ABSTRACT

Title of dissertation: COMPUTER VISION AND DEEP LEARNING WITH APPLICATIONS TO OBJECT DETECTION, SEGMENTATION, AND DOCUMENT ANALYSIS

Xianzhi Du
Doctor of Philosophy, 2017

Dissertation directed by: Professor Larry Davis
Department of Computer Science

There are three work on signature matching for document analysis. In the first work, we propose a large-scale signature matching method based on locality sensitive hashing (LSH). Shape Context features are used to describe the structure of signatures. Two stages of hashing are performed to find the nearest neighbors for query signatures. We show that our algorithm can achieve a high accuracy even when few signatures are collected from one same person and perform fast matching when dealing with a large dataset. In the second work, we present a novel signature matching method based on supervised topic models. Shape Context features are extracted from signature shape contours which capture the local variations in signature properties. We then use the concept of topic models to learn the shape context features which correspond to individual authors. We demonstrate considerable improvement over state of the art methods. In the third work, we present a partial signature matching method using graphical models. In additional to the second work, modified shape context features are extracted from the contour of signatures to describe both full and partial signatures. Hierarchical Dirichlet processes are implemented to infer the number of salient regions needed. The results show the effectiveness of the approach for both the partial and full signature matching.

There are three work on deep learning for object detection and segmentation. In the first work, we propose a deep neural network fusion architecture for fast and robust pedestrian detection. The proposed network fusion architecture allows for parallel processing of multiple networks for speed. A single shot deep convolutional network is trained as an object detector to generate all possible pedestrian candidates of different sizes and occlusions. Next, multiple deep neural networks are used in parallel for further refinement of these pedestrian candidates. We introduce a soft-rejection based network fusion method to fuse the soft metrics from all networks together to generate the final confidence scores. Our method performs better than existing state-of-the-arts, especially when detecting small-size and occluded pedestrians. Furthermore, we propose a method for integrating pixel-wise semantic segmentation network into the network fusion architecture as a reinforcement to the pedestrian detector. In the second work, in addition to the first work, a fusion network is trained to fuse the multiple classification networks. Furthermore, a novel soft-label method is devised to assign floating point labels to the pedestrian candidates. This metric for each candidate detection is derived from the percentage of overlap of its bounding box with those of other ground truth classes. In the third work, we propose a boundary-sensitive deep neural network architecture for portrait segmentation. A residual network and atrous convolution based framework is trained as the base portrait segmentation network. To better solve boundary segmentation, three techniques are introduced. First, an individual boundary-sensitive kernel is introduced by

labeling the boundary pixels as a separate class and using the soft-label strategy to assign floating-point label vectors to pixels in the boundary class. Each pixel contributes to multiple classes when updating loss based on its relative position to the contour. Second, a global boundary-sensitive kernel is used when updating loss function to assign different weights to pixel locations on one image to constrain the global shape of the resulted segmentation map. Third, we add multiple binary classifiers to classify boundary-sensitive portrait attributes, so as to refine the learning process of our model.

COMPUTER VISION AND DEEP LEARNING WITH APPLICATIONS TO OBJECT DETECTION, SEGMENTATION, AND DOCUMENT ANALYSIS

by

Xianzhi Du

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisors Prof. Larry Davis and Dr. David Doermann for the continuous support of my Ph.D study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better mentors for my Ph.D study.

What's more, I would like to thank the rest of my thesis committee: Prof. Rama Chellappa, Prof. Min Wu, Prof. David Jacobs, and Prof. Ramani Duraiswami, for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Dr. Mostafa El-Khamy, Dr. Jungwon Lee, and Dr. Xiaolong Wang, for offering me the research internship opportunities in their groups and leading me working on diverse exciting projects.

I thank my fellow labmates: Le Kang, Peng Ye, Jayant Kumar, Yangmuzi Zhang, Varun Manjunatha, Sungmin Eum, etc. for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years.

Last but not the least, I would like to thank my family for giving birth to me at the first place and supporting me spiritually throughout my life.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADAS	Advanced Driver Assistance Systems
ASPP	Atrous Spatial Pyramid Pooling
AR	Augmented Reality
BB	Bounding Box
BSN	Boundary-sensitive Network
CG	Candidate Generator
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
FCN	Fully Convolutional Network
F-DNN	Fused Deep Neural Network
FPPI	False Positives Per Image
HDP	Hierarchical Dirichlet Process
HOG	Histogram of Gradient Orientations
IoU	Intersection-over-Union
L-AMR	Log-average Miss Rate
LDA	Latent Dirichlet Allocation
LSH	Locality Sensitive Hashing
MS-CNN	Multi-scale Convolutional Neural Network
NN	Nearest Neighbour
PFCN	Portrait Fully Convolutional Network
RPN	Region Proposal Network
ReLU	Rectified Linear Unit
RP-LSH	Random Projection Locality Sensitive Hashing
SC	Shape Context
SGD	Stochastic Gradient Descent
SIFT	Scale-invariant Feature Transform
SLDA	Supervised Latent Dirichlet Allocation
SR	Salient Region
SSD	Single-shot Multi-box Detector

SNF	Soft-rejection based Network Fusion
SS	Semantic Segmentation
SVM	Support Vector Machine
TF	Term-Frequency

Chapter 1: Introduction

1.1 Document Image Analysis and Signature Matching

The continuing growth of digital imaging has lead to a significant increase in the number of larger repositories in all aspects of government and business. For example, the now well known Tobacco Litigation has produced a document database of approximately 50 million pages. The massive increase in document database sizes have rekindled interest in the difficult challenges of indexing, querying and retrieving relevant documents.

In today's electronic world, signatures are still widely used as a method for authorization and identification, and in some cases document retrieval. This is due to the fact that signatures are written in a ballistic manner and will carry discriminative features that are difficult to forge by an amateur. Hence, signature matching is an important problem in document image retrieval and forensic applications [1] [2] [3]. Given an image of a query signature, the problem is to find all documents in a (possibly large) database, which were signed by the same person. Signature matching is usually treated as nearest neighbour problem, with emphasis on the numerical representation of the signature and the similarity measures used to compute distances between these representations.

Document authorization and identification applications which rely on signatures typically involve processes of signature detection [4], signature verification [5] [6] [7], and/or signature matching [4] [8] [9]. Signatures detection is the localization of the signature region in the document image using techniques such as multi-scale structural saliency. Signature verification aims to authenticate signatures as genuine given known samples from the same signer. Several competitions held in conjunction with the ICDAR and ICFHR conference, have focused on signature verification have resulted in datasets containing both genuine and forged signatures [10] [11] [12]. The signature matching problem attempts to match questioned signatures with sample signatures based primarily on shape. The goal of signature matching is simply to identify signatures in large collections that look similar, but not necessarily to authenticate them. Authentication (and/or) verification can be performed once the number of candidate signatures is more manageable.

1.2 Deep Learning with Application to Pedestrian Detection

Pedestrian detection has applications in various areas such as video surveillance, person identification, image retrieval, and advanced driver assistance systems (ADAS) [13]. Real-time accurate detection of pedestrians is a key for adoption of such systems. A pedestrian detection algorithm aims to draw bounding boxes which describe the locations of pedestrians in an image in real-time. However, this is difficult to achieve due to the tradeoff between accuracy and speed [14]. Whereas a low-resolution input will, in general, result in fast object detection but with poor performance, better object detection can be obtained by using a high-resolution input at the expense of processing speed. Other factors such as crowded scene, non-person occluding objects, or different appearances of pedestrians (different poses or clothing styles) also make this problem challenging.

The general framework of pedestrian detection can be decomposed into region proposal generation, feature extraction, and pedestrian verification [15]. Classic methods commonly use sliding window based techniques for proposal generation, histograms of gradient orientation (HOG) [16] or scale-invariant feature transform (SIFT) [] as features, and support vector machine (SVM) [17] or Adaptive Boosting [18] as the pedestrian verification methods. Recently convolutional neural networks have been applied to pedestrian detection. Hosang et al. [19] use SquaresChnFtrs [20] method to generate pedestrian proposals and train a AlexNet [21] to perform pedestrian verification. Zhang et al. [22] use a Region Proposal Network (RPN) [23] to compute pedestrian candidates and a cascaded Boosted Forest [24] to perform sample re-weighting to classify the candidates. Li et al. [25] train multiple Fast R-CNN [26] based networks to detect pedestrians with different scales and combine results from all networks to generate the final results.

1.3 Deep Learning with Application to Portrait Segmentation

Semantic segmentation is a fundamental problem in computer vision community which aims to classify pixels into semantic categories. We target a special binary class semantic segmentation problem, namely portrait segmentation, which generates pixel-wise predictions as foreground (i.e., people) or background. Recently, it is becoming a hot topic and has been widely used in many real-world applications, such as augmented reality (AR), background replacement, portrait stylization, depth of field, etc. Although numerous deep learning based approaches (e.g., [27] [28] [29] [30] [31] [32] [33] [34]) were proposed to solve the general semantic segmentation problem, direct adaptation of these methods cannot satisfy the high precision requirement in the portrait segmentation problem.

In portrait segmentation, precise segmentation around object boundaries is crucial but challenging. For applications like background replacement, accurate and smooth boundary segmentation (such as hair and clothes) is the key for better visual effects. However, this has long been one of the most challenging part of portrait segmentation, especially when using convolutional neural networks (CNN). Since the neighborhood of boundary pixels contains a mixture of both foreground and background labels, convolutional filters fuse information of different classes, which may confuse the network when segmenting boundary pixels. Previous CNN based semantic segmentation methods, which use either the conventional hard-label method or ignore the boundary pixels during training [27] [35] [36] [28] [29], fail to solve this problem. These methods aim to train a better model to separate foreground and background while sacrificing the accuracy when predicting the boundary pixels.

Chapter 2: Signature Matching for Document Analysis

2.1 Large-scale Signature Matching using Multi-Stage Hashing

2.1.1 Introduction

Methods used for shape matching can generally be grouped into two categories based on whether or not a correspondence between the points of the query shape and points of the database shapes is used. Methods that do not explicitly solve a correspondence problem construct global shape representations, using, for example, Fourier descriptors [37] or other structural descriptors [38], and directly compare the shape representations to find the nearest shapes. The main limitation of these methods is that it is typically challenging to extract global descriptor from real images and this therefore affects the matching accuracy. Moreover, extracting global shape representations restricts these methods to shapes with high degrees of rigidity.

Methods that depend on extracting point features on the contours of the shapes being compared, and try to solve a correspondence problem between these points to recover an unknown transformation between the shapes [38]. These methods typically produce better matching performance, since they tolerate lower degrees of rigidity. However, they are generally computationally expensive and require solving the correspondence problem, which renders these methods intractable as the size of the dataset grows. In [38], for example, Shape Context (SC) [39] features are extracted around the contours of the query and test signatures and a correspondence problem is solved using thin-plate splines. Four different similarity measures were computed between the query and test signatures. The method is computationally expensive and as the number of test signatures in the dataset increases, it becomes computationally intractable, since this correspondence problem must be solved between the query signatures and all other signatures in the dataset.

In this work, we introduce a new signature matching method specifically designed to scale to large-scale datasets. We use a Locality Sensitive Hashing (LSH) [40] in a mutlistage approach to avoid comparing the query signature against all other signatures in the dataset. Moreover, we use LSH to reduce the dimensionality of extracted feature vectors, without the need to perform any clustering or vector quantization.

LSH is a probabilistic dimensionality reduction technique. The idea is to hash points that exist in a high-dimensional feature space such that the probability of collision is much higher for near-by points than for those that are far apart. Points whose hashing values collide with each other fall into the same bucket. LSH has been used for nearest neighbour search [40] [41] [42] [43], content-based image retrieval [44] [45] and large-scale clustering [46]. In this work, we employ Random Projection LSH (RP-LSH) [40], which uses randomly generated hyperplanes to partition the higher dimensional feature space. RP-LSH depends on generating a set of random vectors ri and i = 1, . . . , M1, where M1 is the number of generated random vectors, and use them to hash an input vector v, as shown in Equation (2.1).

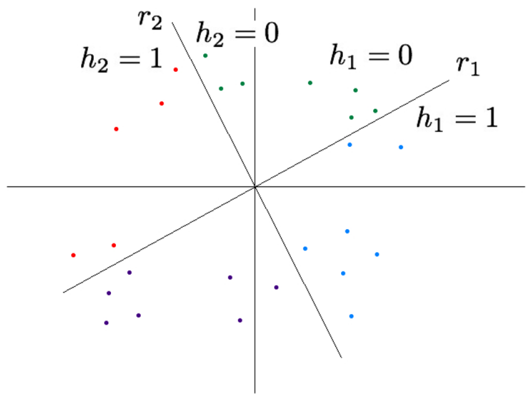

Figure 2.1: Illustration of random projection locality sensitive hashing in 2D feature space. Two hashing vectors r1 and r2 are using to create 2-bit hash codes for the input vectors. Each random vector partitions the feature space into two halves. Points on the left of the random vector are hashed with 0 while points on the right are hashed with 1.

$$h_i = \begin{cases} 0, & \text{if } v \cdot r_i >= 0 \\ \\ 1, & \text{if } v \cdot r_i < 0 \end{cases} \tag{2.1}$$

Each random vector r_i produces a 1-bit hash value for an input vector v. The M1-bits are used to generate a 1D decimal hash code for each input vector. Figure 2.1 illustrates the idea in 2D feature space. Two hashing vectors r_1 and r_2 are using to create 2-bit hash codes for the input vectors. Each random vector partitions the feature space into two halves. Points on the left of the random vector are hashed with 0 while points on the right are hashed with 1.

2.1.2 Multi-stage Signature Matching via Locality Sensitive Hashing

The primary motivation of the method presented In this work is to be able to perform fast signature matching against large-scale signature datasets. We formulate the problem as a nearest neighbour search problem. However, instead of matching the query signature against the entire signature dataset, we limit the search space using Locality Sensitive Hashing (LSH), effectively limiting the similarity measures to a few signature that share the same hash code with the query signature. Moreover, rather than using simple global features to represent the signature, such as Fourier descriptors, and to accurately capture the structure of the signatures, we use Shape Context (SC) features computed on the contour. Furthermore, to avoid solving the computationally expensive point correspondence problem, we use LSH to reduce the dimensionality of the SC features and construct a unified representation for the signature with low computational overhead.

Figure 2.2 shows a sample signature and extracted contour. The contour is evenly

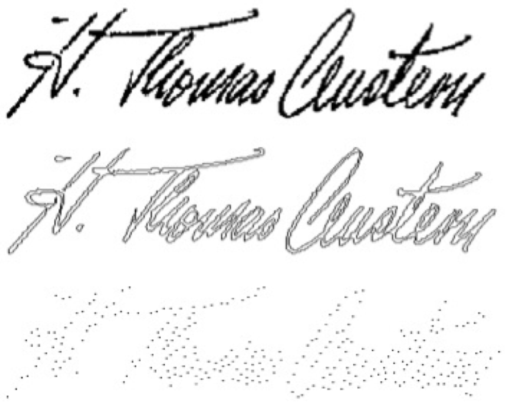
Figure 2.2: Feature extraction from a signature. A sample signature is shown in the top figure. The middle figure shows the contour extracted from the signature. Finally, the lower figure shows a fixed number of points evenly sampled from the contour of the signature. SC features are computed at each one of the sampled contour points.

sampled and a fixed number of points, K, are extracted. SC is calculated around each of contour sample, producing a D-dimensional feature vector, where $D = n_r * n_\theta$ and nr and n are the number of concentric circles and the number of angular sections, respectively. Hence, each signature i is essentially represented using a K-D-dimensional feature vectors, $v^i_1, v^i_2, ..., v^i_K$. Let M_1 be a number of D-dimensional zero-mean, unity-variance random vectors, where M_1 represents the length of the locality-sensitive hash code. For a given signature i, all vectors v^i_j are hashed using the M_1 vectors as shown in Equation (2.2).

$$h_m = \begin{cases} 0, & \text{if } v_j \cdot r_m >= 0 \\ 1, & \text{if } v_j \cdot r_m < 0 \end{cases} \tag{2.2}$$

where $m = 1, 2, ..., M_1$ and $j = 1, 2, ..., K$, which produces M_1-bit has code for each vector. Hash codes for all vectors extracted from the dataset are pooled to construct a dictionary of distinct hash codes. For each signature i, the K hash codes are used to calculate a Term-Frequency (TF) (i.e. histogram), which is used a unified signature representation.

The TF is a higher level representation of the signature and it encodes the structural features of the signature represented by the shape context features. Therefore, signatures created by the same author should have similar TF representations. Given the TF representation of a query signature, in order to find other signatures created by the same author, we must perform a linear search to compute the similarity between the query and test signatures. However, as the size of the signature dataset increases, the linear search becomes computationally inefficient.

We use a second set M_2 of zero-mean, unity-variance random vectors to compute locality-sensitive hash codes for the TF representations of the signatures. However, rather than converting the hash codes to decimal numbers, signatures are compared by computing the Hamming distance between binary hash codes. The Hamming distance is used to find other signatures in the dataset that shares the same M_2 binary hash with the query signature, thus avoiding the linear search.

2.1.3 Experiments

To evaluate the performance of the proposed method, two sets of experiments were conducted. The first set of experiments was on a small dataset to study the performance

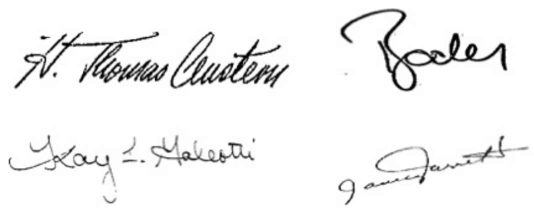

Figure 2.3: Sample signatures from DS-I. The dataset includes 189 signatures from 63 authors. Signatures are relatively clean without any underlying machine printed text.

of our signature matching algorithm when the number of samples of each author is small. The second set of experiments was performed on a larger data set to illustrate the performance of the algorithm as the size of the dataset grows and demonstrate the scalability of our method.

2.1.3.1 Evaluation protocol and datasets

We use a top-N accuracy strategy to assess the performance of the proposed signature matching method. For each query signature, if another signature from the same author appears in the top-N neighbours, selected by our method using the Hamming distance, we regard this is a successful match. The process if repeated for all signatures in the dataset and the average of all signatures is reported. Accuracy statistics were collected on two datasets. The first dataset (DS-I) contains 189 signatures collected from 63 authors with three signatures from each author. Signatures in this dataset are fairly clean, since they are extracted from high-quality document images with little noise or machine-printed text on them. Several signatures in this dataset are shown in Figure 2.3.

The second dataset (DS-II) contains 26,661 signatures, from 890 individuals, ex-

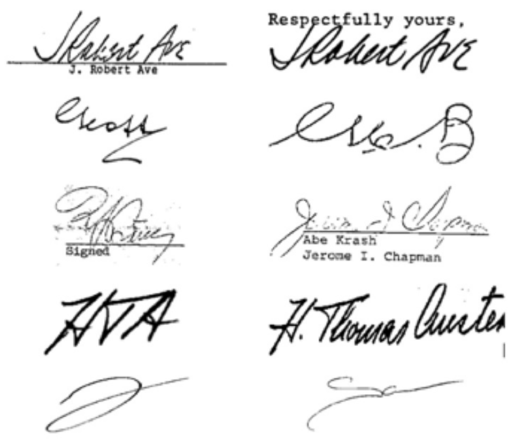

Figure 2.4: Sample signatures from DS-II. The dataset includes 26,661 from 890 authors. Signatures vary in image quality, occlusion, underlying and complexity. The signatures are collected from letters and memorandums from the Tobacco litigation document dataset.

tracted from the Tobacco litigation document dataset. This dataset is more representative of a realistic signature matching problem. Unlike the previous dataset, signatures here are blocks extracted from letters and memorandums. Several common and unrelated content and degradations appear, such as:

- Machine-printed text or lines which do not belong to the signature
- Signatures may be occluded
- Different styles may be used by the same writer, such as, the use of initials in place of names
- Images may be of different sizes and qualities
- Some signatures may be overly simple

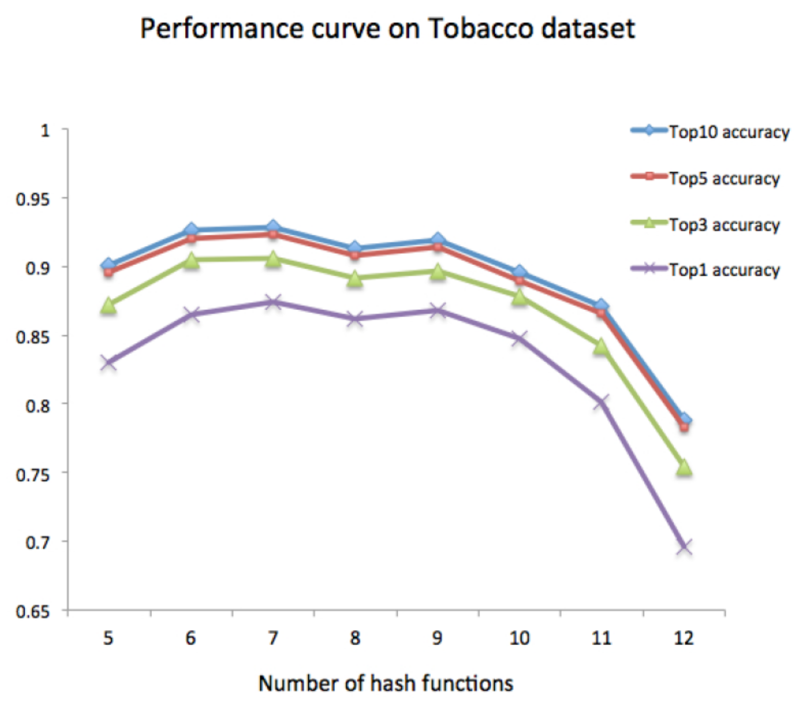

Figure 2.5: Accuracy of the proposed signature matching algorithm on DS-I. The reported accuracy is the average of 10 different runs of the algorithm.

Figure 2.4 shows some samples of the signatures in DS-II illustrating the varying levels of image quality, underlying text, and signature complexity.

2.1.3.2 Results

The first set of experiments was conducted on DS-I. The resolution of the signatures is 480×152 and the number of contour points extracted is $K = 300$. To extract shape context (SC) features, we use $n_r = 5$ and $n_\theta = 12$. Therefore, for each signature in the dataset, we extract $30060 - D$ feature vectors. To counteract the randomization property of hash functions, we repeat hash function generation process 40 times. Each

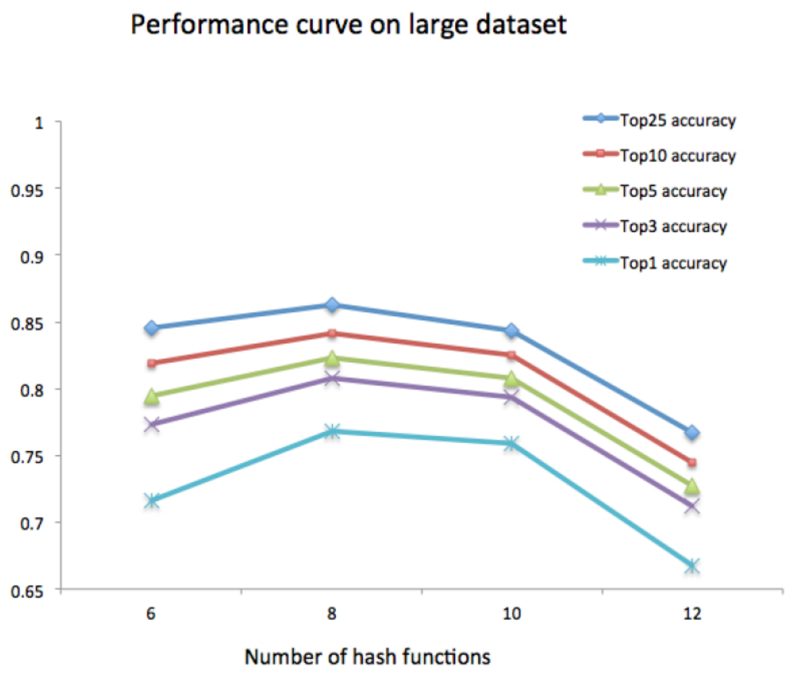

Figure 2.6: Accuracy of the proposed signature matching algorithm on DS-II.

iteration produces a top-N matching result. The nth neighbour is the most frequently appearing term. To compute average accuracy with high confidence, the procedure is repeated 10 times and the mean accuracy is reported. Figure 2.5 shows the performance of the signature matching algorithm on DS-I. The best accuracy is achieved at $M_1 = 7$.

DS-I was used to evaluate the performance of the method proposed by Zhu et al. [4]. They reported a less than 10% accuracy on this dataset due to the small number of signatures from each author. Meanwhile, the execution time for [4] requires approximately 11 hours to train and the same amount of time to find the nearest neighbours for all signatures in the dataset, for a total of 22 hours. On the other hand, the execution time for our algorithm is approximately 1.77 seconds.

The second set of experiments was conducted on DS-II. All signatures were normalized to the mean signature size of 482×182. Other system parameters were set similar to the previous experiment. Figure 2.6 shows the performance of the proposed algorithm. Peak accuracy was obtained at $M_1 = 8$.

The execution time of the proposed algorithm, to find matching signatures for all samples in the dataset, was 24 minutes. We were not able to test the method proposed in [4] due to the computational intractability, since it is estimated that it years to run. Tables 2.1 and 2.2 show the peak performance of the proposed method for various top-N accuracies, for both DS-I and DS-II, respectively.

	Top-10	Top-5	Top-3	Top-1
Accuracy	92.80%	92.28%	90.58%	87.41%

Table 2.1: Peak accuracy of the proposed signature matching algorithm for DS-I.

	Top-25	Top-10	Top-5	Top-1
Accuracy	86.24%	84.15%	82.27%	76.79%

Table 2.2: Peak accuracy of the proposed signature matching algorithm for DS-II.

2.2 Signature Matching using Supervised Topic Models

2.2.1 Introduction

State of the art methods used for signature matching can generally be classified as either global-shape and point-level approaches. Approaches that use global-shape representations such as Fourier descriptor [8] and global shape structure [9] [47] work efficiently and accurately for simple shape matching tasks. However, signatures are shapes with high degree of variability, so as the signatures become distorted or occluded, matching becomes increasingly challenging for these global methods. Other methods depend on the point-level features of the signature. Samples from the spine or contours are often used and features such as Shape Context [48] [49], or FREAK [50] are extracted to form descriptors. By solving the point correspondence problem, a shape transformation between signature and a model can be recovered. These methods tend to be more robust as they can tolerate a low degree of variation, which is a must-solve problem for handwritten content.

Previous work on point-based signature matching problem has focused primarily on pairwise correspondence and comparison. In [4], for example, all pairs of corresponding points between two signatures are compared to form four global weights during training. This method may achieve a high accuracy, but it is computationally expensive and does not scale as the number of signatures grows. In [46], signature matching problem is modeled as a nearest neighbour (NN) search problem. Locality sensitive hashing (LSH) is performed to reduce the dimension of features and to reduce the search radius for the NNs. This method is fast and efficient when dealing with large datasets, but is less accurate as LSH may miss true NNs and this method only uses L1-norm distance between high-level features to determine the matching points.

In this work, we model the signature matching problem using supervised latent Dirichlet allocation (sLDA) [51]. SLDA is a statistical model developed from latent Dirichlet allocation (LDA) [48] and was originally used for labeling documents. Co-occurring observations are combined in latent distributions called topics, which have an unknown distribution over the vocabulary. The collection of documents share a set of topics and a specific mixture of topics are represented by each document. The model is built as follows. First, for each document, we draw a mixture of topics from a Dirichlet distribution parameterization. We then repeatedly choose one topic from the mixture distributed as a multinomial distribution to assign to the current document and draw a word from the corresponding topic. The label for each document is regarded as a response variable from the latent topics in that document. With only the observations known, variational inference is used to learn to rest unknown parameters. (For details please refer to [48] and [51])

Motivated by sLDA, we model signature matching problem in a similar way: a document is a signature, a topic is a salient region, a/an word/observation is the class label of a point-level feature, and a document label is an authorship for one signature. We model each signature as a mixture of salient regions, which are latent groupings for similar observations in feature space. Our point-level SC features are used to generate observations. Then the generative process of each observation is described as follows. First, we draw a mixture of salient regions from a Dirichlet distribution. Second, we draw a specific salient region for a signature and a specific observation from that salient region.

2.2.2 Building the Vocabulary

The first step of our algorithm is to build the TF histogram based on point-level features then to build the vocabulary that will be used in the supervised topic model. Many different features have been used to describe various kinds of images including local point features such as SIFT [51] or SURF [5] or simple global features like Fourier descriptors. To address the local binary and curvature properties of signatures, SC feature provides a reasonable alternative. SC feature preserves the relative position of local points and tolerate simple distortion from similar shapes.

For a new signature, we first extract the contour of the signature and sample K points along the contour. Second, for each point, a $D = N_r \times N_\theta$ dimensional SC feature is computed, where N_r and N_θ represent the two dimensions of log-polar space. This results in a K-D-dimensional feature for each signature. Figure 2.7 shows a sample

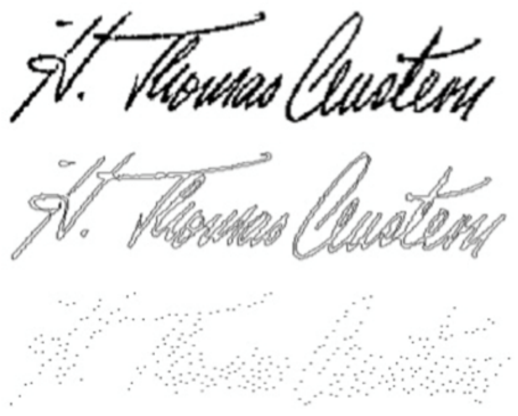

Figure 2.7: Contour extraction and points sampling of a signature. A sample signature is shown in the top figure. The middle figure shows the extracted contour. Points are evenly sampled from contour, and SC features are computed based on the points, which is shown in the bottom figure.

signature, the contour and the sampled K points.

After obtaining the features, we reduce the feature dimension and build codebook by classifying all points and computing the TF histogram based on their class labels. One simple and efficient way to do this is by hashing [8]. M hyperplanes are generated, and all high dimensional feature points are hashed into bins in a lower dimensional space by measuring the relative position of points and hyperplanes. The index of the bin is used as the class label. This is very effective for large-scale datasets, but in practice the accuracy may be reduced due to the fact that the hyperplanes are randomly generated and the number of classes must be 2M.

To increase accuracy, we use a M-class K-means clustering to build the codebook. In K-means, the number of classes can be chosen smoothly. First, features for all training points are classified into M classes. Second, an M-length TF histogram is built to compute the appearance frequency of the M classes among all the points in each signature.

Until now, a signature is represented by the M-dimensional TF vector. The class label of SC feature of a contour point will be used as an observation later in sLDA step. The value of each unit in the TF vector represents how many contour points are classified into this class. This further represents the appearance frequency of one observation in one signature. All the M labels (M different observations, in other words) form the vocabulary.

Currently, the choice of M is closely related to the size of dataset. A smaller M will tend to group different features into same group. A larger M will make the data over-classified and become more time consuming.

2.2.3 Building Supervised Topic Model

In this section, sLDA is used to build model for authors and infer the authorship for query signatures. We build our model as shown in Figure 2.8. α is the Dirichlet parameter; θ_d is the per-signature salient region proportions; Z_d, n is the per-observation salient region assignment; W_d, n is the observed class label of SC feature of one contour point; γ_k is the salient region, which is drawn from Dirichlet process parameterized by η; Y_d is the authorship for one signature, which is a response from all the salient regions in this signature.

The generative process for the observation is as follows:

- Draw salient region proportions θ_d from $Dir(\alpha)$
- For each observation

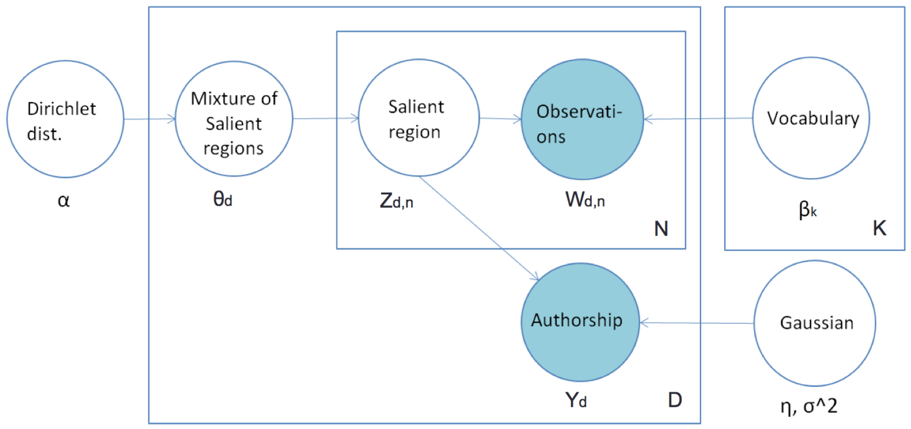

Figure 2.8: SLDA model for our problem. SC features and authorship labels are observed.

  - Draw salient region assignment Z_d, n from $Mult(\theta_d)$
  - Draw observation W_d, n from $Mult(\beta_{Z_{d,n}})$

- Draw authorship variable Y_d from $N(\eta, \sigma^2)$

where $Dir(\alpha), Mult(\theta_d), andN(\eta, \sigma^2)$ represent Dirichlet distribution, multinomial distribution, and normal distribution.

In this model, for each signature, the labels of D-dimensional SC features of the K contour points are used as observations, and they form the TF histogram to represent the appearance frequency of all observations. Points with similar local SC features are more likely to be grouped into a same salient region. So signatures from same author should share similar salient region composition, which makes them neighbors in the feature space. During training, η, the distribution of observations over all salient regions, is estimated. Based on the observed authorships, we infer the salient region distributions for all authors. During testing, ϕ, the salient region assignments of observations, is computed. Using η and ϕ, we can get the salient region distribution for query signatures. The closest salient region distribution among all authors is considered as the resulting authorship.

The choice of the number of salient regions is closely related to the size of vocabulary M, which further depends on the size of dataset.

2.2.4 Experiments and results

2.2.4.1 Datasets

Two datasets are used to evaluate the performance of our method. We first test on a small and clean dataset to illustrate our method can do well in modeling author characteristics. Second we test on a larger dataset to demonstrate the robustness to noisy realistic data even when the number of training samples is limited.

The first dataset is DS-I Tobacco dataset. This dataset contains 189 signatures collected from 63 authors with 3 signatures per author. The signatures are extracted from high quality document so there is limited noise present and machine-printed text rarely appears in the signature blocks. This is the same dataset used in signature hashing work [8]. Samples of signatures from DS-I dataset are shown in Figure 2.9.

The second dataset is DS-II UMD dataset. Signatures from this dataset are extracted from the same Tobacco litigation document dataset [6]. This dataset represents more of a realistic signatures dataset as signatures are extracted from letters and memorandums, and signatures may have various degradations. We choose a 1000-signature set from 100 authors with 10 signatures per author. Signatures from DS-II UMD dataset are shown in Figure 2.10.

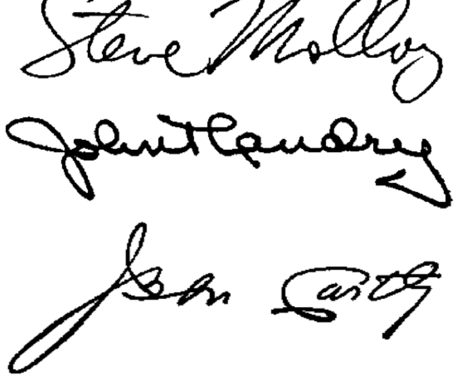

Figure 2.9: Samples signatures from DS-I Tobacco dataset.

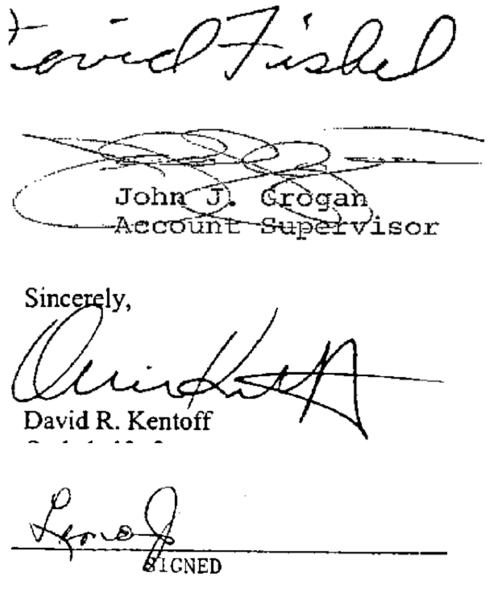
John J. Grogan
Account Supervisor
Sincerely,
David R. Kentoff
SIGNED

Figure 2.10: Sample signatures from DS-II UMD dataset.

2.2.4.2 Evaluation Metrics

For both datasets, we use Top-N accuracy to evaluate the performance. After ranking the inferred authorship for each query signature, if one of the top-N authors is correct, corresponding to the referring labels of testing signatures, the top-N accuracy is set to one for this query. The total top-N accuracy is the mean of all query signatures.

2.2.4.3 Experiment I

In the first experiment on DS-I Tobacco signature dataset, all images are normalized to 495×186 pixels and 300 points are extracted from each contour. A 5×12 SC feature ($N_r = 5, N_\theta = 12$) is used as the feature descriptor resulting in a 300×60-dimensional feature to describe each signature.

Two main parameters that may directly affect the accuracy are M, the number of classes in K-means which corresponds to the size of the vocabulary, and the number of salient regions in sLDA (# SR). The effects of the two parameters on the final results are shown in Figures 2.11 and 2.12. We choose M=700 and the # SR=90 empirically. Since the K-means algorithm uses random initialization points as centers, we repeat the process 20 times to obtain the 20 top-N authorships. The most frequently appearing term at top nth is set as the n_{th} authorship.

To compare with the previous hashing method [8] and Zhus method [4], each combination of two of the three signatures from one author are used to train and remaining one is used to test. This results in three rounds. From Figure 2.13 we can see our new method significantly improves the performance of the previous hashing method on this

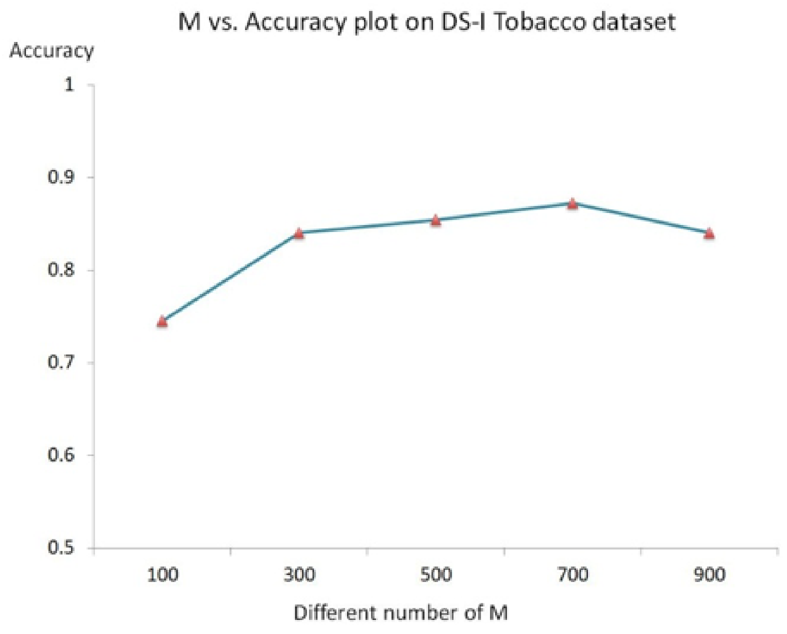

Figure 2.11: Performance curve of different M on DS-I Tobacco dataset. An optimal result is given by 700 classes.

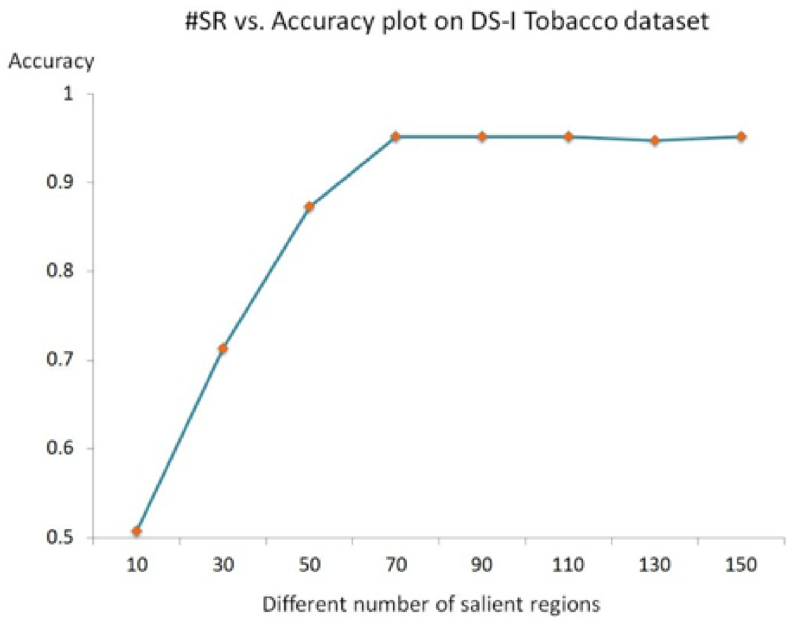

Figure 2.12: Performance curve of different # SR on DS-I Tobacco dataset. The accuracy stops increasing after a number of salient regions near 90.

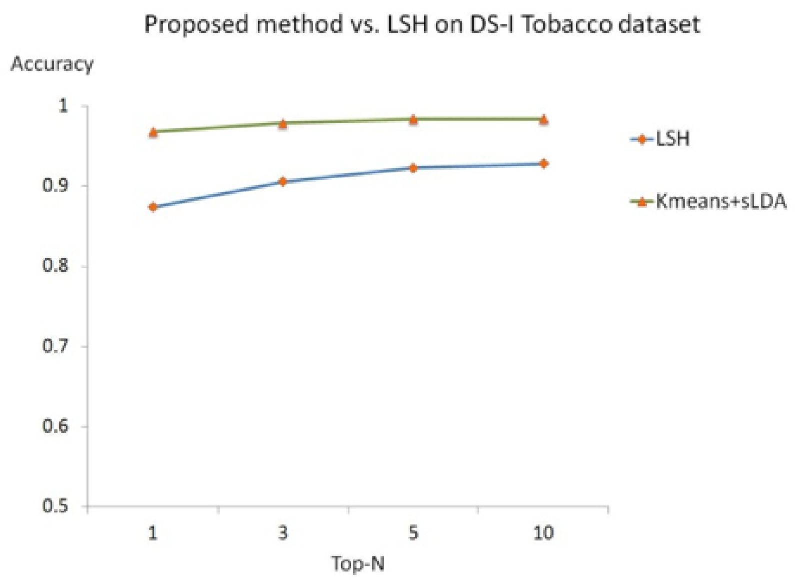

Figure 2.13: Performance curve and accuracy table of the proposed method and state-of-the-art hashing method. We can see the Top-1, Top-3, Top5, Top-10 accuracies are improved a lot.

dataset, from 87.41% to 96.83% on Top-1 accuracy, even when limited signatures are used for training. Due to the limited number of training samples, Zhus method reported less than 10% accuracy.

The execution time for Zhus method requires as long as 11 hours to train and another 11 hours to test. On the other hand, the hashing method requires approximately 1.8 seconds, as it is designed for fast matching. Our method is a little slower than hashing method but still much faster than Zhus method. The execution time of training is about 7 minutes and the execution time of testing is less than 5 seconds.

Top-accuracy	Top-1	Top-3	Top-5	Top-10
Kmeans + sLDA	96.83%	97.88%	98.41%	98.41%
LSH	87.41%	90.58%	92.28%	92.8%

Table 2.3: Accuracy table of the proposed method and state-of-the-art hashing method. We can see the Top-1, Top-3, Top5, Top-10 accuracies are improved a lot.

2.2.4.4 Experiment II

We conducted a second experiment on DS-II UMD dataset. For each author, 5 signatures are used to train and 5 are used to test. The set up of system parameters are similar to the first dataset. Empirically we choose M=1300 and # SR=140 to optimize efficiency and accuracy. Results from different M and # SR are plotted in Figure 2.14 and 2.15.

Even if only five signatures from each author are used in the training set and data is added with different kinds of noise, our 95.2% Top-1 accuracy still outperformed the previous hashing methods 80.6% Top-1 accuracy. The result is given in Figure 2.16.

The execution time of the proposed algorithm is about 20 minutes for training and less than 5 seconds for testing. The hashing method is a little faster (5 minutes). Zhus method would be computational intractable, since it is estimated to take months to run.

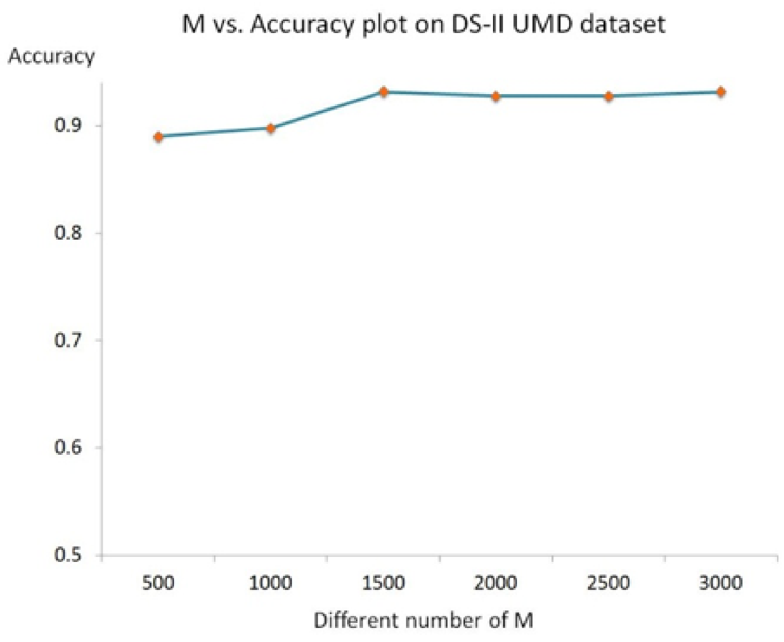

Figure 2.14: Performance curve of different M on DS-II UMD dataset. Optimal result is given by a number between 1300 and 1800.

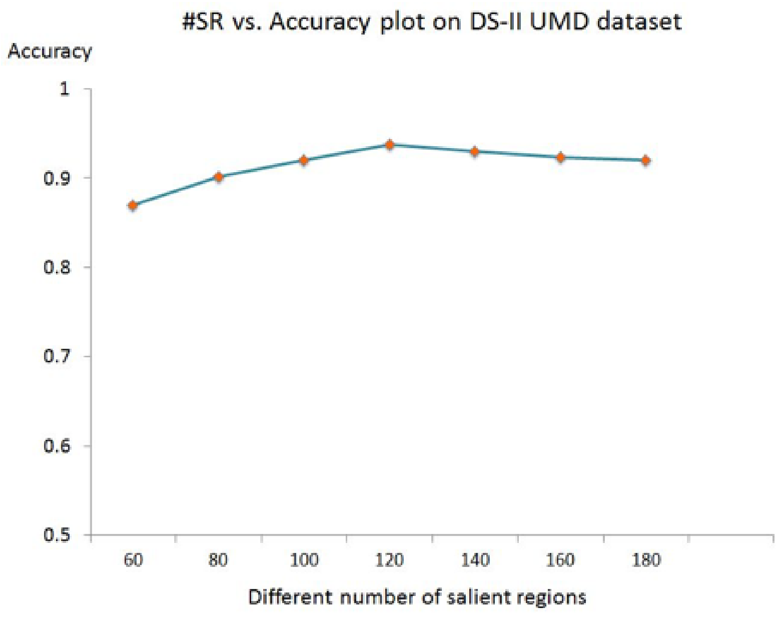

Figure 2.15: Performance curve of different # SR on DS-II UMD dataset. The accuracy stops increasing after a number of salient regions near 140.

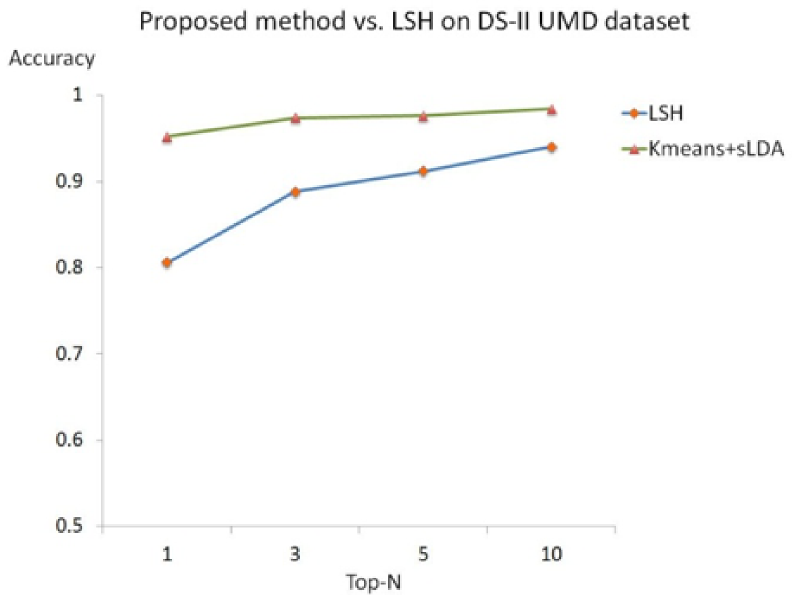

Figure 2.16: Performance curve and accuracy table of the proposed method and state-of-the-art hashing method on DS-II dataset.

Top-accuracy	Top-1	Top-3	Top-5	Top-10
Kmeans + sLDA	95.2%	97.4%	97.6%	98.4%
LSH	80.6%	88.8%	91.2%	94.0%

Table 2.4: Accuracy table of the proposed method and state-of-the-art hashing method on DS-II dataset.

2.3 A Graphical Model Approach for Matching Partial Signatures

2.3.1 Introduction

In this work, we distinguish between two problems—*full signature matching*, which assumes an accurate segmentation and an author who has produced a complete and consistent signature, and *partial signature matching*, where we enroll a full reference signature, but may only have a partial signature or even initials to match against the reference signature. A lot of work has shown great promise for addressing the full signature matching problem using feature descriptors such as shape context [52] [53] [54] [55]. However, partial signature matching remains an open problem. When dealing with real applications, even though full signatures are collected, many query signatures are only part of the authors' full name. This is due to multiple factors including the fact that people sometimes only sign their first/last names or initials, part of a signature may be missing when performing signature extraction from documents, or when the signature is obscured by other information, such as the machine-printed text in a signature block.

To address the partial signature matching problem, we developed a method based on the combination of supervised latent Dirichlet allocation (sLDA) [56] and hierarchical Dirichlet processes (HDP) [57]. SLDA was first developed as a statistical model of labeled documents, derived from latent Dirichlet Allocation (LDA) [58]. Unlike LDA, in sLDA, each document is paired with a label. Topic distributions are estimated over the vocabulary and relation between topics and labels are discovered in the training stage. For an unlabeled document, the label is regressed from its topic structure. In our formulation, ”signature” corresponds to ”document”, ”salient region” corresponds to ”topic”, ”observation” corresponds to ”word”, and ”authorship” corresponds to ”label”.

In this approach, sLDA is first used to discover the salient regions in all training signatures. A salient region is a distribution over the features in the visual vocabulary, which groups similar co-occurring observations. Each author is modeled as a combination of all salient regions with different proportions. For a query signature, classification is performed by computing the salient region proportions for the signature based on observations. Further, instead of guessing the number of salient regions empirically, HDP is used to estimate the number needed for the given dataset.

2.3.2 Observation and vocabulary building

In this section, we describe how signatures are modeled as a group of observations.

2.3.2.1 Partial shape context feature extraction

The first step of modeling signatures as a group of observations is to find a proper feature descriptor. As we are working with 2-D binary shapes, we want to find a feature descriptor that captures the relations between points in the binary shapes while preserving the local information between full and partial signatures. In this case, popular features like SIFT [] and SURF [59] are not suitable since they use gradient information of feature points which is not informative with binary points. Therefore, we use shape context features, a feature designed for 2-D shape, which describes the relations between nearby points while tolerating slight shape distortion. This is especially important for the signature matching problem since even high-quality signatures from same author may have slight differences.

To build observations, we first extract contour points from one signature proportional to the total length of the contour. The result is that partial signature will have similar contour points as its corresponding part in the full signature. For each contour point c_i, a $r * \theta$ log-polar space is formed around it with uniform bins. A histogram s_i

is built by calculating the number of nearby points that fall in each bin in a certain order, based on the relative distance and angle of the two points, as shown in Equation (2.3).

$$\begin{aligned} s_i &= [s_i(1), s_i(2), ..., s_i(n)] \\ s_i(k) &= \# \{p : p{\in}bin(k), p{\neq}c_i\} \end{aligned} \tag{2.3}$$

where $n = r * \theta$ is the total number of histogram bins. p represents the contour points. In order to make shape context applicable to partial signature matching, instead of normalizing the pairwise point distances within each signature, we normalize the pairwise point distances for all possible point pairs in all signatures by dividing by the mean value $\bar{D}$, as shown in Equation (2.4).

$$\begin{aligned} \bar{D} &= (\sum_{l=1}^{N_{sig}} \sum\nolimits_{i,j \in s_l} D(p_i, p_j))/N_{pair} \\ D_{norm}(p_i, p_j) &= D(p_i, p_j)/\bar{D} \end{aligned} \tag{2.4}$$

where $D(p_i, p_j)$ represents the distance between two points p_i and p_j in signature s_l, N_{sig} is the total number of signatures, and N_{pair} is the total number all possible pairs of points in all signatures. Since the size of log-polar space is fixed for all signatures, this makes partial signatures have more similar shape context features to a corresponding full signature, as illustrated in Figure 2.17.

2.3.2.2 Building the visual vocabulary

After extracting contour points and computing shape context features, the next step is to build the visual vocabulary for all signatures. One intuitive way is to cluster all contour points and treat each cluster label as one observation. The vocabulary consists of all the cluster labels. For each signature, the number of contour points being classified

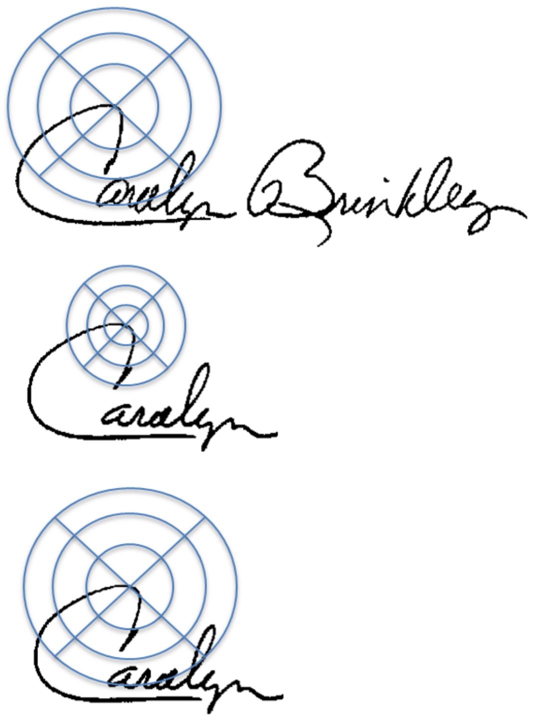

Figure 2.17: Shape context features for three signatures. The first row shows a full signature and the log-polar space of the shape context features. The second row shows its partial signature and the log-polar space by using the standard shape context features. The third row shows the same partial signature and the log-polar space by using our modified shape context features.

into one cluster is regarded as the appearance frequency of this observation.

In our method, we use K-means clustering. A histogram h_t for the t_{th} signature is formed to indicate the appearance frequencies of all observations, as shown in Equation (2.5).

$$\begin{gathered} h_t = [h_t(1), h_t(2), ..., h_t(K)] \\ h_t(k) = \# \{p : p \in Cluster(k), p \in Sig(t)\} \end{gathered} \tag{2.5}$$

Finally, signatures are represented by an observation-frequency histograms.

2.3.3 Supervised topic models for partial signature-matching

In this section, we build a supervised topic model for the partial signature matching problem and briefly review the variational inference solution based on the work of Wang et al. [60].

2.3.3.1 Building supervised topic model

For our problem, the generative process for the n_{th} observation in the t_{th} signature is given as follows:

1. For the t_{th} signature, draw salient region proportions θ_t from $Dir(\alpha)$
2. For the n_{th} observation:
    (a) Draw a salient region assignment $S_{t,n}$ from $Mult(\theta_t)$
    (b) Draw an observation $O_{t,n}$ from $Mult(\beta_{S_{t,n}})$
3. Draw authorship variable A_t from $N(\eta^{\intercal} \bar{S}_t, \sigma^2)$

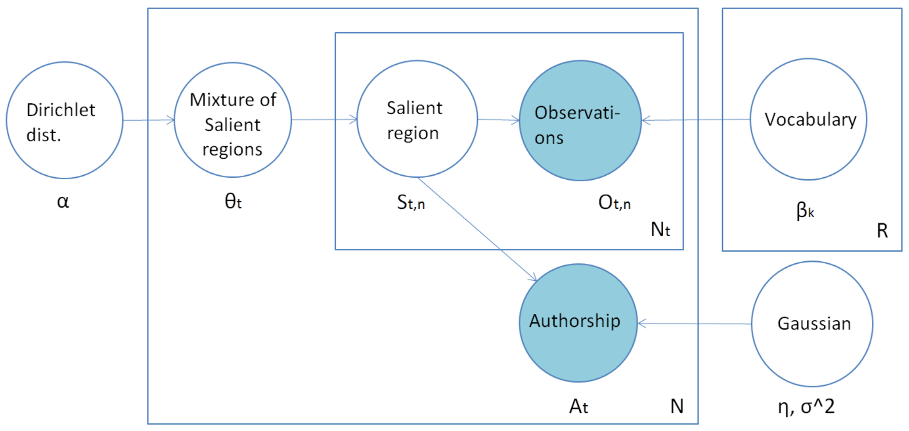

Figure 2.18: SLDA model for our problem.

Where the $Dir(\cdot)$, $Mult(\cdot)$, $N(\cdot)$ represent the Dirichlet distribution, the Multinomial distribution, and the Normal distribution respectively. α is an R-dimensional hyperparameter for the Dirichlet distribution with R being the number of salient regions. $\beta = [\beta_1, \beta_2, ..., \beta_R]$, where each β_r is the distribution of the salient region r over the vocabulary, and $\bar{S}_t$ is the mean of the salient regions of the t_{th} signature. With only observations and authorship given, we want to estimate $\alpha, \beta, \eta, \sigma^2$. The model is given in Figure 2.18.

Variational EM algorithm [60] is used to solve the sLDA model. Here we give a brief description.

By introducing the free variational parameters γ and ϕ, the posterior distribution $p(\theta, S|O, A, \alpha, \beta, \eta, \sigma^2)$ is approximated by Equation (2.6).

$$q(\theta, S|\gamma, \phi) = q(\theta|\gamma)\prod_{n=1}^{N} q(S_n|\phi_n) \tag{2.6}$$

Here γ is a R-dimensional variational Dirichlet hyperparameter that governs salient region distribution of each signature. We can regard it as an approximation for α in the original model. $\phi = [\phi_1, \phi_2, ..., \phi_{N_t}]$ is an approximation to θ, but specific for each observation.

The variational EM algorithm works as follows: In the E-step, γ and ϕ are computed to minimize the KL divergence between the true posterior and the approximation. In the M-step, each of the salient regions β_r is estimated by counting how many times each observation is assigned to this salient region among all signatures. η and σ^2 are estimated by the relationships between salient regions and the labels for all training signatures. For a new signature, the label is predicted by Equation (2.7).

$$\mathbb{E}[A|O, \alpha, \beta, \eta, \sigma^2] = \eta_{est}^{\intercal}\mathbb{E}[S|O, \alpha, \beta] = \eta_{est}^{\intercal}\phi_{est} \tag{2.7}$$

For more details about the algorithm in this section please refer to [58] [60] [56].

2.3.3.2 HDP for salient region estimation

In LDA/sLDA, the number of salient regions need to be prefixed and it is always chosen empirically. When processing new and massive data, it is not possible to easily choose an optimal number and it is also very expensive to reprocess the massive data multiple times to find out the optimal number. To solve this problem, Teh [57] provided a new topic model structure called hierarchical Dirichlet processes, which lets the data estimate the number of salient regions needed.

The main difference between HDP and standard LDA lies in the model structure. In HDP, each group of data (signature in our case) has its own mixture model with random probability measure G_t. G_t's are distributed as a Dirithlet process with a global base distribution G_0. G_0 itself is also distributed as a Dirichlet process.

$$G_t \sim DP(\alpha_0, G_0) \tag{2.8}$$

$$G_0 \sim DP(\gamma, H) \tag{2.9}$$

where α_0, γ are the concentration parameters, and H is the base distribution for G_0.

A straightforward way to explain HDP is the Chinese restaurant franchise. We give a brief introduction as follows: Let $\theta_{t,n}$ be the salient region for the n_{th} observation in the t_{th} signature, $\psi_{t,r}$'s be the existing salient regions for signature t, $N_{t,r}$ be the number of observations in signature t under salient region r, M_t be the number of salient regions used in signature t. For an new observation, the salient region assignment is given as follows:

$$\begin{gathered}\theta_{t,n}|\theta_{t,1}, ..., \theta_{t,n-1}, \alpha_0, G_0 \sim \\ \sum_{r=1}^{M_t} \frac{N_{t,r}}{n-1+\alpha_0}\delta_{\psi_{t,r}} + \frac{\alpha_0}{n-1+\alpha_0}G_0\end{gathered} \tag{2.10}$$

The salient region of a new observation either can be an existing salient region within this signature, which is a draw from the first summation term on the right-hand side of Equation (2.10) with a probability proportional to the number of observations under that salient region, or can be a new salient region with probability given by the second term on the right-hand side of Equation (2.10). If a new salient region is needed for this observation, we draw one salient region $\psi_{t,r}$ from G_0 and increase M_t by one as follows:

$$\psi_{t,r}|\psi_{1,1}, ..., \psi_{t,r-1}, \gamma, H \sim \\ \sum_{r_0=1}^{R} \frac{M_{r_0}}{M+\gamma}\delta_{\phi_{r_0}} + \frac{\gamma}{M+\gamma_0}H \tag{2.11}$$

where M_{r_0} is the number of times salient region r_0 is used in all signatures and M is the total number of times all salient regions are used in all signatures. If the first term of the right-hand side of Equation (2.11) is chosen, the new salient region for $\theta_{t,n}$ is picked among the existing salient regions with a probability proportional to the number of times one salient region is used in all signatures. If the second term is chosen, a new salient region is introduced and the total number of salient regions is increased by one. For more details about the algorithm, please refer to [57] [61].

2.3.4 Experiments and results

2.3.4.1 Datasets

Two popular datasets for signature matching problem are the DS-I Tobacco signature dataset and the DS-II UMD signature dataset. The DS-I Tobacco signature dataset contains 189 relatively clean and high-quality signatures from 63 authors (three per author). All the signatures are full signatures, meaning each contains the full name of the author. The DS-II UMD signature dataset contains 26661 signatures from 887 authors. It is a more challenging and more realistic dataset since the signatures are extracted from the Tobacco litigation corpuses, memos and letters. It is a mixture of both full signatures and partial signatures as many signatures are partial signatures or initials. Moreover, signatures in this dataset have other different kinds of degradations such as: signatures with

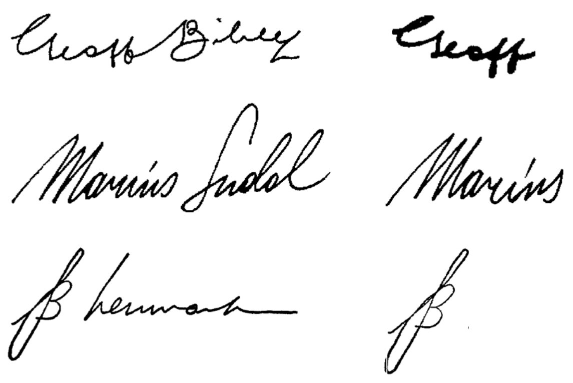

Figure 2.19: Sample signatures in DS-I partial dataset. Left column shows three full signatures. Right column shows their partial signatures.

low-quality; machine-printed texts like address and date appear in the signature block; signatures are occluded; signatures are too simple to be classified.

Our method is tested on two partial signature datasets and one full signature dataset. The two partial signature datasets are built based on the DS-I Tobacco signature dataset and the DS-II UMD signature dataset. We refer to them as the DS-I partial dataset and DS-II partial dataset. The full signature dataset is the DS-I Tobacco signature dataset itself.

DS-I partial dataset: As there are no partial signature datasets which contain clean and high-quality partial signatures, we manually selected the DS-I partial set from the DS-I Tobacco signature dataset by clipping out the first names of all full signatures. Full signatures are used to train and partial signatures are used to test in the experiment. Samples from the DS-I partial set are shown in Figure 2.19.

DS-II partial dataset: This dataset contains 495 full signatures for training from 495 authors and 1732 signatures contain both partial and full signatures for testing. To

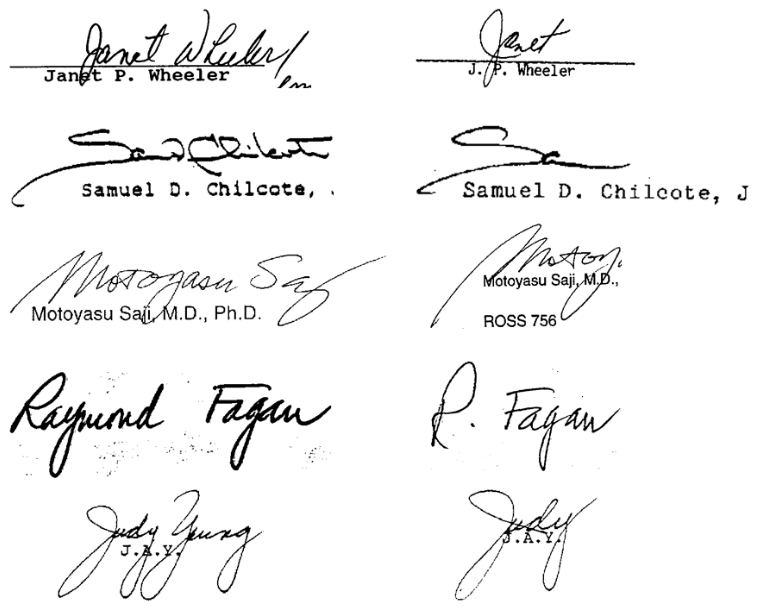

Figure 2.20: Sample signatures in DS-II partial dataset. Left column shows five full signatures. Right column shows their partial signatures with different kinds of degradations.

build the DS-II partial dataset, we use authors in the DS-II UMD dataset with at least one partial signature. Here we define a partial signature as follows: For each signature, when compared to a full sample, if any part that should belong to the full signature is missing, we consider it to be a partial signature. We didn't test on the whole dataset because the number of partial signatures is relatively too small compared to full signatures. Due to the limited samples in this dataset, we only pick one full signature per author to train (which is the usual case in real life applications) and use other signatures (may contain both full and partial signatures) to test. Each author has at least one and at most 30 signatures for testing. Samples from DS-II partial dataset are given in Figure 2.20.

2.3.4.2 Evaluation protocol

In our experiments, the top-N rank accuracy is used for evaluation. Top-N rank strategy means for each query signature, in the prediction stage, if the full signature sample of the true author appears in any position of the first N_{th} ranks, the top-N rank accuracy for that sample is set to 1. The total top-N rank accuracy is the mean of all query signatures.

2.3.4.3 Results on DS-I partial dataset

We normalize the height of each signature to the mean height of all training signatures and set the widths to the aspect ratios. In this experiment, the heights of all signatures are fixed to 182. For shape context features, $r = 5$ and $\theta = 12$ are used. As we have three signatures from each author in this dataset, each time we pick two full signatures to train and the partial signature of the rest one to test. We run three rounds to cover all the signatures in both training and testing stages. The average accuracy is reported.

First we run HDP on the DS-I parital dataset to estimate the number of salient regions. $R = 90$ is suggested after 1000 iterations. For K in K-means algorithm, empirically we choose $K = 1300$. Due to the random initialization of K-means algorithm, we rerun the whole experiment 10 times. The term most frequently appears on the 10 i_{th}'s rank is taken. Our method is compared with Du et al. [53] and Zhu et al. [52]. We refer to them as LSH method and pairwise matching method. Results are given in Figure 2.21 and Table 2.5.

Our method achieves 87.8% top-1 rank accuracy on this partial dataset, which sig-

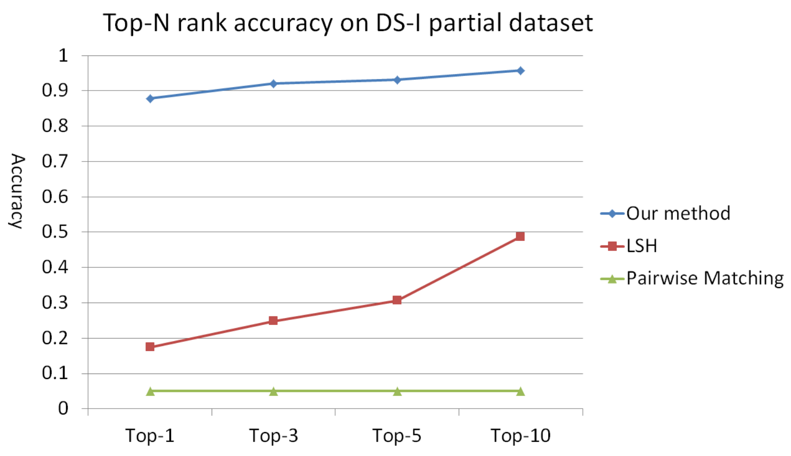

Figure 2.21: Performance curves on DS-I partial dataset.

	Top-1	Top-3	Top-5	Top-10
LSH	0.175	0.249	0.307	0.487
Pairwise matching	<0.05	<0.05	<0.05	<0.05
Our method	**0.878**	**0.921**	**0.931**	**0.958**

Table 2.5: Results on DS-I partial dataset

nificantly outperforms the LSH method by 17.5% and the pairwise matching method. The pairwise matching method reported a less than 5% top-1 rank accuracy on this dataset due to incapability of dealing with partial signature-matching problem and insufficient training samples.

2.3.4.4 Results on DS-I Tobacco full signature dataset

In order to show that our method also works well on full signature dataset, we test on the DS-I Tobacco full signature dataset. We run this experiment three times and report the

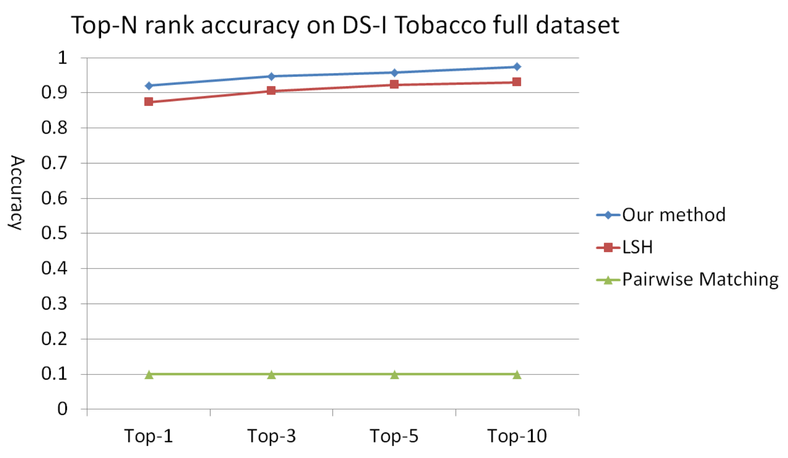

Figure 2.22: Performance curves on DS-I Tobacco full dataset.

	Top-1	Top-3	Top-5	Top-10
LSH	0.874	0.906	0.923	0.928
Pairwise matching	<0.1	<0.1	<0.1	<0.1
Our method	**0.921**	**0.947**	**0.958**	**0.974**

Table 2.6: Results on DS-I Tobacco full dataset

average accuracy as we did in the previous experiment except that the full signatures are used to test. We use the same setups as the previous experiment, since the training stages are identical. The results are compared to the LSH method and the pairwise matching method reported in [62] [63]. Figure 2.22 and Table 2.6 show the results.

Our 92.1% top-1 rank accuracy outperforms both of the previous methods on this dataset. This proves that our method is not only specifically designed for partial signature matching, but also works well on full signature dataset.

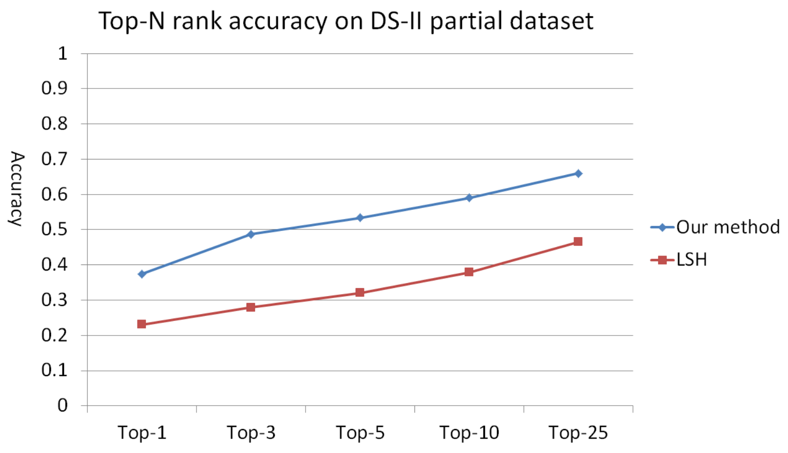

Figure 2.23: Performance curves on DS-II partial dataset.

2.3.4.5 Results on DS-II partial dataset

This dataset is extremely challenging since we only have one training sample per class to perform the 495-class classification task. To setup the experiment, the heights of all signatures are fixed to 166, and $r = 5$ and $\theta = 12$ shape context features are used. 230 salient regions are estimated from HDP after 1000 iterations, and empirically $K = 1500$ is chosen. Figure 2.23 and Table 2.7 show the results from our method and the LSH method. Our algorithm achieve 37.8% top-1 rank accuracy, which outperforms the LSH method by 23.1%. We didn't compare to the pairwise matching method since it is not scalable to large datasets.

2.3.4.6 Effects of parameters

The two parameters that have impact on the performance are the number of centers K in K-means algorithm and the number of salient regions R in sLDA. We test on different values of K and R to see how they change the performance.

Number of clusters: The choice of K depends on the data and the size of the dataset. A larger value of K tends to over-classify the data and make the classification more computationally intensive. A smaller value of K tends group incorrect data together. The results of K versus top-1 rank accuracy graphs on DS-I partial dataset and DS-II partial dataset are given in Figure 2.24 and Table 2.8, and Figure 2.25 and Table 2.9, at fixed $R = 90$ and $R = 230$ respectively.

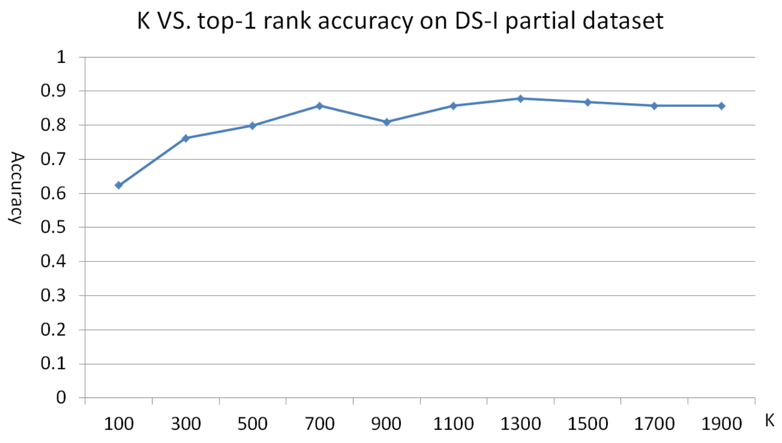

Figure 2.24: The impact of K on DS-I partial dataset.

Number of salient regions: The choice of R depends on the size of vocabulary and the observations. HDP provides us an alternative way to choose R. The top-1 rank accuracy comparisons between R predicted by HDP and other R values on DS-I partial

	Top-1	Top-3	Top-5	Top-10	Top-25
LSH	0.231	0.279	0.321	0.379	0.465
Our method	**0.378**	**0.487**	**0.534**	**0.590**	**0.660**

Table 2.7: Results on DS-II partial dataset

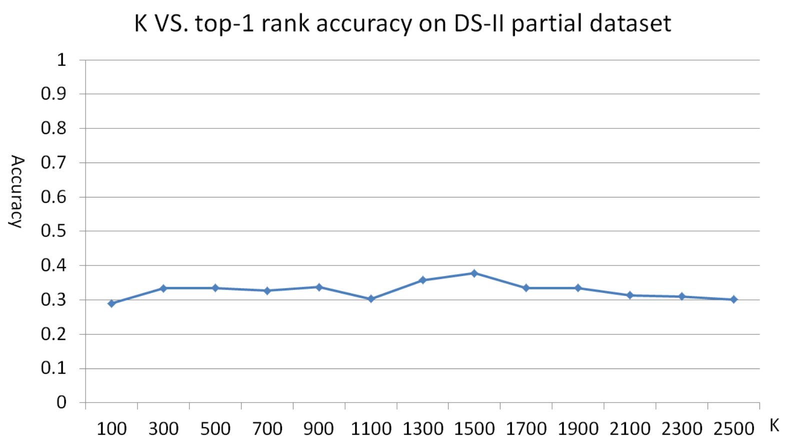

Figure 2.25: The impact of K on DS-II partial dataset.

$\#K$	100	300	500	700	900	1100	1300	1500	1700	1900
Accuracy	0.624	0.762	0.799	0.857	0.809	0.857	**0.878**	0.867	0.857	0.857

Table 2.8: Results from different K at $R = 90$ on DS-I partial set

dataset and DS-II partial dataset are given in Figure 2.26 and Table 2.10, and Figure 2.27 and Table 2.11, at $K = 1300$ and $K = 1500$ respectively.

From Figure 2.27 we see HDP overfit the training samples of DS-II partial dataset with a large value of R. This is due to the variety of the training samples.

$\#K$	300	500	700	900	1100	1300	1500	1700	1900	2100
Accuracy	0.334	0.335	0.327	0.337	0.303	0.358	**0.378**	0.335	0.335	0.314

Table 2.9: Results from different K at $R = 230$ on DS-II partial set

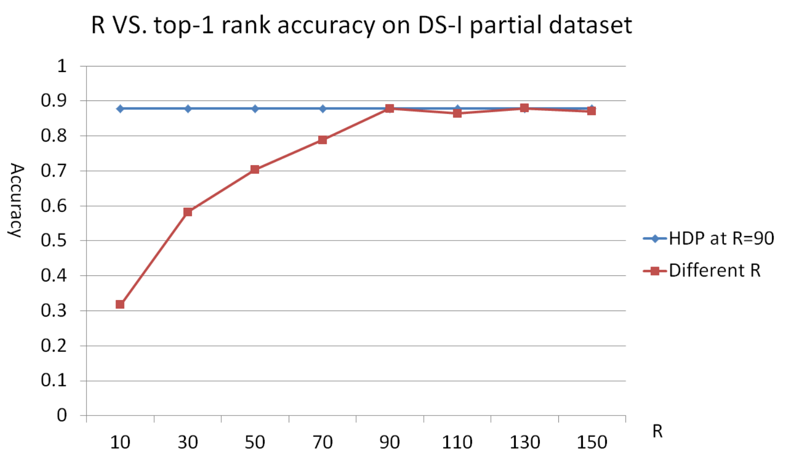

Figure 2.26: The impact of R and comparisons between different R and R predicted by HDP on DS-I partial dataset.

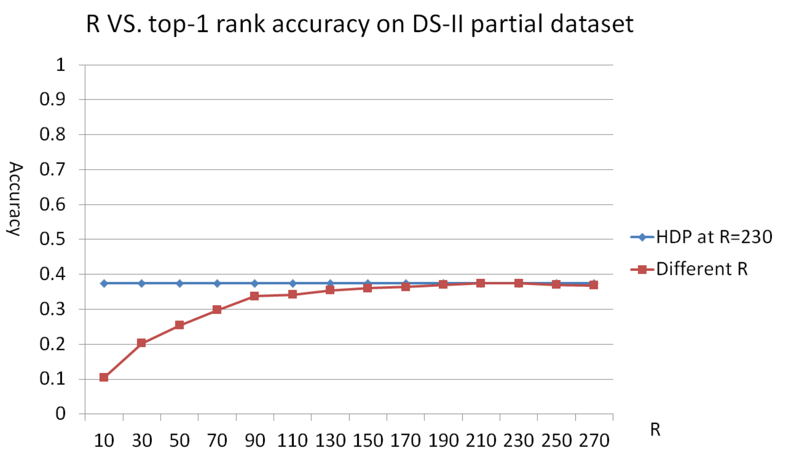

Figure 2.27: The impact of R and comparisons between different R and R predicted by HDP on DS-II partial dataset.

2.3.4.7 Failure examples

There are several common cases in DS-II partial dataset that will lead to a mismatching, such as the full signature of a query partial signature is close to another signature, the partial signature is more similar to the full signature of another author, signatures

$\#R$	10	30	50	70	90	110	130	150
Accuracy	0.317	0.582	0.704	0.788	**0.878**	0.864	0.878	0.870

Table 2.10: Results from different R at $K = 1300$ on DS-I partial dataset

$\#R$	70	90	110	130	150	170	190	210	230	250
Accuracy	0.298	0.337	0.342	0.354	0.360	0.364	0.370	0.378	**0.378**	0.370

Table 2.11: Results from different R at $K = 1500$ on DS-II partial dataset

are not informative because of too many handwritten texts, and signatures are so simple that limited information can be collected. Figure 2.28 shows four failure cases.

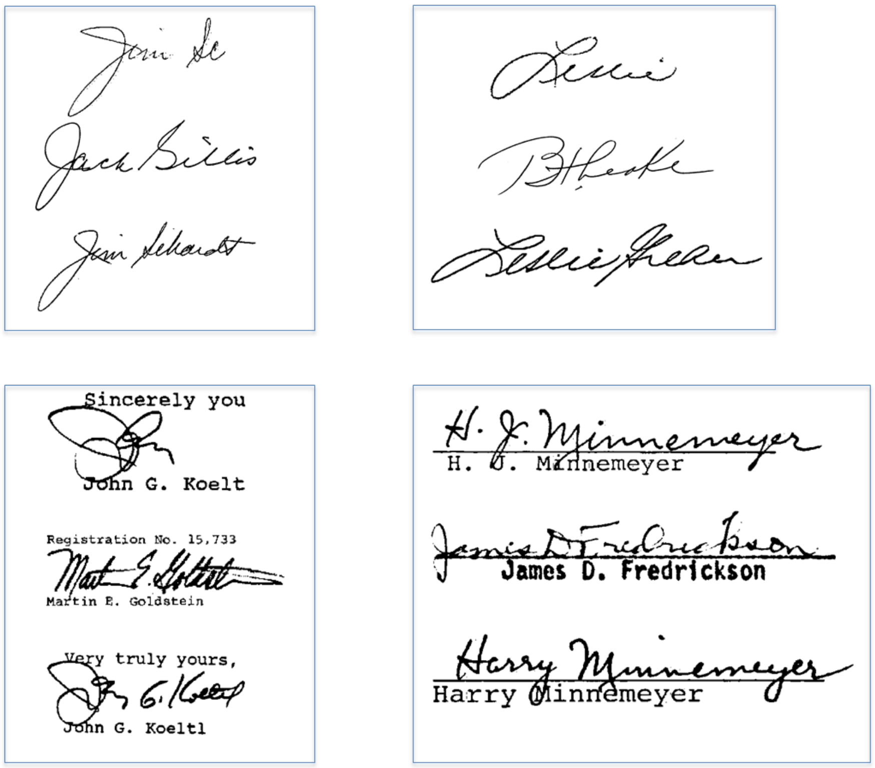

Figure 2.28: Failure examples for four query signatures. For each box, the first signature is a query signature; the middle signature is the incorrect match; the last signature is the true full signature.

Chapter 3: Deep Learning with Applications to Pedestrian Detection and Portrait Segmentation

3.1 Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection

3.1.1 Overview

In this work, we propose a deep neural network fusion architecture to address the pedestrian detection problem, called Fused Deep Neural Network (F-DNN). Compared to previous methods, the proposed system is faster while achieving better detection accuracy. The architecture consists of a pedestrian candidiate generator, which is obtained by training a deep convolutional neural network to have a high detection rate, albeit a large false positive rate. A novel network fusion method called soft-rejection based network fusion is proposed. It employs a classification network, consisting of multiple deep neural network classifiers, to refine the pedestrian candidates. Their soft classification probabilities are fused with the original candidates using the soft-rejection based network fusion method. A parallel semantic segmentation network, using deep dilated convolutions and context aggregation [64], delivers another soft confidence vote on the pedestrian candidates, which are further fused with the candidate generator and the classification network.

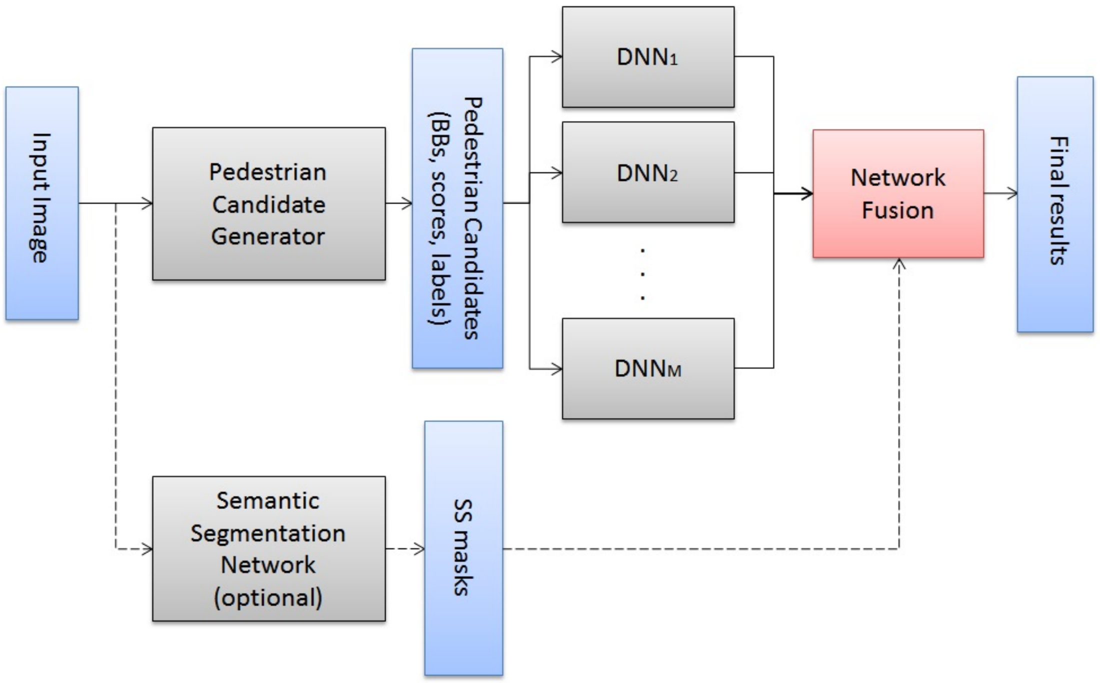
Input Image
Pedestrian Candidate Generator
Pedestrian Candidates (BBs, scores, labels)
DNN1
DNN2
DNNM
Network Fusion
Final results
Semantic Segmentation Network (optional)
SS masks

Figure 3.1: The whole pipeline of our proposed work.

Our work is evaluated on the Caltech Pedestrian dataset [14]. We improve log-average miss rate on the 'Reasonable' evaluation setting from 9.58% (previous best result [22]) to 8.65% (8.18% with semantic segmentation network). Meanwhile, our speed is 1.67 times faster (3 times faster for 'Reasonable' test). Our numerical results show that the proposed system is accurate, robust, and efficient.

3.1.2 The Fused Deep Neural Network

The proposed network architecture consists of a pedestrian candidate generator, a classification network, and a pixel-wise semantic segmentation network. The pipeline of the proposed network fusion architecture is shown in Figure 3.1.

For the implementation described In this work, the candidate generator is a single shot multi-box detector (SSD) [65]. The SSD generates a large pool of candidates with the goal of detecting all true pedestrians, resulting in a large number of false positives. Each pedestrian candidate is associated with its localization box coordinates and a confidence score. By lowering the confidence score threshold above which a detection candidate is accepted, candidates of various sizes and occlusions are generated from the primary detector. The classification network consists of multiple binary classifiers which are run in parallel. We propose a new method for network fusion called soft-rejection based network fusion (SNF). Instead of performing hard binary classification, which either accepts or rejects candidates, the confidence scores of the pedestrian candidates are boosted or discounted based on the aggregated degree of confidence in those candidates from the classifiers. We further propose a method for utilizing the context aggregation dilated convolutional network with semantic segmentation (SS) as another classifier and integrating it into our network fusion architecture. However, due to the large input size and complex network structure, the improved accuracy comes at the expense of a significant loss in speed.

3.1.2.1 Pedestrian Candidate Generator

We use SSD to generate pedestrian candidates. The SSD is a feed-forward convolutional network which has a truncated VGG16 as the base network. In VGG16 base, pool5 is converted to 3×3 with stride one, and fc6 and fc7 are converted to convolutional layers with atrous algorithm [64]. Additional 8 convolutional layers and a global average pooling layer are added after the base network and the size of each layer decreases

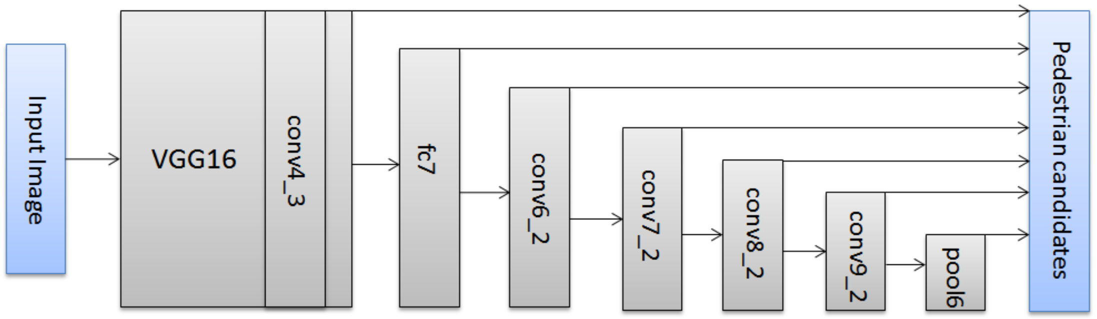

Figure 3.2: The structure of SSD. 7 output layers are used to generate pedestrian candidates in this work.

progressively. Layers 'conv4_3', 'fc7', 'conv6_2', 'conv7_2', 'conv8_2', 'conv9_2', and 'pool6' are used as the output layers. Since 'conv4_3' has a much larger feature scale, an L2 normalization technique is used to scale down the feature magnitudes [66]. After each output layer, bounding box(BB) regression and classification are performed to generate pedestrian candidates. Figure 3.2 shows the structure of SSD.

For each output layer of size $m \times n \times p$, a set of default BBs at different scales and aspect ratios are placed at each location. $3 \times 3 \times p$ convolutional kernels are applied to each location to produce classification scores and BB location offsets with respect to the default BB locations. A default BB is labeled as positive if it has a Jaccard overlap greater than 0.5 with any ground truth BB, otherwise negative (as shown in Equation (3.1)).

$$label = \begin{cases} 1, & \text{if } \frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d} \cup A_{BB_g}} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where A_{BB_d} and A_{BB_g} represent the area covered by the default BB and the ground true

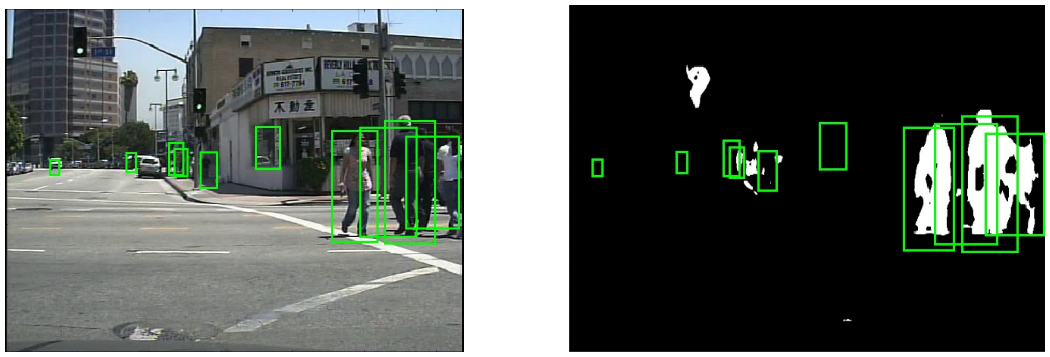

Figure 3.3: Left figure shows the pedestrian candidates' BBs on an image. Right figure shows the SS mask over the BBs. We can visualize several false positives (such as windows and cars) are softly rejected by the SS mask.

BB, respectively. The training objective is given as Equation (3.2):

$$L = \frac{1}{N}(L_{conf} + \alpha L_{loc}) \tag{3.2}$$

where L_{conf} is the softmax loss and L_{loc} is the Smooth L1 localization loss [26], N is the number of positive default boxes, and α is a constant weight term to keep a balance between the two losses. For more details about SSD please refer to [65]. Since SSD uses 7 output layers to generate multi-scale BB outputs, it provides a large pool of pedestrian candidates varying in scales and aspect ratios. This is very important to the following work since pedestrian candidates generated here should cover almost all the ground truth pedestrians, even though many false positives are introduced at the same time. Since SSD uses a fully convolutional framework, it is fast.

3.1.2.2 Classification Network and Soft-rejection based DNN Fusion

The classification network consists of multiple binary classification deep neural networks which are trained on the pedestrian candidates from the first stage. All candidates with confidence score greater than 0.01 and height greater than 40 pixels are collected as the new training data for the classification network. For each candidate, we scale it to a fixed size and directly use positive/negative information collected from Equation (3.1) for labeling.

After training, verification methods are implemented to generate the final results. Traditional hard binary classification results in hard rejection and will eliminate a pedestrian candidate based on a single negative vote from one classification network. Instead, we introduce the SNF method which works as follows: Consider one pedestrian candidate and one classifier. If the classifier has high confidence about the candidate, we boost its original score from the candidate generator by multiplying with a confidence scaling factor greater than one. Otherwise, we decrease its score by a scaling factor less than one. We define "confident" as a classification probability of at least a_c. To prevent any classifier from dominating, we set b_c as the lower bound for the scaling factor. Let p_m be the classification probability generated by the m_{th} classifier for this candidate, the scaling factor is computed as Equation (3.3).

$$a_m = \max(p_m \times \frac{1}{a_c}, b_c) \tag{3.3}$$

where a_c and b_c are chosen as 0.7 and 0.1 by cross validation. To fuse all M classifiers, we multiply the candidate's original confidence score with the product of the confidence scaling factors from all classifiers in the classification network. This can be expressed as Equation (3.4).

$$S_{FDNN} = S_{SSD} \times \prod_{m=1}^{M} a_m \tag{3.4}$$

The key idea behind SNF is that we don't directly accept or reject any candidates, instead we scale them with factors based on the classification probabilities. This is because a wrong elimination of a true pedestrian (e.g. as in hard-binary classification) cannot be corrected, whereas a low classification probability can be compensated for by larger classification probabilities from other classifiers.

3.1.2.3 Pixel-wise semantic segmentation for object detection reinforcement

We utilize an SS network, based on deep dilated convolutions and context aggregation [64], as a parallel classification network. The SS network is trained on the Cityscapes dataset for driving scene segmentation [67]. To perform dense prediction, the SS network consists of a fully convolutional VGG16 network, adapted with dilated convolutions as the front end prediction module, whose output is fed to a multi-scale context aggregation module, consisting of a fully convolutional network whose convolutional layers have increasing dilation factors.

An input image is scaled and directly processed by the SS network, which produces a binary mask with one color showing the activated pixels for the pedestrian class, and the other color showing the background. We consider both the 'person' and 'rider' categories in Cityscapes dataset as pedestrians, and the remaining classes as background. The SS mask is intersected with all detected BBs from the SSD. We propose a method to fuse the SS mask and the original pedestrian candidates. The degree to which each candidate's BB overlaps with the pedestrian category in the SS activation mask gives a measure of the confidence of the SS network in the candidate generator's results. We use the following strategy to fuse the results: If the pedestrian pixels occupy at least 20% of the candidate BB area, we accept the candidate and keep its score unaltered; Otherwise, we apply SNF to scale the original confidence scores. This is summarized in Equation (3.5):

$$S_{all} = \begin{cases} S_{FDNN}, & \text{if } \frac{A_m}{A_b} > 0.2 \\ S_{FDNN} \times \max(\frac{A_m}{A_b} \times a_{ss}, b_{ss}), & \text{otherwise} \end{cases} \tag{3.5}$$

where A_b represents the area of the BB, A_m represents the area within A_b covered by semantic segmentation mask, a_{ss}, and b_{ss} are chosen as 4 and 0.35 by cross validation. As we don't have pixel-level labels for pedestrian detection datasets, we directly implement the SS model [64] trained on the Cityscape dataset [67]. Figure 3.3 shows an example of this method and how we fuse it into the existing model.

SNF with an SS network is slightly different from SNF with a classification network. The reason is that the SS network can generate new detections which have not been produced by the candidate generator, which is not the case for the classification network. To address this, the proposed SNF method eliminates new detections from the SS network.

3.1.3 Experiments and result analysis

3.1.3.1 Data and evaluation settings

We evaluate the proposed method on the most popular pedestrian detection dataset: Caltech Pedestrian dataset. The Caltech Pedestrian dataset contains 11 sets (S0-S10), where each set consists of 6 to 13 one-minute long videos collected from a vehicle driving through an urban environment. There are about 250,000 frames with about 350,000 annotated BBs and 2300 unique pedestrians. Each BB is assigned with one of the three labels: 'Person', 'People' (large group of individuals), and 'Person?' (unclear identifications). The original frame size is 480×640. The log-average miss rate (L-AMR) is used as the performance evaluation metric [14]. L-AMR is computed evenly spaced in log-space in the range 10^{-2} to 10^{0} by averaging miss rate at the rate of nine false positives per image (FPPI) [14]. There are multiple evaluation settings defined based on the height and visible part of the BBs. The most popular settings are listed in Table 3.1.

3.1.3.2 Training details and results

To train the SSD candidate generator, all images which contain at least one annotated pedestrian from Caltech training set, ETH dataset [71], and TudBrussels dataset [72] are used. By using both original and flipped images, it provides around 68,000 images. Among all annotations, only 'Person' and 'People' categories are included. We further classify 'Person' into 'Person_full' and 'Person_occluded'. This results in 109,000 pedestrians in 'Person_full', 60,000 pedestrians in 'Person_occluded', and 35,000 pedestrians

Setting	Description
Reasonable	50+ pixels. Occ. none or partial
All	20+ pixels. Occ. none, partial, or heavy
Far	30- pixels
Medium	30-80 pixels
Near	80+ pixels
Occ. none	0% occluded
Occ. partial	1-35% occluded
Occ. heavy	35-80% occluded

Table 3.1: Evaluation settings for Caltech Pedestrian dataset.

in 'People'. All the images are in their original size 480×640. To set up the SSD model, we place 7 default BBs with aspect ratios [0.1, 0.2, 0.41a, 0.41b, 0.8, 1.6, 3.0] on top of each location of all output feature maps. All default BBs except 0.41b have relative heights [0.05, 0.1, 0.24, 0.38, 0.52, 0.66, 0.80] for the 7 output layers. The heights for 0.41b are [0.1, 0.24, 0.38, 0.52, 0.66, 0.80, 0.94]. Since 0.41 is the average aspect ratio for all annotated pedestrians, we use two default BBs with slightly different heights. The aspect ratio '1.6' and '3.0' are designed for 'People'. By doing so, we can generate a rich pool of candidates so as not to lose any ground truth pedestrians. We then fine-tune our SSD model from the Microsoft COCO [73] pre-trained SSD model for 40k iterations using stochastic gradient descent (SGD), with a learning rate of 10^{-5}. All the layers after each output layer are randomly initialized and trained from scratch.

Method	Reas.	All	Far	Medium	Near	Occ. none	Occ. partial	Occ. heavy
SCF+AlexNet [19]	23.32%	70.33%	100%	62.34%	10.16%	19.99%	48.47%	74.65%
SAF R-CNN [25]	9.68%	62.6%	100%	51.8%	**0%**	7.7%	24.8%	64.3%
MS-CNN [68]	9.95%	60.95%	97.23%	49.13%	2.60%	8.15%	19.24%	59.94%
DeepParts [69]	11.89%	64.78%	100%	56.42%	4.78%	10.64%	19.93%	60.42%
CACT-Deep [70]	11.75%	64.44%	100%	53.23%	3.99%	9.63%	25.14%	65.78%
RPN+BF [22]	9.58%	64.66%	100%	53.93%	2.26%	7.68%	24.23%	69.91%
F-DNN (Ours)	**8.65%**	**50.55%**	**77.37%**	**33.27%**	2.96%	**7.10%**	**15.41%**	**55.13%**
F-DNN+SS (Ours)	**8.18%**	**50.29%**	**77.47%**	**33.15%**	2.82%	**6.74%**	**15.11%**	**53.76%**

Table 3.2: Detailed breakdown performance comparisons of our models and other state-of-the-art models on the 8 evaluation settings. All numbers are reported in L-AMR.

To train the classification network, we use all pedestrian cadidates generated by SSD as well as all ground truth BBs. All the training samples are horizontally flipped with probability 0.5. This results in $69,000$ positive samples and $163,000$ negative samples, with a ratio $1 : 2.4$. As the high similarity between consecutive frames sometimes leads to identical training samples, we scale all samples to a fixed size of 250×250, and then randomly crop a 224×224 patch for data augmentation (center cropping for testing). We then fine-tune two models in parallel: ResNet-50 [74] and GoogleNet [75], from their ImageNet pre-trained models. All models are trained using SGD with a learning rate of 10^{-4}.

For the SS network, since there is a lack of well-labeled SS dataset for pedestrian detection, we directly implement the network trained on the Cityscapes dataset with image size 1024×2048. To preserve the aspect ratio, we scale all images to height 1024, then pad black pixels on both sides.

All the above models are built on Caffe deep learning framework [76].

Evaluation on Caltech Pedestrian : The detailed breakdown performance of our two models (without and with semantic segmentation) on this dataset is shown in Table 3.2. We compare with all the state-of-the-art methods reported on Caltech Pedestrian website. We can see that both of our models significantly outperform others on almost all evaluation settings. On the 'Reasonable' setting, our best model achieves 8.18% L-AMR, which has a 14.6% relative improvement from the previous best result 9.58% by RPN+BF. On the 'All' evaluation setting, we achieve 50.29%, a relative improvement of 17.5% from 60.95% by MS-CNN [68]. The L-AMR VS. FPPI plots for the 'Reasonable' and 'All' evaluation settings are shown in Figure 3.5 and Figure 3.6. F-DNN refers to fusing the SSD with the classification network, whereas F-DNN+SS refers to fusing the SSD with both the classification network and the SS network. Results from VJ [77] and HOG [16] are plotted as the baselines on this dataset.

3.1.3.3 Result analysis

Effectiveness of classification network We explore how effective the classification network refines the original confidence scores of the pedestrian candidates. As many false positives are introduced from SSD, the main goal of the classification network is to decrease the scores of the false positives. By using ResNet-50 with classification probability 0.7 as the confidence scaling threshold, 96.7% scores of the false positives are decreased on the Caltech Pedestrian testing set. This improves the performance on the 'Reasonable' setting from 13.07% to 8.98%. A similar result is obtained by GoogleNet-97.1% scores of the false positives are decreased, which improves performance from 13.07% to 9.41%. As the two classifiers do not decrease scores of the same set of false positives, by fusing their results with SNF, almost all false positives are covered, and the L-AMR is further improved to 8.65%. Concerning limits to performance, if we were able to train an oracle classifier with classification accuracy 100%, the L-AMR would be improved to only 4%. This is shown in Table 3.3.

Method	Reasonable
SSD	13.06%
SSD+GoogleNet	9.41%
SSD+ResNet-50	8.97%
SSD+GoogleNet+ResNet-50 (F-DNN)	8.65%
SSD+Oracle classifier	4%

Table 3.3: Effectiveness of the classification network.

Soft rejection versus hard rejection   SNF plays an important role in our system. Hard rejection is defined as eliminating any candidate which is classified as a false positive by any of the classifiers. The performance of hard-rejection based fusion depends on the performance of all classifiers. A comparison between the two methods is shown in Table 3.4. We also compare against the case when the SSD is fused with the SS network only, labeled as 'SSD+SS'. For the classification network, a 0.5 classification probability threshold is used for hard rejection, while for the SS network, an overlap ratio of 5% is used. We can see that hard rejection hurts performance significantly, especially for the classification network. All numbers are reported in L-AMR on the 'Reasonable' evaluation setting.

Method	Hard rejection	Soft rejection
SSD+SS	13.4%	11.57%
F-DNN	20.67%	8.65%
F-DNN+SS	22.33%%	8.18%

Table 3.4: Performance comparisons on Caltech 'Reasonable' setting between soft rejection and hard rejection. The original L-AMR of SSD alone is 13.06%

Robustness on challenging scenarios The proposed method performs much better than all other methods on challenging scenarios such as the small pedestrian scenario, the occluded pedestrian scenario, crowded scenes, and the blurred input image. Figure 3.7 visualizes the results of the ground truth annotation, our method, and RPN-BF (previous state-of-the-art method). The four rows represent the four challenging scenarios and the four columns represent the BBs from the ground true annotations, the pedestrian candidates generated by SSD alone, our final detection results, and the results from RPN-BF method. By comparing the third column with the second column, we can see that the classification network and the SS network are able to filter out most of the false positives introduced by the SSD detector. By comparing the third column with the last column, we can see our method is more robust and accurate on the challenging scenarios than

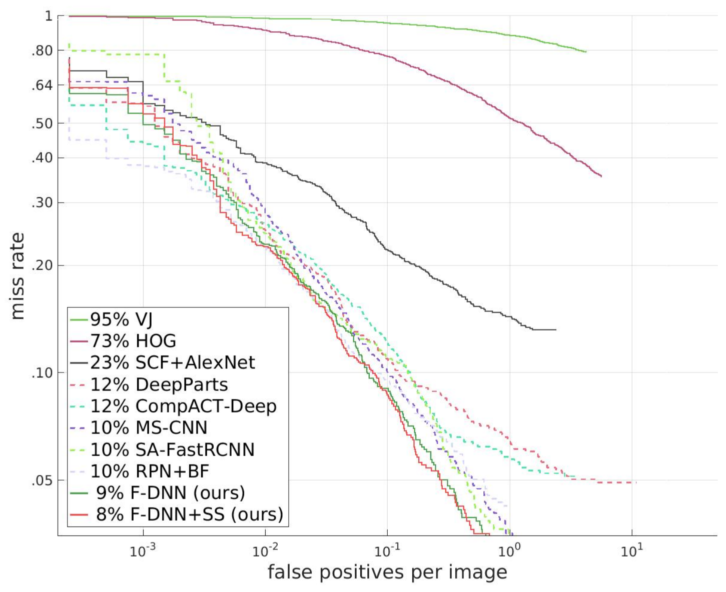

Figure 3.4: L-AMR VS. FPPI plot on 'Reasonable' evaluation setting.

RPN-BF method.

Speed analysis There are 4 networks in the proposed work. As the SSD uses a fully convolutional framework, its processing time is 0.06s per image. For the speed of the classification network, we perform two tests: The first test runs on all pedestrian candidates; The second test runs only on candidates above 40 pixels in height. The second test is targeting only the 'Reasonable' evaluation setting. Using parallel processing, the speed for the classification network equals its slowest classifier, which is 0.24s and 0.1s per image for the two tests. The overall processing time of F-DNN is 0.3s and 0.16s per image, which is 1.67 to 3 times faster than other methods. Note that the speed for GoogleNet on 'Reasonable' test is only 0.05s per image. If we only fuse the SSD with GoogleNet, we can still achieve the state-of-the-art performance while being 0.11s per image. We also

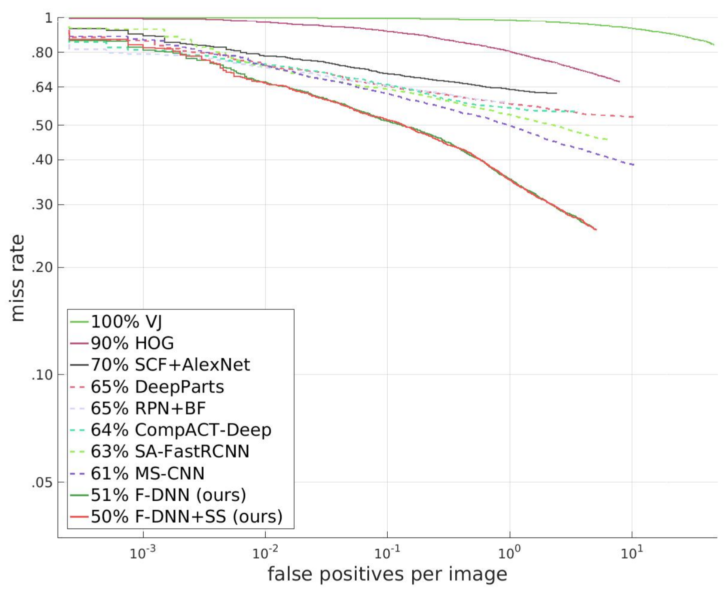

Figure 3.5: L-AMR VS. FPPI plot on 'All' evaluation setting.

Method	Speed on TITAN X (seconds per image)
CompACT-Deep	0.5
SAF R-CNN	0.59
RPN+BF	0.5
F-DNN	**0.3**
F-DNN (Reasonable)	**0.16**
SSD+GoogleNet (Reasonable)	**0.11**
SSD+SqueezeNet (Reasonable)	**0.09**
F-DNN+SS	2.48

Table 3.5: A comparison of speed among the state-of-the-art models.

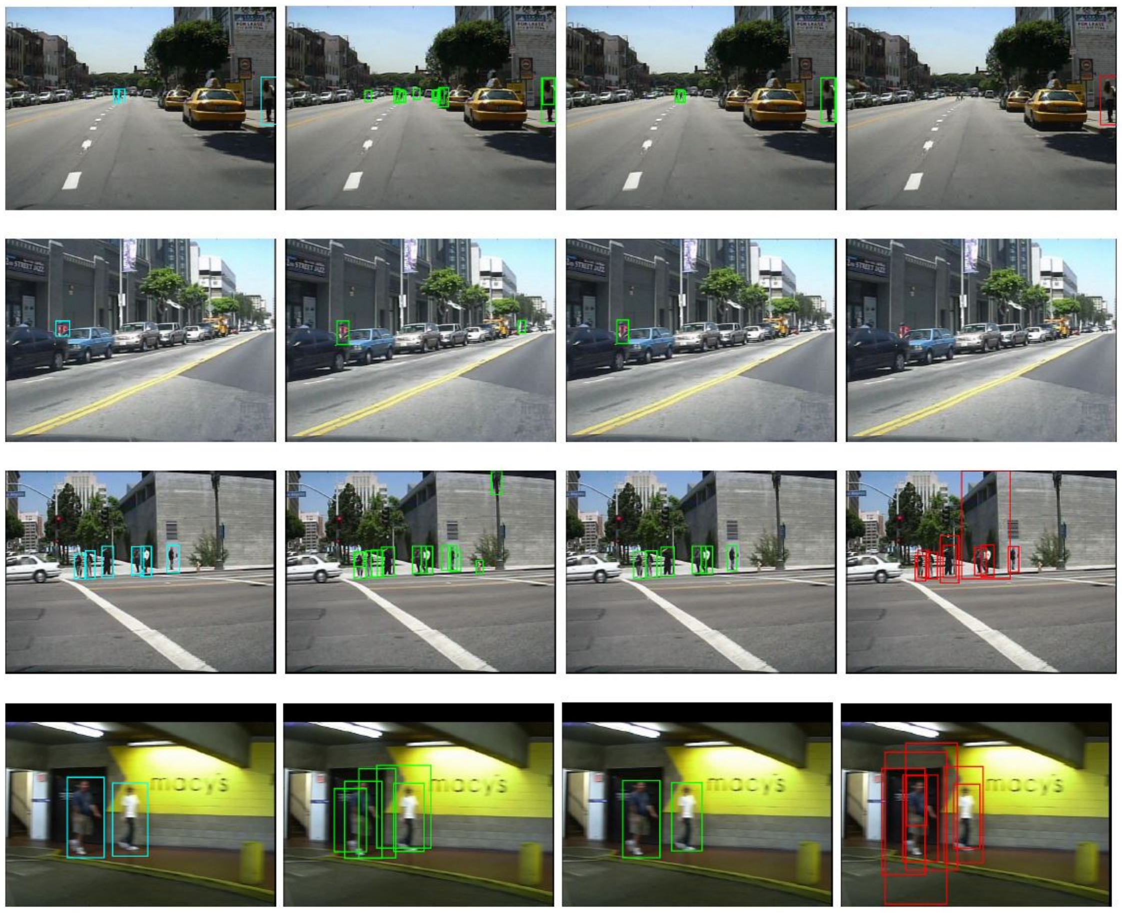

Figure 3.6: Detection comparisons on four challenging pedestrian detection scenarios. The four rows represent the small pedestrian scenario, the occluded pedestrian scenario, the crowed scenes, and the blurred input image. The four columns represent the ground true annotations, the pedestrian candidates generated by SSD alone, our final detection results, and the results from the RPN-BF method.

test fusing SSD with SqueezeNet [78] only to achieve 0.09s per image, which is above 10 frames per second (with a L-AMR around 10.8%). As the SS network uses 1024×2048 size input with a more complex network structure, it's processing time is 2.48s per image. As it can be processed in parallel with the whole F-DNN pipeline, the overall processing time will be limited to 2.48s per image. We use one NVIDIA TITIAN X GPU for all the speed tests. All the classifiers in the classification network are processed in parallel on one single GPU. Table 3.5 compares the processing speed of our methods and the other methods.

3.2 Efficient Pedestrian Detection using Deep Neural Network Fusion

3.2.1 Overview

This work is extended from the work in the previous section [79]. We propose the Fused Deep Neural Network 2 (F-DNN2). The whole architecture of this work consists of two parallel systems: the pedestrian detection system and the semantic segmentation system.

The pedestrian detection system consists of a pedestrian candidate generator, which is trained to have a high detection rate, albeit a lot of false positives are introduced at the same time. To collect the detected bounding boxes as the training data to the following steps, we design a novel soft-label method to assign floating point labels. The value of the soft-label is set to be the largest overlap ratio between the current detected bounding box and all the ground-truth bounding boxes. The candidate generator is followed by a classification system which consists of multiple classification networks and a soft-rejection

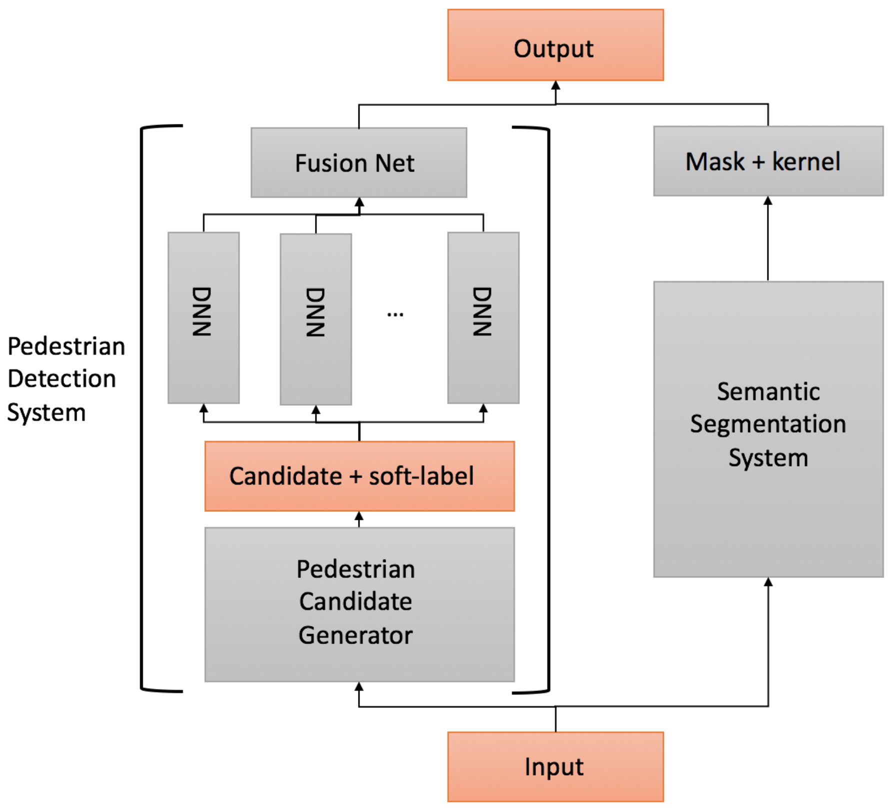
Output
Fusion Net
Mask + kernel
DNN
DNN
...
DNN
Pedestrian Detection System
Semantic Segmentation System
Candidate + soft-label
Pedestrian Candidate Generator
Input

Figure 3.7: The whole pipeline of our proposed work.

based fusion network. Here we use the idea of ensemble learning: Multiple networks with different structures are trained separately and their opinions are fused together at the end by the fusion network.

In parallel with the pedestrian detection system, we run an semantic segmentation network, using deep dilated convolutions and context aggregation [64], to further refine the pedestrian detection results. However, due to the large input size and complex network structure, the improved accuracy comes at the expense of a loss in speed. The whole architecture is shown in Figure 3.7.

The novel ideas In this work are the soft-label method, the soft-rejection based fusion network, the new kernel based method to fuse the results of the semantic segmentation system and the detection system. The new techniques help to significantly increase the performance on Caltech dataset from 8.18% to 7.65%. We also extend the model to work on more classes besides pedestrian, such as car, cyclist. Besides the Caltech Pedestrian dataset, we evaluate on three more popular pedestrian detection datasets: INRIA, ETH, and KITTI. We outperformed all the previous state-of-the-arts on Caltech, INRIA, and ETH in both accuracy and speed, and achieved comparable results on KITTI. More experiment analysis is conducted to explain the effectiveness of our system.

3.2.2 Pedestrian Detection system

3.2.2.1 Pedestrian Candidate Generator

In order to quickly obtain pedestrian candidates in various sizes and aspect ratios at all possible locations of the input image, we use an single shot multi-box detector (SSD)

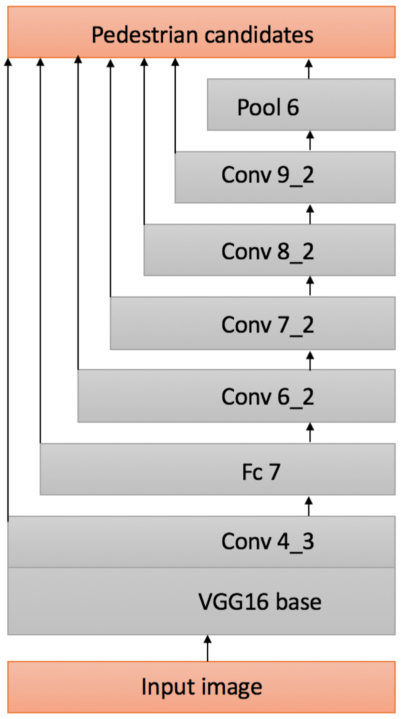

Figure 3.8: The structure of the pedestrian candidate generator.

as the candidate generator. The main reason we select SSD instead of other system is that it uses multiple feature maps as the output layers. By lowering the accepting threshold of the confidence score, it outputs a large number of pedestrian candidates which are very likely to cover all the ground-truth pedestrians. Since it has a fully convolutional framework, it's also very fast in speed.

The SSD network is a feed-forward convolutional network which consists of a VGG16 base, 8 convolutional layers above it, and a global pooling layer at the top. In the VGG16 base, the kernel size of the pool5 layer is set to 3×3 and the stride is set to one, and the fc6 and fc7 layers are converted to dilated convolutional layers. Bounding box regression and classification are performed on the feature maps of 'conv4_3', 'fc7', 'conv6_2', 'conv7_2', 'conv8_2', 'conv9_2', and 'pool6' to generate the pedestrian candidates. A L2 normalization technique is used to scale down the feature magnitudes [66] of 'conv4_3' since it has a much larger feature scale than other output layers. The network structure is shown in Figure 3.8.

To generate the pedestrian candidates, a set of default bounding boxes are placed on top of each output feature map. At every pixel location of the 7 output layers, we place 6 default bounding boxes with aspect ratios [0.1, 0.2, 0.41, 0.8, 1.6, 3.0] and relative heights [0.05, 0.1, 0.24, 0.38, 0.52, 0.66, 0.80]. Since '0.41' is the average aspect ratio of all the pedestrian annotations, we place another set of default bounding boxes with relative heights [0.1, 0.24, 0.38, 0.52, 0.66, 0.80, 0.94] for it. In the training stage, we further categories all the pedestrians into three classes: 'Full pedestrian', 'Occluded pedestrian', and 'People'. The 'People' class is defined as a group of people that close to or overlap with each other. The aspect ratio '1.6' and '3.0' are designed for 'People'. For each default bounding box, a $3 \times 3 \times p$ convolutional kernel is applied to produce classification scores and perform bounding box regression by calculating the location offsets with respect to the default bounding box.

The multi-task training objective of SSD is given by Equation (3.6)

$$L = \frac{1}{N}(L_{conf} + \alpha L_{loc}) \tag{3.6}$$

where L_{conf} is the softmax loss for the classification task and L_{loc} is the smooth L1 localization loss [26], N is the number of positive default boxes, and α is a constant weight term to keep a balance between the two losses. For more details about SSD please refer to [65]. Since SSD uses 7 output layers to generate multi-scale BB outputs, it provides a large pool of pedestrian candidates varying in scales and aspect ratios.

3.2.2.2 Classification System

networks and the soft-label method When preparing the training data for classification system, the hard-label used in common object detection problem assigns a binary label to each pedestrian candidate bounding box by thresholding the overlap ratio between this bounding box and the ground-truth bounding boxes. However, this is not the optimal strategy, especially when the overlap ratio is close to the threshold. In this work, we introduce the soft-label strategy to label the pedestrian candidates. The soft-label method will assign a floating point label to each pedestrian candidate using the largest overlap ratio between the current pedestrian candidate and all the ground-truth bounding boxes. Suppose we have a pedestrian candidate and the ground-truth bounding box it overlaps most. The soft-labels for the pedestrian class $label_{ped}$ and the background class $label_{bg}$

are calculated as Equation (3.7) and (3.8).

$$label_{ped} = \frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d}} \tag{3.7}$$

$$label_{bg} = 1 - \frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d}} \tag{3.8}$$

To update the back-propagation algorithm for this change, we modify the cross entropy loss for the classification networks as Equation (3.9). Note that for the conventional binary labeling method, l_i is the indicator function, which equals to 1 for the correct class and 0 otherwise. Minimizing the cross entropy is equivalent to maximizing the softmax probability of the correct class. In our case, the softmax probabilities of all the classes are used to contribute to the cross entropy loss. The floating point soft-labels will determine how much each class contributes.

$$\epsilon = -logL(\theta|t, z) = -\sum_{i=1}^{c} l_i log(y_i) \tag{3.9}$$

where l_i and y_i are the soft-label and the softmax probability for class i. Note $\sum_i l_i = 1$. When doing back-propagation, the derivative of the cross entropy cost function with respect to class i is calculated as Equation (3.10).

$$\begin{aligned}\frac{\partial \epsilon}{\partial z_i} &= -\sum_{j=1}^{c} l_j \frac{\partial log(y_j)}{\partial z_i} = -\frac{l_i}{y_i}\frac{\partial y_i}{\partial z_i} - \sum_{j\neq i}^{c} \frac{l_j}{y_j}\frac{\partial y_j}{\partial z_i} \\ &= -\frac{l_i}{y_i} y_i(1-y_i) - \sum_{j\neq i}^{c} \frac{l_j}{y_j}(-y_j y_i) \\ &= -l_i + y_i \sum_{j=1}^{c} l_j = y_i - l_i\end{aligned} \tag{3.10}$$

We note that Equation (5) shows the gradient for the conventional method, where a training sample has label 1 for the correct class, and label 0 for the incorrect classes. We also note that it still holds for the soft-label schemes as long as all the soft labels sum to 1.

We devise a hybrid soft-label/hard-label strategy, as follows: If the overlap ratio between the pedestrian candidate and the ground truth bounding box is lower than threshold th_a or greater than threshold th_b, we think it's safe to label the current sample as background or pedestrian with probability one. For the candidates with intermediate confidence, the soft-label method is used to assign floating point labels, and we normalize the range [th_a, th_b] to [0, 1]. Equation (3.11) and (3.12) illustrates the overall idea of the soft-label method.

$$label_{ped} = \begin{cases} 1, & \text{if } \frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d}} > th_b \\ 0, & \text{if } \frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d}} < th_a \\ \frac{\frac{A_{BB_d} \cap A_{BB_g}}{A_{BB_d}} - th_a}{th_b - th_a}, & \text{otherwise} \end{cases} \tag{3.11}$$

$$label_{bg} = 1 - label_{ped} \tag{3.12}$$

After we generate a large pool of pedestrian candidates in the previous step, a lot of false positives are introduced at the same time. In this step, we attempt to eliminate or decrease the confidence in many of the false positives. Here we use the idea of ensemble learning: The classification network consists of multiple classification deep neural networks which have different network structures but trained with the same input data. The opinions of all the networks are combined at the end. By doing so, it's more likely to get a lower error than each of the single network. Since it's hard to bias towards each of the single network, it's also less likely to overfit.

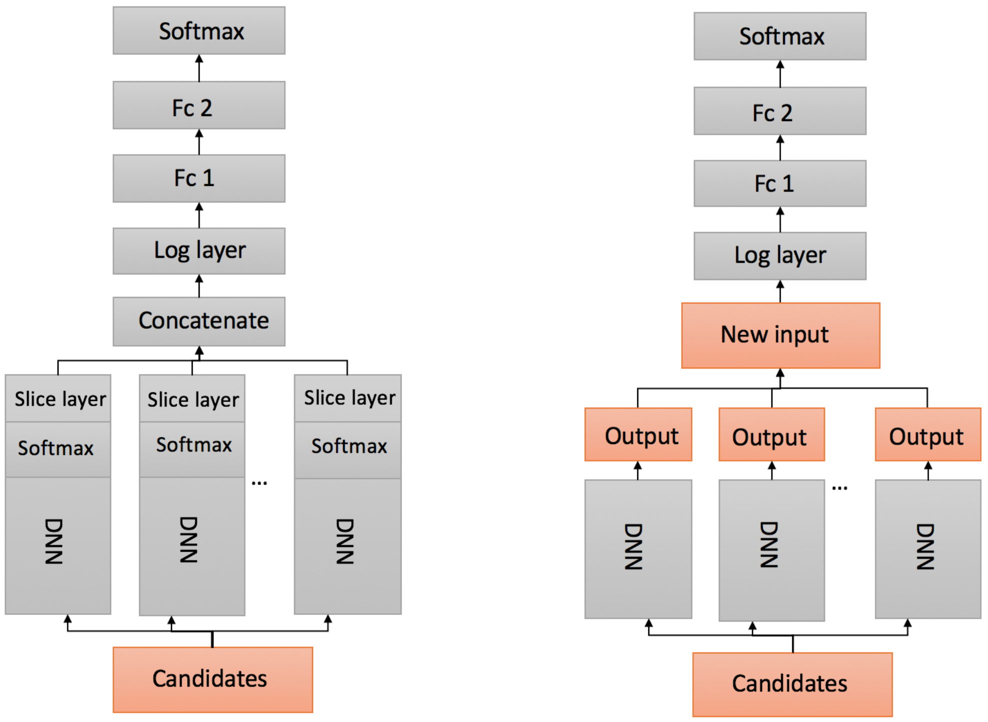

Figure 3.9: The two fusion network designs. The left structure is an end-to-end training scheme. The right structure trains all the networks separately.

Soft-rejection based fusion   The last step of the classification system is to fuse the results of all the classification networks together. There are several conventional methods commonly used such as computing the mean of all results, majority voting, or the hard-rejection method, which classifies the input to be a false positive if any of the network classifies the input as a false positive. In this work, as we use classification networks with different structures, we expect each network to work well in some of the subcategories while performing mediocre in other categories. We introduce the soft-rejection based fusion method. The soft-rejection based fusion method can be described as: For one pedestrian candidate, the k_{th} classification network gives us a classification probability p_k. If p_k is higher than a threshold t_1, we generate a scaling factor s_k greater than one to boost the initial confidence score generated by SSD. Otherwise, we generate a scaling factor less than one to decrease the initial confidence score. To prevent any of the classification network from dominating the final results, we set a lower bound t_2 to the scaling factors. The scaling factors coming out from all the classification networks are further multiplied together with the initial confidence score to produce the final score. The idea behind this is that instead of accepting or rejecting any candidate, we boost or decrease their scores instead. This is because a poor classification network can be compensated by other good ones, whereas, a wrong elimination of a true pedestrian cannot be corrected. The idea is illustrated in Equation (3.13) and Equation (3.14).

$$s_k = \max(p_k \times \frac{1}{t_1}, t_2) \tag{3.13}$$

$$S_{FDNN} = S_{SSD} \times \prod_{k=1}^{K} s_k \tag{3.14}$$

Soft-rejection based fusion network The values of all the parameters in the soft-rejection based fusion method are selected by cross-validation in the previous work. This is ad hoc and difficult to generalize to new datasets. Instead of hand-craft all parameters, we use a neural network to train them while keeping the idea of the soft-rejection based fusion method. We call the new method soft-rejection based fusion network.

To keep the idea of the previous hand-craft method, we design the fusion network as follows: Suppose we have $p_1, p_2, ..., p_K$ as the inputs to the fusion network, where p_k is the output of the k_{th} classification network. The input layer is followed by two fully connected layers, each has 500 neurons, and one softmax layer to predict the classification probability. By the nature of the neural network, the final result is scaled by the exponential function of the weighted sum of all classification probabilities. To follow the soft-rejection based fusion method, where the final result is scaled by the multiplication of all classification probabilities, we add one log layer after the input layer. At last, we have the form of Equation (3.15)

$$\begin{aligned} S_{FDNN} &= S_{SSD} * \exp(\sum_{k=1}^{K} w_k * log(p_k)) \\ &= S_{SSD} * \prod_{k=1}^{K} p_k^{w_k} \end{aligned} \tag{3.15}$$

where the w_k are the new parameters to be learnt by the fusion network.

Train the classification system There are two ways to train the classification system. The first method is to train an end-to-end system. For all the classification networks, we remove their loss layers and concatenate the output neurons for the pedestrian class from the softmax layers to form the input layer to the fusion network. This system has classification networks as branches and merged together by the fusion network at the end. This is shown at the left of Figure 3.9. However, since we train all the networks together, the structure grows huge and is prone to overfitting. What's more, since all the branches have different structures, they require different settings of optimal hyper-parameters and have different converging speed, it's much more difficult to train than the first method.

The second method is to train the classification networks first, and use the output probabilities to train the fusion network separately. Since this is straightforward and easy to implement, as shown at the right figure of Figure 3.9, this method is finally used In this work.

3.2.3 Pixel-wise Semantic Segmentation System

The recent pixel-wise semantic segmentation systems have shown great performance on tasks working with high-resolution digital images. In our work, we utilize an semantic segmentation system based on deep dilated convolutions and context aggregation [64] running in parallel with the pedestrian detection system to further refine the detection results at the end of the whole system. The network is trained on the Cityscapes dataset for driving scene segmentation [67]. To perform dense prediction, the network consists of a fully convolutional VGG16 network, adapted with dilated convolutions as the front end prediction module, whose output is fed to a multi-scale context aggregation module, consisting of a fully convolutional network whose convolutional layers have increasing dilation factors. We consider both the ”person” and ”rider” classes in Cityscapes dataset as pedestrians, and the remaining classes as background.

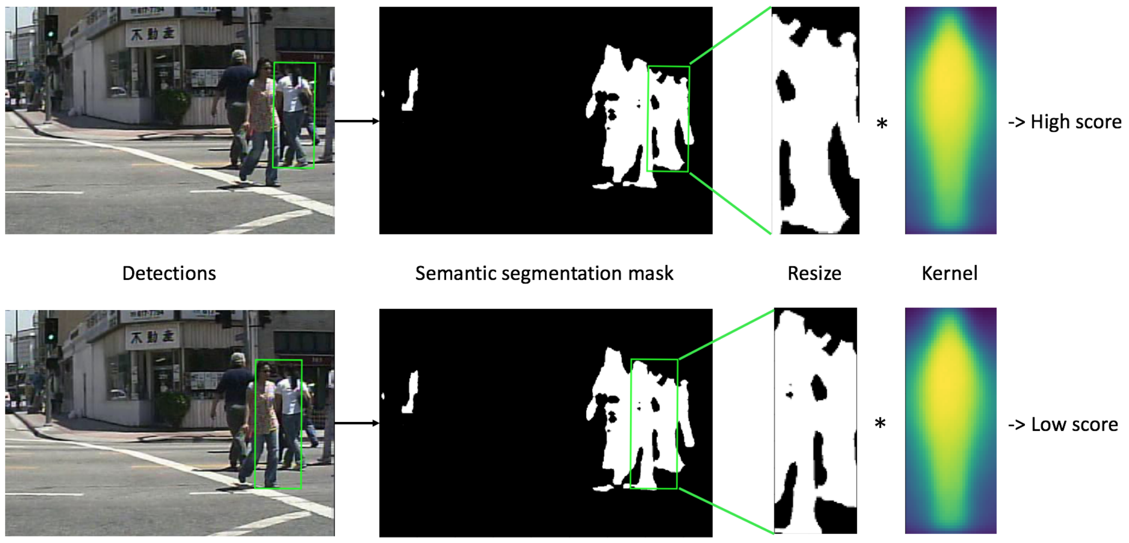

Figure 3.10: The idea of kernel-based method to fuse the semantic segmentation system and the detection system.

To fuse the results of the semantic segmentation system and the pedestrian detection system, we do it as follows: First, we process the same original input image using the semantic segmentation system. This produces a binary mask where the foreground pixel is set to 1 to represent pedestrian class and the background pixel is set to 0. Then, for each of the pedestrian bounding box detected by the detection system, we analysis the pixels at the same locations on the binary mask. A scaler is computed as the weighted sum of the foreground pixels and a weight matrix within the bounding box on the binary mask. We call the weight matrix as a kernel. The kernel is trained as the mean of the semantic segmentation binary masks of all ground-truth pedestrian bounding boxes in the training set and normalized to have sum 1. To perform element-wise multiplication, we rescale all detected bounding boxes to the same size as the kernel. The fusion method is described in Equation (3.16) and (3.17).

$$s_{SS} = \sum_{i,j}^{A_{BB}} mask(i,j) * kernel(i,j) \tag{3.16}$$

$$S_{all} = S_{FDNN} \times s_{SS} \tag{3.17}$$

where the A_{BB} is the area of the bounding box. $mask(i,j)$ and $kernel(i,j)$ are the pixel value of the binary mask and the kernel at location (i, j). From the visualization of the weight mask, we see that the pixels at the center of the kernel tend to have higher values than the pixels at the boundary. This agrees the fact in a perfect detection, the person tends to appear at the center of the bounding box. The idea is illustrated in Figure 3.10. We can see that the kernel will boost the score of a detection whose bounding box fits a person and decrease the score of a detection whose bounding box is not well located.

3.2.4 Experiments and result analysis

3.2.4.1 Training settings

The proposed method is trained on the training sets of the Caltech Pedestrian dataset, the ETH dataset, and the TudBrussels dataset.

To train the pedestrian candidate generator, both the original images and the horizontally flipped images which contain at least one annotated bounding box are used, which grives us around 68,000 training images in total. Among all the annotated bounding boxes, there are about 109,000 annotated bounding boxes in 'Person_full' class, 60,000 annotated bounding boxes in 'Person_occluded' class, and 35,000 bounding boxes in 'People' class. All the images are in size 480×640. The model is fine-tuned from the Microsoft COCO [73] pre-trained SSD model for 40k iterations using the standard stochastic gradient descent (SGD) algorithm and the back-propagation algorithm at a learning rate of 10^{-5}.

To train the classification system, all the ground-truth annotations and the pedestrian candidates generated from the previous stage with height greater than 40 in pixel and confidence score larger than 0.01 are selected and rescaled into size 250×250 as the training samples. For data augmentation, a 224×224 patch is randomly cropped out of each training sample and horizontally flipped with probability 0.5. To label to training samples, a combination of the soft-label method and the hard-label method as described in Equation (9) and Equation (10) is implemented. The thresholds th_a and th_b are set to 0.4 and 0.6, respectively. To build the classification networks, one ResNet-50 [74] and one GoogleNet [75] are used as the classification networks. Both of the classifiers are fine-tuned from the ImageNet pre-trained models using the standard SGD algorithm and the back-propagation algorithm at a learning rate of 10^{-4}.

To incorporate the semantic segmentation network, the dilated convolution model [64] trained on the Cityscapes dataset is directly implemented. All the classes are disabled as background except the ’Person’ and ’Rider’. Since a lack of well-labeled pedestrian dataset for our problem, no fine-tune is involved in this step. All the input images are rescaled from 480×640 into 1024×2048. To preserve the aspect ratio so as to preserve the human body shape, the image’s height is firstly scaled to 1024 and then blank patches are padded on both left and right sides.

All the above mentioned models are built with Caffe deep learning framework [76].

3.2.4.2 Evaluation settings and results

We evaluate the proposed method on four most popular pedestrian detection datasets: the Caltech Pedestrian dataset, the INRIA dataset, the ETH dataset, and the KITTI dataset. The log-average miss rate (L-AMR) is used as the performance evaluation metric [14]. L-AMR is computed evenly spaced in log-space in the range 10^{-2} to 10^{0} by averaging miss rate at the rate of nine false positives per image (FPPI) [14]. There are multiple evaluation settings defined based on the height and visible part of the bounding boxes. The most popular settings are listed in Table 3.6. Descriptions of each dataset and the evaluation results are given blow. We refer our two models as F-DNN2, which is the proposed pedestrian detection system, and F-DNN2+SS, which is F-DNN2 system plus the semantic segmentation system.

Setting	Description
Reasonable	50+ pixels. Occ. none or partial
All	20+ pixels. Occ. none, partial, or heavy
Far	30- pixels
Medium	30-80 pixels
Near	80+ pixels
Occ. none	0% occluded
Occ. partial	1-35% occluded
Occ. heavy	35-80% occluded

Table 3.6: Evaluation settings for Caltech Pedestrian dataset.

Method	Reasonable	All	Far	Medium	Near	Occ. none	Occ. partial	Occ. heavy
SCF+AlexNet [19]	23.32%	70.33%	100%	62.34%	10.16%	19.99%	48.47%	74.65%
SAF R-CNN [25]	9.68%	62.6%	100%	51.8%	**0%**	7.7%	24.8%	64.3%
MS-CNN [68]	9.95%	60.95%	97.23%	49.13%	2.60%	8.15%	19.24%	59.94%
DeepParts [69]	11.89%	64.78%	100%	56.42%	4.78%	10.64%	19.93%	60.42%
CompACT-Deep [70]	11.75%	64.44%	100%	53.23%	3.99%	9.63%	25.14%	65.78%
RPN+BF [22]	9.58%	64.66%	100%	53.93%	2.26%	7.68%	24.23%	69.91%
F-DNN+SS [80]	8.18%	50.29%	77.47%	**33.15%**	2.82%	6.74%	**15.11%**	53.76%
F-DNN2 (ours)	8.12%	51.86%	77.99%	36.72%	1.68%	6.75%	17.51%	40.84%
F-DNN2+SS (ours)	**7.67%**	**49.80%**	**75.83%**	35.09%	1.51%	**6.35%**	16.17%	**39.84%**

Table 3.7: Detailed breakdown performance comparisons of our models and other state-of-the-art models on the 8 evaluation settings. All numbers are reported in L-AMR.

Evaluation on the Caltech Pedestrian data: The Caltech Pedestrian dataset contains 11 sets (S0-S10), where each set consists of 6 to 13 one-minute long videos collected from a vehicle driving through an urban environment. There are about 250,000 frames with about 350,000 annotated BBs and 2300 unique pedestrians. Each bounding box is assigned with one of the three labels: 'Person', 'People' (large group of individuals), and 'Person?' (unclear identifications). The detailed breakdown performances of our two models (detection system only and detection system plus semantic segmentation system) on this dataset is shown in Table 3.7. We compare with all the state-of-the-art methods reported on Caltech Pedestrian website. We can see that both of our models significantly outperform others on almost all evaluation settings. On the 'Reasonable' setting, our best model achieves 8.18% L-AMR, which has a 14.6% relative improvement from the previous best result 9.58% by RPN+BF. On the 'All' evaluation setting, we achieve 50.29%, a relative improvement of 17.5% from 60.95% by MS-CNN [68]. The L-AMR VS. FPPI

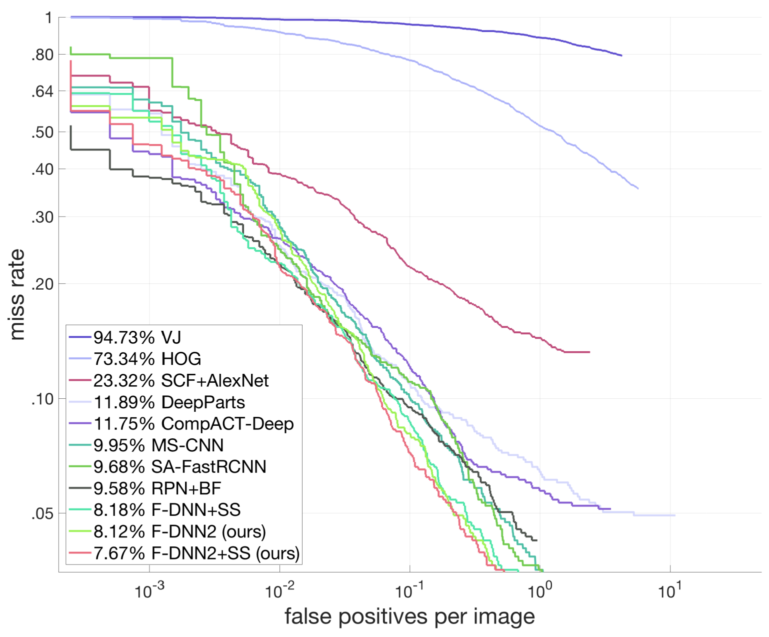

Figure 3.11: L-AMR VS. FPPI plot on ’Reasonable’ evaluation setting on Caltech Pedestrian dataset.

plots for the ’Reasonable’ and ’All’ evaluation settings are shown in Figure 3.11 and Figure 3.12. Except the VJ [77] and the HOG [16] methods, which are plotted as the baselines, all the other results are CNN-based methods.

Evaluation on INRIA: We evaluate the proposed method using the converted INRIA pedestrian dataset provided by Caltech Pedestrian group. There are 614 full positive training images and 288 full positive testing images in the INRIA dataset. At least one pedestrian is annotated in each image. To test the generalization capability of our model, we directly test our Caltech-pretrained model on the INRIA test set without any fine-tuning on the INRIA training set. On the ’Reasonable’ setting, our method achieves 6.78% L-AMR, outperforming the previous best result 6.9% by RPN+BF. Table 3.8 shows

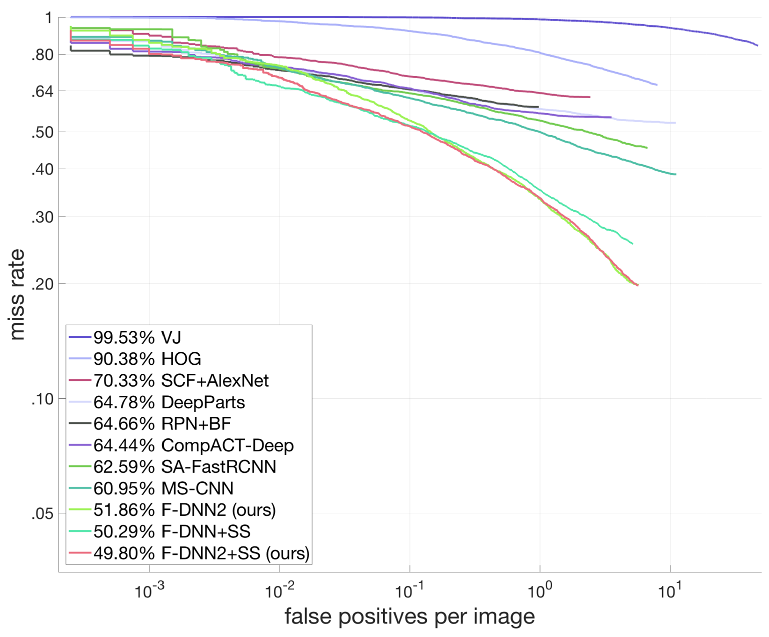

Figure 3.12: L-AMR VS. FPPI plot on 'All' evaluation setting Caltech Pedestrian dataset.

the best results reported on INRIA dataset and Figure 3.13 shows the L-AMR VS. FPPI plot. Results from VJ and HOG are plotted as the baselines.

Method	RPN+BF	SketchTokens	SpatialPooling	RandForest	VJ	HOG	F-DNN2+SS
L-AMR	6.88%	13.32%	11.22%	15.37%	72.48%	63.49%	**6.78%**

Table 3.8: Performance comparisons of our models and other state-of-the-art models on the INRIA dataset.

Evaluation on ETH: There are 1804 images from three video sequences in the ETH pedestrian dataset. As we used the ETH images to train our model, int order to test on the ETH dataset, we removed all the training images from the ETH dataset in our training set and retrained our model. On the 'Reasonable' setting, our method achieves 30.02% L-AMR, outperforming the previous best result 30.23% by RPN+BF. Table 3.9

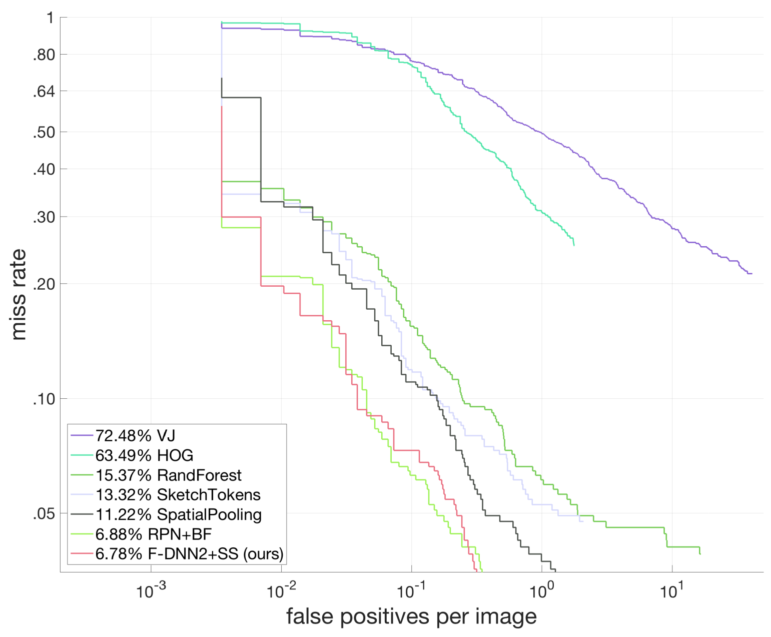

Figure 3.13: L-AMR VS. FPPI plot on 'Reasonable' evaluation setting on INRIA dataset.

shows the best results reported on the ETH dataset and Figure 3.14 shows the L-AMR VS. FPPI plot. Results from VJ and HOG are plotted as the baselines.

Method	RPN+BF	TA-CNN	SpatialPooling	RandForest	VJ	HOG	F-DNN2+SS
L-AMR	30.32%	34.98%	37.37%	45.04%	74.69%	89.89%	**30.02%**

Table 3.9: Performance comparisons of our models and other state-of-the-art models on the ETH dataset.

Evaluation on KITTI: We further generalize our method to multi-class detection problem and test on KITTI object detection dataset. KITTI object detection dataset contains 7481 training images and 7518 test images. All the annotations are split into 7 classes such as cars, vans, trucks, pedestrians, cyclists, trams, and 'Don't care'. Only

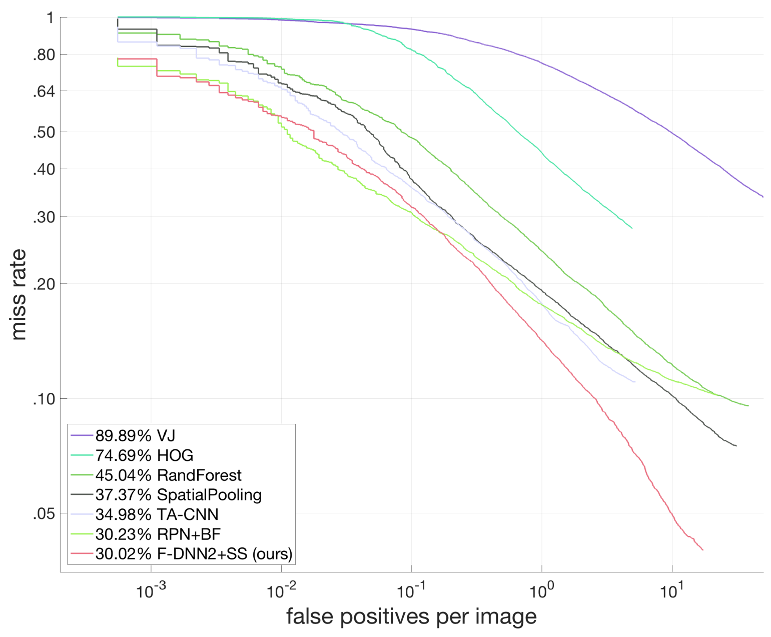

Figure 3.14: L-AMR VS. FPPI plot on 'Reasonable' evaluation setting on ETH dataset.

cars, pedestrians, and cyclists are evaluated. There are three evaluation settings as shown in Table 3.10. Following [68], we split the training data into training set and validation set. We fine-tune three models using all training annotations for the three evaluating classes respectively. For the three models, we set the main aspect ratio to the mean aspect ratio of each class, which is 0.4 for pedestrians, 0.7 for cyclists, and 1.6 for cars. Table 3.11 shows the results on KITTI object detection dataset. We achieve comparable results on all classes. Since the Caltech Pedestrian dataset doesn't distinguish between pedestrians and cyclists, while the KITTI object detection dataset does, it degrades our performance on the pedestrians class and the cyclists class on KITTI.

Setting	Description
Easy	Min. BB height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15%
Moderate	Min. BB height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30%
Hard	Min. BB height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50%

Table 3.10: Evaluation settings for KITTI object detection dataset.

Benchmark	Easy	Moderate	Hard
Car	89.68 %	85.11 %	70.35 %
Pedestrian	74.05 %	61.17 %	57.15 %
Cyclist	67.06 %	51.85 %	46.67 %

Table 3.11: Evaluation results on KITTI object detection dataset.

3.2.4.3 Result analysis

Effectiveness of the network fusion technique   In this subsection, we analysis the performance increases step by step from the candidate generator (CG) to the whole system. The L-AMR is 11.52% by using the candidate generator alone, due to the large number of false positives. By fusing the candidate generator with GoogLeNet, we can improve the performance to 9.41%. By fusing the candidate generator with ResNet-50, we can improve the L-AMR to 8.5%. Furthermore, by fusing the candidate generator with GoogLeNet and ResNet-50 using our proposed fusion net, we can achieve the lowest L-AMR so far at 8.12%. Finally, by fusing the semantic segmentation network into our system, we can achieve the best performance at 7.67%. The analysis shows the capability of the network fusion framework and the advantages of using the idea of ensemble learning. The performance increases of the reasonable setting on the Caltech Pedestrian dataset is given in Table 3.12.

Method	Reasonable
CG	13.06%
CG+GglNet	9.41%
CG+ResNet	8.97%
CG+GglNet+ResNet	8.65%
CG+GglNet+ResNet+Fusion net (F-DNN2)	8.12%
CG+GglNet+ResNet+Fusion net+SS (F-DNN2+SS)	7.67%

Table 3.12: Effectiveness of the network fusion technique.

GoogLeNet VS. ResNet-50 We explore how each part of the classification system contributes to our final results. The breakdown performance comparisons of all evaluation settings on the Caltech Pedestrian dataset between fusing with GoogLeNet alone and fusing with ResNet-50 alone are given in Table 3.13. From the results we can see that GoogLeNet works better in partial/heavy occluded pedestrians while ResNet-50 works better in non-occluded pedestrians. By analyzing the weights learnt in Equation (3.15), we see that the weight for GoogLeNet is 1.11 and the weight for ResNet-50 is 2.22%, which means that our fusion network values the ResNet-50 more than the GoogLeNet. This is reasonable since there are more non-occluded pedestrians in the training data.

Method	Reasonable	All	Far	Medium	Near	Occ. none	Occ. partial	Occ. heavy
CA+GglNet	8.64%	51.59%	76.87%	37.75%	1.72%	7.18%	18.05%	41.19%
CA+ResNet	8.38%	49.58%	74.60%	34.88%	1.70%	6.94%	19.26%	42.71%

Table 3.13: .

Soft-label method versus hard-label method To test how effectively the soft-label method improves the performance, we compare with the conventional hard-label method on the ETH dataset. Since we use the overlap ratio between the candidate bounding box and the ground-truth annotation to assign labels, the soft-label method gives us not only the information of the existence of a pedestrian in each candidate's bounding box, but also how much of the bounding box belongs to the pedestrian. This feature benefits even more in cases where the overlap ratio is around 0.5: e.g. it is too risky to directly assign a hard label 1 or 0 to a bounding box with overlap ratio 0.49 or 0.51. We give the performance comparisons between the hard-label method and the soft-label method in Table 3.14.

Method	Reasonable
CG + ResNet + hard-label	33%
CG + GglNet + hard-label	32.82%
CG + ResNet + soft-label	30.8%
CG + GglNet + soft-label	30.42%
CG + GglNet + ResNet + soft-label	30.02%

Table 3.14: Effectiveness of the soft-label method compared to the conventional hard-label method on ETH dataset.

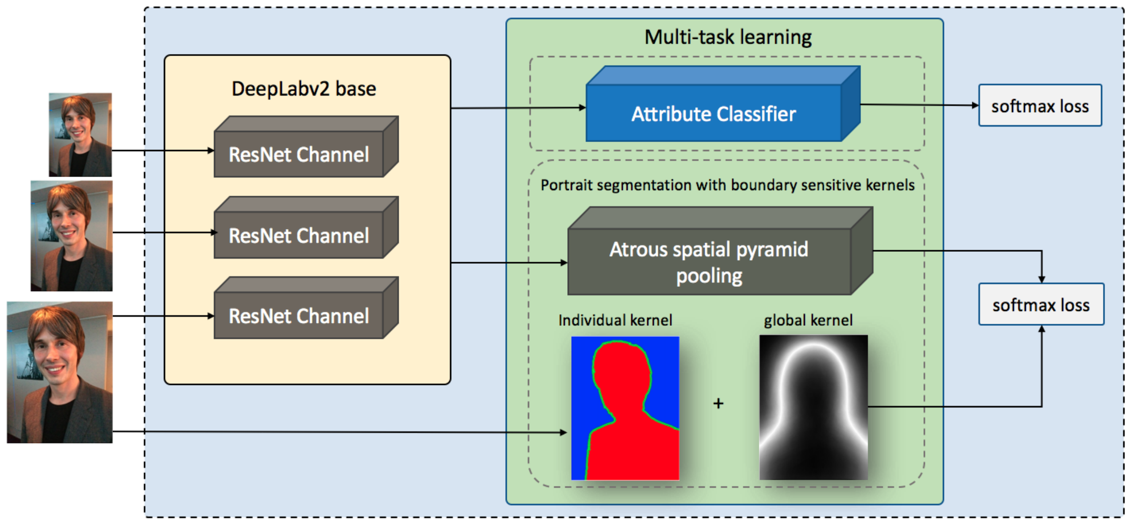

Figure 3.15: The whole architecture of our framework.

3.3 Boundary-sensitive Network for Portrait Segmentation

3.3.1 Overview

In this work, we propose a new boundary-sensitive network (BSN) for more accurate portrait segmentation. In contrast to conventional semantic image segmentation systems, we dilate the contour line of the portrait foreground and label the boundary pixels as the third class with the proposed soft-label method. Two boundary-sensitive kernels are introduced into the loss function to help the network learn better representations for the boundary class as well as govern an overall shape of the portrait. The first boundary-sensitive kernel is designed for each training image such that a floating point vector is assigned as a soft label for each pixel in the boundary class. The second boundary-sensitive kernel is a global kernel where each location in the kernel indicates the probability of the current location belonging to the boundary class. Furthermore, a

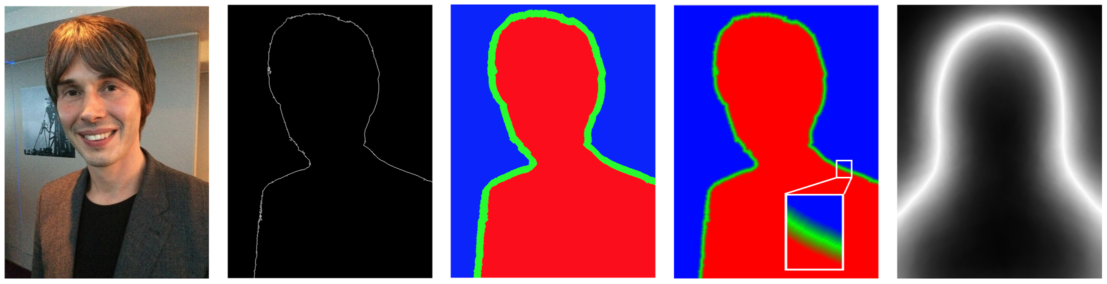

Figure 3.16: The kernel generating process in our method: (a) represents the original image; (b) represents the detected contour line; (c) shows the three class labels: foreground, background, and dilated boundary; (d) shows the individual boundary-sensitive kernel; (e) shows the global boundary-sensitive kernel.

boundary-sensitive attribute classifier is trained jointly with the segmentation network to reinforce the training process. We evaluate our method on PFCN [81], the largest available portrait segmentation dataset. Our method achieves the best quantitative performance in mean IoU at 96.7%. In order to show the effectiveness and generalization capability of our method, we further test on the portrait images collected from COCO [82], COCO-Stuff [83], PASCAL VOC [84] and the experiment results demonstrate that our method significantly outperforms all other state-of-the-art methods.

3.3.2 Related work

Semantic segmentation systems can be categorized as unsupervised methods and supervised methods. Unsupervised methods solve the semantic segmentation problem with classic machine learning techniques include thresholding, K-means clustering, graph-cut [85], etc. On the other hand, conventional supervised methods treat the semantic segmentation problem as a pixel-wise classification problem which first build hand-crafted features and then train classifiers such as Support Vector Machines [17], Random Forest [86], etc.

In recent years, convolutional neural network (CNN) based methods have been successfully applied to semantic segmentation. In 2014, Long et al. [87] introduced the end-to-end Fully Convolutional Networks (FCN) which takes a natural image as input and performs dense pixel-wise predictions to generate a segmentation map of the same size as the input image. Fully connected layers are removed from this network to preserve the spatial information and deconvolutional layers are proposed for up-sampling to recover the full image size. This paradigm popularized the CNN based method and was quickly adopted by subsequent approaches. In traditional CNN architectures, pooling layer was introduced to increase the receptive field as the network goes deeper. However, it also decreases the resolution of feature map. Yu et al. [88] proposed the dilated convolutional layer to replace the pooling layer, which allows for increasing the size of the receptive field without losing resolution in feature maps. Chen et al. [89] proposed the DeepLab system which passes multiple rescaled input images to different network branches in parallel and combines the features maps with max operation at the end.

Portrait segmentation is generally regarded as a sub-problem of semantic segmentation, and it is different from traditional segmentation in two aspects. First, the foreground object is limited to only people which provides additional prior information. Meanwhile, portrait segmentation has higher precision requirement on boundary area. Shen et al. [81] fine-tuned a portrait segmentation system from a pre-trained FCN network with portrait images. To provide more portrait-specific information to the network, two normalized x and y position channels and one mean mask shape channel are added to the input image. Shen et al. [90] proposed a joint correspondence and segmentation estimation method by using extra information provided by dual-lens camera.

While most methods can easily generate a rough segmentation, they generally fail to provide precise segmentation near the object boundaries. For refining the predictions near the boundaries, the most commonly used solution is employing Conditional Random Fields (CRF) along with CNN. Deeplab [89] employs dense CRF after CNN as a post processing method to smooth out the predictions. However, CRF is generally used as a post-precessing step and may be quite time-consuming.

3.3.3 Boundary-sensitive portrait segmentation

The architecture of our framework is shown in Figure 3.15. We use DeepLabv2_ResNet101 model as the base segmentation network. DeepLabv2_ResNet101 consists of three ResNet101 branches at the base which process different scales of the input image. Then the three branches are followed by the atrous spatial pyramid pooling (ASPP) at different dilation rates and fused together at the end. For more details please refer to [89]. To make the model more sensitive to a portrait's boundary, during training, we label the training samples with three non-overlapping classes: foreground, boundary, and background, using the soft-label method described below. One individual boundary-sensitive kernel and one global boundary-sensitive kernel are introduced when updating the loss function, which affect both the forward pass and the back-propagation. The generation process of the two kernels are shown in Figure 3.16. Furthermore, an attribute classifier which shares the base layers with BSN is trained jointly with the segmentation task to reinforce the training process.

3.3.3.1 The individual boundary-sensitive kernel and the soft-label method

To better address the boundary prediction problem, we introduce the individual boundary-sensitive kernel. We label the boundary class as a third class to separate from foreground and background classes and assign soft-labels to pixels in the boundary class as follows. First, the portrait's contour line is identified in the ground truth segmentation map with the Canny edge detector [91]. The contour is then dilated to be P-pixels in width and that map is overlayed onto the ground truth segmentation map. We call the new label map the individual boundary-sensitive kernel. For each pixel in the kernel, a 1×3 floating-point vector $K^{indv} = [l^{fg}, l^{bdry}, l^{bg}]$ is assigned as the soft-label to represent how likely the current pixel belongs to each class. The K^{indv} is computed as Equations (3.18) (3.19) (3.20).

$$l_i^{bdry} = \begin{cases} \dfrac{\min\limits_{\forall C_j \in C} ||I_i - C_j||}{\sum\limits_k \min\limits_{\forall C_j \in C} ||I_k - C_j||} & \text{, if } i \in \text{boundary} \\ 0 & \text{, if } i \in \text{foreground} \\ 0 & \text{, if } i \in \text{background} \end{cases} \tag{3.18}$$

$$l_i^{fg} = \begin{cases} \mathbb{1}(M_i \in fg)(1 - l_i^{bdry}) & \text{, if } i \in \text{boundary} \\ 1 & \text{, if } i \in \text{foreground} \\ 0 & \text{, if } i \in \text{background} \end{cases} \tag{3.19}$$

$$l_i^{bg} = \begin{cases} \mathbb{1}(M_i \in bg)(1 - l_i^{bdry}) & \text{, if } i \in \text{boundary} \\ 0 & \text{, if } i \in \text{foreground} \\ 1 & \text{, if } i \in \text{background} \end{cases} \tag{3.20}$$

where $\min_{\forall C_j \in C} ||I_i - C_j||$ represents the distance from the current pixel I_i to the nearest point on the contour line C. M_i represents the binary label of the current pixel in the original label map M. We can see that pixels in the foreground/background class are labeled as $[1, 0, 0]$/$[0, 0, 1]$ and pixels in the boundary class are labeled with a floating-point vector. The soft-label method computes l^{bdry} as the normalized distance from the current pixel to the nearest point on the contour and sets l^{fg} and l^{bg} to either $(1 - l^{bdry})$ or 0 based on the class label of the current pixel in the ground truth segmentation map. During the forward pass for each pixel in one sample, the new formula for updating the loss function can be expressed as Equation (3.21):

$$\epsilon = -\sum_{j=1}^{c} K_j^{indv} \times log(\frac{e^{z_j}}{\sum_k e^{z_k}}) = -\sum_{j=1}^{c} K_j^{indv} \times log(y_j) \tag{3.21}$$

where l_j denotes the soft-label for class j and $y_j = e^{z_j} / \sum_k e^{z_k}$ denotes the softmax probability for this class. c represents all the three classes. The new back-propagation for this sample can be derived as in Equation (3.22):

$$\begin{aligned}
\frac{\partial \epsilon}{\partial z_i} &= -\sum_{j=1}^{c} K_j^{indv} \times \frac{\partial log(y_j)}{\partial z_i} \\
&= -(\frac{K_i^{indv}}{y_i} \times \frac{\partial y_i}{\partial z_i} + \sum_{j \neq i}^{c} \times \frac{K_j^{indv}}{y_j} \frac{\partial y_j}{\partial z_i}) \\
&= -(\frac{K_i^{indv}}{y_i} \times y_i \times (1 - y_i) - \sum_{j \neq i}^{c} \frac{K_j^{indv}}{y_j} \times (y_j \times y_i)) \\
&= -(K_i^{indv} - y_i \times \sum_{j=1}^{c} l_j) = -(K_i^{indv} - y_i)
\end{aligned} \tag{3.22}$$

The last step holds since the soft-label vector sums to one.

By using the soft-label method, we can see that boundary pixels contribute not only to the boundary class but also to the foreground/background class in a weighted manner based on how close it is to the contour line.

3.3.3.2 The global boundary-sensitive kernel

By the nature of aligned portrait images, it is likely that some locations in the image, such as the upper corner pixels, should belong to the background with very high probabilities while some other locations, such as the middle bottom pixels, should belong to the foreground with high probabilities. These pixels should be more easily classified, while pixels in between should be harder to classify. We estimate a position sensitive prior from the training data.

We design a global boundary-sensitive kernel to guide the network to learn a better shape prediction specifically for portrait images. The global kernel is designed as follows. First, a mean mask $\overline{M}$ is computed using the average of all ground truth segmentation maps from the training samples. This generates a probability map where the value at each location indicates how likely the current location belongs to foreground/background. Second, Equation (3.23) is employed to generate the global boundary-sensitive kernel. All the values are normalized to range $[a, b]$. A larger value close to b in the global kernel indicates that the current location has a higher probability to be boundary. In other words, this location should be more difficult for the network to classify. To force the network to focus more on the possible boundary locations, we weight the locations with their corresponding kernel values when updating the loss function. When performing the forward pass for one pixel location in one sample, we update the loss function as equations (3.24)

$$K^{global} = b - (1 - \frac{|\overline{M} - 0.5|}{0.5}) \times (b - a) \tag{3.23}$$

$$\epsilon -= K_s^{global} \times \sum_j \mathbb{1}(j = c) \times log(y_j) \tag{3.24}$$

where K_s^{global} denotes the global kernel value at the pixel location s. g denotes the ground truth class label for the current pixel location. During back-propagation, the new gradient is computed as Equation (3.25):

$$\begin{aligned}\frac{\partial \epsilon}{\partial z_i} &= -K_s^{global} \times \sum_j \mathbb{1}(j = g) \times \frac{\partial log(y_j)}{\partial z_i} \\ &= -K_s^{global} \times (\frac{1}{y_i} \times \frac{\partial y_i}{\partial z_i}) \\ &= -K_s^{global} \times (\frac{1}{y_i} \times y_i \times (\mathbb{1}(i = g) - y_i)) \\ &= -K_s^{global} \times (\mathbb{1}(i = g) - y_i)\end{aligned} \tag{3.25}$$

From the new forward pass and back-propagation functions we can see that the pixels that are more likely to be located in the boundary (e.g., the pixels lying within the brighter region in Figure 3.16 (e)) are weighted higher so that they contribute more to the loss. This guides the network to be more sensitive to the difficult locations.

3.3.3.3 The boundary-sensitive attribute classifier

Portrait attributes such as long/short hair play an important role in determining a portrait's shape. Training a network which is capable of classifying boundary-sensitive attributes will give more prior information to the system, which further makes the system more accurate and efficient on boundary prediction. Motivated by this idea, we train an attribute classifier jointly with the portrait segmentation network for multi-task learning. An example of how the hair style attribute changes the boundary shape is shown in Figure 3.17.

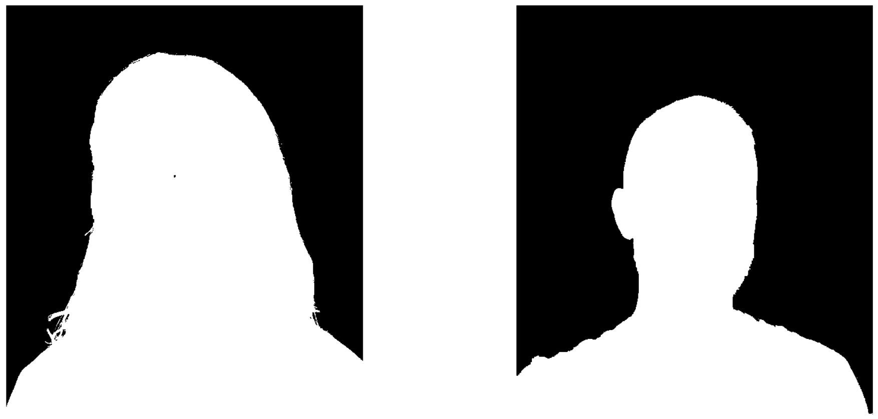

Figure 3.17: An example of how boundary-sensitive attributes affect the portrait's shape: long hair vs. short hair.

To design the attribute classifier, the base layers from "conv1_1" to "pool5" are shared between the segmentation network and the classifier. Above this, for each channel, we add three more fully connected layers. The first two fully connected layers have 1024 neurons and are followed by a dropout layer and a ReLU layer. The last fully connected layer has two neurons for binary classification.

3.3.4 Experiments and results analysis

3.3.4.1 Training settings and evaluation settings

Model details: To train our portrait segmentation system, we fine-tune the DeepLabv2_ResNet101 model using the training set of the PFCN dataset. We will introduce this dataset in the next subsection. There are three ResNet branches in DeepLabv2. In each branch, 4 atrous convolution layers are added in parallel with dilation factors $[6, 12, 18, 24]$ and then summed together to produce the final feature map. Element-wise max operation is performed at the end over the three branches to produce the final prediction. To generate the individual kernel, we dilate the contour line to 10-pixels in width and label the dilated boundary using the soft-label method. We select the weight range in the global kernel as $[0.9, 1]$. Following PortraitFCN+, in addition to the three RGB channels, we add two normalized x and y position channels and one mean mask shape channel into the input. For more details please refer to [81]. At each iteration, a random patch of size 400×400 is cropped out from the original image and randomly flipped with probability 0.5 for data augmentation. Then the input image is rescaled by factors of $[0.5, 0.75, 1.0]$ as the new input images to the three branches of the DeepLabv2 network. To train the attribute classifier, we label the training images into long/short hair classes.

We use Stochastic Gradient Descent (SGD) with a learning rate of $2.5e^{-4}$ to train the model for 20K iterations without the attribute classifier. Then we decrease the learning rate by a factor of 10 and add the attribute classifier to train the model for another 20K iterations. The whole network is built with the Caffe deep learning framework [92].

During testing, we ignore the boundary class and the attribute classifier. Only probabilities from foreground and background classes are used for segmentation.

Mean IoU: The standard mean Intersection-over-Union (IoU) metric is used to evaluate the segmentation performance. The mean IoU is computed as following.

$$\overline{\text{IoU}} = \frac{1}{N} \times \sum_{i}^{N} \frac{A_i^{seg} \cap A_i^{gt}}{A_i^{seg} \cup A_i^{gt}} \tag{3.26}$$

where A_i^{seg} and A_i^{gt} represent the area of the segmentation results and the ground-truth label mask for the i_{th} testing sample, respectively.

3.3.4.2 Results on the PFCN dataset

We evaluate the proposed method on the largest publicly available portrait segmentation dataset [81]. This dataset is collected from Flickr and manually labeled with variations in age, pose, appearance, background, lighting condition, hair style, accessory, etc. Most of the portrait images are captured by the frontal cameras of mobile phones. This dataset consists of 1800 portrait images which are split into 1500 training images and 300 testing images. All the images are scaled and cropped into size 800×600. In one portrait image, the pixels are labeled as either “foreground” or “background”. We will refer to this dataset as PFCN dataset. Some sample images from the PFCN dataset are

Figure 3.18: Sample images from the PFCN dataset.

given in Figure 3.18.

We compare with the state-of-the-art method reported on this dataset: PortraitFCN+ [81] and the DeepLabv2_ResNet101 fine-tuned model , which we will refer to as PortraitDeepLabv2. The PortraitDeepLabv2 model is fine-tuned using the same 6-channel training data as PortraitFCN+ and the same training settings as BSN. For ablation study, we report results of four models from our work: train with the attribute classifier only (BSN_AC), train with the global boundary-sensitive kernel only (BSN_GK), train with the individual boundary-sensitive kernel only (BSN_IK), and the all-in-one model (BSN). Our final model achieves the state-of-the-art mean IoU at 96.7%. The quantitative result comparison is given in Table 3.15. Result from graph-cut [85] is shown as the baseline.

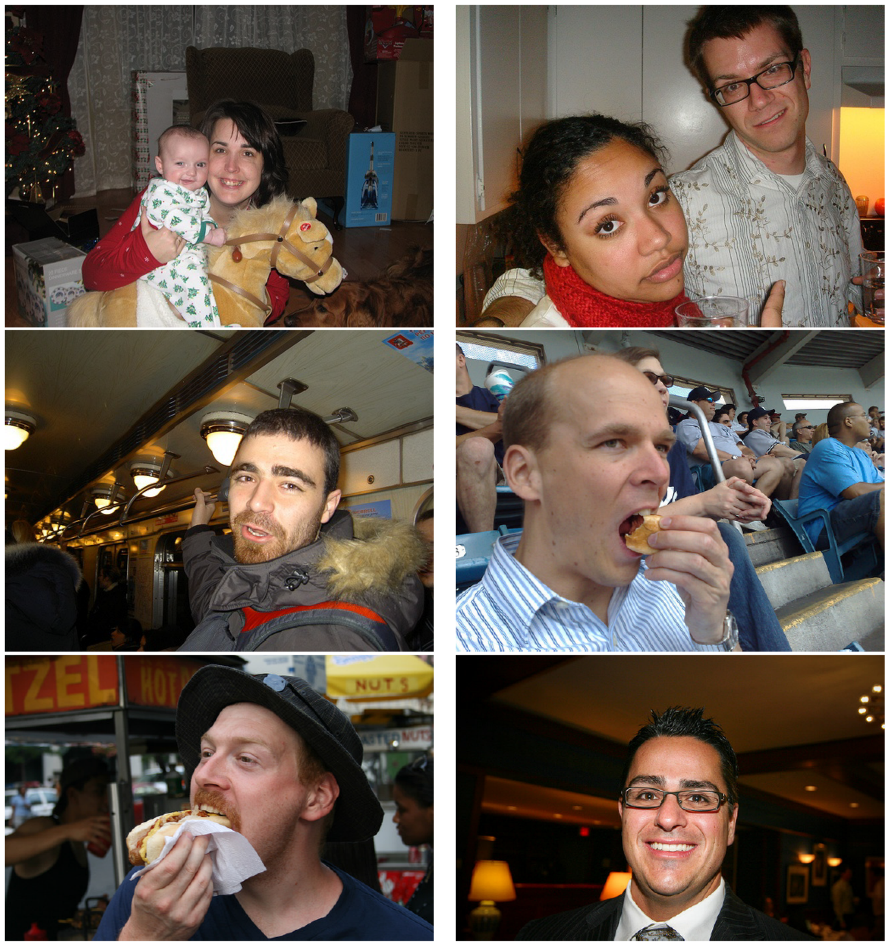

Figure 3.19: Sample images from COCO, COCO-Stuff, and Pascal VOC portrait datasets.

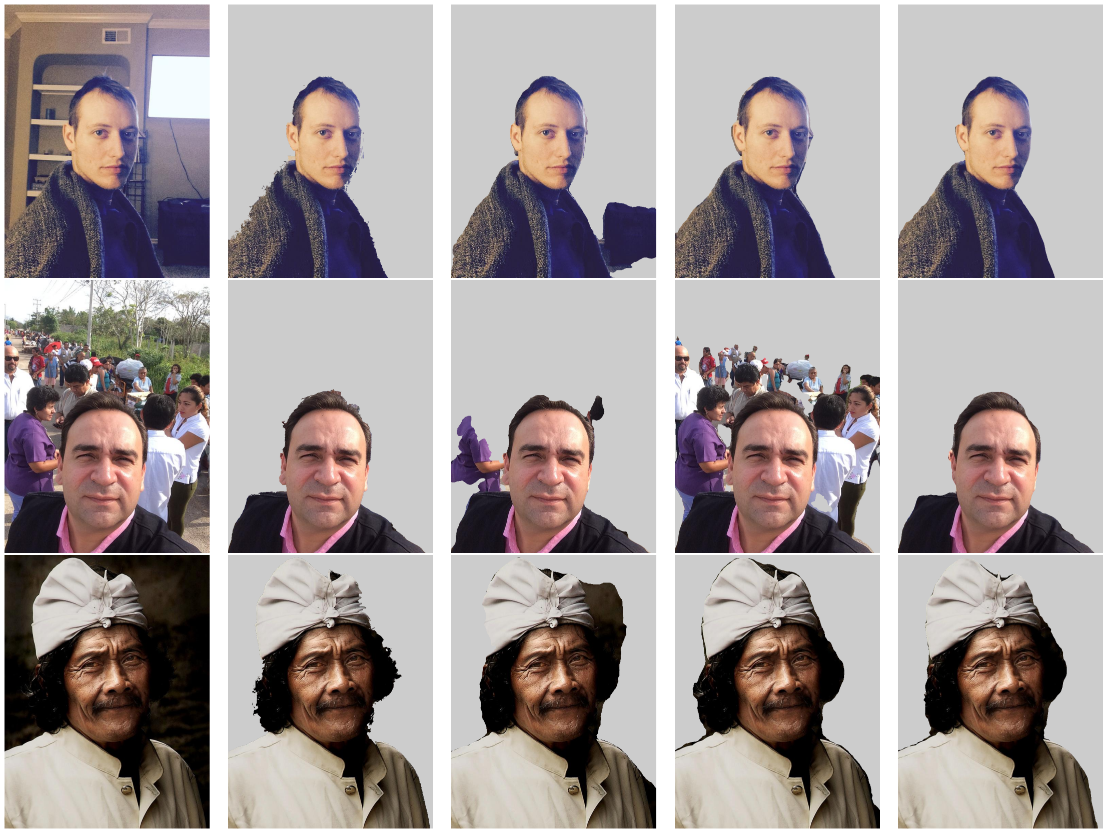

Figure 3.20: Result visualizations of three challenging examples. The first row shows contains confusing objects in the background; the second row includes multiple people in the background; in the third row the background color is close to the foreground.

Method	Mean IoU
Graph-cut	80.0%
PortraitFCN+	95.9%
PortraitDeepLabv2	96.1%
BSN_AC (ours)	96.2%
BSN_GK (ours)	96.2%
BSN_IK (ours)	96.5%
BSN (ours)	**96.7%**

Table 3.15: Quantitative performance comparisons on the PFCN dataset.

3.3.4.3 Evaluation on other datasets:

Since the performance on the PFCN dataset is pretty high and data for the boundary class is unbalanced compare to foreground/background, a good performance on boundary segmentation may only lead to marginal improvement in mean IoU on this dataset. Thus we further test our method on the portrait images collected from three more popular semantic segmentation datasets to evaluate the effectiveness of our boundary-sensitive techniques.

COCO portrait: We automatically collect all the portrait and portrait-like images from the COCO dataset. We run a face detector over the dataset and keep the images only containing one person where the face area covers at least 10% of the whole image. There are 626 images in total with ground truth segmentation maps. We will refer to this dataset as COCO portrait. COCO portrait is more challenging than the PFCN data in various ways such as large pose variations, large occlusions, unlabeled individuals appear on the background, large portion of background, different kinds of accessories, etc.

COCO-Stuff portrait: The COCO-Stuff dataset augments the COCO dataset with refined pixel-level stuff annotations on $10K$ images. We collect 92 portrait and portrait-like images from this dataset. The quality of images in this dataset are same as COCO portrait. We will refer to this dataset as COCO-Stuff portrait.

Pascal VOC portrait: We use the same method to collect portrait and portrait-like images from the Pascal VOC 2007, 2008, and 2012 datasets. Due to the lack of ground truth segmentation maps on this dataset, 62 images are collected. The images in this dataset are also challenging and unconstrained. We will refer to this dataset as PASCAL VOC portrait. Some sample images from the three datasets are illustrated in Figure 3.19 and the statistics are given in Table 3.16.

To test the generalization capability of our model, we directly test on these three datasets without fine-tuning. We achieve 77.7% mean IoU, 72.0% mean IoU, and 75.6% mean IoU on COCO portrait, COCO-stuff portrait, and PASCAL VOC portrait, respectively. We significantly outperform PortraitFCN+ on all the three datasets. The result comparisons are illustrated in Table 3.17. Since the DeepLabv2 model is trained on these dataset, we can not compare with it directly.

Dataset	Num. of Portrait
COCO portrait	626
COCO-Stuff portrait	92
PASCAL VOC portrait	62

Table 3.16: Statistics of the three portrait datasets.

Method	COCO	COCO-Stuff	PASCAL VOC
PortraitFCN+	68.6%	60.8%	59.5%
BSN (ours)	**77.7%**	**72.0%**	**75.6%**

Table 3.17: Quantitative performance comparisons on COCO portrait, COCO-Stuff portrait and Pascal VOC portrait datasets.

3.3.4.4 Result analysis

Visualization on challenging scenarios We visualize the overall performance of our BSN model compared to DeepLabv2 and PortraitFCN+ using three challenging scenarios: confusing objects in the background, multiple people appear in the image, and the background color theme is close to the foreground. Figure 3.20 shows that our model is more accurate and robust than other methods even under challenging conditions.

Accurate boundary segmentation Our method also delivers more precise boundary predictions thanks to its novel boundary-sensitive segmentation techniques. Figure 3.21 shows the comparison of our method with DeepLabv2 and PortraitFCN+ in three challenging scenarios: hair segmentation, accessory segmentation and ear segmentation. Results reveal that while other methods have difficulty in segmenting accessories and small body parts, our method can provide a smooth and accurate segmentation.

Generating trimap for image matting Since our method can deliver an accurate boundary prediction, it is a natural extension to generate trimaps for image matting models. After performing segmentation, we use the same technique during training to dilate the boundary pixels to 10-pixels in width. Several examples are shown in Figure 3.22.

Applications of portrait segmentation Portrait segmentation has been widely used in various image processing applications such as background replacement, depth of field, augmented reality, image cartoonization, etc. We show some applications in Figure 3.23.

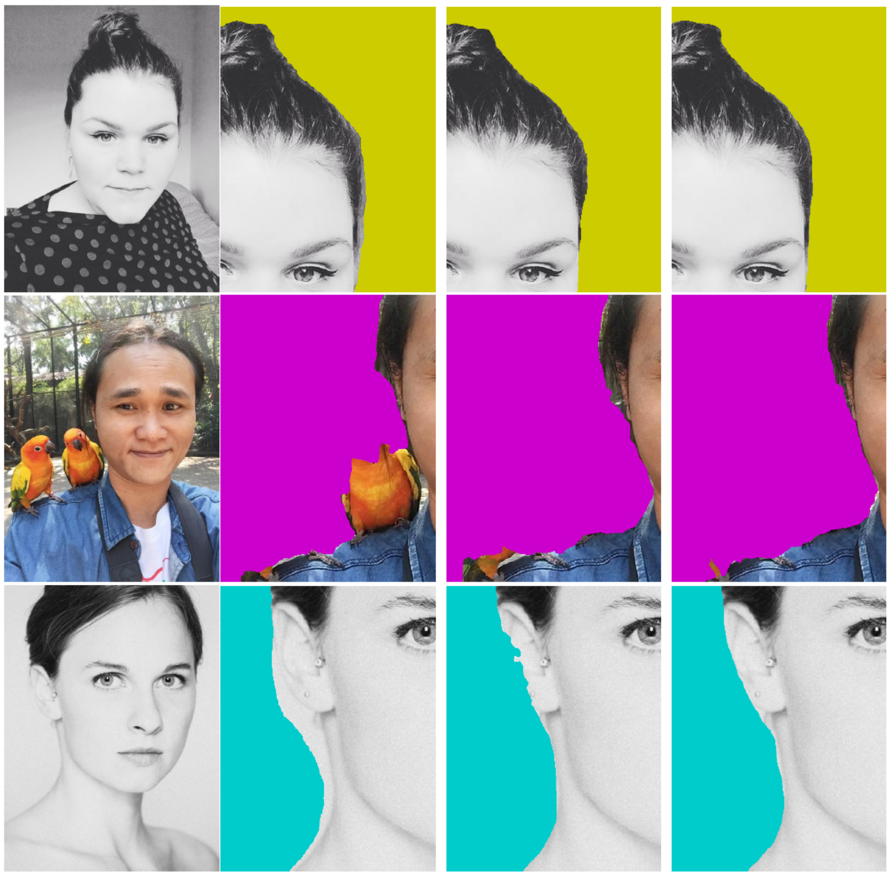

Figure 3.21: Boundary segmentation comparisons. The first column are the original images. The three subsequent columns represent the results from the PortraitFCN+ method, the fine-tuned DeepLabv2 model with the attribute classifier, and our final model (magnified for best viewing).

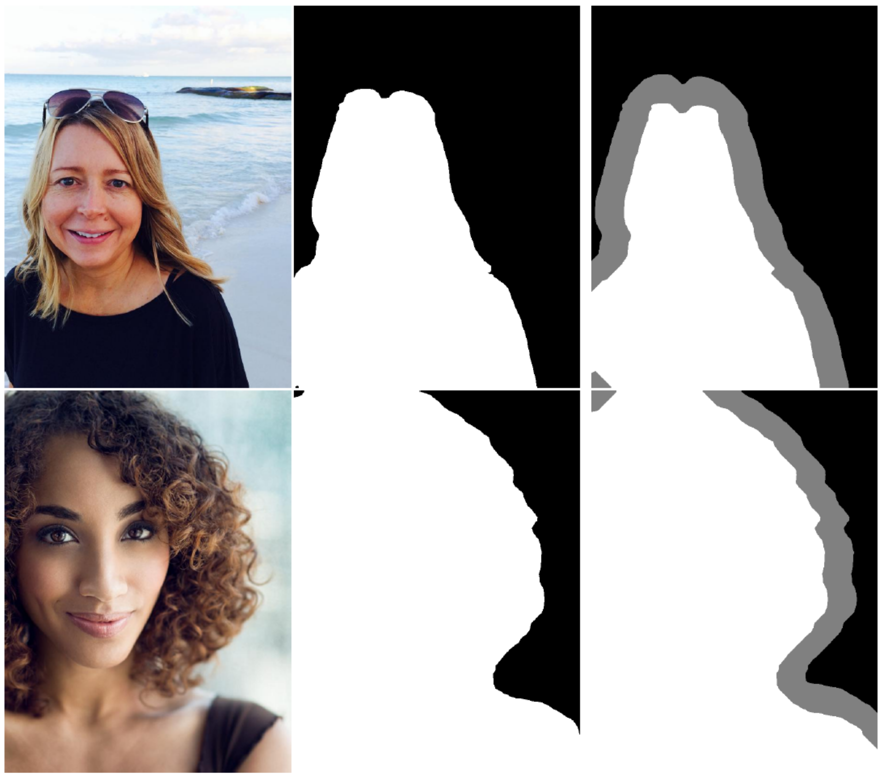

Figure 3.22: Trimaps generated from our segmentation maps.

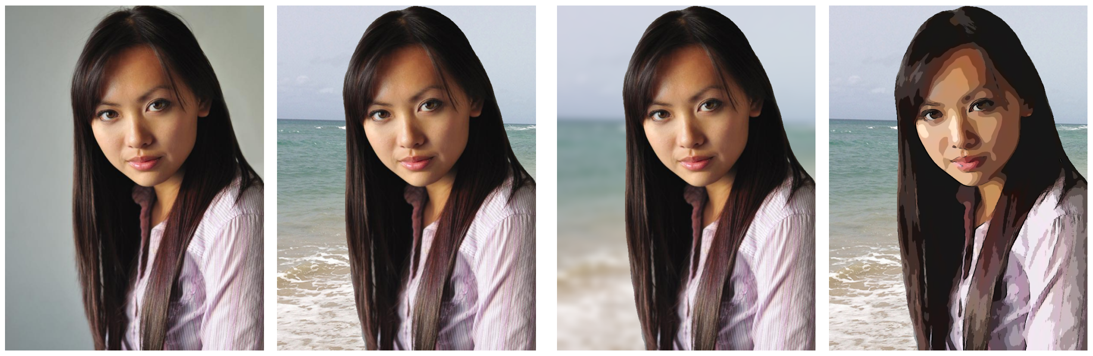

Figure 3.23: Some applications of portrait segmentation.

Chapter 4: Publications

1. **Xianzhi Du**, Xiaolong wang, Dawei Li, Jingwen Zhu, Serafettin Tasci, Larry Davis. ”Boundary-sensitive Network for Portrait Segmentation.” Anonymous conference submission, 2018.

2. **Xianzhi Du**, Mostafa El-Khamy, Jungwon Lee, Larry S. Davis. ”Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection.” IEEE Winter Conf. on Applications of Computer Vision (WACV), March 2017.

3. **Xianzhi Du**, Mostafa El-Khamy, Vlad Morariu, Jungwon Lee, Larry S. Davis. ”Efficient Pedestrian Detection using Deep Neural Network Fusion.” To be submitted to IEEE Transactions on Image Processing.

4. **Xianzhi Du**, David S. Doermann, Wael Abd-Almageed. ”A graphical model approach for matching partial signatures.” IEEE Conf. On Computer Vision and Pattern Recognition (CVPR), June 2015.

5. **Xianzhi Du**, David S. Doermann, Wael Abd-Almageed. ”Signature Matching Using Supervised Topic Models.” Intl. Conf. on Pattern Recognition (ICPR), August 2014.

6. **Xianzhi Du**, Wael Abd-Almageed, David S. Doermann. ”Large-Scale Signature

Matching Using Multi-stage Hashing." Intl. Conf. on Document Analysis and Recognition (ICDAR), 2012.

Bibliography

[1] J. F. Vlez, Snchez, and A. B. Moreno. Robust offline signature verification using compression networks and positional cuttings. *In Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, 1:627636, 2003.

[2] E. J. R. Justino, F. Bortolozzi, and R. Sabourin. Off-line signature verification using hmm for random, simple and skilled forgeriess. *In International Conference on Document Analysis and Recognition*, 1:105–110, 2001.

[3] B. Zhang, M. Fu, and H. Yan. Handwritten signature verification based on neural gas based vector quantization. *In IEEE International Joint Conference on Neural Networks*, page 18621864, 1998.

[4] G. Zhu, Y. Zheng, and D. Doermann. Signature detection and matching for document image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[5] J. F. Vlez, . Snchez, and A. B. Moreno. Robust signature verification using compression networks and positional cuttings. *In Proceedings of IEEE workshop on Neural Networks for Signal Processing*, 1:627–636, 2008.

[6] E. J. R. Justino, F. Bortolozzi, and R. Sabourin. Off-line signature verification using hmm for random, simple and skilled forgeries. *In international conference on Document analysis and Recognition*, 1:105–110, 2008.

[7] B. Zhang, M. Fu, and H. Yan. Handwritten signature verification based on neural gas based vector quantization. *In IEEE international Joint Conference on Neural Networks*, 1998.

[8] X. Du, W. AbdAlmageed, and D. Doermann. Large-scale signature matching using multi-stage hashing. *International conference on document analysis and recognition*, 2013.

[9] S. Srihari, S. Shetty, S. Chen, H. Srinivasan, C. Huang, G. Agam, and O. Frieder. Document image retrieval using signatures as queries. *In Proc. Intl Conf. Document Image Analysis for Libraries*, pages 198–203, 2006.

[10] M. Liwicki, M. Blumenstein, E. Heuvel, C. E.H. Berger, R. D. Stoel, B. Found, X. Chen, and M. Malik. Sigcomp11: Signature verification competition for on- and offline skilled forgeries. *Proc. 11th Int. Conference on Document Analysis and Recognition*, 2011.

[11] M. Liwicki, M. Malik, L. Alewijnse, E. Heuvel, and B. Found. Icfhr2012 competition and automatic forensic signature verification (4nsigcomp 2012). *Proc. 13th Int. Conference on Frontiers in Handwriting Recognition*, 2012.

[12] X. Du, D. Doermann, and W. AbdAlmageed. A graphical model approach for matching partial signatures. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1465–1472, June 2015.

[13] B. Chen, Z. Yang, S. Huang, X. Du, Z. Cui, J. Bhimani, and N. Mi. Cyber-physical system enabled nearby traffic flow modelling for autonomous vehicles. *36th IEEE International Performance Computing and Communications Conference, Special Session on Cyber Physical Systems: Security, Computing, and Performance (IPCCC-CPS). IEEE*, 2017.

[14] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.

[15] Markus Enzweiler and Dariu M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2179–2195, December 2009.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[17] Chris J.C. Burges. A tutorial on support vector machines for pattern recognition. volume 2, pages 121–167, January 1998.

[18] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[19] Jan Hendrik Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele. Taking a deeper look at pedestrians. *CoRR*, abs/1501.05790, 2015.

[20] Rodrigo Benenson, Mohamed Omran, Jan Hendrik Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? *CoRR*, abs/1411.4304, 2014.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[22] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. Is faster R-CNN doing well for pedestrian detection? *To appear in ECCV 2016*, 2016.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[24] Piotr Dollar. Quickly boosting decision trees - pruning underachieving features early. In *ICML*. International Conference on Machine Learning, June 2013.

[25] Jianan Li, Xiaodan Liang, ShengMei Shen, Tingfa Xu, and Shuicheng Yan. Scale-aware fast R-CNN for pedestrian detection. *CoRR*, abs/1510.08160, 2015.

[26] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

[27] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[28] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[29] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters - improve semantic segmentation by global convolutional network. *CoRR*, abs/1703.02719, 2017.

[30] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[31] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612, 2016.

[32] Dawei Li, Xiaolong Wang, and Deguang Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices. *CoRR*, abs/1708.04728, 2017.

[33] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah. Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 64–76, Oct 2016.

[34] Haoyu Ren, Mostafa El-Khamy, and Jungwon Lee. CT-SRCNN: cascade trained and trimmed deep convolutional neural networks for image super resolution. *CoRR*, abs/1711.04048, 2017.

[35] H. Ren, M. El-Khamy, and J. Lee. Image super resolution based on fusing multiple convolution neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1050–1057, July 2017.

[36] X. Wang, Y. Zhou, D. Kong, J. Currey, D. Li, and J. Zhou. Unleash the black magic in age: A multi-task deep neural network approach for cross-age face verification. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 596–603, May 2017.

[37] C. Lin and R. Chellappa. Classification of partial 2d shapes using fourier descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1987.

[38] S. Srihari, S. Shetty, S. Chen, H. Srinivasan, C. Huang, G. Agam, and O. Frieder. Document image retrieval using signatures as queries. *In Proc. Intl Conf. Document Image Analysis for Libraries*, pages 198–203, 2006.

[39] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[40] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. *In Proceedings of the 25th Very Large Data Bases Conference*, 2003.

[41] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *In Symposium on the Theory of Computing*, 1998.

[42] Wei Cai, Cheng Li, and ShiWei Luan. SOI RF switch for wireless sensor network. *CoRR*, abs/1701.01763, 2017.

[43] Wei Cai, Jian Xu, and Liang Huang. Low power SI class E power amplifier and RF switch for health care. *CoRR*, abs/1701.01771, 2017.

[44] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[45] Long Cheng, Avinash Malik, Spyros Kotoulas, Tomas E Ward, and Georgios Theodoropoulos. Efficient parallel dictionary encoding for rdf data. In *in Proc. 17th Int. Workshop on the Web and Databases*, 2014.

[46] F. Gao, W. AbdAlmageed, and M. Hefeeda. Distributed approximate spectral clustering for large-scale datasets. *In ACM International Sym. On High Performance Parallel and Distributed Computing*, 2012.

[47] Yizheng Cao and Kho Pin Verian. A veda simulation on cement paste: using dynamic atomic force microscopy to characterize cellulose nanocrystal distribution. *MRS Communications*, 7, 2017.

[48] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003.

[49] Miao Guo, Abhinav Mishra, Robert L. Buchanan, Jitender P. Dubey, Dolores E. Hill, H. Ray Gamble, Jeffrey L. Jones, Xianzhi Du, and Abani K. Pradhan. Development of dose-response models to predict the relationship for human toxoplasma gondii infection associated with meat consumption. *Risk Analysis*, 36(5):926–938, 2016.

[50] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: fast retina keypoint. *CVPR*, 2012.

[51] D. Blei and J. McAuliffe. Supervised topic models. *NIPS*, 2008.

[52] G. Zhu, Y. Zheng, and D. Doermann. Signature detection and matching for document image retrieval. *IEEE Trans. PAMI*, 31(11):2015–2031, 2009.

[53] X. Du, W. AbdAlmageed, and D. Doermann. Large-scale signature matching using multi-stage hashing. *In ICDAR*, pages 976–980, 2013.

[54] Yizheng Cao, Pablo Zavattieri, Jeffrey Youngblood, Robert Moon, and Jason Weiss. The influence of cellulose nanocrystal additions on the performance of cement paste. *Cement and Concrete Composites*, 56:73–83, 2015.

[55] Yizheng Cao, Pablo Zavattieri, Jeffrey Youngblood, Robert Moon, and Jason Weiss. The relationship between cellulose nanocrystal dispersion and strength. *Construction and Building Materials*, 119:71–79, 2016.

[56] D. Blei and J. McAuliffe. Supervised topic models. *In NIPS*, 2008.

[57] Y. W. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, pages 1566–1581, 2006.

[58] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[59] H. Bay, A. Ess, T. Tuytelaars, and L. Gool. Surf: Speeded up robust features. *In CVIU*, 110(3):346–359, 2008.

[60] C. Wang, D. Blei, and F. Li. Simultaneous image classification and annotation. *In CVPR*, 2009.

[61] C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical dirichlet process. *In AISTATS*, 2011.

[62] X. Du, W. Abdalmageed, and D. Doermann. Large-scale signature matching using multi-stage hashing. In *2013 12th International Conference on Document Analysis and Recognition*, pages 976–980, Aug 2013.

[63] X. Du, D. Doermann, and W. Abd-Almageed. Signature matching using supervised topic models. In *2014 22nd International Conference on Pattern Recognition*, pages 327–332, Aug 2014.

[64] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *To appear in ICLR 2016*, 2016.

[65] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. *arXiv:1512.02325*, 2015.

[66] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015.

[67] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[68] Zhaowei Cai, Quanfu Fan, Rogerio Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016.

[69] Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning strong parts for pedestrian detection. In *ICCV*, 2015.

[70] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*, 2015.

[71] A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press, June 2008.

[72] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009.

[73] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

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

[77] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, May 2004.

[78] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *arXiv:1602.07360*, 2016.

[79] X. Du, M. El-Khamy, J. Lee, and L. Davis. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 953–961, March 2017.

[80] Xianzhi Du, Mostafa El-Khamy, Jungwon Lee, and Larry S. Davis. Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection. *CoRR*, abs/1610.03466, 2016.

[81] X. Shen, A. Hertzmann, J. Jia, , S. Paris, B. Price, E. Shechtman, and I. Sachs. Automatic portrait segmentation for image stylization. *Computer Graphics Forum*, 35(3):93–102, 2016.

[82] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[83] Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. *CoRR*, abs/1612.03716, 2016.

[84] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[85] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[86] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[87] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[88] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[89] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

[90] Xiaoyong Shen, Hongyun Gao, Xin Tao, Chao Zhou, and Jiaya Jia. High-quality correspondence and segmentation estimation for dual-lens smart-phone portraits. *arXiv preprint arXiv:1704.02205*, 2017.

[91] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

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