Statistical Pattern Recognition
Final Project Report

Maximum Likelihood Approximate Nearest Neighbor Method in Real-time Face Recognition

Submitted in partial fulfillment of the degree

Bachelor of Science
in
Software Engineering

Submitted by
Ali Gholami

Under the guidance of
Prof. Mohammad Rahmati

Department of Computer Engineering and Information Technology
Amirkabir University of Technology

Spring Semester 2018

Abstract

Image recognition is one of the fundamental tasks in computer vision. Modern face recognition which is one of the substantial subtasks of the image recognition, in most cases, uses *brute-force* nearest neighbor method to iterate through the reference faces database to find the desired identity of the input image. Nowadays, most of these methods extract feature maps of the images using Convolutional Neural Networks. In this project, we'll be using Convolutional Neural Networks to extract image feature maps. We'll also analyze the implementation process of the *Approximate Nearest Neighbor* method to find the best ID for the input image.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

One of the well-known issues of vision systems building, is a processing of large databases. Unfortunately, nearest neighbor (NN) rule and exhaustive search usually cannot be implemented in real-time applications[1]. To overcome these problems, we'll analyze some of the purposed methods in this this topic.

1.2 Recent Methods of Image Recognition in Large Scale Databases

In this section we'll take a brief look at the recent researches on image recognition systems and their improvements for large scale databases.

1.2.1 Approximate Nearest Neighbor Shape Indexing

This method relies on the rapid recovery of the nearest neighbors from the index. In high-dimensional databases, standard k-d tree search often performs poorly. having to examine a large fraction of the points in the space to find the exact nearest neighbor. However, a variant of this search which efficiently finds approximate neighbors will be used to limit the search time. The algorithm, which we have called Best Bin First (BBF) search, finds the nearest neighbor for a large fraction of queries, and finds a very good neighbor the remaining times [2].

1.2.2 Directed Enumeration Method

Directed enumeration method is proposed to improve image recognition performance. The method is applied with similarity measures which do not met metric properties. This method increases performance in 3 to 12 times in comparison with nearest neighbor. Recognition speed using *DEM* is increased when many neighbors are located at similar distances [3].

1.2.3 Directed Enumeration Alternatives Modification

In this method, a new modification of the method of directed alternatives enumeration using the Kullback Leibler discrimination information is proposed for half-tone image recognition. Results of an experimental study in the problem of face images recognition with a large database are presented. It is shown that the proposed modification is characterized by increased speed of image recognition (5-10 times vs exhaustive search) [4].

Chapter 2

Problem Definition and Algorithm Definition

In this chapter, we'll take a close look at the core algorithm of *MLANN* method. The next is dedicated to the implementation procedure of the algorithm.

2.0.1 MLANN; General Idea

Despite the most of known fast approximate NN algorithms, the proposed method is not heuristic. The joint probabilistic densities (likelihoods) of the distances to previously checked reference objects are estimated for each class. The next reference instance is selected from the class with the maximal likelihood. To deal with the quadratic memory requirement of this approach, the author has proposed its modification, which processes the distances from all instances to a small set of pivots chosen with the farthest-first traversal. Experimental study in face recognition with the histograms of oriented gradients and the deep neural network-based image features shows that the proposed method is much faster than the known approximate NN algorithms for medium databases [5].

2.0.2 How Statistical Face Recognition Works

In face recognition, we are required to assign an observed image X to one of R classes which is specified by the database of reference images. In this method, feature maps extracted from observed and reference images are treated as probability distributions. The following subsection provides baseline assumptions for this task.

Key Assumptions

In face recognition task with statistical method, we assume that

1. The reference images from different classes are independent;
2. The probability distributions of the feature vectors from the same class are identical [5].

2.0.3 Where the Problem Arises

The core algorithm of many face recognition implementations use *Nearest Neighbor* method as the main approach to find the most similarity between the input image and the reference images in the database. Let X be the observed image and X_r $r \in 1, ...R$ be each of the images in the reference database. In that case, the optimal maximal likelihood solution of the face recognition task is achieved by

$$W_v : v = arg \min_r \ \rho(X, X_r) \tag{2.1}$$

where r is selected in a *brute-force* manner by the nearest neighbor algorithm and ρ is mean to be the *distance* of two image feature maps.

2.0.4 Speeding up the Search Process

Approximate Nearest Neighbor

To speed-up the search process, we can use *approximate* techniques. As an example, an approximate method is provided in [6]. This method is based on the following criterion:

$$W_v : \rho(X, X_v) < \rho_0 \tag{2.2}$$

which is the termination condition of the *ANN* method with respect to a bound of of ρ_0.

Best-Bin First kd-tree

In this method, the first reference image $X_{r_1},\ r_1 \in \{1, ..., R\}$ is randomly chosen. Next, it is put into the priority queue if reference images sorted by the distance to X. Next, the highest priority item X_i is pulled from the queue and the set of reference images $X_i^{(M)}$ is determined by using the following expression:

$$(\forall X_k \in X_i^{(M)})(\forall X_j \notin X_i^{(M)})|\rho_{i,k} - \rho(X, X_k)| \leq |\rho_{i,j} - \rho(X, X_j)| \tag{2.3}$$

the distance matrix $P = [\rho_{i,j}]$ is computed at the beginning of the algorithm.

Maximum Likelihood Approximate Nearest Neighbor Method

Here is the core algorithm of this paper. The method is proposed for the next step reference image choice. Let the reference images $X_{r_1}, ..., X_{r_k}$ have been checked before the k-th step, meaning that the distances $\rho(X, X_{r_1}), \rho(X, X_{r_2}), ...\rho(X, X_{r_k})$ have been computed. We can choose the next most probable reference image $X_{r_{k+1}}$ as

$$r_{k+1} = argmax(p_v . \prod_{i=1}^{k} f(\rho(X, X_{r_i})|W_v)) \quad v \in \{1, ...R\} - \{r_1, ...r_k\} \tag{2.4}$$

where p_v is the prior probability of class v which is the class of the observed image. $f(\rho(X, X_{r_i})|W_v)$ is the conditional density of the distance $\rho(X, X_{r_i})$ if the hypothesis W_v is true. We can acquire the likelihood $f(\rho(X, X_{r_i})|W_v)$ using a *Homogeneity Testing Probabilistic Neural Network*. Using an small modification proposed in the paper, we can obtain

$$r_{k+1} = argmin(\sum_{i=1}^{k} \phi_\mu(r_i) - \ln p_\mu) \quad \mu \in \{1, ...R\} - \{r_1, ...r_k\} \tag{2.5}$$

where $\phi_\mu(r_i)$ satisfies all we need for a normal distribution:

$$\phi_\mu(r_i) = \frac{(\rho(X, X_{r_i}) - \rho_{\mu,r_i} - \frac{N-1}{UV}))^2}{\rho_{\mu,r_i}} + \frac{4}{UV} \ln(2\rho_{\mu,r_i} + \frac{N-1}{UV}) \tag{2.6}$$

since the average image size is usually much higher than the number of parameters $UV >> (N-1)$, $\phi_\mu(r_i)$ can be simplified:

$$\phi_\mu(r_i) = \frac{(\rho(X, X_{r_i}) - \rho_{\mu,r_i}))^2}{\rho_{\mu,r_i}} \tag{2.7}$$

If the distance $\rho(X, X_{r_{k+1}})$ is lower than the threshold ρ_0, then the search procedure is terminated at the $K_{checks} = k+1$ step. Otherwise, the reference image $X_{r_{k+1}}$ is put into the set of previously checked reference images and the maximal likelihood search procedure is repeated.

2.0.5 MLANN; Disadvantages

Though the proposed MLANN method makes it possible to minimize the average number of distance computations K_{checks}, it has two significant disadvantages:

- The necessity to compute and store distances between all reference images makes MLANN to use *quadratic memory space.*
- The complexity of extra computation at each step is rather high. For instance, $\phi_\mu(r_i)$ is calculated for each unchecked μ-th reference image.

2.0.6 Pivot Based MLANN; An Improvement to MLANN Method

In this section a modified version of MLANN is proposed. We'll choose $p << R$ reference images (pivots) from the database. The results show that the performance increases if we choose images far from each other. Thus it is necessary to choose the pivots as distant as possible. This task can be solved using known *clustering* algorithms. Another approach is the incremental selection of pivots which is better compared to the clustering techniques which requires calculating the distances between all the images in database. We'll use the *farthest-first traversal* procedure. The first pivot X_{r_1} is chosen randomly. Next, the distances between this reference image and all other reference images are computed, and the most distant reference image is selected as the second pivot. The procedure is repeated so that every pivot is characterized with the highest sum of distances to the previous pivots:

$$r_{i+1} = argmax \sum_{j=1}^{i} \rho(X_\mu, X_{r_j}), i \in \{1, ..., p-1\} \tag{2.8}$$

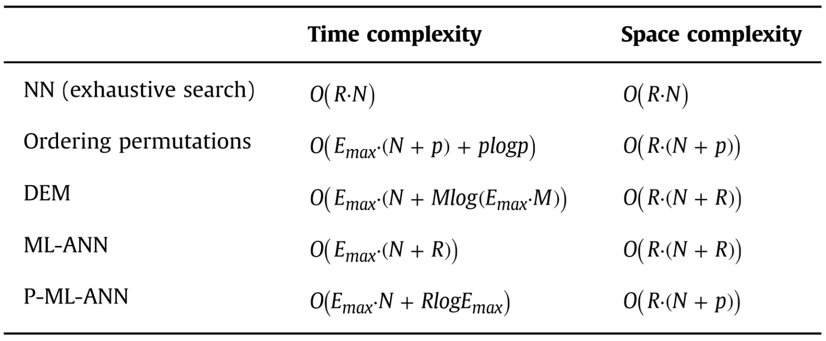
where

$$\mu \in \{1, ...R\} - \{r_1, ...r_i\}$$

The result of the initialization phase is a list of pivots and $R * p$ matrix of distances between all reference images and the pivots.

2.0.7 Time and Space Complexity Analysis

Figure 2.1 demonstrates the time and space complexities of *P-ML-ANN* method.



	Time complexity	**Space complexity**
NN (exhaustive search)	$O(R \cdot N)$	$O(R \cdot N)$
Ordering permutations	$O(E_{max} \cdot (N + p) + plogp)$	$O(R \cdot (N + p))$
DEM	$O(E_{max} \cdot (N + Mlog(E_{max} \cdot M))$	$O(R \cdot (N + R))$
ML-ANN	$O(E_{max} \cdot (N + R))$	$O(R \cdot (N + R))$
P-ML-ANN	$O(E_{max} \cdot N + RlogE_{max})$	$O(R \cdot (N + p))$

Figure 2.1: Time and space complexities of the P-ML-ANN algorithm.

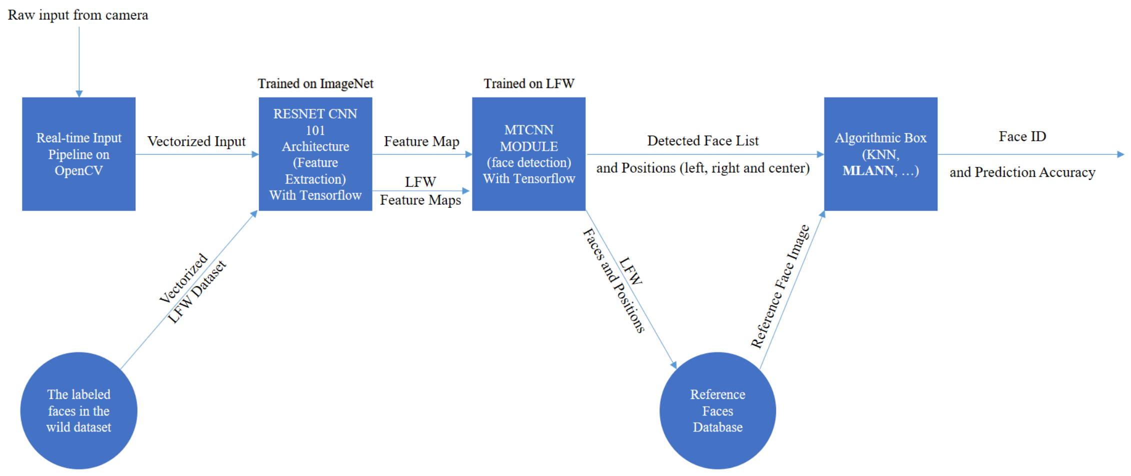
Chapter 3

Implementation and Performance Analysis

The core implementation is purposed at the official repository of the MLANN paper by Professor Savchenko. The algorithm is already implemented in *C++* and Caffe by Professor Savchenko and is available via this link. I've also provided the *C++* implementation in the directory of this project. In this section, we'll analyze the implementation of MLANN method using Tensorflow and OpenCV in Python.

3.1 The Architecture

Figure 3.1 illustrates the overall architecture of implementation.



Raw input from camera
Real-time Input Pipeline on OpenCV
Vectorized Input
Trained on ImageNet
RESNET CNN 101 Architecture (Feature Extraction) With Tensorflow
Feature Map
LFW Feature Maps
Trained on LFW
MTCNN MODULE (face detection) With Tensorflow
Detected Face List and Positions (left, right and center)
Algorithmic Box (KNN, MLANN, ...)
Face ID and Prediction Accuracy
Vectorized LFW Dataset
The labeled faces in the wild dataset
LFW Faces and Positions
Reference Face Image
Reference Faces Database

Figure 3.1: The architecture of real-time face recognition system.

3.2 Performance Analysis

Figure 3.2 compares methods of face recognition on LFW dataset. The brute-force method is the worst algorithm possible for face recognition. The proposed method *Pivot based MLANN* has the best performance on this dataset and it is 3.5-5.5 times faster than the brute-force method.

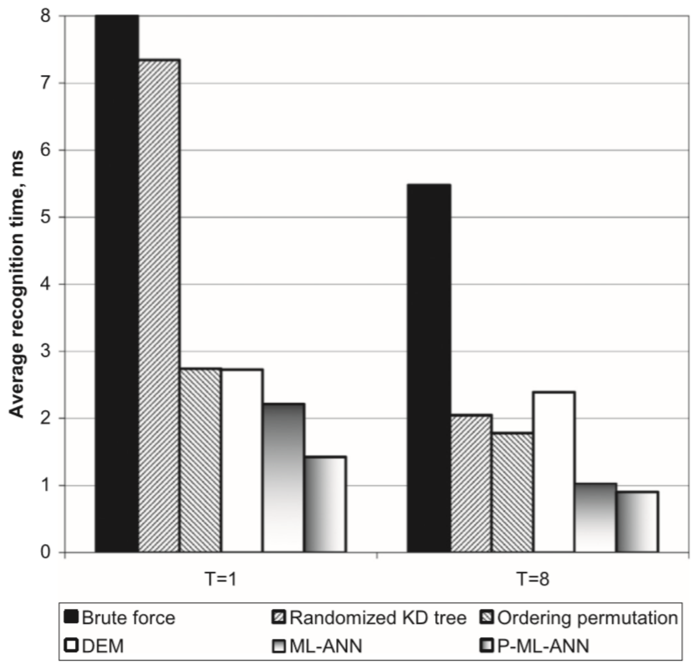


Figure 3.2: Average face recognition time (ms), Euclidean distance, DNN features, LFW dataset.

3.3 Implementation Dependencies

3.3.1 Anaconda

This project has been tested with Python 3.6 which is provided in the Anaconda Python package. Anaconda can be downloaded and installed from the official site.

3.3.2 Tensorflow

For this project, we've used the amazing *Tensorflow* framework. The place of usage is illustrated in figure 3.1. To run the project you need the latest version of the *Tensorflow* which can be downloaded using Python package manager, *PIP* using *pip install tensorflow*.

3.3.3 OpenCV-Python

For real-time image pipelining we have used the Python distribution of the OpenCV package. Python OpenCV can be installed with Python package manager with command: *pip install python-opencv*.

3.3.4 Pre-trained Model Dependency; Inception Resnet V1

This model has been trained on the *ImageNet* dataset for feature extraction only. The pre-trained model should be downloaded under the directory */models*. The model is provided with this report. The model has been implemented by the *Tensorflow* community and is available on the official GitHub repository of Tensorflow.

3.3.5 MTCNN-Detect Module

This module has been implemented by the *David Sandberg* as a part of *Facenet* face recognition and is available via this link for copyright purposes. The implementation is provided in the directory as *MTCNN-DETECT* module.

3.3.6 Core Algorithm

The core algorithm implementation in Python is provided in the class *Nearest Neighbor*. At the time of writing this paper, the whole method has not been implemented completely in Python and is still in progress, but you can refer to the *ann.cpp* under the *Savchenko* directory for *C++* implementation of the MLANN algorithm.

3.3.7 How to Run the Project

Adding a New Reference Face

You can simply open a terminal and type *python main.py –mode input*. Make sure to move your face gradually to extract the features of it!

Real-time Face Recognition

You can run the face recognition module using the command *python main.py –mode camera*.

Chapter 4

Future Work

We admit the possible drawbacks and inefficiencies existing in this project, thus we have the following key improvement to-do list:

- Improvement and optimization of Python implementation of MLANN algorithm.
- Using newer trained models like *Resnet-V2* for the feature extraction phase.
- Implementation of *face alignment* module under *MTCNN-Detect* using parallel programming languages; i.e. CUDA.
- Implementation of the *Pivot-based* improvement to the MLANN method for faster face recognition times.
- Using one-shot learning to reduce the per-person number of reference images in LFW dataset.

Chapter 5

Conclusion

In this project, we implemented the novel ANN method by Professor *Savchenko* with the maximal likelihood search of the next reference instance in order to improve the performance of the NN-based image recognition methods. We analyzed its modification P-ML-ANN to make this approach suitable for practical applications. Several pivots are chosen at the preprocessing stage of the P-ML-ANN method. Hence, the main cycle of the search procedure becomes as fast as the brute force. Although this approach does not achieve the lowest average number of computed distances as the ML-ANN, it is more computationally efficient in most cases. Our modification requires only linear memory space to store the distances between pivots and all reference images. An obvious drawback of this procedure is the need for a proper choice of the number of pivots.e Invariant Feature Transform.

References

[1] Tan, Xiaoyang, et al. Face Recognition from a Single Image per Person: A Survey. Pattern Recognition, vol. 39, no. 9, 2006, pp. 17251745.

[2] Beis, Jeffrey S., and David G. Lowe. "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces." cvpr. IEEE, 1997.

[3] Savchenko, Andrey V. "Real-time image recognition with the parallel directed enumeration method." International Conference on Computer Vision Systems. Springer, Berlin, Heidelberg, 2013.

[4] Savchenko, Andrey V. Image Recognition with a Large Database Using Method of Directed Enumeration Alternatives Modification. RSFDGrC11 Proceedings of the 13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, 2011, pp. 338341.

[5] Savchenko, A. V. "Maximum-likelihood approximate nearest neighbor method in real-time image recognition." Pattern Recognition 61 (2017): 459-469.

[6] Arya, Sunil, et al. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions." Journal of the ACM (JACM) 45.6 (1998): 891-923.