Smart Library: Recognizing Books in a Library using Deep Learning Techniques

A B.Tech Dissertation report submitted in fulfilment of the requirements for the degree of Bachelor of Technology

by

ADARSH KUMAR JAIN (2015UCP1547),

RAVI KUMAR VERMA (2015UCP1536),

AMIT KUMAR MEENA (2015UCP1734)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

May 2019

Certificate

We,

ADARSH KUMAR JAIN (2015UCP1547),

RAVI KUMAR VERMA (2015UCP1536),

AMIT KUMAR MEENA (2015UCP1734),

Declare that this thesis titled, “Smart Library: Recognizing Books in a Library using Deep Learning Techniques” and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).
- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed.Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.
- We have acknowledged all main sources of help.

Signed:

____

Date:

____

Dr. Neeta Nain

May 2019

Professor/Associate Professor/ Assistant Professor

Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

Name of the student: **ADARSH KUMAR JAIN (2015UCP1547), RAVI KUMAR VERMA (2015UCP1536), AMIT KUMAR MEENA (2015UCP1734)**

Degree for which submitted: **B.Tech.** Department: **Computer Science and Engineering**

Thesis title: **Smart Library: Recognizing Books in a Library using Deep Learning Techniques**

Thesis supervisor: **Dr. Neeta Nain**

Month and year of thesis submission: **May 2019**

Physical library collections are valuable and long standing resources for knowledge and learning. However, managing books in a large bookshelf and finding books on it often leads to tedious manual work, especially for large book collections where books might be missing or misplaced. Recently, deep neural models, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have achieved great success for scene text detection and recognition. Motivated by these recent successes, we aim to investigate their viability in facilitating book management, a task that introduces further challenges including large amounts of cluttered scene text, distortion, and varied lighting conditions.

We present a library inventory building and retrieval system based on scene text reading methods. Our proposed system has the potential to greatly reduce the amount of human labor required in managing book inventories as well as the space needed to store book information.

Acknowledgements

We would like to give our sincere thanks and gratitude to our esteemed mentor

Dr. Neeta Nain for providing her valuable guidance and encouragement to learn new things and enhancing our knowledge in the field of machine learning and sound processing. Her kind cooperation and suggestions throughout the course of this research guided us with an impetus to work, and successfully complete the project.

We express our sincere gratitude to Prof. Girdhari Singh, HOD, Department of Computer Science and Engineering, MNIT, Jaipur for all the department facilities provided to us.

We would also like to thank Prof. U.R.Yaragatti Director, Malaviya National Institute of Technology, Jaipur for providing all the institutional facilities.

Adarsh Kumar Jain

Ravi Kumar Verma

Amit Kumar Meena

Contents

List of Figures

Chapter 1

Introduction

Despite the increasing availability of digital books in the modern age of information, many people still favor reading physical books. The large libraries that house them require great amount of time and labor to manage inventories that number in the millions. Manually searching bookshelves is time-consuming and often not fruitful depending on how vague the search is. To solve this issue, research groups have developed automated management systems to identify books on the shelves. One way to recognize the books is to tag each book with an identier such as an RFID or barcode and read the tag using a specialized reader. Another way is to use images from a digital camera for identifying books. Deploying camera-based book recognition solutions is more cost-effective because there is no need to attach physical tags to individual books. We propose a deep neural network-based system that can viably solve this problem.

Our work has two goals: 1) build a book inventory from only the images of bookshelves; 2) help users quickly locate the book they are looking for. Once the book is identied, additional stored information could also be retrieved, e.g. a short description of the book. The intent of this work is to make what was previously a tedious experience (i.e., searching books in bookshelves) much more user-friendly. In addition, our system also offers the potential for efciently building a large database of book collections.

The main contributions of our work are as follows:

- Design and implementation of a book spine recognition system which achieves superior recognition accuracy compared to low-level image feature-based recognition system.

- Development of a method to robustly recognize spine texts from bookshelf images for spine recognition: We extract book spines from the bookshelf image. Text is localized using a deep learning model and recognized using Optical Character Recognition (OCR) and a Convolutional Recurrent Neural Network (CRNN). Recognized texts are used as keywords to search from the online book databases.

1.1 Objective

Deep learning has achieved great success in both research and application during the past decade. Today Deep Learning is used in autonomous vehicles, Retail, Personal Assistance, Language Processing and many more. Our objective in this project is to use the Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in Deep learning to solve the problem of manually generating book inventories in libraries and searching of a book in the library.

1.2 Report Structure

In the remaining portion of the report, chapter-2 tells about the libraries and frameworks used for the project. Chapter-3 explains various algorithms and deep learning models used throughout in the program. Chapter-4 gives the details of what we have to study for the project. Chapter-5 introduces to our proposed work. Chapter-6 shows the results and discussion. Chapter-7 sums up the report and gives possible future work that can be done.

Chapter 2

Libraries And Frameworks Used

In computer science, a library is a collection of non-volatile resources used by computer programs, often for software development. These may include configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications.

A framework, or software framework, is a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform. For example, a framework may include predefined classes and functions that can be used to process input, manage hardware devices, and interact with system software. This streamlines the development process since programmers don't need to reinvent the wheel each time they develop a new application.

2.1 Libraries

2.1.1 OpenCV-Python

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

2.1.2 SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms, and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multidimensional container of data with arbitrary datatypes. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favor of the newer NumPy array code.

2.1.3 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

2.2 Frameworks

2.2.1 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. We can use Keras if we need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

User friendliness - Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent and simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

Modularity - A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.

Easy extensibility - New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

Work with Python - No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Chapter 3

Algorithms and Deep Learning Models

3.1 Algorithms

In computer science, an algorithm is an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing, automated reasoning, and other tasks.

The Following Algorithms are used in our project : -

3.1.1 Canny Edge Detection Algorithm

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages.

1. **Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

2. **Finding Intensity Gradient of the Image**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction ( ) and vertical direction ( ). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3. **Non-maximum Suppression** After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

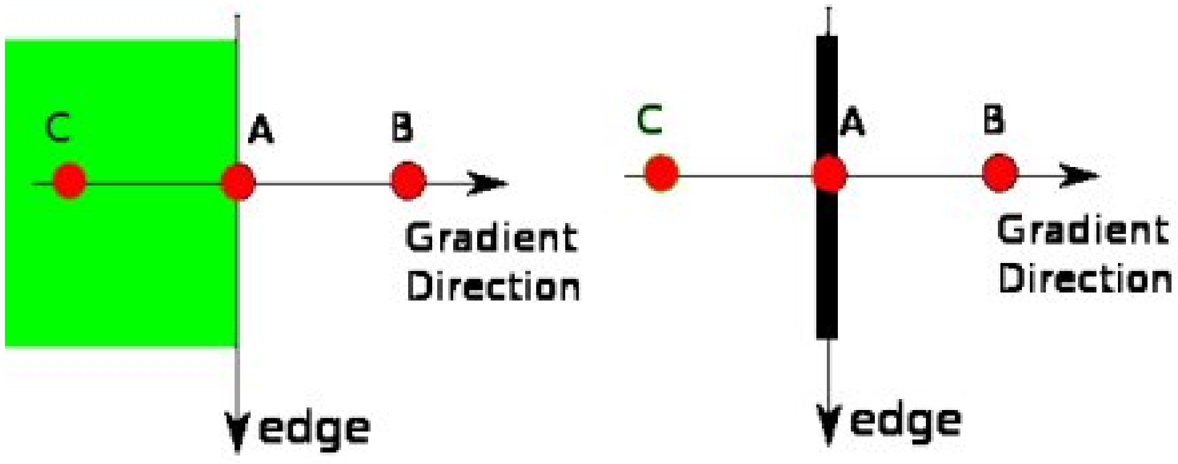


FIGURE 3.1: Non-Maximum Suppression

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with “thin edges”.

4. **Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to “sure-edge” pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

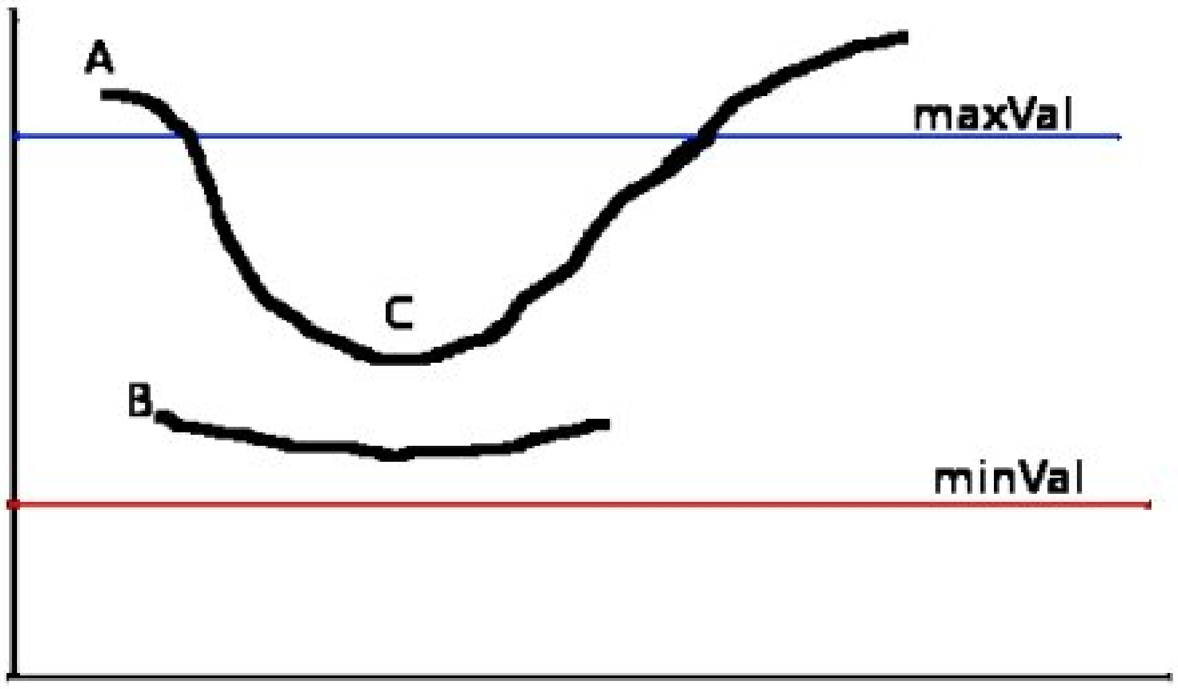


Figure 3.2: Hysteresis Thresholding

The edge A is above the maxVal, so considered as “sure-edge”. Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any “sure-edge”, so that is discarded. So it is very important that we have to select minVal and maxValaccordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

3.1.2 Hough Transform

The Hough Transform (HT) is a robust method for finding lines in images that was developed by Paul Hough.

The main idea for the HT is as follows:

1. For each line L, there is a unique line.

2. L?perpendicular to L which passes through the origin.L?has a unique distance and angle from the horizontal axis of the image. This angle and distance define a point in the parameter space, sometimes known as Hough space.

3. A point in image space has an infinite number of lines that could pass through it, each with a unique distance and angle.

4. This set of lines corresponds to a sinusoidal function in parameter space. Two points on a line in image space correspond to two sinusoids which cross at a point in parameter space.

5. That point in parameter space corresponds to that line in image space, and all sinusoids corresponding to points on that line will pass through that point.

The real solution to implement this algorithm is to quantize the parameter space by using a 2D array of counters, where the array coordinates represent the parameters of the line; this is commonly known as an accumulator array.

The HT method for finding lines in images generally consists of the following three stages:

1. Perform accumulation on the accumulator array using the binary edge image.

2. Find peak values in the accumulator array

3. Verify that the peaks found correspond to legitimate lines, rather than noise.

3.2 Deep Learning Models

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised.

3.2.1 InceptionV3

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1 percent accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

A high-level diagram of the model is shown below:

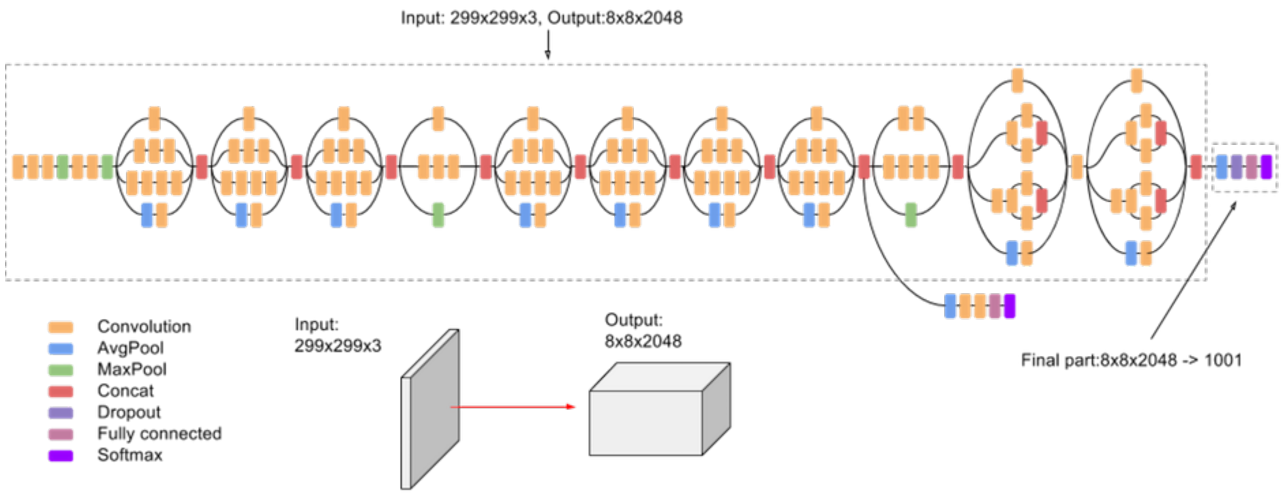


FIGURE 3.3: InceptionV3 Model

3.2.2 ResNet50

ResNet-50 is a convolutional neural network that is trained on more than a million images from the ImageNet database [1]. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

3.2.3 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7 percent top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 33 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

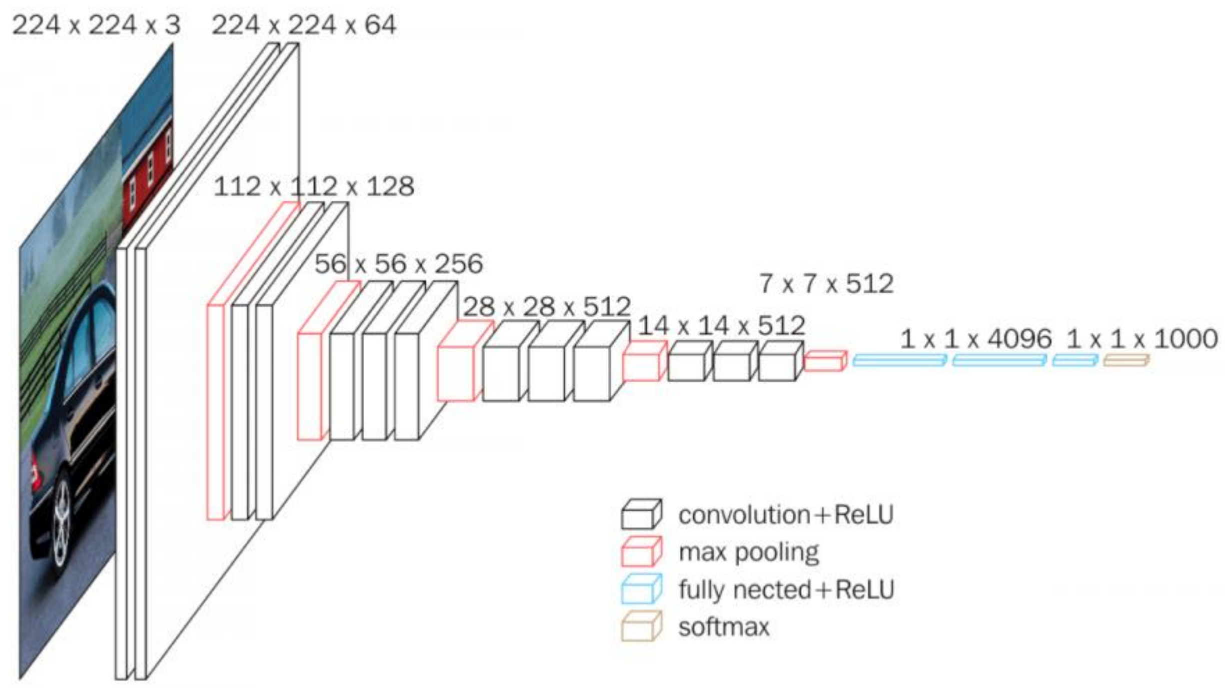


FIGURE 3.4: VGG16 Model

3.2.4 VGG19

VGG-19 is a convolutional neural network that is trained on more than a million images from the ImageNet database [1]. The network is 19 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

Chapter 4

Literature

4.1 Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

The major advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

4.2 Convolutional neural network (CNN)

A convolutional neural network (CNN) contains one or more convolutional layers, pooling or fully connected, and uses a variation of multilayer perceptrons . Convolutional layers use a convolution operation to the input passing the result to the next layer. This operation allows the network to be deeper with much fewer parameters. Convolutional neural networks show outstanding results in image and speech applications.

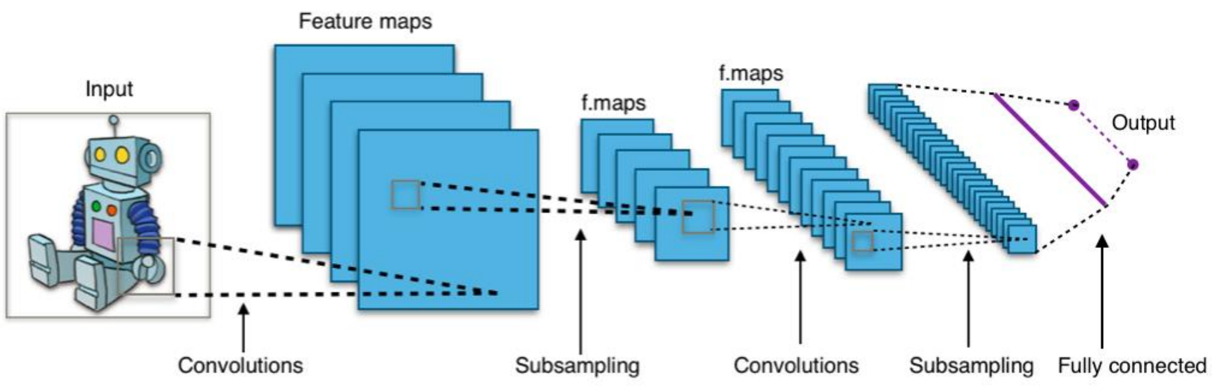


Figure 4.1: Typical CNN

4.3 Recurrent neural network (RNN)

A recurrent neural network (RNN), unlike a feed forward neural network, is a variant of a recursive artificial neural network in which connections between neurons make a directed cycle. It means that output depends not only on the present inputs but also on the previous step's neuron state. This memory lets users solve NLP problems like connected handwriting recognition or speech recognition. In a paper, Natural Language Generation, Paraphrasing and Summarization of User Reviews with Recurrent Neural Networks, authors demonstrate a recurrent neural network (RNN) model that can generate novel sentences and document summaries.

4.3.1 Long short-term memory (LSTM) Network

Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs . LSTM does not use activation function within its recurrent components, the stored values are not modified, and the gradient does not tend to vanish during training. Usually, LSTM units are implemented in "blocks" with several units. These blocks have three or four "gates" (for example, input gate, forget gate, output gate) that control information flow drawing on the logistic function.

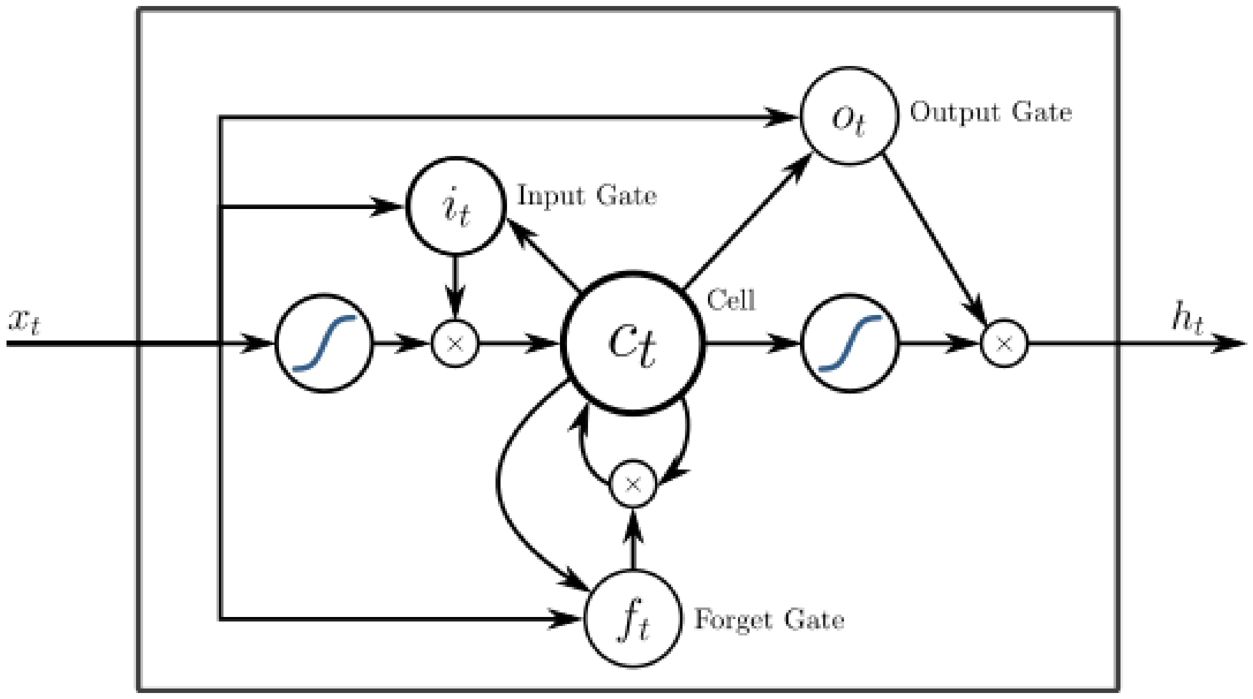


Figure 4.2: LSTM block with input, output, and forget gates

Chapter 5

Proposed Work

The proposed work mainly contains three steps - segmenting book spines, apply text detection procedure on these extracted spines and then recognize all the extracted text regions using a deep learning model

We also evaluate the performance of image feature-based system. Previous work has been done using image features but that is mainly based on low-level feature extraction method. We propose a image-feature based method using neural networks.

After text recognition, the recognized keywords are searched through online book databases to get the information about query spine image. The returned results are matched with the recognized one using similarity measures and then book are marked as correctly identified or incorrectly identified.

5.1 Book Spine Segmentation

Book spine segmentation is a critical component of our system since each book is expected to be recognized, stored, and queried independently. Recognizing book spines in a photo of a bookshelf is very challenging because each spine has a small surface area containing few image features and the other spines' image features act as clutter in feature-based image matching.

To segment out individual book spines in the query image, we exploit the fact that often books are densely packed together on a bookshelf and have roughly the same orientation. First, a Canny edge map is extracted from the query image. In the Canny edge map, the crevices between book spines appear as long and straight edges, while the text and graphics on the spines appear as short and curved edges.

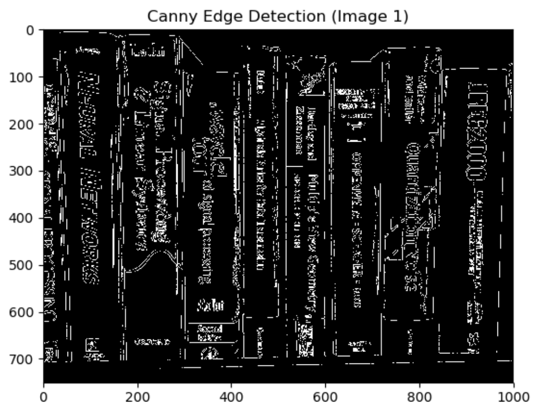


FIGURE 5.1: Canny Edge Detection

Connected components are detected in the edge map, and all components containing fewer pixels than a lower threshold are removed.

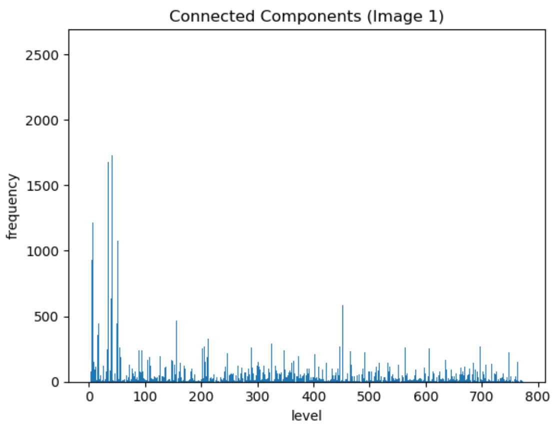


FIGURE 5.2: Remove Short Clusters

Next, a Hough transform is calculated from the edge map with short edges removed. It is found that removal of short edges improves the resolution of peaks in the Hough transform. Fig 5.3 displays the Hough transform calculated for the query image.

To estimate the dominant angle of the spines, the Hough peak occurrences are binned in an angular histogram as shown. The bin attaining the highest count in the angular histogram yields the spines' dominant angle.

Hough transfrom gives multiple lines. So the lines which are closer, are merged into a single one.

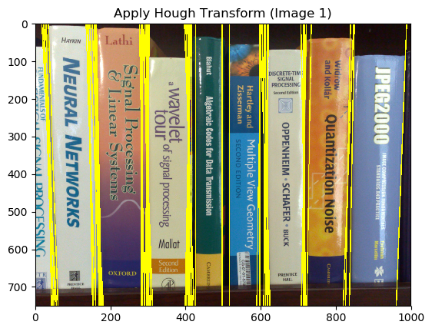


Figure 5.3: Hough Transfrom

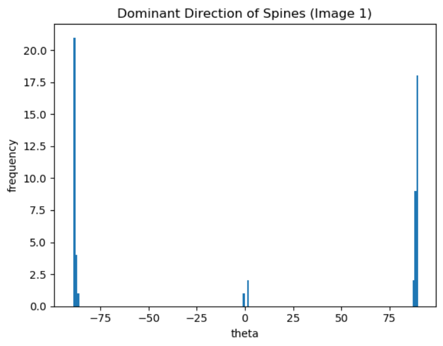


Figure 5.4: Dominant Direction

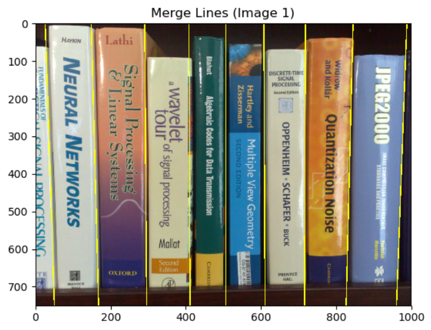


Figure 5.5: Dominant Direction

We illustrate the steps to extract the individual book spines in Fig 5.6.

5.2 Book Spine Recognition

We use two recognition system that consists of a text-based recognition and an image feature-based recognition. From each book spine image, we detect and recognize the text on the spines and use it as keywords to search a book spine text database. Similarly, image features are extracted from the book spine images and matched to a already extracted book spine image features.

5.2.1 Recognizing Spines with Image Features

A common and highly effective approach to deep learning on small image datasets is to leverage a pre-trained network. A pre-trained network is simply a saved network previously trained on a large dataset, typically on a large-scale image classification task. If this original dataset is large enough and general enough, then the spatial feature hierarchy learned by the pre-trained network can effectively act as a generic model of our visual world, and hence its features can prove useful for many different computer vision problems, even though these new problems might involve completely different classes from those of the original task. For instance, one might train a network on ImageNet (where classes are

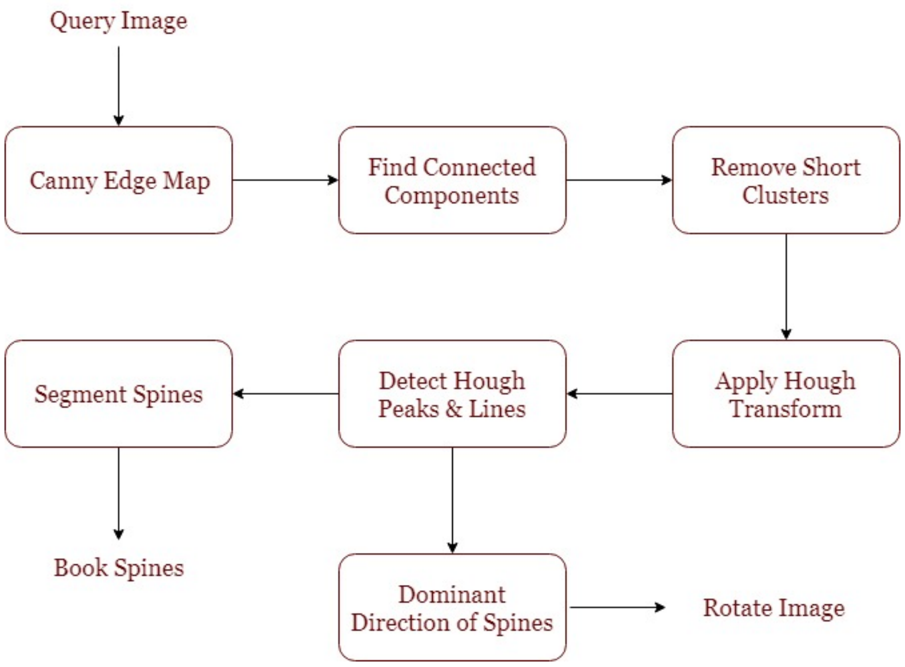


FIGURE 5.6: Operations for Book Spine Segmentation

mostly animals and everyday objects) and then re-purpose this trained network for something as remote as identifying books in images. Such portability of learned features across different problems is a key advantage of deep learning compared to many older shallow learning approaches, and it makes deep learning very effective for small-data problems.

Feature extraction consists of using the representations learned by a previous network to extract interesting features from new samples. These features are then run through a new classifier, which is trained from scratch.

As we saw previously, convnets used for image classification comprise two parts: they start with a series of pooling and convolution layers, and they end with a densely-connected classifier. The first part is called the "convolutional base" of the model. In the case of convnets, "feature extraction" will simply consist of taking the convolutional base of a previously-trained network, running the new data through it, and training a new classifier on top of the output.

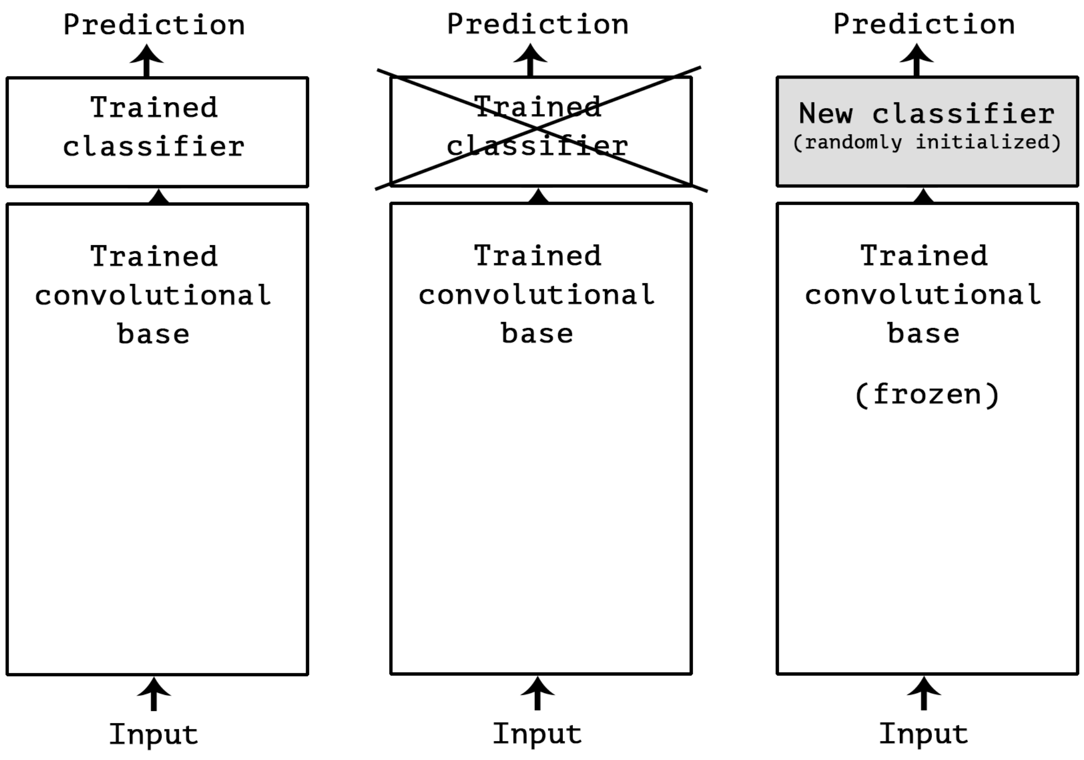

Prediction
Trained classifier
Trained convolutional base
Input
Prediction
~~Trained classifier~~
Trained convolutional base
Input
Prediction
New classifier (randomly initialized)
Trained convolutional base (frozen)
Input

Figure 5.7: Feature Extraction

The representations learned by the convolutional base are likely to be more generic and therefore more reusable: the feature maps of a convnet are presence maps of generic concepts over a picture, which is likely to be useful regardless of the computer vision problem at hand. On the other end, the representations learned by the classifier will necessarily be very specific to the set of classes that the model was trained on – they will only contain information about the presence probability of this or that class in the entire picture.

Here we use four different models for feature extraction:

- VGG16
- VGG19
- ResNet50
- InceptionV3

First, features from book spine database are extracted using all these four models. Then we applied dimensionality reduction technique to visualize these extracted features as shown in fig 5.8 and fig 5.9.

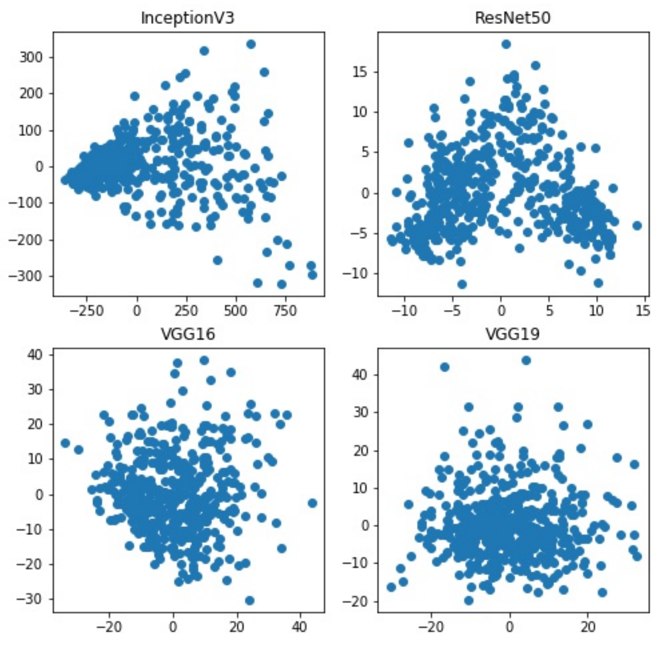


FIGURE 5.8: PCA Algorithm

Fig. 5.10 illustrates the steps used in the recognition process.

5.2.2 Recognizing Spines with Text

The text on the book spines typically contains the title and the author names, which can provide effective keywords to search for the book. To use the text on the book spines, the text within the extracted book spine image has to be automatically recognized.

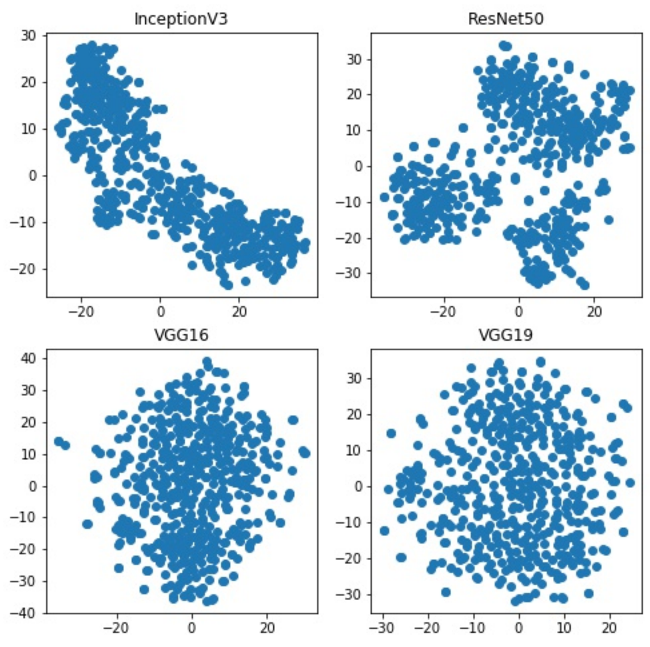


FIGURE 5.9: tSNE Algorithm

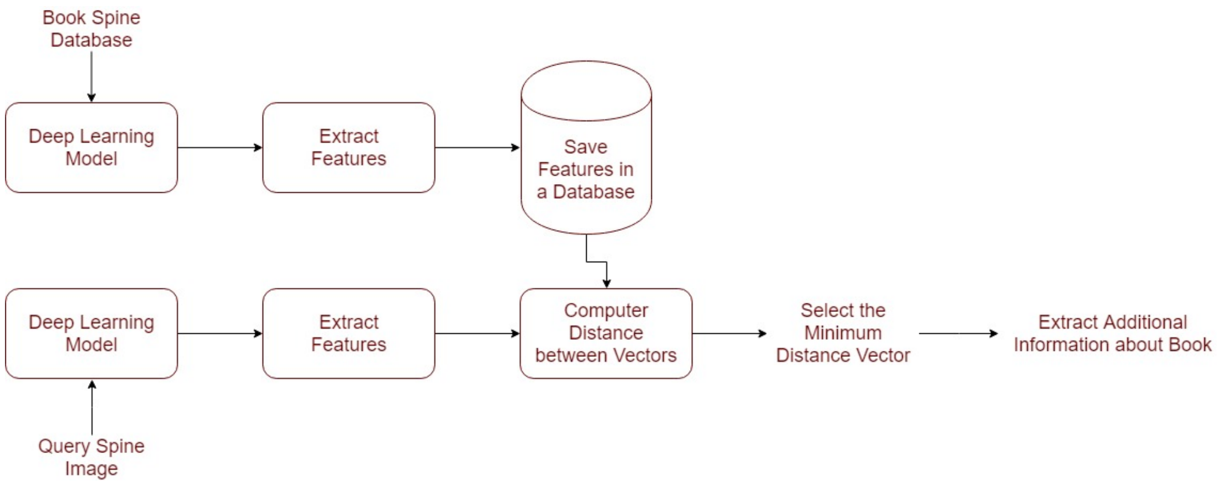

Book Spine Database
Deep Learning Model
Extract Features
Save Features in a Database
Deep Learning Model
Extract Features
Computer Distance between Vectors
Select the Minimum Distance Vector
Extract Additional Information about Book
Query Spine Image

FIGURE 5.10: Operations for Book Spine Recognition

5.2.2.1 Text Detection

The key component of the the algorithm is a neural network model, which is trained to directly predict the existence of text instances and their geometries from full images. The model is a fully-convolutional neural network adapted for text detection that outputs dense per-pixel predictions of words or text lines. This eliminates intermediate steps such as candidate proposal, text region formation and word partition. The post-processing steps only include thresholding and NMS on predicted geometric shapes.

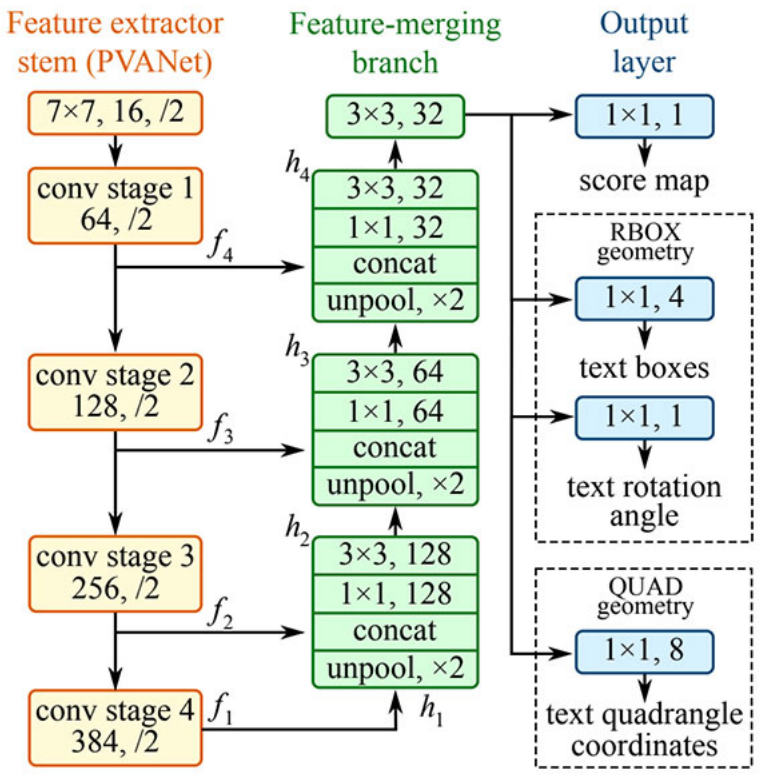


Figure 5.11: Text Detection Model

Network architecture is illustrated in Fig. 5.11. The algorithm follows the general design of DenseBox, in which an image is fed into the FCN and multiple channels of pixel-level text score map and geometry are generated. One of the predicted channels is a score map whose pixel values are in the range of [0,1]. The remaining channels represent geometries that encloses the word from the view of each pixel. The score stands for the condence of the geometry shape predicted at the same location. Thresholding is then applied to each predicted region, where the geometries whose scores are over the predened threshold is considered valid and saved for later nonmaximum-suppression. Results after NMS are considered the nal output of the pipeline.

Because of the multi-oriented nature of the text lines in book spines, we need to further decide whether a text line is upside down or not. To address this issue, we train a CNN classier on 32 64 image patches. The binary classier tells us whether we need to ip the text lines in order to provide the text recognition module with a correct sequence. During testing, we rst resize the height of all the text lines to 32 pixels. Then pass it through the trained network.

5.2.2.2 Text Recognition

In the text recognition step, a common approach is to rst segment and recognize each character, then output a word level prediction based on a language model or a combination of heuristic rules.However,these approaches are highly sensitive to various distortions in images, making character level segmentation imperfect and sometimes even impossible. To bypass the segmentation step, our model is trained to recognize a sequence of characters simultaneously. Similar to He et al. and Shi, Bai, and Yao (2015), we adopt a hybrid approach that combines a CNN with an RNN, casting scene text recognition problem as a sequential labeling task. A CNN at the bottom learns features from the images which are then composed as feature sequences that are subsequently fed into an RNN for sequential labeling.

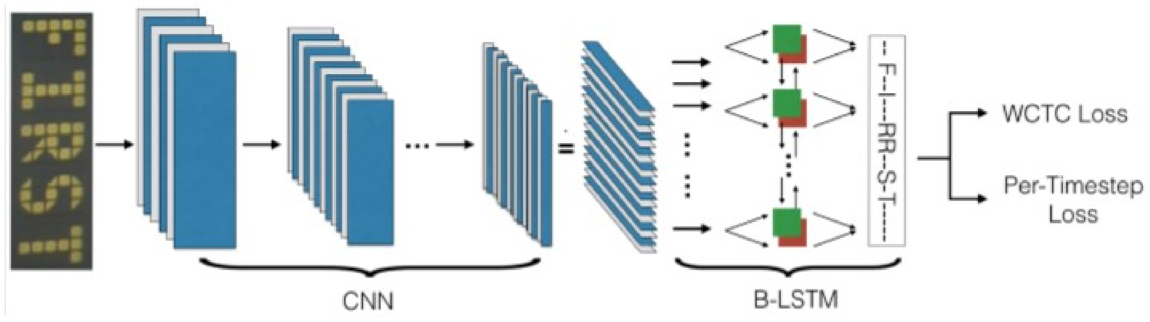


Figure 5.12: Text Recognition Model

Figure 5.12 summarizes the architecture of the text recognition model. We rst generate a sequence of deep CNN features F = f1,f2,··· ,fT from an image I. To further exploit the interdependence among features, a bidirectional Long Short-Term Memory (B-LSTM) model (Hochreiter and Schmidhuber, 1997) is stacked on top of generated sequential CNN features, yielding another sequence X = x1,x2,··· ,xT as nal outputs. Each one of the xi is normalized through a softmax function and can be interpreted as the emission of a specic label for a given time-step. On the other hand, the target word Y can be viewed as a sequence of characters, Y = y1,y2,··· ,yL

5.3 Getting Results

Once the text is recognized, the keywords are used to query an online database of book. We used google books api to search the books.

Chapter 6

Results and Discussion

To test recognition accuracy, we constructed a database of 497 book spine images. The book spine images are taken at the MNIT College Library.

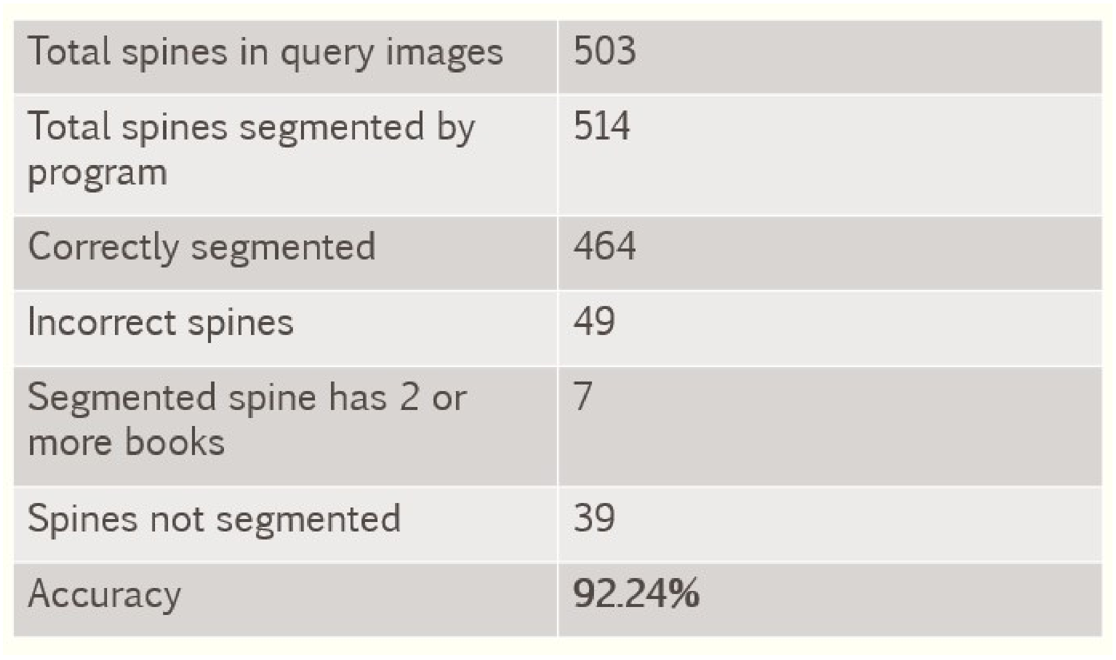
6.1 Segmentation Accuracy

We took 59 query images from MNIT library to test the segmentation accuracy. Segmentation process is applied on each query image. We manually checked the correctly segmented spines, incorrect segmented spines and other parameters. These are shown fig 6.1.

6.2 Book Spine Recognition Accuracy

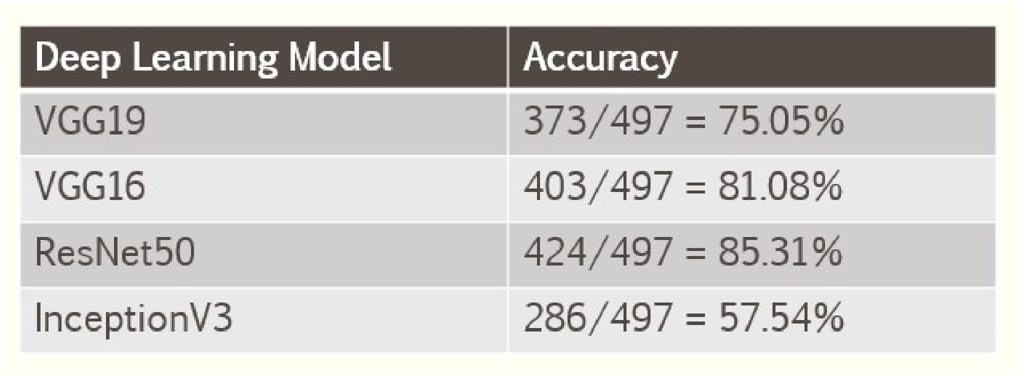
From a database of 497 book images, features are extracted. During testing these features are matched with the extracted one. Accuracy is shown in the fig. 6.2

Our system mainly relies on text spotting. As a result, we only need the textual information of each book and do not require storing and querying spine images, dramatically reducing the volume of data we need to store and manage. Moreover, for libraries with limited resources, one cannot always assume that book images are already available. For each book spine image, text is detected, recognized, and corrected.



Total spines in query images	503
Total spines segmented by program	514
Correctly segmented	464
Incorrect spines	49
Segmented spine has 2 or more books	7
Spines not segmented	39
Accuracy	**92.24%**

FIGURE 6.1: Segmentation Accuracy



Deep Learning Model	Accuracy
VGG19	373/497 = 75.05%
VGG16	403/497 = 81.08%
ResNet50	424/497 = 85.31%
InceptionV3	286/497 = 57.54%

FIGURE 6.2: Recognition Accuracy

Chapter 7

Conclusion and Future Scope

We have presented a new recognition system for identifying books on a bookshelf for use in book management systems. The book image first goes to the server. From a query image of a bookshelf, we extract the individual book spine images t by analyzing the line structures. Text is detected from the book spine images and used as keywords to search through a book spine text database. We extract image features from the book spine images and use them to match the spines to a book spine image database.

7.1 Future Scope

- Developing an android app.
- After spine recognition, the user can select individual spines in the phone's viewnder to obtain more information about a particular book, without ever having to take that book off the bookshelf.
- Use the digital compass and accelerometer on the smartphone to estimate location of the identified books. The digital compass tells us which direction the phone is facing when a book is photographed, while a temporal trace of the accelerometer informs us how far vertically and horizontally the phone has moved from the last anchor point.
- Once the book is identied, additional stored information could also be retrieved, e.g. a short description of the book.
- Find the location of books on the bookshelves.

Bibliography

[1] D. Chen, S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod, "Streaming mobile augmented reality on mobile phones," *Computer Networks*, pp. 181–182, 2009.

[2] S. S. Tsai, D. Chen, H. Chen, C.-H. Hsu, K.-H. Kim, J. P. Singh, and B. Girod, "Combining image and text features : A hybrid approach to mobile book spine recognition," 2011.

[3] D. Chen, S. Tsai, K.-H. Kim, C.-H. Hsu, J. P. Singh, and B. Girod, "Low-cost asset tracking using location-aware camera phones," 2010.

[4] T. Yeh and B. Katz, "Searching documentation using text, ocr, and image," 2009.

[5] N. Quoc and W. Choi, "A framework for recognition books on bookshelves," 2009.

[6] K. Matsushita, D. Iwai, and K. Sato, "Interactive bookshelf surface for in situ book searching and storing support," 2011.

[7] X. Shi, B.; Bai and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," 2015.

[8] R. Smith, "An overview of the tesseract ocr engine.," 2007.

[9] Y. A. J. K. Lee, D.-J.; Chang and C. Pitzak, "Matching book-spine images for library shelfreading process automation," 2008.

[10] S. B. C.-H. K.-H. Chen, D.M.;Tsai and J. P. Singh, "Building book inventories using smartphones. in proceedings of the 18th acm international conference on multimedia," 2010.

[11] M. Nevetha and A. Baskar, "Automatic book spine extraction and recognition for library inventory management. in proceedings of the third international symposium on women in computing and informatics," 2015.