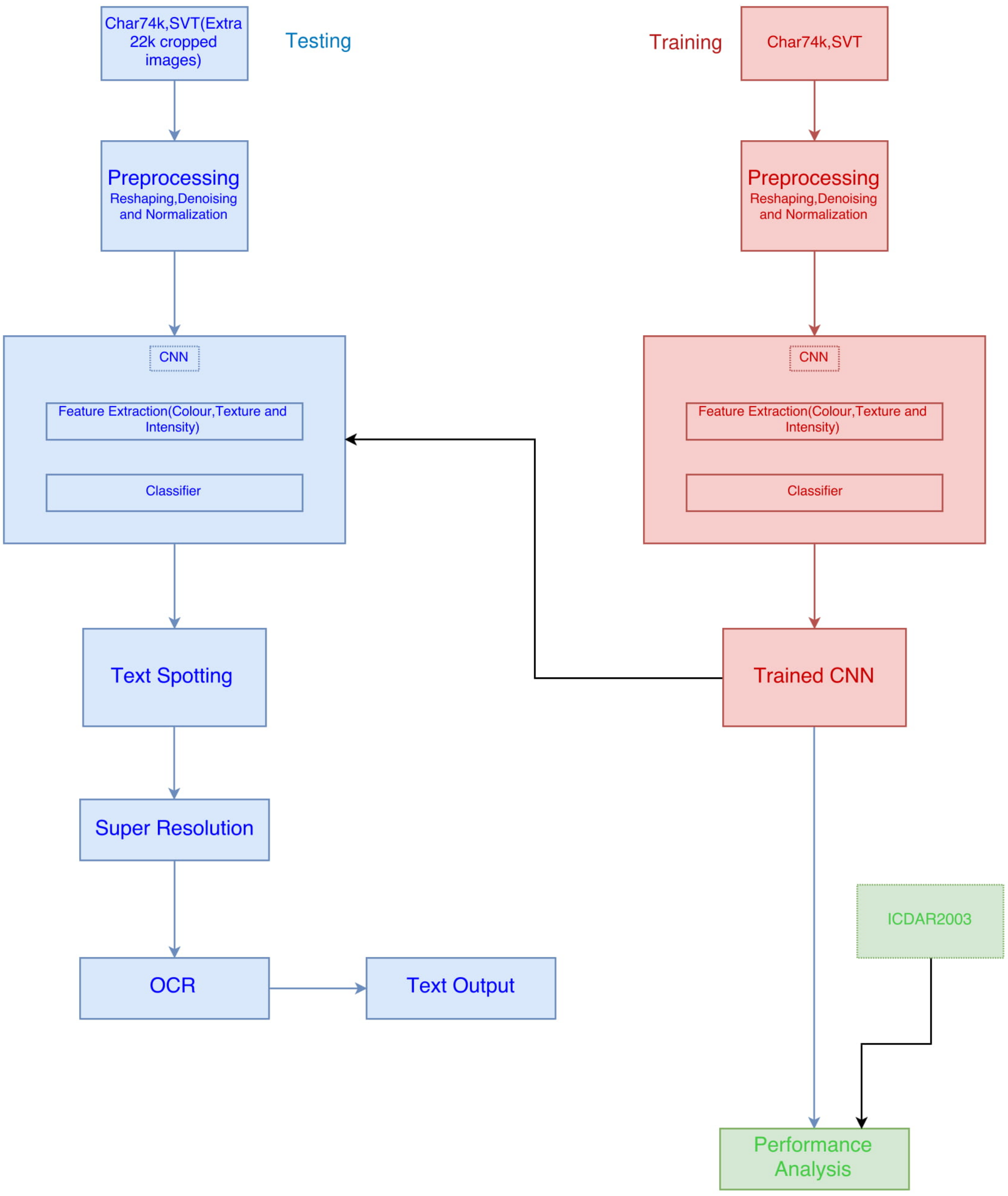

Char74k,SVT(Extra 22k cropped images)
Testing
Training
Char74k,SVT
Preprocessing
Reshaping,Denoising and Normalization
Preprocessing
Reshaping,Denoising and Normalization
CNN
Feature Extraction(Colour,Texture and Intensity)
Classifier
CNN
Feature Extraction(Colour,Texture and Intensity)
Classifier
Text Spotting
Trained CNN
Super Resolution
ICDAR2003
OCR
Text Output
Performance Analysis

FIGURE 4.1: Architectural Diagram

CHAPTER 5

Proposed System

5.1 Preprocessing

The process of enhancing the image, which should be used for further processing, is called preprocessing.Preprocessing is the major step in handwriting recognition system.

5.1.1 Reshaping

Reshaping the dimensions of image matrix by giving size as argument.

5.1.2 Denoising

Mean Filter: Mean filtering is a simple, intuitive and easy to implement method of smoothing images, i.e. reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images.The idea of mean filtering is simply to replace each pixel value in an image with the mean ('average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 33 square kernel is used, as

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

FIGURE 5.1: 3 X 3 averaging kernel often used in mean filtering

shown in Figure 5.1, although larger kernels (e.g. 55 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar but not identical effect as a single pass with a large kernel.)

Computing the straightforward convolution of an image with this kernel carries out the mean filtering process.

5.1.3 Normalisation

Normalization is scaling technique or a mapping technique or a pre processing stage where, we can find new range from an existing one range. It can be helpful for the prediction or forecasting purpose a lot. As we know there are so many ways to predict or forecast but all can vary with each other a lot. So to maintain the large variation of prediction and forecasting the normalization technique is required to make them closer. Standard deviation is found along dimensions to normalize the data.

5.2 Feature Learning using Convolutional Neural Networks

The workhorse of a text-spotting system is the character classifier. The output of this classifier is used to recognize words and, in our system, to detect image regions that contain text. Text-spotting systems appear to be particularly sensitive to the performance of character classification; for example, in [8] increasing the accuracy of the character classifier by 7 led to a 25 percent increase in word recognition. In this section we therefore concentrate on maximizing the performance of this component.

To classify an image patch x in one of the possible characters (or background),we extract a set of features $\phi(x)$.

A binary classifier fc for each character c of the alphabet C is learnt.

Classifiers are learnt to yield a posterior probability distribution p(c/x) = fc($\phi(x)$).

Traditionally, features are manually engineered and optimized through a laborious trial and error cycle involving adjusting the features and re-learning the classifiers.In this work, we propose instead to learn the representation using a CNN, jointly optimizing the performance of the features as well as of the classifiers. As noted in the recent literature, a well designed learnable representation of this type can in fact yield substantial performance gains [5].CNNs are obtained by stacking multiple layers of features. A convolutional layer consist of K linear filters followed by a non-linear response function.

The input to a convolutional layer is a feature map z$_i(u,v)$

where (u,v) $\varepsilon\Omega are spatial coordinates$

and z$_i(u,v)\varepsilon R^C contains C scalar features$

The output is a new feature map zi+1 such that

z$_k i+1 = h(W_i k * z_i + b_i k);....(1)$

where

W$_i k and b_i k$

denote the k-th filter kernel and bias respectively,

and h is a non-linear activation function such as the Rectified Linear Unit (ReLU)

h(z) = max$\{$0,z$\}$.

Convolutional layers can be intertwined with normalization, subsampling, and max-pooling layers which build translation invariance in local neighborhoods. The process starts with z1 = x and ends by connecting the last feature map to a logistic regressor for classification. All the parameters of the model are jointly optimized to minimize the classification loss over a training set using Stochastic Gradient Descent (SGD), back-propagation, and other improvements.

Instead of using ReLUs as activation function hi, in our experiments it was found empirically that maxout yields superior performance. Maxout, in particular when used in the final classification layer, can be thought of as taking the maximum response over a mixture of n linear models, allowing the CNN to easily model multiple modes of the data. The maxout of two feature channels z1i and z2i is simply their pointwise maximum. More generally, the k0-th maxout operator hk' is obtained by selecting a subset Gk'i={1,2,....,K} of feature channels and computing the maximum over them: hk'i(zi(u,v)) = maxk(Gk'),zi(u,v). While different grouping strategies are possible, here groups are formed by taking g consecutive channels of the inputmap: G1i = {1,2,...,g}, G2i = {g +1,g +2, 2g} and so on. Hence, given K feature channels as input, maxout constructs K' = K/g new channels.

5.3 Training and implementation details

This section discusses the details of learning the character classifiers. Training is divided into two stages. In the first stage, a case-insensitive CNN character classifier is learned. In the second stage, the resulting feature maps are applied to other classification problems as needed. The output is four state-of-the-art CNN classifiers: a character/background classifier, a case-insensitive character classifier, a case-sensitive character classifier, and a bigram classifier.

Stage 1: Bootstrapping the case-insensitive classifier: The case-insensitive classifier uses a four-layer CNN outputting a probability p(c/x) over an alphabet C including all 26 letters, 10 digits, and a noise/background (no-text) class, giving a total of 37 classes. The input z1 = x of the CNN are grayscale cropped character images of 24 x 24 pixels, zero-centered and normalized by subtracting the patch mean and dividing by the standard deviation. Due to the small input size, no spatial pooling or downsampling is performed. Starting from the first layer, the input image is convolved with 96 filters of size 9 x 9, resulting in a map of size 16 x 16 (to avoid boundary effects) and 96 channels. The 96 channels are then pooled with maxout in group of size g = 2, resulting in 48 channels. The sequence

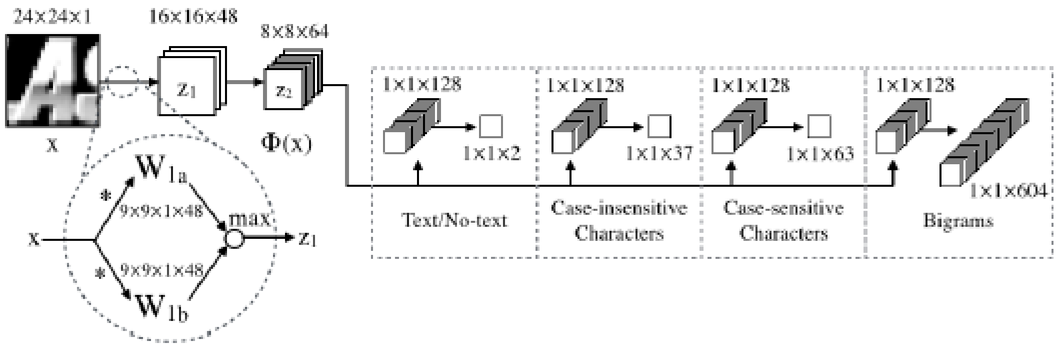


FIGURE 5.2: CNN module

continues by convolving with 128, 512,148 filters of side 9, 8, 1 and maxout groups of size g = 2,4,4, resulting in feature maps with 64, 128, 37 channels and size 8 x 8, 1 x 1, 1 x 1 respectively. The last 37 channels are fed into a soft-max to convert them into character probabilities.In practice we use 48 channels in the final classification layer rather than 37 as the software we use, based on cuda-convnet, is optimized for multiples of 16 convolutional filters - we do however use the additional 12 classes as extra no-text classes, abstracting this to 37 output classes.

We train using stochastic gradient descent and back-propagation, and also use dropout in all layers except the first convolutional layer to help prevent overfitting. Dropout simply involves randomly zeroing a proportion of the parameters; the proportion we keep for each layer is 1, 0.5, 0.5, 0.5. The training data is augmented by random rotations and noise injection. By omitting any downsampling in our network and ensuring the output for each class is one pixel in size, it is immediate to apply the learnt filters on a full image in a convolutional manner to obtain a per-pixel output without a loss of resolution.

Stage 2: Learning the other character classifiers: Training on a large amount of annotated data, and also including a no-text class in our alphabet, means the

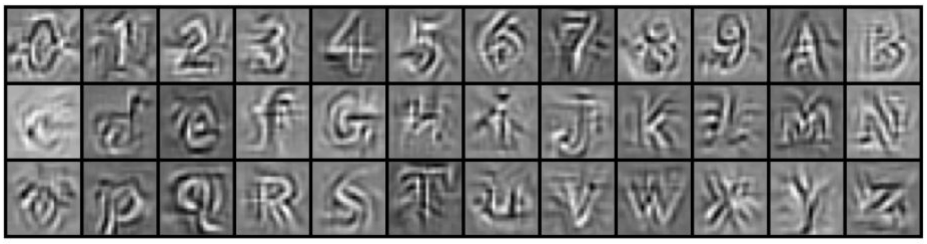


FIGURE 5.3: Visualizations of each character class learnt from the 37-way case-insensitive character classifier CNN

hidden layers of the network produce feature maps highly adept at discriminating characters, and can be adapted for other classification tasks related to text. We use the outputs of the second convolutional layer as our set of discriminative features, $\phi(x) = z2$.

From these features, we train a 2-way text/no-text classifier, a 63-way case-sensitive character classifier, and a bigram classifier, each one using a two-layer CNN acting on $\phi(x)(Fig.5.2)$.

The last two layers of each of these three CNNs result in feature maps with 128-2,128-63, and 128-604 channels respectively, all resulting from maxout grouping of size g = 4. These are all trained with $\phi(x) as input$.

The bigram classifier recognises instances of two adjacent characters, e.g. Fig 5.4. These CNNs could have been learned independently. However, sharing the first two layers has two key advantages. First, the low-level features learned from case-insensitive character classification allows sharing training data among tasks, reducing overfitting and improving performance in classification tasks with less informative labels (text/no-text classification), or tasks with fewer training examples (case-sensitive character classification, bigram classification). Second,it allows sharing computations, significantly increasing the efficiency.

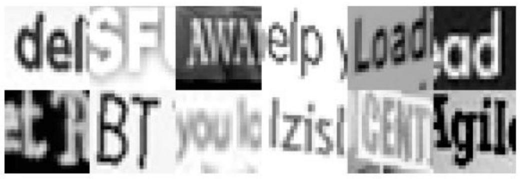


FIGURE 5.4: Bigram classification

5.4 End-to-End Pipeline

This section describes the various stages of the proposed end-to-end text spotting system, making use of the features learnt in Sect. 5.3. The pipeline starts with a detection phase (Sect. 5.4.1) that takes a raw image and generates candidate bounding boxes of words, making use of the text/no-text classifer. The words contained within these bounding boxes are then recognized against a fixed lexicon of words (Sect. 5.4.2), driven by the character classifiers, bigram classifier,and other geometric cues.

5.4.1 Text Detection

The aim of the detection phase is to start from a large, raw pixel input image and generate a set of rectangular bounding boxes, each of which should contain the image of a word. This detection process is tuned for high recall, and generates a set of candidate word bounding boxes. The process starts by computing a text saliency map by evaluating the character/background CNN classifier in a sliding window fashion across the image, which has been appropriately zero-padded so that the resulting text saliency map is the same resolution as the original image. As the CNN is trained to detect text at a single canonical height, this process is repeated for 16 different scales to target text heights between 16 and 260 pixels by resizing the input image.

Given these saliency maps, word bounding boxes are generated independently at each scale in two steps. The first step is to identify lines of text. To this end,the probability map is first thresholded to find local regions of high probability. Then these regions are connected in text lines by using the run length smoothing algorithm (RLSA): for each row of pixels the mean and standard deviation of the spacings between probability peaks are computed and neighboring regions are connected if the space between them is less than 3μ.

Finding connected components of the linked regions results in candidate text lines. The next step is to split text lines into words. For this, the image is cropped to just that of a text line and Otsu thresholding is applied to roughly segment foreground characters from background. Adjacent connected components (which are hopefully segmented characters) are then connected if their horizontal spacings are less than the mean horizontal spacing for the text line, again using RLSA. The resulting connected components give candidate bounding boxes for individual words, which are then added to the global set of bounding boxes at all scales. Finally,these bounding boxes are filtered based on geometric constraints (box height, aspect ratio, etc.) and undergo non-maximal suppression sorting them by decreasing average per-pixel text saliency score.

5.4.2 Word Recognition

For each word hypothesis w the optimal location of breakpoints are determined using dynamic programming and the word with best score is

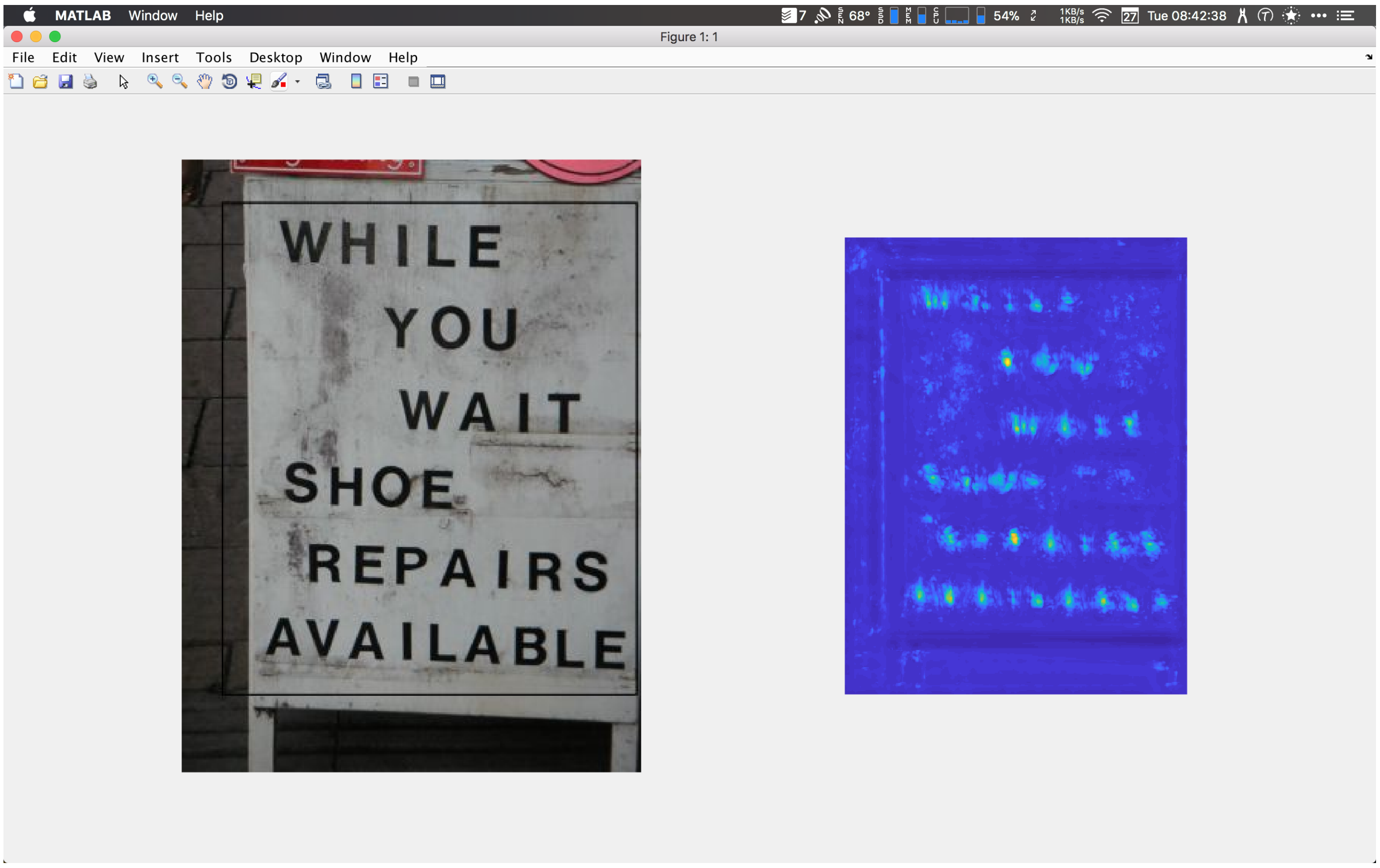


FIGURE 5.5: CNN generates text saliency map using that text/no-text classifier, after the run length smoothing phase, after the word splitting phase

s(w,I) = $\max_b^w s(w, b^w, P, Q, R)....(2)$

The unary scores m(bw,R) combine the following cues:distance from expected breakpoint placement, distance to out of image bounds, no-text class score, the bigram score, and, for the first and last breakpoint, the distance from the edge of the image. The pairwise score f(bw, bwi-1, P,Q,R) combines:the character score at the midpoint between breakpoints, the no-text score at the character center, the deviation from the average width of the character, and a dynamic contribution from the left and right bigram responses relative to the character score this allows bigram responses to take control when it is difficult to classify the character in focus, but easy to classify characters on either side. Also it is ensured that there is no violation of the sequence of characters in the word and that the character centers are all in the region of the word image.

Each score is weighted and linearly combined, with the parameters found by grid search on a validation set. Given the recognized word, the bounding box is adjusted to match the estimated breakpoints and added to a list of candidate recognized word regions.

The final step is to perform non-maximal suppression on this set of bounding boxes in order to eliminate duplicate detections.

5.5 Super-Resolution

Image Resolution depends on the physical characteristics of sensor: the optics and the density and spatial response of the detector elements.Increasing the resolution by sensor modification may not be an available option. A increase int he sampling rate could however be achieved by obtaining more samples of the scene from a sequence of displaced pictures.An estimate of the sensor's spatial response helps obtain a sharper image.

An iterative algorithm to increase image resolution,together with a method for image registration with sub-pixel accuracy,is presented. The approach described here is based on the resemblance of problem to the reconstruction of a 2-D object from its 1-D projections in CAT. In SR, each low-resolution pixel is a projection of a region in the scene whose size is determined by the imaging blur. The high resolution image is constructed using an approach similar to the back-projection method used in CAT.

Accurate knowledge of relative scene locations sensed by each pixel in the observed images is necessary for SR. This information is available in image regions where local deformation can be described by parametric function.They describe perspective transformation.

The image process yielding an observed monochrome image sequence gk, is modeled by

gk(m,n)=$\sigma(h(f(x,y))+\eta k(x,y),....(3)$

where gk is kth observed image frame

f is the original scene

h is blurring operator

ηkis

additive noise term

σis

nonlinear function which digitalizes and decimates the image into pixels and quantizes the image into pixels

(x,y) is the center of receptive field of detector.

The scene location is computed by

x=$x_k^0+s_x m\cos\theta k-s_y n\sin\theta k;$

y=$y_k^0+s_x m\sin\theta k-s_y n\cos\theta k.......(4)$

where

$(x_k, y_k) is$

the translation of kth frame

θis

the rotation of the kth frame about origin

$s_x and s_y are$

the sampling rates in the x and y directions,respectively

5.5.1 The Imaging Process

This section describes the two preliminary tasks of obtaining the parameters of the imaging process. The relative displacements of the input images at subpixel accuracy, as well as the blur in the imaging process, are computed.

Horizontal shift a, vertical shift b, and rotation angle 9 between images gi and g2 can be written as

g2(x, y)= g1(x cos $\theta - y sin\theta + a, y cos\theta + x sin\theta + b).....(5)$

5.5.2 Iterative Refinement

As images are recorded in discrete time intervals, the displacements between them may not be sufficiently small for the motion recovery method. We therefore iterate the following process for two given images g, and g2

- Initially assume no motion between the frames.

- Compute approximations to the motion parameters by solving Eqs. (6). Add the computed motion to the existing motion estimate.

- Warp frame g2 toward gi using the current motion estimates, and return to Step 2 with the warped image g2.

g2 gets closer to gi at every iteration, and as the residual corrections to (a, b, 6) computed in Step 2 get smaller, the motion parameters become more accurate. The process terminates when the corrections to (a, b, $\theta) approachzero.$

Since frame gi remains unchanged, 9 of the 12 coefficients in the set of equations are computed only once, and only 3 coefficients, depending on g2, need to be recomputed every iteration. This saves time in the iterative process.

$\Sigma g_x^2 a + \Sigma g_x g_y b + \Sigma A g_x \theta = g_x g_t,$

$\Sigma g_x g_y a + \Sigma g_y^2 b + \Sigma A g_y \theta = g_y g_t,$

$\Sigma A g_x a + \Sigma A g_y b + \Sigma A^2 \theta = A g_t, ...(6)$

In order to speed up the process and improve accuracy, a Gaussian pyramid data structure is used. First, the motion parameters are computed for a reduced resolution image in the pyramid, where even large translations become small. The computed motion parameters are then interpolated into a larger image, the motion estimate is corrected through a few iterations, and again interpolated to the next resolution level. This process is continued until the original full-size image is reached.

5.5.3 Recovering the Blur

Some knowledge of the digitization process is necessary to simulate the imaging process. Images used in our experiments were scanned by a flatbed scanner, and its blurring function was evaluated by scanning a small, white dot on a black background. We have approximated the point spread function by a 3 x 3 kernel. When the imaging process cannot be applied to control images, similar to the above mentioned white dot, the blur can be estimated from the degradation of features that are originally small points or sharp edges.

Experimental results are also given. The presented algorithm for solving the super-resolution problem is iterative. Starting with an initial guess for the high-resolution image, the imaging process is simulated to obtain a set of low-resolution images corresponding to the observed input images. If f were the the correct high-resolution image, then the simulated images should be identical to the observed images. The difference images are then computed, and used to improve the initial guess by back- projecting each value in the difference images onto its receptive field in.

This process is repeated iteratively to minimize the error function.

$e^n = \sqrt{\Sigma_k \Sigma(x,y)(g_k(x,y) - g_k^n(x,y))^2} \ldots.(7)$

The algorithm is described schematically in Fig. 5.6.

Definition 1:A low-resolution pixel y is influenced by a high-resolution pixel x, if x is in y 's receptive field.

Definition 2:A low-resolution image g is influenced by a high-resolution pixel x, if g contains a pixel y such that y is influenced by x.

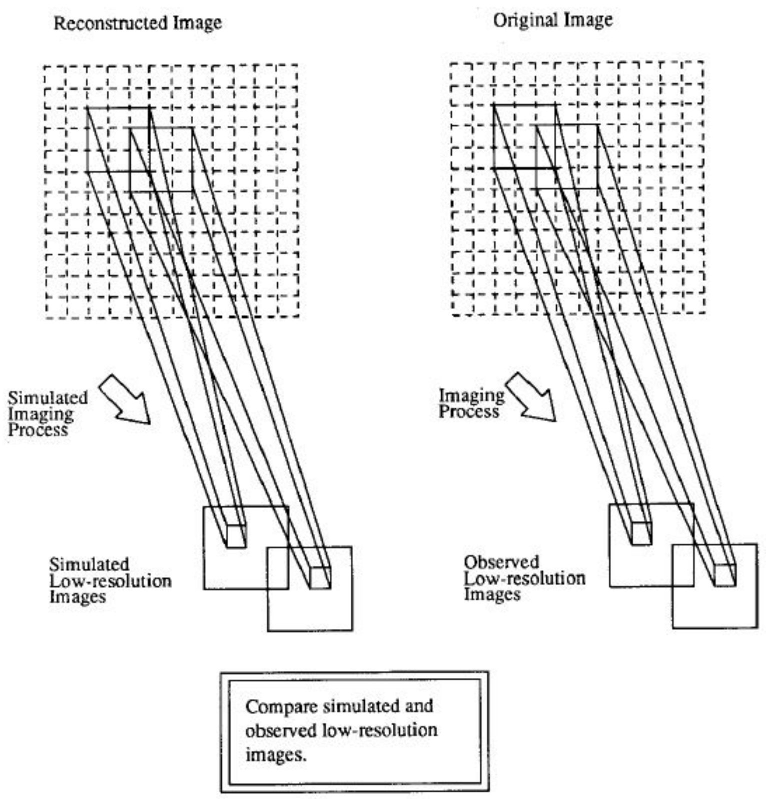

Reconstructed Image
Original Image
Simulated Imaging Process
Imaging Process
Simulated Low-resolution Images
Observed Low-resolution Images
Compare simulated and observed low-resolution images.

FIGURE 5.6: Schema Diagram

It is important to bear in mind that the original high- resolution frequencies may not always be fully restored. For example, if the blurring function is an ideal low-pass filter, and its Fourier transform has zero value at high frequencies, it is obvious that the frequency components which have been filtered out cannot be restored. In such cases, there is more than one high-resolution image which gives the same low-resolution images after the imaging process. Therefore, there are several possible solutions, and the algorithm may either converge to one of them or oscillate among some of them. The choice of initial guess does not influence the performance of the algorithm (speed or stability). It may, however, influence which of the possible solutions is reached first. A good choice of initial guess is the average of the low-resolution images. The average image is computed by registering all the low-resolution images over a fixed finer grid. Each high-resolution pixel in the fine grid is taken to be the average of all the low-resolution pixels stacked above it. Such an initial guess leads the algorithm to a smooth solution, which is usually a desired one.

This super-resolution algorithm performs well on both real and computer-simulated images. Improvement in resolution is clearly observed even when a very small number of low-resolution images are available. The algorithm converges rapidly (usually within less than five iterations) and is very stable. The complexity of the algorithm is low; there are O(KN minM, log N) operations per iteration, where N is the size of the high-resolution image f, M is the size of the blurring kernel, and K is the number of low-resolution pictures. Since the number of iterations is very small, this is also a good estimate of the complexity of the complete algorithm.The iterative update scheme to estimate the high-resolution image is expressed by

f(n+1) (x) = f$^n(x)+\Sigma(g_k(y)-g_k^n(y))(h_x^B)^2/c\Sigma_y h_x^B ......(8)$

where

c is a constant normalizing factor

h$_x^B = h^B(x-z_y)$

The algorithm has parallel characteristics; the contributions (to be averaged) of the low-resolution pixels to the high-resolution pixels, within a single iteration, may all be computed independently.Synchronization is needed only at the end of each iteration, when the values have to be averaged to obtain the new value.

5.5.4 Color Super-resolution

The color images are first transformed into YIQ representation, since in this representation most of the energy is concentrated in the luminance (Y) component,with little energy in the chrominance (I,Q) components. It is therefore sufficient to apply the iterative monochrome super-resolution algorithm only to the Y component, and use simpler processing of the I and Q components. In our experiments it was sufficient to average each of the chrominance component sequences after registration. Applying super-resolution to the Y component enables the mixing of color and gray-level images, as the gray-level images could be mixed with the Y component images.

The superresolution algorithm for color images is therefore the following:

Step 1: Transform the color images into YIQ representation.

Step 2: Apply the monochrome super-resolution algorithm to the Y component images.

Step 3: Register the images at the two chrominance image sequences using the same registration parameters obtained in Step 2 for the Y component. Create an average image for each of the 1 and Q components. Step 4. Use the high-resolution Y component and the low-resolution 1 and Q components to generate a high-resolution RGB image.

The sampling rate was doubled in both directions. Since it is hard to reproduce color images, only the green image is displayed.

5.5.5 Deblurring

Restoration of degraded images, when a model of the degradation process is given, is considered as an ill-conditioned problem. In this section deblurring of a single image is shown to be a special case of super-resolution, and convergence conditions of the algorithm with stability analysis are given for this case.

Deblurring a single image is achieved by applying the algorithm to a single input image, without increasing the sampling rate. One of the main benefits of the algorithm is in the freedom of choice of the auxiliary filter and the normalizing constant c so that it holds for as many frequencies Since the speed of convergence is accelerated as the term approaches 0, there is a trade-off between stability and speed of convergence.as possible, ensuring optimal conditions for convergence.Equation (8) then reduces to

f (n+1) (x) = f$^{n}(x)+\Sigma(g(y)-g^{n}(y))(h^{B}(x-y))^{2}/c......(9)$

Using convolutions,this can be written as

f(n+1) = f$^{n}+(g-g^{n})*h^{A}/c......(10)$

This method, therefore, has the advantage of being stable even in neighborhoods of zero-valued frequency components of the blur. This compared well with other deblurring methods, such as inverse filtering, which tend to amplify noise.

See *Appendix* for improvements in super-resolution.

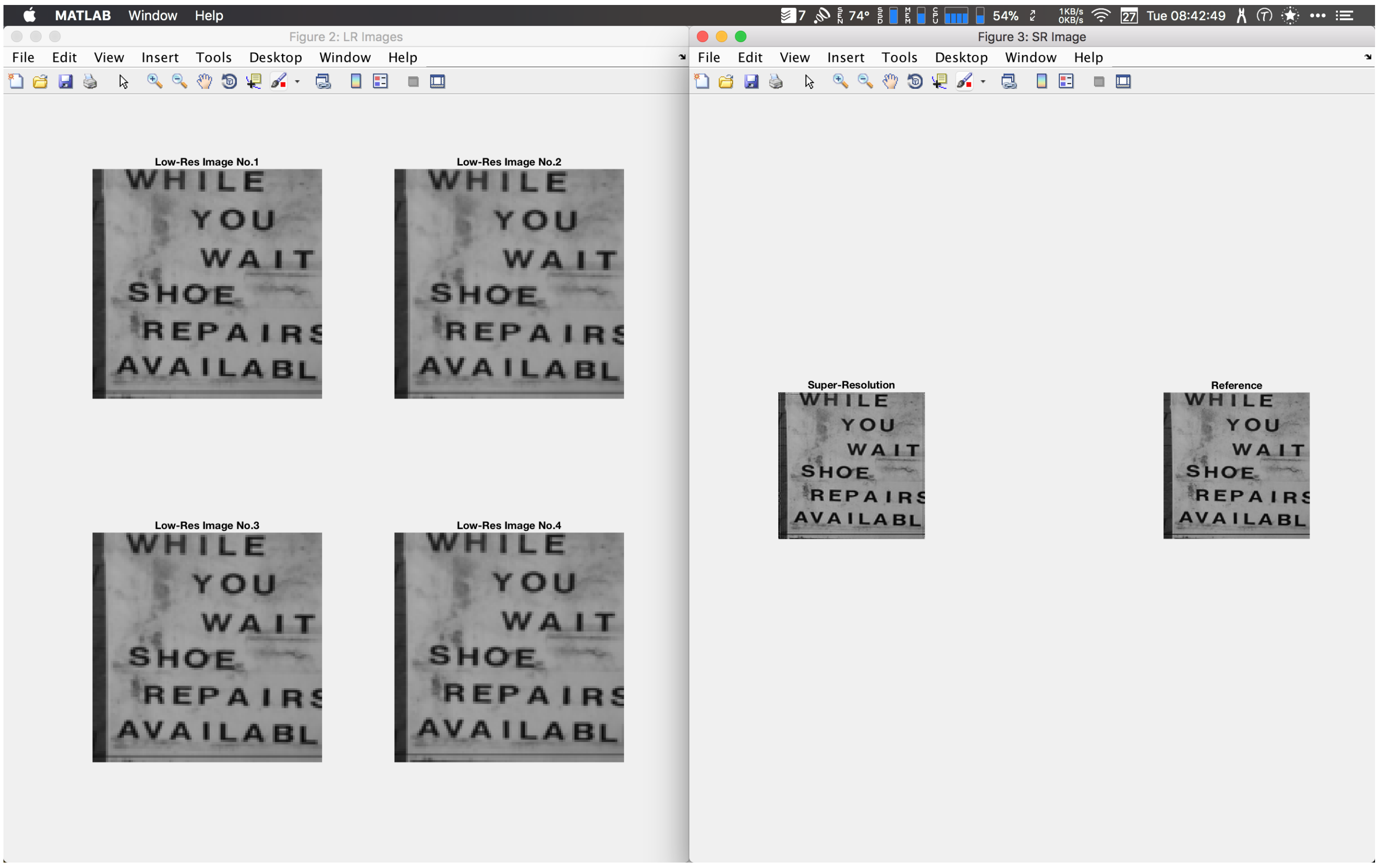


FIGURE 5.7: Left: A high-resolution reconstructed image is sought, which gives simulated low-resolution images that are as close as possible to the observed low-resolution images. Right:Shows reference and super-resoluted image

5.6 Text Recognition

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as ”pattern matching”, ”pattern recognition”, or ”image correlation”. This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new
]

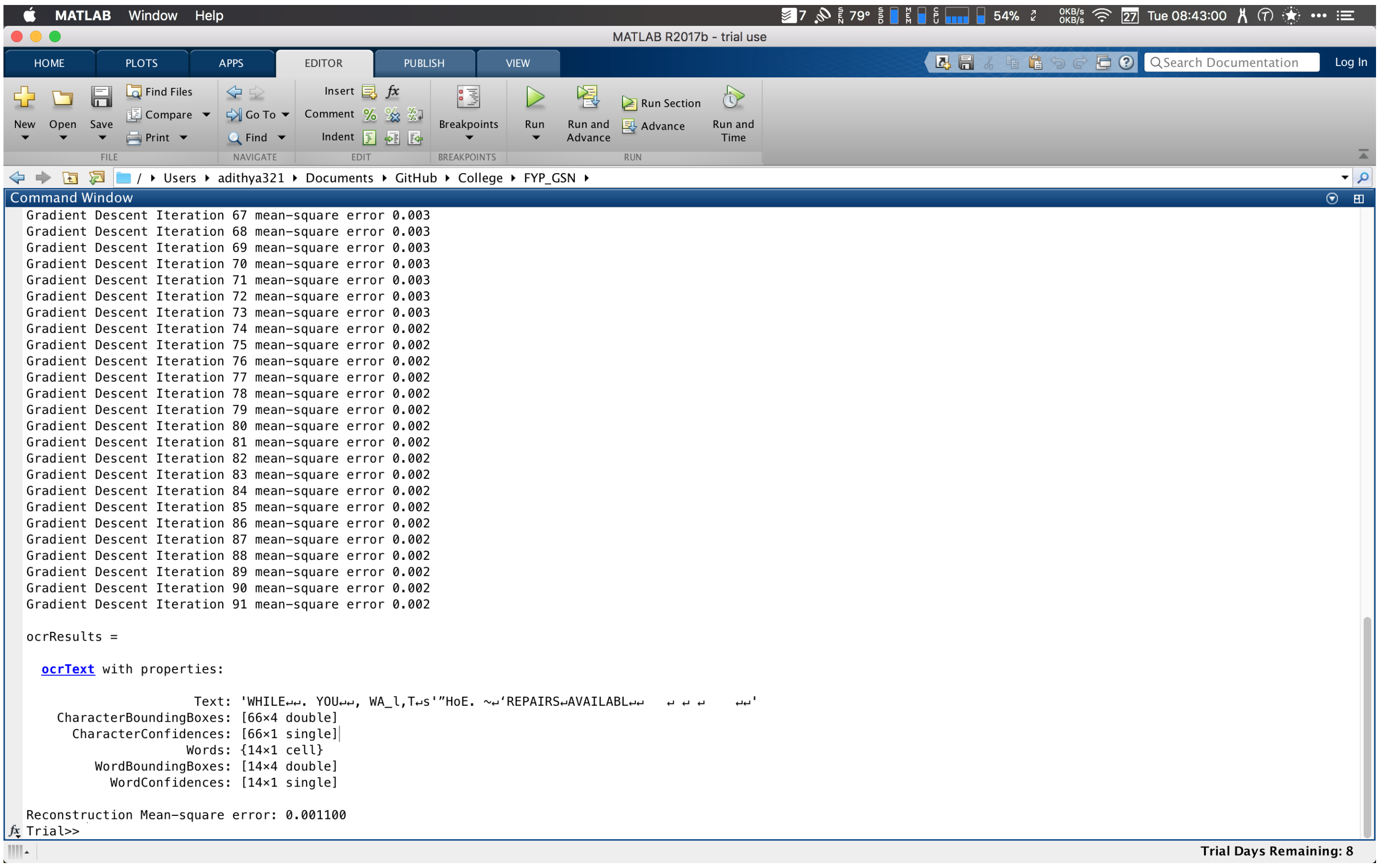
fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Feature extraction decomposes glyphs into ‘features’like lines, closed loops, line direction, and line intersections. The extraction features reduces the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in ”intelligent” handwriting recognition and indeed most modern OCR software.

OCR accuracy can be increased if the output is constrained by a lexicon a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

We use inbuilt OCR function to accept the natural-image as input and print the text in the console.



```
Gradient Descent Iteration 67 mean-square error 0.003
Gradient Descent Iteration 68 mean-square error 0.003
Gradient Descent Iteration 69 mean-square error 0.003
Gradient Descent Iteration 70 mean-square error 0.003
Gradient Descent Iteration 71 mean-square error 0.003
Gradient Descent Iteration 72 mean-square error 0.003
Gradient Descent Iteration 73 mean-square error 0.003
Gradient Descent Iteration 74 mean-square error 0.002
Gradient Descent Iteration 75 mean-square error 0.002
Gradient Descent Iteration 76 mean-square error 0.002
Gradient Descent Iteration 77 mean-square error 0.002
Gradient Descent Iteration 78 mean-square error 0.002
Gradient Descent Iteration 79 mean-square error 0.002
Gradient Descent Iteration 80 mean-square error 0.002
Gradient Descent Iteration 81 mean-square error 0.002
Gradient Descent Iteration 82 mean-square error 0.002
Gradient Descent Iteration 83 mean-square error 0.002
Gradient Descent Iteration 84 mean-square error 0.002
Gradient Descent Iteration 85 mean-square error 0.002
Gradient Descent Iteration 86 mean-square error 0.002
Gradient Descent Iteration 87 mean-square error 0.002
Gradient Descent Iteration 88 mean-square error 0.002
Gradient Descent Iteration 89 mean-square error 0.002
Gradient Descent Iteration 90 mean-square error 0.002
Gradient Descent Iteration 91 mean-square error 0.002

ocrResults =

  ocrText with properties:

                      Text: 'WHILE↵↵. YOU↵↵, WA_l,T↵s'"HoE. ~↵'REPAIRS↵AVAILABL↵↵   ↵ ↵ ↵    ↵↵'
    CharacterBoundingBoxes: [66×4 double]
      CharacterConfidences: [66×1 single]
                     Words: {14×1 cell}
         WordBoundingBoxes: [14×4 double]
           WordConfidences: [14×1 single]

Reconstruction Mean-square error: 0.001100
fx Trial>>
```

FIGURE 5.8: Gradient descent error for each iteration along with reconstruction mean-square error is calculated.Recognized text from super-resoluted image is displayed

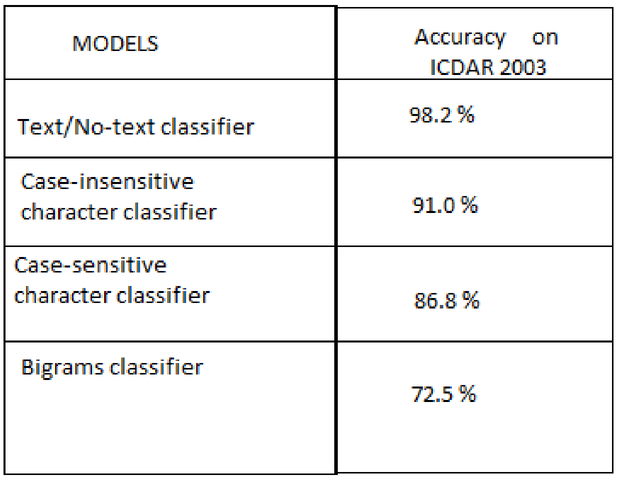
CHAPTER 6

Performance Evaluation

CNN Classifiers: The 37-way case-insensitive character classifier and case-sensitive classifier both achieve state-of-the-art performance, as does the text/no-text classifier used for detection. Our bigram classifier gives a recognition accuracy of 72.5 percent, a good result for a problem with 604 classes.

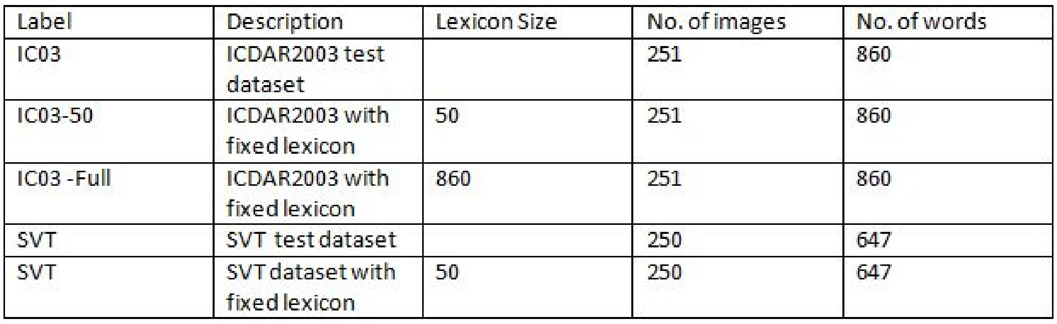
Although the CNNs are large (2.6 million parameters for the case-insensitive classifier), the method does not require any unsupervised pre-training as is done in [2], and incorporating the unsupervised approach described in [2,6] gives no improvement. Empirically, maxout and dropout were found to be essential to achieve this performance. For example, replacing maxout with ReLU non-linearities (this equates to reducing the number of filters to give the same layer output dimensionality) causes slight overfitting hence worse accuracy (-3.3 percent accuracy for case-insensitive character classification). We also found experimentally that pooling and downsampling have no effect on classifier accuracy.Sharing feature maps between tasks improves results compared to learning independent models: +3 percent accuracy for text/no-text, +1 percent accuracy for the case-sensitive character classifier, and +2.5 percent accuracy for the bigram classifier.

Including the FlickrType mined data also gives an extra 0.8 percent accuracy for the case-insensitive character classifier, illustrating the importance of more training data. On the contrary, learning on more synthetic data from Chars74k dataset(black and white renderings of different fonts) and Wang et al. [21] harmed recognition accuracy by causing the CNN to overfit to the synthetic data.



MODELS	Accuracy on ICDAR 2003
Text/No-text classifier	98.2 %
Case-insensitive character classifier	91.0 %
Case-sensitive character classifier	86.8 %
Bigrams classifier	72.5 %

FIGURE 6.1: Accuracy of Classifiers



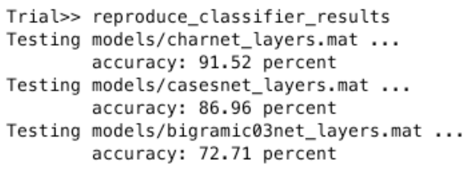
Label	Description	Lexicon Size	No. of images	No. of words
IC03	ICDAR2003 test dataset		251	860
IC03-50	ICDAR2003 with fixed lexicon	50	251	860
IC03 -Full	ICDAR2003 with fixed lexicon	860	251	860
SVT	SVT test dataset		250	647
SVT	SVT dataset with fixed lexicon	50	250	647

FIGURE 6.2: Description of Datasets

As per Fig 5.5, the accuracy of classifiers for 36-way character classification, 62-way case-sensitive character classification, 2-way text detection, and 604-way bigram classification on ground-truth cropped patches. For the SVT dataset the character-level annotation is automatically generated by our system.

Fixed Lexicon Cropped Word Recognition: For each word, a set of hypothesis is formed adding to the ground-truth text a small number of distractors. These distractors are: in IC03-full the full lexicon,in IC03-50 the 50 words from [21], and in SVT, 50 selected words.

Our word recognition system gives state of the art accuracy on the ICDAR2003



```
Trial>> reproduce_classifier_results
Testing models/charnet_layers.mat ...
        accuracy: 91.52 percent
Testing models/casesnet_layers.mat ...
        accuracy: 86.96 percent
Testing models/bigramic03net_layers.mat ...
        accuracy: 72.71 percent
```

FIGURE 6.3: Case-insensitive character classifier (models/charnetlayers.mat); Case-sensitive character classifier (models/casesnetlayers.mat); ICDAR 2003 test-bigrams classifier (models/bigramic03netlayers.mat)

benchmarks, improving on state of the art by 3.1 percent for the 50 word lexicon and 2.4 percent for the full lexicon. The total recognition accuracy of 96.2 percent for the 50 word lexicon makes only 33 mistakes out of 860 test images, and many of the misclassified examples can be very difficult to classify even for a human.

On the SVT dataset, we achieve an accuracy of 86.1 percent, which while 4.3 percent off state-of-the-art, improves on the next best result by 8.8 percent. This is a competitive result considering our method is trained on two orders of magnitude less data,and so must use a smaller model than the state-of-the-art PhotoOCR method.

End-to-End Word Recognition: The results of our end-to-end system are evaluated on the ICDAR 2003 test set with different sized lexicons and the SVT dataset (Fig. 5.7). A recognition result is considered to be correct if the bounding box has at least 50 percent overlap with the ground truth and the text is correct. The detector described in Sect. 5.4.1 is tuned for high recall and generates word bounding boxes with localization P/R (precision/recall) of 0.19/0.80 (IC03) and 0.04/0.72 (SVT). After word recognition, detection results are re-ranked by word recognition score, P/R curves generated, and the P/R/F-measure at the maximum F-measure working point is reported [21].

Our end-to-end pipeline outperforms previous works, with P/R/F-measure of 0.90/0.73/0.80 for IC03-50,0.89/0.66/0.75 for IC03-Full, and 0.73/0.45/0.56 for

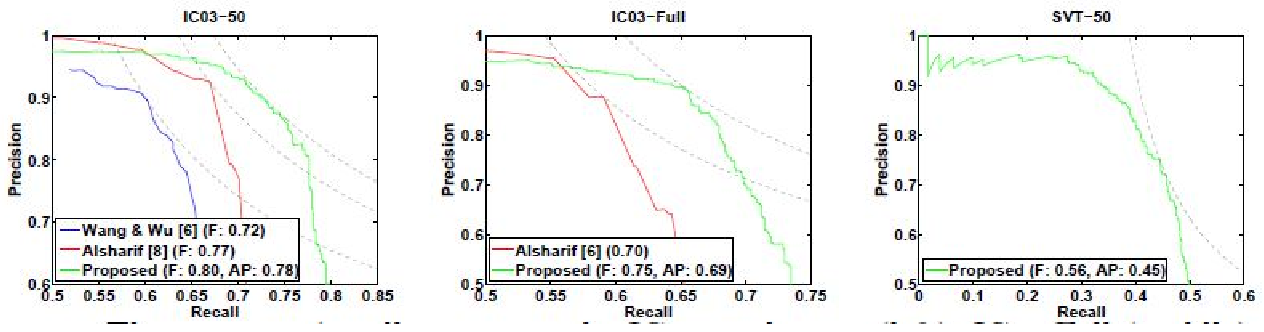


FIGURE 6.4: Precision recalls

SVT-50. Interestingly, due to the fact that our pre-recognition bounding boxes are generated by a detector trained from the same data as the character recognizer, we find that the difference between the localization and recognition scores to be inline with the cropped word recognition results: at maximum recall, 95 percent of correctly localized words are subsequently correctly recognized in IC03-50. When removing the bigram response maps from the word recognition process, F-measure drops significantly from 0.8 to 0.76 for IC03-50 and from 0.56 to 0.50 for SVT-50.

CHAPTER 7

Conclusion and Future Work

We have presented a text spotting system using a single set of rich, learnt features, that achieve state-of-the-art performance on a number of benchmarks. These results illustrate the power of jointly learning features to build multiple strong classifiers as well as mining additional training data in publicly available resources such as Flickr. This framework is not only limited to fixed lexicon recognition,as the major contributions are agnostic as to whether a lexicon is used or not.

Super-resolution is shown to be feasible for mono-chrome and color image sequences, when the relative displacements can be computed, and with approximate knowledge of the imaging process. An iterative algorithm for computing super-resolution has been presented. It was shown that when the algorithm is applied to a single image without increasing the sampling rate, super-resolution reduces to deblurring.

The suggested algorithm performed well for both computer-simulated and real images, and has been shown, theoretically and practically, to converge quickly. The algorithm can be executed in parallel for faster hardware implementation. Accurate knowledge of the relative displacements of scene regions is essential when using image sequences. We have assumed a simple uniform motion of translation and rotation for the entire image, and implemented an accurate method for computing this displacement. This method, however, can also be applied to other types of motion, such as perspective transformation and multiple motions in the image. As long as the image can be divided into regions such that each region undergoes some uniform motion, resolution can be improved in the regions.

All images used in this paper were digitized using uniform sampling. This is, however, not necessary; the process can be applied to images sampled in any arbitrary, nonuniform manner. Samples not on a uniform sampling grid, as well as blur that varies between sample locations, can be accommodated.

Future work: One additional potential advantage,to be explored in future work, is that by implementing sliding window detection as a byproduct of the convolutional network, our method allows the use of CNN speedup methods to dramatically accelerate detection. In addition, it would be interesting to continue classifier learning by self-supervision using the system to continually and automatically mine more data, and find other sources for this.

In future, we will continue to improve the text recognition accuracy by incorporating with a high level language model while maintaining its efficiency. These ideas can also be used to address deep recurrent neural networks, especially for long short-term memory, as they are viable deep-learning models to deal with such time sequence-based problems. We have planned to investigate various filter-operations in CNNs, intended to further improve system efficiency in applications like pattern analysis and machine intelligence.

Appendix A

Super- Resolution

A.1 Heuristic Improvements

A.1.1 Fixing Stable Pixels

To increase speed and stability of the algorithm, a high-resolution pixel is fixed when it is assigned the same value (or nearly the same value) two iterations in a row. This pixel will not be considered again in future iterations.

This fixing process increases the speed of convergence, since at later iterations fewer pixels are examined. It also prevents harmful salt-and-pepper type of noise from contaminating large surrounding areas. It is unlikely for a noisy pixel to be fixed as its gray level usually differs from those of its neighboring pixels.

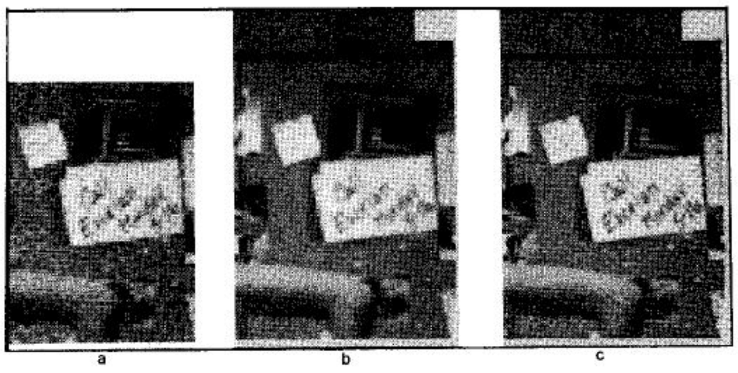


FIGURE A.1: SR of monochrome images

A.1.2 Noise Reduction

Referring back to the updating scheme of Eqn. (6), the new value of a high-resolution pixel in each iteration is computed by taking the average of all contributions of the various low-resolution pixels. Taking an average in itself already handles additive noise. In order to handle multiplicative noise as well, contributions having extreme (high and low) values are eliminated before taking the average.Only contributions whose values are neither maximal nor minimal are averaged, eliminating both additive and multiplicative noise.For such noise cleaning a reasonable number of low-resolution images is needed.

Figure A.1 shows the result of applying the algorithm to 10 monochrome low-resolution images, contaminated by zero-mean Gaussian noise of standard deviation 10. The sampling rate was doubled in each direction.

REFERENCES

1. Max Jaderberg, Andrea Vedaldi, Andrew Zisserman:Deep Features for Text Spotting.s. In: NIPS,European Conference on Computer Vision(2014)

2. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Proc. CVPR. pp. 2963 - 2970. IEEE (2010)

3. Anthimopoulos, M., Gatos, B., Pratikakis, I.: Detection of artificial and scene text in images and video frames. Pattern Analysis and Applications pp. 1 - 16 ,(2011)

4. Alsharif, O., Pineau, J.: End-to-End Text Recognition with Hybrid HMM Maxout Models. ICLR (2014)

5. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: Pattern Recognition (ICPR), 2012 21st International Conference on. pp. 3304 - 3308. IEEE (2012)

6. Mathieu, M., Henaff, M., LeCun, Y.: Fast training of convolutional networks through FFTs. CoRR abs/1312.5851 (2013)

7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. vol. 1, p. 4 (2012)

8. Michal Irani and Shamuel Peleg:Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.:Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (NIPS 2012)

9. Jaehwa Park, V. Govindaraju, S.N. Srihari: OCR in a hierarchical feature space.In:IEEE(2014)

10. . Shahab, A., Shafait, F., Dengel, A.: Reading text in scene images. In: Proc. ICDAR. pp. 1491 - 1496. IEEE(2011)

11. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Proc. CVPR. vol. 3, pp. 1187 - 1190. IEEE (2012)

12. Chen, H., Tsai, S., Schroth, G., Chen, D., Grzeszczuk, R., Girod, B.: Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: Proc. International Conference on Image Processing (ICIP). pp. 2609 - 2612 (2011).

13. Mishra, A., Alahari, K., Jawahar, C., et al.: Scene text recognition using higher order language priors. In: BMVC 2012-23rd British Machine Vision Conference(2012)

14. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing features: efficient boosting procedures for multiclass object detection. In: Proc. CVPR. pp. 762 - 769 (2004)

15. K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In: ECCV, 2014.

16. M.Elad and A,Feuer:Super-Resolution Reconstruction of an Image. In: IEEE(2011)

17. Neumann, L., Matas, J.: Text localization in real-world images using efficiently pruned exhaustive search. In: Proc. ICDAR. pp. 687 – 691. IEEE (2011)

18. Goel, V., Mishra, A., Alahari, K., Jawahar, C.: Whole is greater than sum of parts:Recognizing scene text words. In: Document Analysis and Recognition (ICDAR),2013 12th International Conference on. pp. 398 - 402. IEEE (2013)

19. Andrei Polzounov, Artsiom Ablavatski, Sergio Escalera, Shijian Lu, Jianfei Cai,::WordFence: Text Detection in Natural Images with Border Awareness,arXiv:1705.05483v1,IEEE 2017

20. https://www.flickr.com/

21. https://www.flickr.com/groups/types