การส่งค่าย้อนกลับสู่โครงข่ายประสาทเทียมคอนโวลูชั่นของซัพพอร์ทเวกเตอร์แมชชีนเชิงโครงสร้างนำมาใช้กับปัญหาการประมาณท่าทางของมนุษย์

นายพีระจักร์ วิฑูรชาติ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์ดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559

STRUCTURED SVM BACKPROPAGATION TO CONVOLUTIONAL NEURAL NETWORK APPLYING TO HUMAN POSE ESTIMATION

Mr. Peerajak Witoonchart

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2016

Thesis Title	STRUCTURED SVM BACKPROPAGATION TO CONVOLUTIONAL NEURAL NETWORK APPLYING TO HUMAN POSE ESTIMATION
By	Mr. Peerajak Witoonchart
Field of Study	Computer Engineering
Thesis Advisor	Professor Prabhas Chongstitvatana, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

.... Dean of the Faculty of Engineering

(Associate Professor Supot Teachavorasinskun, D.Eng)

THESIS COMMITTEE

.... Chairman

(Professor Boonserm Kijsirikul, Ph.D.)

.... Thesis Advisor

(Professor Prabhas Chongstitvatana, Ph.D.)

.... Examiner

(Nattee Niparnan, Ph.D.)

.... Examiner

(Assistant Professor Thanarat Chalidabhongse, Ph.D.)

.... External Examiner

(Associate Professor Bunyarit Uyyanonvara, Ph.D.)

พีระจักร์ วิฑูรชาติ: การส่งค่าย้อนกลับสู่โครงข่ายประสาทเทียมคอนโวลูชั่นของซัพพอร์ทเวกเตอร์แมชชีนเชิงโครงสร้างนำมาใช้กับปัญหาการประมาณท่าทางของมนุษย์. (STRUCTURED SVM BACKPROPAGATION TO CONVOLUTIONAL NEURAL NETWORK APPLYING TO HUMAN POSE ESTIMATION) อ.ที่ปรึกษาวิทยานิพนธ์ : ศ. ดร. ประภาส จงสถิตย์วัฒนา .

ในงานนี้ เราแสดงให้เห็นเป็นครั้งแรกถึงวิธีการที่จะแปลงสูตรของซัพพอร์ตเวคเตอร์แมชชีนเชิงโครงสร้างให้เป็นโครงข่ายประสาทเทียมคอนโวลูชั่นสองชั้น ชั้นบนของโครงข่ายประสาทเทียมคอนโวลูชั่นเป็นชั้นการอนุมาณความสูญเสียช่วยเหลือเพิ่มเติม และชั้นล่างของโครงข่ายประสาทเทียมคอนโวลูชั่นเป็นชั้นคอนโวลูชั่น เราแสดงให้เห็นว่าโมเดลหลายส่วนที่บิดเบี้ยวได้สามารถถูกเรียนรู้ได้ด้วยโครงข่ายประสาทเทียมเชิงคอนโวลูชั่นซัพพอร์ตเวคเตอร์แมชชีนเชิงโครงสร้างที่เพิ่งประดิษฐ์ขึ้นมาใหม่โดยการส่งค่าย้อนกลับของความผิดพลาดของโมเดลหลายส่วนที่บิดเบี้ยวได้ไปสู่โครงข่ายประสาทเทียมเชิงคอนโวลูชั่น การส่งค่าไปข้างหน้าคำนวนการอนุมาณความสูญเสียช่วยเหลือเพิ่มเติม การส่งค่าย้อนกลับคำนวณความลาดเอียงของการอนุมาณความสูญเสียช่วยเหลือเพิ่มเติมสู่ชั้นคอนโวลูชั่นโดยการกระทำเช่นนั้น เราได้สร้างโครงข่ายประสาทเทียมคอนโวลูชั่นชนิดใหม่:โครงข่ายประสาทเทียมเชิงคอนโวลูชั่นซัพพอร์ตเวคเตอร์แมชชีนเชิงโครงสร้างที่ซึ่งถูกนำไปใช้กับปัญหาการประมาณท่าทางของมนุษย์ การประดิษฐ์ใหม่นี้เป็นโครงข่ายประสาทเทียมที่สามารถนำไปใช้เป็นชั้นสุดท้ายของการเรียนรู้เชิงลึก วิธีของเราเรียนตัวแปรของโมเดลเชิงโครงสร้างและโมเดลรูปลักษณ์ไปพร้อมกัน สูตรการส่งค่ากลับของเรายังสามารถนำไปใช้กับปัญหาการแบ่งแยกหลายชนิดของซัพพอร์ตเวคเตอร์แมชชีนเชิงโครงสร้างที่ซึ่งผลการทดลองของเราเหนือกว่าตัวแบ่งแยกหลายชนิดที่ชื่อว่าซอฟต์แมกซ์ที่ใช้กันอย่างแพร่หลายเมื่อเปรียบเทียบบนฐานข้อมูลมาตราฐาน MNIST. เราเขียนโปรแกรมวิธีของเราในฐานะชั้นชนิดใหม่ของห้องสมุดโปรแกรมชื่อว่าCaffeที่มีอยู่แล้ว

ภาควิชา วิศวกรรมคอมพิวเตอร์ . . ลายมือชื่อนิสิต . . . .
สาขาวิชา วิศวกรรมคอมพิวเตอร์ . . ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์ . . . .
ปีการศึกษา 2559 . . . .

5471446721: MAJOR COMPUTER ENGINEERING

KEYWORDS: STRUCTURED SVM/ CONVOLUTIONAL NEURAL NETWORK / NEURAL NETWORK

PEERAJAK WITOONCHART : STRUCTURED SVM BACKPROPAGATION TO CONVOLUTIONAL NEURAL NETWORK APPLYING TO HUMAN POSE ESTIMATION. ADVISOR : Professor Prabhas Chongstitvatana, Ph.D. .

In this work, we show, for the first time, how to formulate Structured SVM as two layers of Convolutional Neural Network, the top layer of which is loss augmented inference layer, and the bottom is normal convolutional layer. We show that Deformable Part Model can be learned with newly created Structured SVM neural network by propagating the error of Deformable Part Model back propagate to Convolutional Neural Network. The forward propagation calculates loss augmented inference. The back propagation calculates the gradient from the loss augmented inference layer to convolutional layer. By doing so, we create a new type of convolutional neural network: Structured SVM Convolutional Neural Network, which is then applied to Human Pose Estimation problem. This new creation is a neural network, which can be used as the last layers of deep learning. Our method jointly learns structural model parameters and appearance model parameters. Our back propagation formulation can also be applied to Multiclass Classification Structured SVM where our result outperformed widely used Softmax classifier on standard MNIST dataset. We implement our method as a new layer of existing Caffe library.

Department : Computer Engineering Student's Signature ....

Field of Study : Computer Engineering Advisor's Signature ....

Academic Year: 2016 ....

Acknowledgements

Five years of my life has been dedicated to this Ph.D study. It would have been financially impossible to dedicate such an amount of time and resource to this thesis without financial help from my family. I have been very fortunate to be raised by parents who emphasized the importance of education. Now that I myself am a father of three children, I know how hard it is to maintain a good environment and provide resources for the next generation so they can stay focused on what they wish to do. I thank my parents, namely, Wongwai and Gasiniee Witoonchart, for providing such an environment and resources for my Ph.D study. This is a privilege not every family can give.

The next in the line I wish to thank is my wife, Aiping Witoonchart. She given good support to my children. Without her agreement to five years of my life dedicated to this Ph.D study, I would have been unable to dedicate such an amount of time and resources to this thesis. To raise a child, a great amount of labor and attention must be paid. To raise three children, the amount of labor is more than tripled. Yet she has maintained good support and standards so I can dedicate my time to my research.

Finally, I must thank my advisor, Prof. Prabhas Chongstitvatana. He has assisted in answering all of my inquiries. He has been my guide during these years. Through these years, I have had the feeling of swimming on an ocean of methods, parameters, algorithms. It was through my adviser that I could see where the shore lie.

Here in this thesis is the fruit of their contribution. Thank you all. Please enjoy reading this thesis.

I would like to dedicate this thesis to my loving parents, namely Wongwai Witoonchart, and Gasinee Witoonchart …

Thesis is our hope of my loving family, namely Aiping Witoonchart, Kanrawee Witoonchart, Kaonah Witoonchart, and Kanteera Witoonchart …

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Peerajak Witoonchart

Academic Year 2016

Acknowledgements

Five years of my life has been dedicated to this Ph.D study. It would have been financially impossible to dedicate such an amount of time and resource to this thesis without financial help from my family. I have been very fortunate to be raised by parents who emphasized the importance of education. Now that I myself am a father of three children, I know how hard it is to maintain a good environment and provide resources for the next generation so they can stay focused on what they wish to do. I thank my parents, Wongwai and Gasiniee Witoonchart, for providing such an environment and resources for my Ph.D study. This is a privilege not every family can give.

The next in the line I wish to thank is my wife, Aiping Witoonchart. She given good support to my children. Without her agreement to five years of my life dedicated to this Ph.D study, I would have been unable to dedicate such an amount of time and resources to this thesis. To raise a child, a great amount of labor and attention must be paid. To raise three children, the amount of labor is more than tripled. Yet she has maintained good support and standards so I can dedicate my time to my research.

Finally, I must thank my advisor, Prof. Prabhas Chongstitvatana. He has assisted in answering all of my inquiries. He has been my guide during these years. Through these years, I have had the feeling of swimming on an ocean of methods, parameters, algorithms. It was through my adviser that I could see where the shore lie.

Here in this thesis is the fruit of their contribution. Thank you all. Please enjoy reading this thesis.

Abstract

Deep learning and feature learning have been a current trend in finding features for classification , detection, and segmentation. Some examples are: Deep convolutional Neural Network can do facial point detection,Deep Network for pedestrian detection, Pose Estimation with Deep Network, Facial Feature Tracking with Restricted Boltzmann machine, Shape Prior using Deep learning for object segmentation, and Object Detection with Deep Network.

Our approach starts from Ross Girshick's notice that Convolutional Neural Network (CNN) is Deformable Part Model (DPM). However, this did not back propagate the error from DPM model to CNN. Should the DPM be a CNN, they must back propagate the error to the lower layer. To this end, this thesis back propagates the error of DPM back to CNN under Structured SVM Loss function.

The usual method of applying Latent SVM to Deep learning is to extract features with deep neural network and perform Latent SVM learning as two distinct stages on the same pipeline, meaning that one must do feature extraction, then cache the result on the first stage, then submit to Latent SVM learning algorithm as the second stage. For example, extracting pyramid of features from convolutional neural network on the feature extraction stage, caching the extracted features and then learning Latent SVM during the second stage. On second stage, the latent SVM then learns all model parameters by switching between SVM optimization and inference combinatorial optimization. This kind of method has an inherent problem in that they cannot learn the deep learning feature extraction learnable parameters from the error of inference optimization because they are on two different distinct stages. The learnable feature extraction parameters cannot be not updated by the error of Latent SVM. Seeing this shortcoming, we proposed the Structured SVM Convolutional Nueral Network.

In this work, we show, for the first time, how to formulate Structured SVM as two layers of Convolutional Neural Network, the top layer of which is loss augmented inference layer, and the bottom is normal convolutional layer. We show that Deformable Part Model can be learned with newly created Structured SVM neural network by propagating the error of Deformable Part Model back propagate to Convolutional Neural Network. The forward propagation calculates loss augmented inference. The back propagation calculates the gradient from the loss augmented inference layer to convolutional layer. By doing so, we create a new type of convolutional neural network: Structured SVM Convolutional Neural Network, which is then applied to Human Pose Estimation problem. This new creation is a neural network, which can be used as the last layers of deep learning. Our method jointly learns structural model parameters and appearance model parameters. Our back propagation formulation can also be applied to Multiclass Classification Structured SVM where our result outperformed widely used Softmax classifier on standard MNIST dataset. We implement our method as a new layer of existing Caffe library. Our source code is available for download.

Table of contents

List of figures

List of tables

Listings

Chapter 1

Introduction

1.1 Motivation

Since the dawn of the computer era, mankind has dreamed of a robot which can see and understand the world like a human. The story of such a wonder is the subject of many science fiction stories. Some of those stories have inspired other writers. The story of Skynet in "The Terminator," films, in which a computer gains consciousness, sees the world through cameras, and responds to threats, has inspired this author to formulate a Ph.D thesis.

Many decades have passed since then, and our dream of a Skynet-type network is still unrealized. What happened? Why is it so hard to write a program that detects objects around us? Despite decades of research, this is still an open question. Lately, with new research results from the machine learning community, and with advances in computer speed, we are now able to write a program that detects objects like never before. These new methods rely heavily on statistical theory. Data-driven solutions seems to be the core idea upon which these methods rely. In most cases, the more data-driven the algorithm, the better the detection accuracy. These statistical methods, however, need a representation of images in order to work correctly. One must at least normalize each training data point on the same domain. The training data must be engineered in such a way that those which are not distinguished during classification should be removed before being sent to statistical classifiers. Such data preprocessing prior to classification are called feature extraction and the preprocessed data is called feature. These requirements gives rise to “feature” engineering, trying to find a way to pre-process the data such that the intended statistical classifier gives high accuracy. For example, HOG [23], SIFT [67], LBP [71], Haar wavelet[104], Shapelet[81], Shape Context[5], Covariance[102] Features are extracted from training images and used to train classifiers such as SVM[103] and Adaboost[40]. Previous decades saw many Computer Vision researchers trying to find features-classifier pairs which gave the greatest accuracy for image classification. Object detection can be a sliding windows on all possible scales of image classification. For example, to detect a human is to slide a window and then determine if within that window there is a human or not over all possible locations and scales [107]. Engineering such a feature is very difficult task. Many researchers spend a decade or two of their career attempting to engineer features. Their results are impressive, yet improvable. There were many speculations that if hand engineered features were replaced with learned features, which are features obtained by statistical methods to learn the preprocessed data directly from data, one could significantly improve the accuracy. Because the dataset requirement is very large, learning the feature directly from the data is computationally intensive. The accuracy gain from feature learning inspires researchers to learn their features. It is undeniable that the feature learning method requires data engineering, and additional preprocessing of data is still required before feature learning. However, such data engineering is much easier than feature engineering, and additional preprocessing of data is usually very simple. Ideas from pioneering Computer Vision researchers such as Pictorial Structure are then incorporated into this new statistical method. The idea of Pictorial Structure is that objects are formed by object parts. These parts have relative locations which can be modeled as if there are springs among them. Therefore, a static structure model is created for a particular object class. For example, a face must have two eyes, one nose, one mouth, and a hair area. We could outline the image and place springs among these part locations. Once this was done, a static structure model for face could be created. This model is called Pictorial Structure. For a computer program to use this face pictorial structure for face detection, one needs to quantify the Pictorial Structure model and the matching quality of a test image. Together with statistical learning methods, and with engineered features, Pictorial Structure became a powerful tool for object detection since it not only performed object detection, but also a localization of object parts. Using PS with statistical learning gives rise to face detection involving eyes, lips localization, Human Pose-Estimation, and text detection. Since quantifying Pictorial Structure results in an entire structure of hand engineered features refined through statistical learning processes, the method can be said to use hand-designed features over structured learning. The detection is thus called structured inference. How could one improve the Pictorial structure? Currently, there are not many researchers using Feature Learning methods on Pictorial Structure. Since many objects fall under PS framework, they are capable of being detected by this framework using the same technique, with the only change being the dataset. Application of this method is object detection. All objects which fall under PS framework are applicable. Human Pose-Estimation, Face Detection, and Object Detection with deformable object parts are some particular instances of PS framework object detection upon which we are currently focusing. In Human Pose-Estimation, Part articulation is difficult because the same part can be shown from many different angles. Previous research attempted to find an articulation of a mixture of part types. Each part on the dataset was categorized into K clusters with K-mean clustering, thus

K type for that part. Detection is therefore attempting to recover the part position and part type for each and every part. Previous research defined the detection problem as a structural prediction problem under PS framework. However, there results on arms and limbs has not yet reached sufficient accuracy. It should be noted that learned features can handle variation of image quite well. Therefore feature learning methods should be applied to Pose-Estimation Problems as an instance of PS framework.

The author believes that, for computers to understand the world, we need to understand the functioning of the brain, and imitate these function. The author believes that human brains learn to recognize images. Therefore, the author wishes to research deep learning algorithms due to the claims that they imitate brain functions. In this thesis the author has come up with a new layer of deep learning.

The author believes that research is related to the era in which one lives. Human understanding of Artificial Intelligence(AI) has passed both the boom time and the bust time. Within the bust time there is no basic research theories, or no calculation power. Instead of using GPU specifically for image rendering, researchers are now performing general purpose calculation with the General Purpose Graphic Processing Unit (GPGPU) technology. With these developments, the calculation can now power skyrockets. Hence, the author believes that the boom time of AI research is now.

–

1.2 Literature Review on Human Pose Estimation learning algorithms

The Generic Structural Prediction of object parts is similar to the Human Pose-Estimation Problem. Some examples are, labeling and bounding car parts with boxes, labeling and bounding face parts with boxes, and labeling and bounding bus parts with boxes. The Pose Estimation problem on 2D still images is defined as the process of finding human joints or parts on an image that contains one human. This is a difficult problem due to the change in colors of suits and because parts are often partially, if not totally, occluded. Previous state-of-the-art Pose Estimation solutions were based heavily on the success of Pictorial Structure first proposed by [38], which was then further developed in the age of statistical classification by [34]. Pictorial Structure tree inference was made efficient by [32]. The method quickly became the standard for object localization. [77] adopted this method for Human Pose Estimation (HPE), thus making it the standard for HPE. Ramanan's method of solving HPE is to cluster the subparts into a mixture of appearances for each part, then incoperating the co-occurance model and deformable model with a mixture of the appearance model. In the Histogram of Oriented Gradients (HOG) [23], features of each subpart are extracted and the appearance model filters learned the SVM [21] filters for each and every subpart. Pictorial structures were then created and populated the part and subpart filters with these SVM filters. They were then relearned structurally. The result was very successful. However, some problems persisted, for example, double counting, shown in Figure 1.1, where limbs or arms are counted twice on the same limb or arm of the test image. This problem led to a wide variety of improvements. One improvement was to add more training data [51]. Further methods also tried to improve the detection model, such as adding a prior model to [74], incorperateing the geometrical model. Some attempted to use information from the subsequent video frame for loopy inference [17]. Some attempted to impose symmetry betweeen limbs, adding repulsive edges on different arms[50]. [106] used latent tree models where the latent structure was learned from observation without making the assumption of physical constraints. There are other methods in which a puppet is utilized instead of bounding boxes to represent a human[122]. The tests [13] increased the unary potential

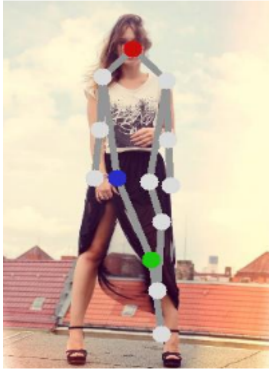


Fig. 1.1 Double Couing problem happens on the limbs where limbs of the stickman are couting on one limb of the test image twice, image from [25]

accuracy by random forest[11] Plausible poses can be modeled with greater fidelity. One way is to find better structure, for example, in [96]. Pictorial Structure tree model is added with spatial hierarchical of hidden nodes, [95] carefully designed leaf node variation and latent node, which control variation on leaf nodes, or [93][88][95] used loopy model to inference. [25] tries to focus on part clustering into multimodal decomposible models. Some of them try to solve multiple instances of a human part with a multimodal Hierarchical model with approximate inference on a single prototype and among local poses [29].[74] use well defined parts which are often encountered in appearance space and configurational space, and use pre-trained Adaboost classifier [40] on SIFT [66] or shape context [5] feature function. [30] adds prior model for head to torso connection in upper body Human Pose Estimation. [17] try to improve PS with better prior model by parameterize geometric variables. However the models are improved, all of these methods must learn structural model parameters. Latent SVM [113] has become the standard for learning these model parameters. Structured SVM [100] can also be used to learn Human Pose Estimation [14]. This research also uses standard Structured SVM as a model learning algorithm. This research differs from [14] in that back propagate Structured Loss Augmented Inference back to neural network is conducted prior.

Apart from using an SVM base to learn the pictorial structure model, Random Forest [11] is used to learn the unary potential filters, joint location prediction, and co-occurance model in [25]. [82] uses Structured SVM on mode-specific submodule for flexible submodule selection.

The morning ring rang since [20] shows near human accuracy in traffic sign recognition and hand written digit bench-marks. Deep learning and feature learning, however, have been recent trends for finding features for classification, object detection[91], and segmentation[119]. The following are examples of such trends: Deep convolutional Neural Network, which can perform facial point detection [90][59],Deep Network for pedestrian detection [84], and Pose Estimation with Deep Network [72], In [72], appearance model, co-occurance model, and deformable model were inserted as three different types of input to neural network, and back propagate the sum of cross-entropy error, square error, and regularization. [97] used spatial dropout and heat map regression on Deep Convolutional Neural Network. [12] fed back gradient error of the approximation of the joint positions to Coonvolutional Neural Network. Key point regression with convolutional neural network can perform both Human Pose Estimation and Action Recognition impressively [44]. These regression based methods differ from our approach in that discriminiative classification with Structured SVM Loss Augmented Inference is used as the last layer. Most similar to our work is the current work of [112], which results in a great Percentage of Correct Parts (PCP) improvement. Their method achieves as high as 81% PCP accuracy. They formulate the problem as end-to-end back propagation of SVM hinge loss back to convolutional neural network. This is analogeous to the work of [105] on the object detection application. Our work differs in that we formulate the problem as Structured SVM (SSVM) [100]. Other similar work is [16], and their extension [15], in which articulated pose estimation was utilized as pictorial structure as can be seen in our work. In addition, Convolutional Neural

Network to learn the appearance weight was utilized as in ours. They used SVM to learn the model structurally, which is a similar method but not the same as ours. The biggest difference is that they did not back propagate the error from Structured SVM, which is the key to our contribution.

Our approach starts from Ross Girshick's notice that Convolutional Neural Network (CNN) [54][55] is Deformable Part Model (DPM) [42]. However, the error was not back propagated from DPM model to CNN. Should the DPM be a CNN, the error must be back propagated to the lower layer. [99],[48],[98], and [47] used regression of joint positions over deep Convolutional Neural Network. To this end, this thesis back propagates the error of DPM back to CNN under Structured SVM Loss function. There have been similar tests conducted in other fields using the method of back propaging the error from Structured SVM to Neural Network. In the Acoustic Signal processing field, [87] the method of back propagating the Structured SVM error from state prediction of acoustic signal to Neural Network is quite similar to the method in this thesis. However, the domain is Acoustic Signal Processing, while ours is Computer Vision. Their loss formulation is also different from ours in that they use square hinge loss function. Their loss function is also different since the nature of Acoustic Signal and Visual Signal is different. They back propagated to neural network, while ours back propagates to convolutional neural network. On Human Pose Estimation field, [112], also doing back propgation to Convolutional Neural network from SVM hinge loss. Our method differs in that we back propagate the Loss Augmented Inference loss.

The usual method of applying Latent SVM [113] with Deep learning is to extract features with deep neural network and perform Latent SVM learning as two distinct stages on the same pipeline, meaning that one must do feature extraction, then cache the result on the first stage, then submit to Latent SVM learning algorithm as the second stage. For example, Girshick [42], and [16] extract pyramid of features from convolutional neural network on the feature extraction stage, cache the extracted features and then learn Latent SVM during the second stage. On the second stage, the latent SVM then learns all model parameters by switching between SVM optimization and inference combinatorial optimization. This kind of method has an inherent problem in that they cannot learn the deep learning feature extraction parameters from the error of inference optimization, because they are on the two different distinct stages. The learnable feature extraction parameters cannot be updated by the error of Latent SVM. Seeing this shortcoming, we proposed the Structured SVM Convolutional Neural Network.

In this work, we show for the first time how to formulate Structured SVM as two layers of Convolutional Neural Network, the top layer of which is loss augmented inference layer, and the bottom is normal convolutional layer. We show that Deformable Part Model can be learned with newly created Structured SVM neural network by propagating the error of Deformable Part Model back propagate to Convolutional Neural Network. The forward propagation calculates loss augmented inference. The back propagation calculates the gradient from the loss augmented inference layer to convolutional layer. By doing so, we create a new type of convolutional neural network: Structured SVM Convolutional Neural Network, which is then applied to Human Pose Estimation problem. This new creation is a neural network, which can be used as the last layers of deep learning. Our method jointly learns structural model parameters and appearance model parameters. Our back propagation formulation can also be applied to Multiclass Classification Structured SVM where our result outperformed widely used Softmax classifier on standard MNIST dataset. We implement our method as a new layer of existing Caffe library. We built a very large back propagating system with 373,978,502 back propagating elements. Our source code is available for download.

1.3 Literature Review on Structured SVM

Why Structured SVM? The ultimate goal of computer vision is to "enable a computer to see things like humans". That means that a computer should be able to do "segmentation[24][64], detection[117][120][37][46][36][121][3], and tracking[115][116]". In the domain of part based detection and Pictorial Structure, a variant of SVM, called LSVM[31] is widely used in Human Pose Estimation [113] [106] [96] [4] [110], [80], and [75], and in other part based detection problems. For example, facial detection[121][3], scene text recognition[86],and object detection[117][120]. In the field of segmentation, Markov Random Field[60](MRF) are widely used[78]. These MRFs can be learned with Structured SVM[24][65]. Since its introduction by Joachim et al[100], Structured SVM has become a common tool in computer vision, ranging from object detection [43] and human pose estimation [18] to CRF Segmentation learning [64]. Structured SVMs are widely used to jointly learn Segmentation and Detection. The work [85] shows that Structured SVM can learn both segmentation and detection. In the case of [108] Structured SVM can further extend from Part based detection model to hierarchical poselet. Structured SVM can incorperate stochastic context free grammar to And-Or Graph for Object Detection works[63]. [37] uses Segmentation clues to help Object Detection. To do tracking, an online structured SVM can be used [115][116]. The work [69] sees Structured SVM perform learning for both Object Detection and Semantic Segmentation in the Wild environment. The work of online Structured SVM learns both Human interactive labeling and pose labeling for bird pose estimation[10] shows that Structured SVM is very capable of performing real different tasks. The work of [111] jointly learn saliency map and dictionary. The most convincing example is perhaps from the work of [28][27][68], which sees Human Pose Estimation and Segmentation model parameters learned by Structured SVM.

Structured SVM requires graphical model inference. This requirement creates a wide variety of graphical model incorperation to Structured learning for computer vision problems. For example, [117] used Bayasian optimization, and Gaussian Process as a probabilistic model, and used local fine grain search to perform inference to improve Convolutional Neural Network result on object detection. In the work [109], distributed convex believe propagation [83] is used to perform scene segmentation. Greedy algorithm for object detection work is done by [26]. The branch-and-bound strategy is used for object detection in work[7], and is learned by Structured SVM regressor. We see Markov Chain Monte Calo[41] type of inference with Bayesian Probabilistic modeling on the computer vision task of Scene Segmentation and parsing in the work of [118]. In the work of [2],RGB-D input on Scene Sementic Labeling problem, we see Integer Programming inference[70] on Structured SVM learning. And-Or Structure for Object Detection[58][62][114] tasks are another type which typically uses Structured SVM learning as their learning method. [61] use Linear Programming to solve the Visual Sementic Search problem, the model parameters of which are learned by Structured SVM.

Structured SVM can also use Kernel trick to combine multiple computer vision tasks to learn co-segmentation-detection. For example, [6] creates co-object-segmentation-detection by having object similarity kernel, mask similarity kernel, shape kernel, and local color model kernel.

When compared to LSVM, there are also works on LSVM with co-segmentation [89], but they are less widely used. It is because of the Structured SVM ability to learn graphical models in a principled way that inspired the author to work on Structured SVM. The author believes that Structured SVM learning brings us closer to the ultimate goal of co-segmentation, detection and tracking than does LSVM.

1.4 Impact of our work

In this work, the author shows, in the computer vision field, that a type of Markov Random Field can be learned with neural network. The work of [34] shows that Pictorial Structure learned by LSVM is a type of Markov Random Field. It is established knowledge that Markov Random Field can be learned with Structured SVM. In this work, the author shows that the unary potential of Markov Random Field, if learned with deep learning, can be back propagated during the structure learning phase. This is important because there are currently many works that follow the following procedure. The first step is to extract the feature with feature extractors. The second step is to use the extracted feature to learn multilcass classifier. This results in initial Unary weights of MRF. Finally. the third step is to learn MRF structurally as a distinct stage. The author is aware that learning unary potential of MRF before joining the MRF structure could help the system to escape local minima. Our work shows that it is *possible* to jointly learn all these steps in one single step. It is *possible* to jointly learn the parameters of deep learning feature extractor, MRF unary potential parameters and MRF pairwise potential parameters. Our work back propagates the error of MRF structured prediction to the deep learning unary potential weights in a principled way. The impact of our work is not only for Human Pose Estimation, but for other projects that have been using this pipeline. Researchers can now use Structured SVM back propagation for their work. For example, In the work of Scene Text Detection,Recognition [52] Convolutional Neural Network is used to extract the unary potential of each character by Random Ferns[73] (Step 1,2), then learns Pictirial Structure MRF [34] to understand the text (step 3). Given their inference algorithm are also Pictorial Structure, our Structured SVM back propagation to Convolutional Neural Network could be applied to their work to create an end to end deep learning system.

It is now established knowledge that by doing end-to-end deep learning on computer vision tasks results in high accuracy. For example [112] use of end to end in Human Pose Estimation with CNN, or in scene text recognition [52] where end-to-end CNN helps to achieve high accuracy. Our work helps others achieve end-to-end success if one has MRF, which can be learned with Structured SVM, and has deep Neural Network as Unary Potential parameters.

Currently, the few ways of solving Structured SVM are, namely, Cutting plane algorithm [100], Subgradient Method[79], Multi Sample Online Dual Ascent[9], and Block-Coordinate Frank-Wolfe Optimization[53]. Our work can be added to this list as a new way of solving Structured SVM: Back Propagation Algorithm. Although Back propagation algorithm is not different from Subgradient algorithm[79] for single layer perceptrons, it is clearly different from subgradient algorithm in multilayer perceptrons, and deep learning.

1.5 Reading this Thesis

The paper is organized as follows: Chapter 1 is the introduction and literature reviews. Chapter 2 explains basic theories and knowledge required to read the Thesis. Before we can work on complex Structured SVM back propagation to Convolutioinal Neural Network, our work must begin with formulation of Structured SVM back propagation to neural network applying to Multiclass Classification problems. We explain this in Chapter 3. Our formulation of Multiclass Structured SVM back propagate to deep learning in Chapter 3 also reports better performance than widely used deep learning multiclass Classifier Softmax[8] in standard MNIST dataset[57]. Chapter 4 is the main section, where the original work of Structured SVM Convolutional Neural Network is formulated for the first time. In Chapter 5, we compare our Structured SVM trained with Back propagation with base line LSVM, and our implementation of Cutting plane algorithm Structured SVM[101]. We conclude our work in Chapter 6.

Chapter 2

Background

Introduction

This chapter introduces common knowledge needed to understand this thesis. The author starts with a description of Pictorial Structure idea from [34]. Pictorial Structure idea was conceived as an attempt to perform object recognition whose object structure is well defined components. For example, a face object is well defined to have components such as two eyes, a nose, and a pair of lips. Their locational relationship is also well defined. These relationships of well defined components create a structure of face class. This chapter investigates the Pictorial Structure idea, how to realize Pictorial Structure as a model for detection, how to calculate Pictorial Structure detection efficiently, how to learn Pictorial Structure's model parameters with Structured SVM, and how to formulate Pictorial Structure model as an instance of Structured SVM inference.

Applying efficient calculation to Pictorial Structure for Human Pose Estimation problem is not easy to understand. This chapter starts with very simplified part localization problem, so that one can understand the Max-Sum Algorithm, also known as Viterbi Algorithm [39], which facilitates efficient calculation. This is followed by a description of how the Max-Sum algorithm can be used for a toy-based example of a computer vision part-based detection problem. After readers understand the Max-Sum Application to computer vision toy example, the more complex Pictorial Structure model for Human Pose Estimation problem is described.

To understand model learning, which is crucial to understanding this thesis, this chapter begins with Structured SVM framework. This chapter also provides the basic understanding of Structured SVM to those who are familiar with binary SVMs, but do not yet have a clear understanding of Structured SVM. After readers understand the application of Structured SVM to a multiclass classification problem, the more complex formulation of Pictorial Structure model learning into Structured SVM problem will be described.

To understand the main part of this thesis, a prerequisite understanding of Support Vector Machine(SVM), and Neural Networks are required, and will not be described in this chapter. This chapter describes forward and backward opearation of Convolutional layer of Convolutional Neural Network [55]. This is important because in Chapter 4, the readers need to know its operation to fully understand all Chapter 4's formulation.

Detection or prediction inference can be used interchangeably. The former has a straight forward meaning. The latter has its root in a probabilistic graphical model. In our system, there are two types of inference, prediction inference, and loss augmented inference. To understand loss augmented inference, one must understand the Structured SVM framework.

2.1 Pictorial Structure

The Pictorial structure Model was proposed during the 1970s. Figure 2.3 is a picture from the 1970s where the relationship of parts are being modeled in pairwise connection. Parts are connected by springs between them. The key idea of pictorial structure is to change the relationship modeling to a global optimization problem. During test time,

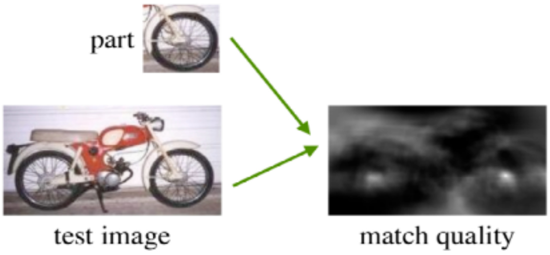


Fig. 2.1 Matching Quality on Motorbike image using a wheel as similarity filter. Courtesy of [34]

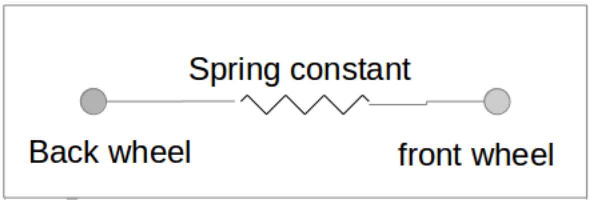


Fig. 2.2 Pictorial Structure for two wheels

part placements are the best positions of parts on an image where the springs are not too stretched or too compressed, while also having the best matching quality.

To see how the idea can be realized, for each part, there are costs of part placements on all positions of the test image. For example, if the test image is a motorcycle, some of its parts are its two wheels. In Figure 2.1, the matching quality shows the cost of placing each part on each position of the test image. White in matching quality shows places where cost of placements are low. One can see that there is more than one position to place a wheel because there is a front wheel and a back wheel, both of which look the same. Modeling of a relationship between these two wheels can be transformed into a global optimization problem. If one is not concerned with pictorial Structure, placement would occur in the whitest positions of the match quality image. Notice that there are many white places. It is likely to turn out that the two best maximum similarity scores may not match the position which the two wheels may be thought to be. These two positions are unrelated except for appearance similarity to the filters.

If we are to have structural prior information for matching, the above pictorial model could be used. The structural model of this model is as shown in Figure 2.2. The overall structural placement is the minimum cost of all placements, which are the sum of the match quality of 2 wheels and the cost of the spring. We can now write the structural placement as a global optimization problem.

$$[\hat{y_1}, \hat{y_2}] = \arg\max_{y_1, y_2} m_1(y_1) + m_2(y_2) + c_{12}\left(y_1, y_2\right) \tag{2.1}$$

,where y_1 represents the pixel location for the front wheel, and y_2 represents the pixel location for the back wheel, and $\hat{y_1}$ represents the best location for the front wheel, and $\hat{y_2}$ represents the best location for the back wheel, $m_1\left(a\right), m_2\left(a\right)$ are two matching qualities evaluating at position a, and $c_{12}\left(a, b\right)$ is the cost of stretching or compressing the spring term. At this stage, the mathematical term is not for precise definition. The main purpose in showing the above equation is to present the idea of Pictorial Structure.

Matching quality, $m_1\left(y_1\right)$,$m_2\left(y_1\right)$, in linear cases, are the filter responses of the image. This can be seen as finding the dot product of each sliding window all over the image. This is sometimes called *unary potential* because the term is a score on a single node. The spring term $c_{12}\left(y_1, y_2\right)$ is called *pairwise potential* because the score needs the values of two nodes. The value which maximize eq. 2.1 is called max-marginal of the optimization problem. The underlying assumption of pictorial structure is that **"Deformation Cost only depends on displacement among parts"**. The Figure 2.3 shows the Pictorial Structure model of a face. In an example model of a face, there are many parts of a connected graph, each of which represents a nose, an eye, an ear, etc. The dynamic programming idea is we trim each leaf from the tree by

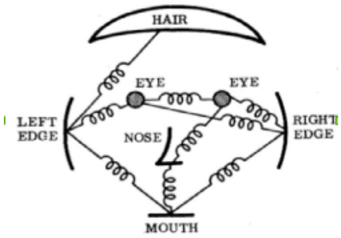


Fig. 2.3 Inference on graphs

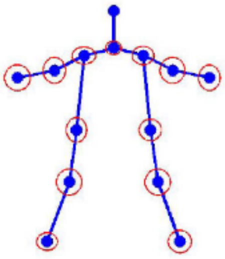


Fig. 2.4 Human Pose Estimation tree graphs

memoizing the best location and score, of each leaf with respect to its parent position. For example, we trim an eye and memoize the best location and score of an eye as a function of each location of the nose. The best score of each nose location is calculated by adding the matching score of every eye with the deformable score with respect to the nose location. Once that is done, the eye is trimmed from the tree. Following this procedure, one trims the tree until reaching the root node. One can then find the best score and position of the root node. Once the best score and position is found, one can read the memoization and retrieve the best configuration that provides the best score for eyes, nose, lips. The described algorithm is referred to as Max Sum Algorithm.

The combinatorial optimization defined by eq. 2.1 has 2 nodes. An arbitrary Pictorial Structure may have n nodes, which may look like Figure 2.4. In this thesis, we focus only on tree graph. The max-sum algorithm seeks to find the solution of the combinatorial optimization of the form,

$$\hat{L} = \arg\max_{L} \sum_{i \in V} m_i \left(l_i\right) + \sum_{ij \in E} g \left(l_i, l_j\right) \tag{2.2}$$

,under Graph $G = \{V, E\}$. To solve this problem, the naïve method is shown in Listing 2.1. From the Listing we can see that the naïve method uses $O\left(p^n\right)$ calculations, where p is the number of pixels in feature space, and n is the number of nodes. This is prohibitively expensive to calculate. There is, however, a dynamic programming method which can calculate in quadratic time provided that our graph is a tree with exact solution. The dynamic programming is the max-sum algorithm.

The max-sum algorithm gives *exact inference* to the combinatorial optimization problem. The word *exact inference* has a special mathematics meaning. It means that the algorithm gives the exactly correct answer to the maximization over the search domain, as opposed to an approximate inference which gives an approximately correct answer. The max-sum algorithm can accelerate the calculation from exponential time complexity to $O(np^2)$. Running the naïve methods has total of $16^6 = 16777216$ calculations and takes 1109 seconds on AMD Phenom2 1090t CPU (released year 2010), while the max-sum algorithm has total $5x16x16 = 1285$ calculations and takes 0.0967 seconds on the same machine.

Listing 2.1 Maximizing Objective Function 2.2 with Naive Method

```
image_size=16;
for i=1:image_size
    best_energy= -inf;
    for j_c1 =1:image_size
        for j_c2 = 1:image_size
            for j_c3 = 1:image_size
```

```
                for j_c4 = 1:image_size
                    for j_c5 = 1:image_size
energy = ...
   cal_energy(i,j_c1,j_c2,j_c3,j_c4,j_c5,MapC(:,:,1),MapC(:,:,2),
MapC(:,:,3),MapC(:,:,4),MapC(:,:,5),MapRoot,distEu);
no_cal = no_cal+1;
                        if best_energy < energy
                        best_conf_root(i,:) = [energy i j_c1 j_c2 ...
                           j_c3 j_c4 j_c5];
                        best_energy = energy;
                        end
                    end
                end
            end
        end
    end best_energy_this_root(i) = best_energy;
    if best_energy > best_energy_of_all_root
        best_energy_of_all_root = best_energy;
        best_root_of_all = i;
    end
end
```

2.2 Max Sum Algorithm

This is how the algorithm works. For ease of understanding, we shall begin with a simple tree graph. Moving from the leaves to the root, the algorithm calculates each possible answer for parent's value and labels the children which have the maximum contribution to the score of the parent. To find the label which gives maximum contribution to the root, one starts by calculating all possible scores from every child's label to each score of root's label. One can then find the best score and position of the root node by maximizing the root score. Once the best root score and best root position is found, one can read the memoization and retrive the configuration that provides the best score.

Here is an example. Suppose our image has the size of $(4, 4)$, and the pixel location l is a scalar, which is a row major order index of the matrix $(4, 4)$. Let us define all possible locations in this image as $\mathbb{L} = \{1, ..., 16\}$. MapP, MapT, MapQ, MapR, MapS maps pixel postion $l \in \mathbb{L}$ to a scalar real value. Therefore $\forall l \in \mathbb{L}$, these mapping functions are tables of the domain $\mathbb{L}$. Thus each of these MapP, MapT, MapQ, MapR, MapS is a vector of size 16 or a matrix of size $(4, 4)$. From the Figure 2.5 one can see there is a total of 16x16 connections between all labels of P to all labels of Q. Here, the Q node updates its unary potentials. For each label of Q, it selects the maximum unary potential updates it can get from every label of P adding the distance score during traversing through its edge. The message from the two leaves are defined as

$$\underset{P\to Q}{Msg}(j) = \max_{i}\left(MapP\left(i\right) + dist\left(i, j\right)\right) \tag{2.3}$$

$$\underset{T\to Q}{Msg}(j) = \max_{t}\left(MapT\left(t\right) + dist\left(t, j\right)\right) \tag{2.4}$$

, where $dist\left(v_i, v_j\right) = \lambda\left(\left(x_{v_i} - x_{v_j}\right)^2 + \left(y_{v_i} - y_{v_j}\right)^2\right)$, and x_{v_i}, y_{v_i} means row, and column number of position of node v_i. The parameter λ in this case is an arbitrary hyperparameter constant. The value $\underset{P\to Q}{Msg}(j)_{1x1}$ is a scalar which memorize (OR MEMOIZE?) $\max MapP\left(\hat{i}\right) + dist\left(\hat{i}, j\right)$ where $\hat{i}$ is the i^{th} label of P which gives THE maximum value of all labels i of P. Each label of Q, j, has different $\hat{i}$ For all the labels of Q, we get the matrix $\underset{P\to Q}{Msg}_{16x1}$. Generally Dynamic Programming needs to memorize the configuration, and the value, which maximize each subproblem. Here the

$$\underset{Q\to P}{belovedChild}_{16x1}(j) = \arg\max_{i} MapP\left(i\right) + dist\left(i, j\right) \tag{2.5}$$

$$\underset{Q\to T}{belovedChild}_{16x1}(j) = \arg\max_{t} MapT\,(t) + dist\,(t, j) \tag{2.6}$$

are recorded. Simply speaking, each parent's label records which label provides the best contribution to parent. This is shown in Listing 2.2 line 55.

To be consistent with eq. 2.2, we can think of node P, T, Q, R, S as node $i = \{1, 2, 3, 4, 5\}$. In this case, MapP is our $m_1\,(L)$, MapT is our $m_2\,(L)$, MapQ is our $m_3\,(L)$, MapR is our $m_4\,(L)$, MapS is our $m_5\,(L)$, where $L = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$.

Next, the message from Q to R is defined as

$$\underset{Q\to R}{Msg}(k) = \max_{j}\left(MapQ\,(j) + \underset{P\to Q}{Msg}(j) + \underset{T\to Q}{Msg}(j) + dist\,(j, k)\right) \tag{2.7}$$

,and then node R memoization

$$\underset{R\to Q}{belovedChild}_{16x1}(k) = \arg\max_{j} MapQ\,(j) + \underset{P\to Q}{Msg}(j) + \underset{t\to q}{Msg}(j) + dist\,(j, k) \tag{2.8}$$

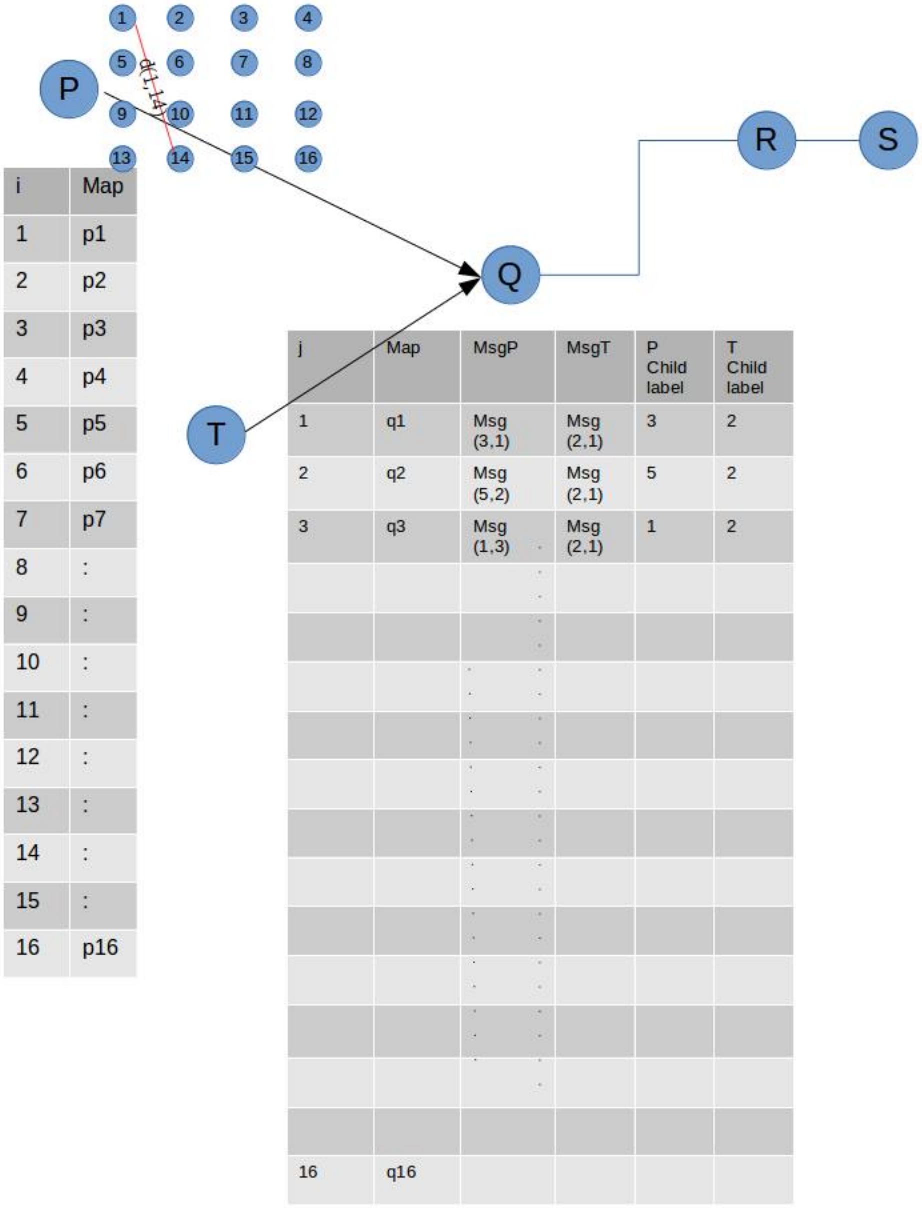
In the next step, the Message Passing from R to S is calculated as follows,

$$\underset{R\to S}{Msg}(s) = \max_{k}\left(MapR\,(k) + \underset{Q\to R}{Msg}(k) + dist\,(k, s)\right) \tag{2.9}$$

,and then node S memoization

$$\underset{S\to R}{belovedChild}_{16x1}(l) = \arg\max_{k} MapR\,(k) + \underset{Q\to R}{Msg}(k) + dist\,(k, s) \tag{2.10}$$

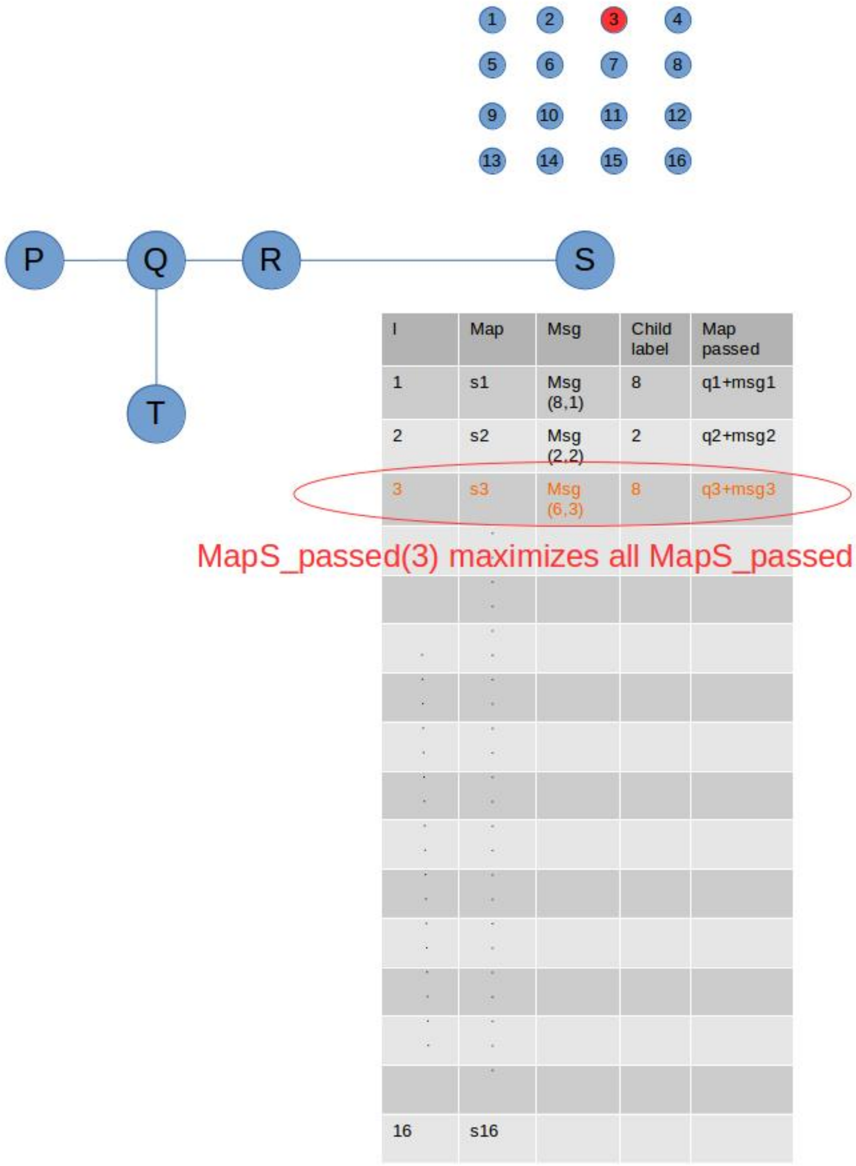
. These steps are shown in Figure 2.5. In this Figure, $\underset{P\to Q}{Msg}$, and $\underset{T\to Q}{Msg}$ is shown, with its memoized table eq. 2.5, and 2.6, respectively.



i	Map
1	p1
2	p2
3	p3
4	p4
5	p5
6	p6
7	p7
8	:
9	:
10	:
11	:
12	:
13	:
14	:
15	:
16	p16

j	Map	MsgP	MsgT	P Child label	T Child label
1	q1	Msg (3,1)	Msg (2,1)	3	2
2	q2	Msg (5,2)	Msg (2,1)	5	2
3	q3	Msg (1,3)	Msg (2,1)	1	2
16	q16				

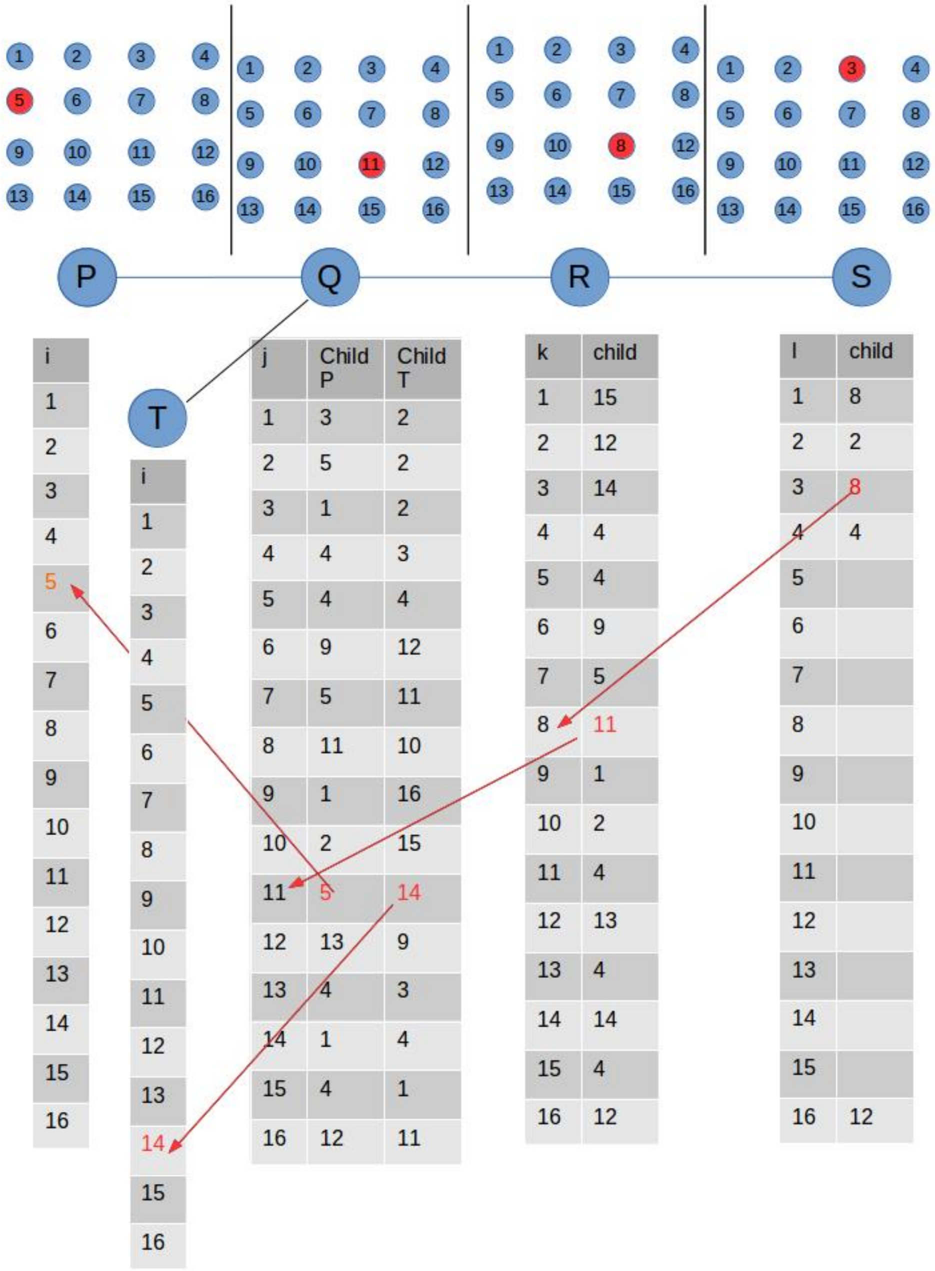
Fig. 2.5 Message Passing from many children to a parent



I	Map	Msg	Child label	Map passed
1	s1	Msg (8,1)	8	q1+msg1
2	s2	Msg (2,2)	2	q2+msg2
3	s3	Msg (6,3)	8	q3+msg3
	⋮			
	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
⋮	⋮			
	⋮			
16	s16			

MapS_passed(3) maximizes all MapS_passed

Fig. 2.6 Message Passing has reached root node. The max-marginal score is found



i
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

i
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

j	Child P	Child T
1	3	2
2	5	2
3	1	2
4	4	3
5	4	4
6	9	12
7	5	11
8	11	10
9	1	16
10	2	15
11	5	14
12	13	9
13	4	3
14	1	4
15	4	1
16	12	11

k	child
1	15
2	12
3	14
4	4
5	4
6	9
7	5
8	11
9	1
10	2
11	4
12	13
13	4
14	14
15	4
16	12

l	child
1	8
2	2
3	8
4	4
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	12

Fig. 2.7 Backtrack with stored tables from root to leafs

Once the messages from all root branches are passed to the root node, we maximize the root score to find the best configuration for the root node. This is called finding the maximum value of our dynamic programming objective eq. 2.2, sometimes called max-marginal. To calculate the max-marginal, one simply adds all the passed messages to the root unary potential and finds the maximum.

$$G\left(\hat{s}\right) = max_s \underset{R\to S}{Msg}(s) + MapS\left(s\right) \tag{2.11}$$

This is line 33 in Listing 2.2, also shown in Figure 2.6. We call $G\left(\hat{s}\right)$ max-marginal. The $\hat{s}$ is the solution of the combinatorial optimization at root position. To get the configuration $\hat{l} = \{\quad \hat{s} \quad \hat{k} \quad \hat{j} \quad \hat{t} \quad \hat{\imath} \quad\}$, which maximizes our initial problem eq. 2.2, we can use table lookup from our previously memoized table. $\hat{s} = \arg\max_{s\in\{1,2,3,\ldots,16\}} \underset{r\to s}{MsgS}(s) + MapS\left(s\right)$ Once $\hat{s}$ value is taken, we look upwards for its best child label $\hat{k} = \underset{S\to R}{belovedChild}_{16x1}\left(\hat{s}\right)$. Once $\hat{k}$ value is taken, we look upwards for its best child label $\hat{\jmath} = \underset{R\to Q}{belovedChild}_{16x1}\left(\hat{k}\right)$. Once $\hat{\jmath}$ value is taken, we look up for its best child label $\hat{\imath} = \underset{P\to Q}{belovedChild}_{16x1}\left(\hat{\jmath}\right)$, and $\hat{t} = \underset{P\to T}{belovedChild}_{16x1}\left(\hat{\jmath}\right)$. Finally, we answer our original question eq. 2.2 as $\hat{l} = \{\quad \hat{s} \quad \hat{k} \quad \hat{j} \quad \hat{t} \quad \hat{\imath} \quad\}$. This is line 82 of Listing 2.2, also shown in Figure 2.7.

In Figure 2.6, if the 3rd label of the root score is the maximum of all root scores across the labels, then the root position, $\hat{s}$, is set to 3. The value of the root score represents $\max_L \sum_{i\in V} m_i\left(l_i\right) + \sum_{ij\in E} g\left(l_i, l_j\right)$. This completes the *forward pass.* Still we need to find the argument which maximizes it. This can be found by looking back the beloved child tables.

Figure 2.7 shows how this can be done. Once the label of the best root ancestor is found, the algorithm follows the memoization by recursively looking at the most beloved child table. In our example, from S to R to Q, and to P. This completes the *backward pass*. This completes the algorithm and the solution $\hat{l}$ is found, thus completes our objective.

Listing 2.2 Maximizing Objective Function 2.2 with Max-Sum Algorithm

```
test_yi = [9 10 11 12 13 14];
loss_fn = @(y) (double(sum(y≠test_yi)));
MapC(:,:,1) = 5*[1,2,3,4;4 3 2 2;2 4 3 2; 3 1 2 1];
MapC(:,:,2) = 2*[4 3 1 3; 1 3 4 2; 2 3 3 1; 3 3 1 5];
MapC(:,:,3) = 5*[1 2 1 3; 1 4 4 2; 3 3 4 2; 2 3 4 3];
MapC(:,:,4) = 3*[3 2 3 3; 3 4 4 3; 3 4 4 3; 3 3 3 3];
MapC(:,:,5) = 5*[5 5 6 2; 5 1 1 5; 5 1 1 6; 4 5 5 3];
Mxr = [4;2;1;4];
%MapC(:,:,1)=Mxr*Mxr';
MapRoot=2*Mxr*Mxr';
%MapRoot=zeros(4);
distEu=@(x,y) (-1*(norm(x-y))^2);
%distEu = @(x,y) 0;


% Tree structure prior and its plot
tstr = [0 1 5 6 1 1];
%tstr=[0 1 1 1 2 3];
t_name =0:5;
%show_tree(tstr,t_name);
Msg = zeros(16,size(tstr,2));
bt = zeros(16,size(tstr,2));
% Tree structure prior and its plot
tstr = [0 1 5 6 1 1];
%tstr=[0 1 1 1 2 3];
t_name =0:5;
```

```
%show_tree(tstr,t_name);
Msg = zeros(16,size(tstr,2));
bt = zeros(16,size(tstr,2));
T_iter = tstr;
T_name = t_name;
tree_depth=0;
num_cal=0;
zeta= zeros(16,16);
while size(T_iter,2)>1
    leafs = find_leaves(T_iter,T_name)
    tree_depth=tree_depth+1;
     book_leaves_depth{tree_depth} = leafs;
     for leaf_j=1:size(leafs,1)
     leafs(leaf_j) ;
       for i=1:16  %parent
           for j=1:16 %current leaf_j
               zeta(i,j) =   distEu(PostMap(i),PostMap(j));
               num_cal = num_cal+1;
           end
           unary_j = MapC(:,:,leafs(leaf_j));
           unary_j = unary_j(:)';
           children_of_leaf_j = find_children(tstr, t_name, ...
              leafs(leaf_j));
           if(¬isempty(children_of_leaf_j))
               sumChdrenMsg = sum( Msg(:,children_of_leaf_j)',1);
           else
               sumChdrenMsg = zeros(1,16);
           end
           parNode = t_name(T_iter(find(T_name == leafs(leaf_j))));
          [Msg(i,leafs(leaf_j)) bt(i,leafs(leaf_j))] = ...
             max(unary_j+ zeta(i,:)+ sumChdrenMsg );
       end
```

```
            out = sprintf('working node %i, children %d, parent ...
               node%d, msg %d \n',...
            leafs(leaf_j),children_of_leaf_j, parNode, ...
               Msg(i,leafs(leaf_j)));
            disp(out);

            T_iter(find(T_name == leafs(leaf_j)))=[]
            T_name(find(T_name == leafs(leaf_j)))=[]

     end
end
children_of_root = find_children(tstr, t_name, 0)
if(¬isempty(children_of_root))
       sumChdrenMsg =sum( Msg(:,children_of_root)',1);
 else
       sumChdrenMsg = zeros(1,16);
 end

[max_marginal arg_max_marginal] = max(MapRoot(:)' + sumChdrenMsg);
RootScore = reshape(MapRoot(:)'+sumChdrenMsg,[4 4]);
best_energy_of_all_root = max_marginal;
best_root_of_all = arg_max_marginal;

vi_pos(1) = best_root_of_all;
 out = sprintf('==================\nbest configuration at ...
    root-%d',...
     best_root_of_all);
 disp(out);
for i=tree_depth:-1:1
    bp_nodes = book_leaves_depth{i};
    for j=1:size(bp_nodes,1)
    parNodeIdx =  tstr(find(t_name == bp_nodes(j)));
    vi_pos(bp_nodes(j)+1) =  bt( vi_pos(parNodeIdx) ,bp_nodes(j)) ;
    num_cal = num_cal+1;
    end
end
for(b=2:size(tstr,2))
    out = sprintf('best v%d is at pos %d',b-1,vi_pos(b));
    disp(out);
end
```

2.3 Toy example of Part base detection

In this section, the toy-based example of part base detection is explained. Suppose we have a Swastika symbol whose part filters is exactly those parts of Swastika symbol. Our choice of this symbol is because it is structured image with respect to its axis appearance. Our choice is of pure technical reason, and has nothing to do with politics or history. By defining the part filters to be the axis of the Swastika symbol itself, we ensure that the similarity score is high when the subwindow score looks similar. This can be measured by means of convolution.

Let $\mathbb{T} = \{\mathbb{V}, \mathbb{E}\}$ denote a tree graph whose structure prior we wish to match. Let $\bar{\Theta}_{p\in\{1..K\}}$, where K is the number of parts, be the unary potential such that $\bar{\Theta}_p(l)$ is the scalar unary potential of pixel position l. Therefore $\bar{\boldsymbol{\Theta}}_p$ is a vector of size $|\mathbb{L}|$, where $\mathbb{L}$ is a set all pixel position. Let $\Delta(l_i, l_j)$ denote pairwise function between i^{th} part, and j^{th} part. Let image intensity be denoted by $\mathbf{X}$, and the image intensity at the position l of p^{th} part denotes $\mathbf{X}(l_p)$. This $\mathbf{X}(l_p)$ is not a scalar pixel intensity of position l_p but a matrix of cropped image $\mathbf{X}$ at the top left position l_p. Let $S_p \in \mathbb{Z}^2$ be the number of rows, and columns of the part p's filter, then the image intensity $\mathbf{X}(l_p)$ is a matrix of size S_p, with l_p at the top-left position. Our $\mathbb{Z}$ denotes a set of all

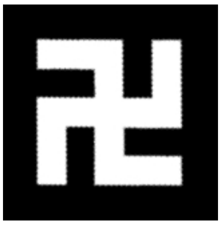


Fig. 2.8 Original Image

integers. Let $\mathbf{w}_p \in \mathbb{R}^{S_p}$ be the similarity filter of part p. In our example, we just crop the intensity value of axis of the Swastika symbol as the filter.

Our requirement is to find a solution $\bar{l}^*$ such that

$$\bar{l}^* = \arg\max_{l} \sum_{i \in \mathbb{V}} \bar{\Theta}_i \left(l_i\right) + \sum_{ij \in \mathbb{E}} \Delta \left(l_i, l_j\right) \tag{2.12}$$

, where $\Delta \left(l_i, l_j\right) = \lambda \left(\left(x_{l_i} - x_{l_j}\right)^2 + \left(y_{l_i} - y_{l_j}\right)^2\right)$, and x_{l_i}, y_{l_i} means row, and column number of position of node l_i.

Let our similarity filter be a matrix of size $23, 46$, which looks like an axis. The unary potentials of each part can be calculated by

$$\bar{\Theta}_i \left(v_i\right) = \sum_{\bar{x}} \sum_{\bar{y}} \mathbf{w} \left(\bar{x}, \bar{y}\right) \mathbf{X} \left(x + \bar{x}, y + \bar{y}\right) \tag{2.13}$$

, or in the compressed form

$$\bar{\Theta}_i \left(v_i\right) = \sum_{\bar{l}} \mathbf{w} \left(\bar{l}\right) \mathbf{X} \left(l + \bar{l}\right) \tag{2.14}$$

. Figure 2.9 shows $\bar{\Theta}_1$,.

We follow the message passing algorithm as described in previous section to find the best position of placing the white bar, on the Swastika symbol. Figure 2.10 shows

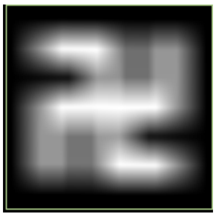


Fig. 2.9 The first part's scores

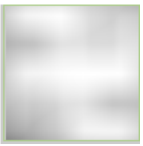


Fig. 2.10 Root Scores plus all passed messages

the root potential after all messages have passed. We hope that the root location of the detection will be in the middle of the image.

Finally, after backward operation, we obtain the best position for placing our tree on the Swastika symbol as shown in Figure 2.11. This is the expected result because the best position of the similarity filters are at the horizontal axes of Swastika symbol, and since the Swastika symbol is symmetrical, the best place to place the root similarity filter is the middle of the flag.

2.4 Structured SVM

The purpose of Structural SVM is to produce Structural Prediction by learning maximum margin classifier for each training data. Probabilistic Graphical Model such as Markov Random Fields (MRF), or Conditional Random Fields (CRF) can use Structured SVM during the learning phase to learn their weight parameters. Structured SVM is not an algorithm in which one can simply plug in the data and conduct learning or classification like SVM, but a framework for which an inference, loss, and feature modules must be plugged in first. For example, if a Structured SVM is to be applied

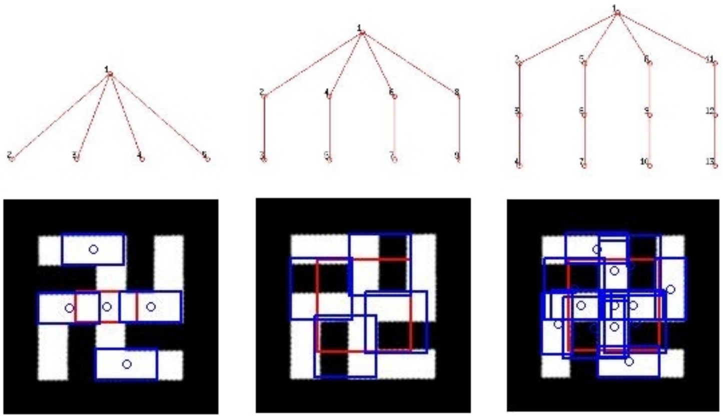


Fig. 2.11 Inference on different tree graphs

with MRF, one must specify the MRF structure, which the Structured SVM will learn, the MRF inference algorithm, the MRF feature function, the loss function, and the loss augmented inference algorithm. The loss augmented inference algorithm is the inference algorithm with loss function.

The propose of SSVM is to learn the structural prediction function of the form

$$\hat{y} = \arg\max_{\hat{y} \in \mathbb{Y}} \mathbf{w} \cdot \mathbf{\Phi_a}\left(x, y\right) \tag{2.15}$$

Suppose we are given the training set $(\mathbf{x_i}, \mathbf{y_i})$, for $i = 1, ..., N$. Where $\mathbf{x_i}$ is i^{th} the input feature, and $\mathbf{y_i}$ are vectored label. Our goal is to find the assignment of position $\hat{y}$ to the a image, whose label is unknown. This is called structured prediction.

The above detection function can be seen as structural prediction where one finds the position $\hat{y}$ of a node on the Figure 2.4 such that inference eq. 2.15 is satisfied. We also need to learn this structural prediction function a weight matrix that maximizes training accuracy. In other words, we want to find $\mathbf{w}^*$, by structural learning, such that, for all images in the training data, the eq. 2.15 provides good training accuracy. The $\mathbf{w}^*$ is the argument $\mathbf{w}$ which maximizes the objective function eq. 2.16.

The SSVM has the optimization of the form [100]

$$\begin{aligned} &\min_{w,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ &s.t. \quad \forall i \quad , \forall y \in Y \setminus y_i : \quad \mathbf{w} \cdot \delta\Phi_{ai}\left(\mathbf{x_i}, \mathbf{y}\right) \geq \Delta\left(y_i, y\right) - \xi_i \end{aligned} \tag{2.16}$$

,where $\delta\Phi_a\left(\mathbf{x_i}, \mathbf{y}\right) = \Phi_a\left(\mathbf{x_i}, \mathbf{y_i}\right) - \Phi_a\left(\mathbf{x_i}, \mathbf{y}\right)$

The above explanation is quite generic. Let us start with simple explanation. Since readers are more familiar with SVM, we shall first explain how to formulate Linear SVM as an instance of Structured SVM.

In a linear binary SVM problem, we have a feature vector $\mathbf{x} \in \mathbb{R}^n$, and label $y \in \{-1, 1\}$. Let's define $\bar{y}_i = -1 \times y_i$ to be an incorrect label. If we define a function $\mathbf{\Phi}\left(\mathbf{x}, y\right) = y \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$, and $\mathbf{w}_{wb} = \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}$. Previous Linear SVM prediction is correct when $y_i\left(\mathbf{w^T x_i} + b\right) \geq 0$ or $\mathbf{w}_{wb} \cdot \mathbf{\Phi}\left(\mathbf{x}, y\right) \geq 0$. That is, suppose $\mathbf{w}^*_{wb}$ was learned, such that the previous test gave 100% correct training accuracy. One can see that, in such a case, for all i, any misclassified samples will have $\bar{y}_i\left(\mathbf{w}^* \cdot \mathbf{x}_i + b\right) \leq 0$. Therefore, one may say that the objective of previous hard-margin Linear SVM

$$\begin{aligned} &\min_{\mathbf{w}_{wb},\xi} \quad \left\|\mathbf{w}_{wb}\right\|^2 \\ &s.t. \quad \forall i : \quad \mathbf{w}_{wb} \cdot \Phi_{ai}\left(\mathbf{x_i}, \mathbf{y_i}\right) \geq 1 \end{aligned} \tag{2.17}$$

is to find $\mathbf{w}^*_{wb}$ such that $\forall i, \quad y_i\left(\mathbf{w}^* \cdot \mathbf{x_i} + b\right) \geq 0$ is satisfied. This is shown in Figure 2.12. The correct classification score $y_i\left(\mathbf{w}^* \cdot \mathbf{x_i} + b\right) \geq 0$, and the incorrect classification score $-y_i\left(\mathbf{w}^* \cdot \mathbf{x_i} + b\right) \leq 0$, therefore the correct classification score $y_i\left(\mathbf{w}^* \cdot \mathbf{x_i} + b\right)$ must ≥ 0 incorrect classification score $\bar{y}_i\left(\mathbf{w}^* \cdot \mathbf{x_i} + b\right)$ must ≤ 0. Since a number that belongs to a set of non-negative real numbers is always greater than or equal to a number

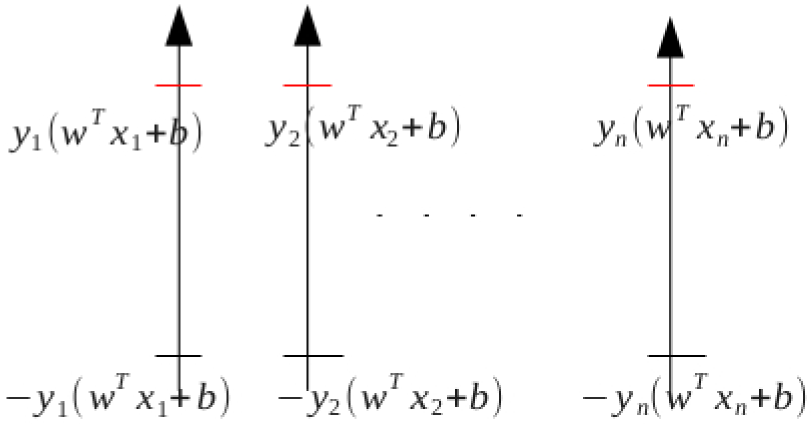


Fig. 2.12 Linear SVM as sorting. The arrows are the real value axes, representing each training data. If the classifications are all correct, such that the training error is 0%, then the correct prediction score $y_i \left(\mathbf{w}^T\mathbf{x_i} + b\right) \geq 0$, and the incorrect prediction score $-y_i \left(\mathbf{w^T}\mathbf{x_i} + b\right) \leq 0$

belonging to a set of non-positive real number, the correct score is always greater than or equal to the incorrect score. Thus, by choosing the maximum number from among the two scores, one can choose the correct prediction $\hat{y} = y_i$. This explanation changes the mindset from hyperplane to sorting.

2.5 Convolutional Neural Network

To understand the proposed algorithm, one needs to understand how the convolutional layer of a convolutional neural network differs from a conventinonal neural network layer. The idea presented here is that the Convolutional Neural Network is simply the sliding windows of a Neural Network.

Convolutional neural network is simply the conventional linear algebra neural network with shared weights across the board. For example, the linear algebra neural network operation $y = \begin{pmatrix} \mathbf{w} \\ w_0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$

Now if $\mathbf{x}$ has more rows and columns than $\mathbf{w}$, both output y_1, and y_2 are fed forward neural network with the same $\mathbf{w}$ value. $y_1 = \begin{pmatrix} \mathbf{w} \\ w_0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}\,(1 : R, 1 : C) \\ 1 \end{pmatrix}$, and $y_2 = \begin{pmatrix} \mathbf{w} \\ w_0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}\,(1 : R+1, 2 : C+1) \\ 1 \end{pmatrix}$. Hence the name *shared weights*. For the sake of simplicity, let us remove the bias w_0 from our calculation. The convolution layer is defined by

$$\mathbf{y}_{ij}^{(2)} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} x_{(a+i)(b+j)}^{(1)}$$

Suppose our example has 2 nodes on the upper layer, denoted $\mathbf{y}_{00}^{(2)}$, and $\mathbf{y}_{01}^{(2)}$. Suppose our weights $\mathbf{w}$ are of size (5,5) suppose our lower layer $x_{(a)(b)}^{(1)}$ is size (6,6) such that $a \in \{0, ..., 5\}$, and $b \in \{0, ..., 5\}$. Then the Forward Operation is calculated by

$$\mathbf{y}_{00}^{(2)} = \sum_{a=0}^{5-1} \sum_{b=0}^{5-1} w_{ab} x_{(a)(b)}^{(1)}$$

$$\mathbf{y}_{01}^{(2)} = \sum_{a=0}^{5-1} \sum_{b=0}^{5-1} w_{ab} x_{(a)(1+b)}^{(1)}$$

or in the matrix form,

$$\mathbf{y}_{00}^{(2)} = \begin{pmatrix} w_{00} & w_{01} & w_{02} & w_{03} & w_{04} \\ w_{10} & w_{11} & w_{12} & w_{13} & w_{14} \\ w_{20} & w_{21} & w_{22} & w_{23} & w_{24} \\ w_{30} & w_{31} & w_{32} & w_{33} & w_{34} \\ w_{40} & w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} \cdot \begin{pmatrix} x_{00}^{(1)} & x_{01}^{(1)} & x_{02}^{(1)} & x_{03}^{(1)} & x_{04}^{(1)} \\ x_{10}^{(1)} & x_{11}^{(1)} & x_{12}^{(1)} & x_{13}^{(1)} & x_{14}^{(1)} \\ x_{20}^{(1)} & x_{21}^{(1)} & x_{22}^{(1)} & x_{23}^{(1)} & x_{24}^{(1)} \\ x_{30}^{(1)} & x_{31}^{(1)} & x_{32}^{(1)} & x_{33}^{(1)} & x_{34}^{(1)} \\ x_{40}^{(1)} & x_{41}^{(1)} & x_{42}^{(1)} & x_{43}^{(1)} & x_{44}^{(1)} \end{pmatrix} \tag{2.18}$$

$$\mathbf{y}^{(2)}_{01} = \begin{pmatrix} w_{00} & w_{01} & w_{02} & w_{03} & w_{04} \\ w_{10} & w_{11} & w_{12} & w_{13} & w_{14} \\ w_{20} & w_{21} & w_{22} & w_{23} & w_{24} \\ w_{30} & w_{31} & w_{32} & w_{33} & w_{34} \\ w_{40} & w_{41} & w_{42} & w_{43} & w_{44} \end{pmatrix} \cdot \begin{pmatrix} x^{(1)}_{01} & x^{(1)}_{02} & x^{(1)}_{03} & x^{(1)}_{04} & x^{(1)}_{05} \\ x^{(1)}_{11} & x^{(1)}_{12} & x^{(1)}_{13} & x^{(1)}_{14} & x^{(1)}_{15} \\ x^{(1)}_{21} & x^{(1)}_{22} & x^{(1)}_{23} & x^{(1)}_{24} & x^{(1)}_{25} \\ x^{(1)}_{31} & x^{(1)}_{32} & x^{(1)}_{33} & x^{(1)}_{34} & x^{(1)}_{35} \\ x^{(1)}_{41} & x^{(1)}_{42} & x^{(1)}_{43} & x^{(1)}_{44} & x^{(1)}_{45} \end{pmatrix} \tag{2.19}$$

During the Backward Operation, the CNN back propagate errors according to the following equations.

$$\frac{\partial E}{\partial x^{(1)}_{ij}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} \frac{\partial E}{\partial y^{(2)}_{(i-a)(j-b)}} \tag{2.20}$$

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{I-m} \sum_{j=0}^{J-m} x^{(1)}_{(i+a,j+b)} \frac{\partial E}{\partial y^{(2)}_{ij}} \tag{2.21}$$

where I, J is the number of rows and columns of the image x. If we set the following two matrices,

$$\begin{pmatrix} \frac{\partial E}{\partial x_{00}} & \frac{\partial E}{\partial x_{01}} & \cdots & \frac{\partial E}{\partial x_{04}} \\ \frac{\partial E}{\partial x_{10}} & \frac{\partial E}{\partial x_{11}} & \cdots & \frac{\partial E}{\partial x_{14}} \\ \frac{\partial E}{\partial x_{20}} & \frac{\partial E}{\partial x_{21}} & \cdots & \frac{\partial E}{\partial x_{24}} \\ \frac{\partial E}{\partial x_{30}} & \frac{\partial E}{\partial x_{31}} & \cdots & \frac{\partial E}{\partial x_{34}} \\ \frac{\partial E}{\partial x_{40}} & \frac{\partial E}{\partial x_{41}} & \cdots & \frac{\partial E}{\partial x_{44}} \end{pmatrix}_{\mathbf{yoo}} = \begin{pmatrix} \frac{\partial E}{\partial y^{(2)}_{00}} & \frac{\partial E}{\partial y^{(2)}_{00}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{00}} \\ \frac{\partial E}{\partial y^{(2)}_{00}} & \frac{\partial E}{\partial y^{(2)}_{00}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{00}} \\ \frac{\partial E}{\partial y^{(2)}_{00}} & \frac{\partial E}{\partial y^{(2)}_{00}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{00}} \\ \frac{\partial E}{\partial y^{(2)}_{00}} & \frac{\partial E}{\partial y^{(2)}_{00}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{00}} \\ \frac{\partial E}{\partial y^{(2)}_{00}} & \frac{\partial E}{\partial y^{(2)}_{00}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{00}} \end{pmatrix} \odot \begin{pmatrix} w_{00} & w_{01} & \cdots & w_{04} \\ w_{10} & w_{11} & \cdots & w_{14} \\ w_{20} & w_{21} & \cdots & w_{24} \\ w_{30} & w_{31} & \cdots & w_{34} \\ w_{40} & w_{41} & \cdots & w_{44} \end{pmatrix} \tag{2.22}$$

$$\begin{pmatrix} \frac{\partial E}{\partial x_{01}} & \frac{\partial E}{\partial x_{02}} & \cdots & \frac{\partial E}{\partial x_{05}} \\ \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} & \cdots & \frac{\partial E}{\partial x_{15}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} & \cdots & \frac{\partial E}{\partial x_{25}} \\ \frac{\partial E}{\partial x_{31}} & \frac{\partial E}{\partial x_{32}} & \cdots & \frac{\partial E}{\partial x_{35}} \\ \frac{\partial E}{\partial x_{41}} & \frac{\partial E}{\partial x_{42}} & \cdots & \frac{\partial E}{\partial x_{45}} \end{pmatrix}_{\mathbf{y01}} = \begin{pmatrix} \frac{\partial E}{\partial y^{(2)}_{01}} & \frac{\partial E}{\partial y^{(2)}_{01}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{01}} \\ \frac{\partial E}{\partial y^{(2)}_{01}} & \frac{\partial E}{\partial y^{(2)}_{01}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{01}} \\ \frac{\partial E}{\partial y^{(2)}_{01}} & \frac{\partial E}{\partial y^{(2)}_{01}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{01}} \\ \frac{\partial E}{\partial y^{(2)}_{01}} & \frac{\partial E}{\partial y^{(2)}_{01}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{01}} \\ \frac{\partial E}{\partial y^{(2)}_{01}} & \frac{\partial E}{\partial y^{(2)}_{01}} & \cdots & \frac{\partial E}{\partial y^{(2)}_{01}} \end{pmatrix} \odot \begin{pmatrix} w_{00} & w_{01} \cdots & w_{04} \\ w_{10} & w_{11} \cdots & w_{14} \\ w_{20} & w_{21} \cdots & w_{24} \\ w_{30} & w_{31} \cdots & w_{34} \\ w_{40} & w_{41} \cdots & w_{44} \end{pmatrix} \tag{2.23}$$

,then eq. 2.20 can be written as ,where $\odot$ is the element-wise multiplication.

$$\frac{\partial E}{\partial x^{(1)}_{ij}} =$$

$$\begin{pmatrix} \frac{\partial E}{\partial x_{00}}\Big|_{\mathbf{y00}} & \frac{\partial E}{\partial x_{01}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{01}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{02}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{02}}\Big|_{\mathbf{y01}} & \cdots & \frac{\partial E}{\partial x_{04}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{04}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{05}}\Big|_{\mathbf{y01}} \\ \frac{\partial E}{\partial x_{10}}\Big|_{\mathbf{y00}} & \frac{\partial E}{\partial x_{11}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{11}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{12}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{12}}\Big|_{\mathbf{y01}} & \cdots & \frac{\partial E}{\partial x_{14}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{14}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{15}}\Big|_{\mathbf{y01}} \\ \frac{\partial E}{\partial x_{20}}\Big|_{\mathbf{y00}} & \frac{\partial E}{\partial x_{21}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{21}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{22}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{22}}\Big|_{\mathbf{y01}} & \cdots & \frac{\partial E}{\partial x_{24}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{24}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{25}}\Big|_{\mathbf{y01}} \\ \frac{\partial E}{\partial x_{30}}\Big|_{\mathbf{y00}} & \frac{\partial E}{\partial x_{31}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{31}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{32}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{32}}\Big|_{\mathbf{y01}} & \cdots & \frac{\partial E}{\partial x_{34}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{34}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{35}}\Big|_{\mathbf{y01}} \\ \frac{\partial E}{\partial x_{40}}\Big|_{\mathbf{y00}} & \frac{\partial E}{\partial x_{41}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{41}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{42}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{42}}\Big|_{\mathbf{y01}} & \cdots & \frac{\partial E}{\partial x_{44}}\Big|_{\mathbf{y00}} + \frac{\partial E}{\partial x_{44}}\Big|_{\mathbf{y01}} & \frac{\partial E}{\partial x_{45}}\Big|_{\mathbf{y01}} \end{pmatrix}$$

One neural network back propagation is the green part, and the other is the red. One can see the true nature of back propagation of convolutional layer of Convolutional Neural Network. The Convolutional Layer of Convolutional Neural Network is simply the sliding window of conventional linear algebra neural network layer $\mathbf{y} = \mathbf{w} \cdot \mathbf{x}$. During the forward operation, the sliding linear algebra neural network is performed. During the backward operation, the sliding back propagation of linear algebra neural network is performed. This idea has a profound impact in later derivations of our Structured SVM Convolutional Neural Network.

Chapter 3

Multiclass Structured SVM backpropagation to Deep Learning

3.1 Introduction

In this work, we created a new multiclass classifier layer for deep learning classification problems. Currently, Softmax classifier[8] is the widely used classifier for deep learning multiclass classification. We introduce a new classifier: Structured SVM multiclass classifier, which out-performs Softmax classifier[8] in MNIST dataset. Our new classifier is simple to use and can replace softmax layer with our structured SVM multiclass classifier. To the best of our knowledge, there has never been any formulation of multiclass Structured SVM back propagation to Deep learning image classification. Our implementation is on Caffe library and the source code is available for download.

Currently, deep learning is widely used for image classification, and data classification problems. For example, Caffe library[49], TensorFlow[1] library and Theano libary[94] are using Softmax classifier for their Hand written digit classification examples. The most prominent of these example is the Multicolumn deep neural network[19], which is the state of the art MNIST dataset benchmark of year 2012. The loss layer of deep learning multiclass classification of these examples is softmax classifier. Though multiclass Structured SVM classifier had been in use for some time, there has not yet been any usage of it as multiclass classifier on deep learning. As Tang et al.[92] shows, SVM based classifier outperforms Softmax Classifier by a small but consistent margin. Tang wonders how other formulations of SVM based multiclass classifiers would perform. Therefore, in this work, we introduce Structured SVM multiclass classifier back propagation to lower layers of neural networks. This is an extension of the work of Tang[92].

The closest work in the liturature, to the best of our knowlede, is done in Acoustic Signal Processing domain, in which Multiclass Structured SVM is formulated and back propagated for Acoustic Signal Processing work [87]. This formulation differs from ours in that it uses square hinge loss, while we use linear loss. In this work, the multiclass Structured SVM back propagation to Deep Convolutional Neural Network is applied to deep learning image classification problem for the first time.

This work formulates multiclass Structured SVM as a two layer Neural Network. Our method can be used with deep learning. Our study is an extension of Tang[92]. We show that our multiclass Structured SVM formulations also outperform Softmax by a small margin.

3.2 Multiclass Classification with Structured SVM

To formulate a problem under SSVM framework, one needs to delineate the feature vector $\mathbf{x}$, which describes the input quantitatively, the structured label $\mathbf{y}$, the structured weights $\mathbf{w}$, and the graph structure $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where $\mathbf{V}$ denotes the set of all nodes, and $\mathbf{E}$ denotes the set of all edges. Once those are delineated, the next step is to theorize the set $\mathbf{Y}$, the set of all possible values $\mathbf{y}$ could take. The domain of all possible $\mathbf{w}$ is usually, but not always, $\mathbf{R}^{|\mathbf{w}|}$. For example, in our HPE problem, some of the dimensions of $\mathbf{w}$ are restricted to be positive numbers. To prepare for SSVM framework, one needs to specify 2 functions and 2 algorithms. They are:

1. Joint Feature, Label function, or what I have termed the feature at label function $\Phi\left(\mathbf{x},\mathbf{y}\right)$. This function must be specified in such a way that the next 2 algorithms are correctly specified. This function output is usually a vector value.

2. Loss function. This function provides the discrepancy measurement between the guessing answer $\hat{\mathbf{y}}$ to the label $\mathbf{y}$. The function is of the form $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$. This function output is a scalar.

3. The prediction inference algorithm, also known as the test inference algorithm. This algorithm's objective is to find optimal solution $\hat{y}$ from $\mathbf{Y}$ which satisfies $\hat{y} = \arg\max_{\mathbf{y}\in\mathbf{Y}} \mathbf{w}\cdot\mathbf{\Phi_a}\left(\mathbf{x_i},\mathbf{y_i}\right)$. where $\hat{\mathbf{y}}$ is the prediction result, the quantitative answer to the problem, and $\Phi\left(\mathbf{x},\mathbf{y}\right)$ is defined in Joint Feature, Label function.

4. The Loss Augmented Inference Algorithm. This algorithm's objective is to find the optimal solution $\hat{\mathbf{y}}$ from $\mathbf{Y}$ which satisfies $\hat{\mathbf{y}} = \arg\max_{\hat{\mathbf{y}}\in Y} \mathbf{w}\cdot\mathbf{\Phi_a}\left(x_i,\mathbf{y}\right) + \Delta\left(\mathbf{y},\mathbf{y_i}\right)$ The quantity $\hat{\mathbf{y}}$ is the *most violated constraint.* One can see a similarity between the test inference and the loss augmented inference. Even though $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$ is a scalar, both $\mathbf{y}$ and $\mathbf{y_i}$ are generally not. Since one must find this maximum over $\mathbf{Y}$, if one wishes to use exhaustive search to find $\hat{y}$, then finding the loss augmented inference with $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$ is trivial. However, if one needs to use dynamic programming, which finds $\hat{\mathbf{y}}$ through subdivision of each element of $\mathbf{y}$ to solve test inference, one must make sure that the define $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$ is also able to go through the same subdivision and memoization to find $\hat{\mathbf{y}}$ and $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$. Having explained the method of formulating the problem under SSVM framework, now let us focus on the multiclass classification problem.

Suppose we have a set of training feature and their corresponding labels of dimension m, we first define an $\mathbf{X}^{train}$ as a feature vector in $\mathbf{R}^m$. The training label is the class of the feature. We define the label as $\mathbf{y}^{train}$. There are a discrete, finite set of values

$\mathbf{y}$ can take. That is all-possible-class-number, or, in other words, number of classes K. Therefore the set $\mathbf{Y}$ is the set of $\{1, 2, ..., K\}$. We wish to find, for a set of unseen features $[\mathbf{X^{test}}]$ a set of predictions $\left[\mathbf{y}^{pred}\right]$, such that $\left[\mathbf{y}^{pred}\right]$ correctly predicts $[\mathbf{y}^{test}]$ with the highest accuracy possible. A $\mathbf{y}_i^{pred}$ is called correctly predicting $\mathbf{y}_i^{test}$ at ith sample if $\mathbf{y}_i^{pred} = \mathbf{y}_i^{test}$.

One can see that under the multiclass problem formulation, the feature $\mathbf{x}$, label y and the set of all possible y, $\mathbf{Y}$, are now defined. If there are not many classes, i.e K is a small number, then $\mathbf{Y}$ is small. In this case, what is our $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$? the $\mathbf{V}$ is one node structure, the $\mathbf{E}$ is an empty set. In one-vs-all SVM, there are total K hyperplanes. Suppose these K hyperplanes' weights are trained, how can one classify the test feature $\mathbf{X}_{unseen}^{test}$?

In 1-vs-all, one classifies the test with the highest score. Therefore, y_{unseen}^{test} chooses number $1, 2, 3, 4$ from the subscript of $\{\mathbf{w}_1^T\mathbf{X}_{unseen}^{test}, \mathbf{w}_2^T\mathbf{X}_{unseen}^{test}, \mathbf{w}_3^T\mathbf{X}_{unseen}^{test}, \mathbf{w}_4^T\mathbf{X}_{unseen}^{test}\}$. We can rewrite the above procedure as $y_{unseen}^{test} = argmax_{j\in\{1,2,3,4\}}, \mathbf{w}_j^T\mathbf{X}_{unseen}^{test}$

One can see that the above sorting is of a similar form to the test algorithm eq. 2.15 required by SSVM framework. To make 1-vs-all SVM testing match the form of SSVM test inference,[22] define the weights and the joint feature-label function as follows: $\mathbf{w} = \begin{pmatrix} \leftarrow \mathbf{w_1} \rightarrow \\ \leftarrow \mathbf{w_2} \rightarrow \\ \vdots \\ \leftarrow \mathbf{w_k} \rightarrow \end{pmatrix}^T$, $\Phi\left(\mathbf{x}, j\right) = \begin{array}{c} \begin{array}{cccccc} column & 1^{st} & 2^{nd} & \cdots & j^{th} & \cdots & K^{th} \end{array} \\ \begin{pmatrix} \uparrow & \uparrow & \cdots & \uparrow & \cdots & \uparrow \\ 0 & 0 & \cdots & \mathbf{x} & \cdots & 0 \\ \downarrow & \downarrow & \cdots & \downarrow & \cdots & \downarrow \end{pmatrix} \end{array}$ by defining in this form, $\mathbf{w} \cdot \mathbf{\Phi_a}\left(x_i, y_i\right) = \mathbf{w}_j^T\mathbf{X}$. Hence the 1-vs-all test inference $y^{test} = argmax_{j\in\{1,2,3,4\}}, \mathbf{w}_j^T\mathbf{X}^{test}$ is now written fully in the form of SSVM test inference.

$$\hat{y} = \arg\max_{y\in Y} \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, y_i\right) \tag{3.1}$$

Note that in [100], $\Phi\left(\mathbf{x}, y\right)$ defined this way can be written as

$$\Phi\left(\mathbf{x}, y\right) = \mathbf{x} \otimes \Lambda^{c}\left(y\right) \tag{3.2}$$

, where $\Lambda^{c}\left(y\right) = \begin{pmatrix} \Delta\left(1, y\right) \\ \vdots \\ \Delta\left(K, y\right) \end{pmatrix}$. The Loss Function $\Delta\left(a, b\right)$ is defined as

$$\Delta\left(a, b\right) = \begin{cases} 1 & a \neq b \\ 0 & otherwise \end{cases}$$

, and $\otimes$ is the direct tensor product, ie

$$\otimes : \mathbf{R}^{D} \times \mathbf{R}^{K} \rightarrow \mathbf{R}^{D \cdot K}, \left(a \otimes b\right)_{i+(j-1)D} = a_{i} \cdot b_{j}$$

. The loss function of multiclass SSVM is just normal 0-1 loss $\Delta\left(y, y_{i}\right)$. To find y_{pred} from the test inference equation 3.1, one can simply use exhaustive search through all possible values of j.

The loss augmented inference

$$\hat{y} = \arg\max_{y \in Y} \mathbf{w} \cdot \mathbf{\Phi_{a}}\left(\mathbf{x_{i}}, y\right) + \Delta\left(y, y_{i}\right) \tag{3.3}$$

can also use exhaustive search for trivial reasons.

3.2.1 SSVM as two-layer Neural Network

$$\Phi_{r}\left(\mathbf{x}, \hat{y}\right) = \mathbf{W_{hk}^{T}} \Phi\left(\mathbf{x}, y\right) \tag{3.4}$$

$$L_{i} = \max_{y \in \mathrm{Y}} \Delta\left(y, y_{i}\right) + \Phi_{r}\left(\mathbf{x_{i}}, \hat{y}\right) - \Phi_{r}\left(\mathbf{x_{i}}, y_{i}\right) \tag{3.5}$$

The term $\mathbf{\Phi_r}\left(\mathbf{x_i},\mathbf{y}\right)-\mathbf{\Phi_r}\left(\mathbf{x_i},\mathbf{y_i}\right)$ can be seen in Neural Network sense as picking two scalars from the matrix of lower layer, and do subtraction. This eq.3.5 defined the upper layer of SSVM as a two-layer Neural Network. We perform exhaustive search over K possible classes, the value of $\hat{y}$ in eq.3.3.

Here is our back propagation rules for our two-layer Neural Network. The Gradient of Top layer, the Loss Augmented Inference Layer, is

The gradient of objective function w.r.t respond map layer, $\mathbf{\Phi_r}\left(\mathbf{x_{iL}}\right)$, is

$$\frac{\partial L_i}{\partial \mathbf{\Phi_r}}=\delta\left(y_v,\hat{y}_v\right)-\delta\left(y_v,y_{iv}\right),\forall y\in \mathrm{Y},\forall v\in \mathrm{V} \tag{3.6}$$

,where $\delta\left(a,b\right)=1$, if $a=b$, and $\delta\left(a,b\right)=0$, otherwise. This can be realized by creating a blank matrix of the size $\mathbf{\Phi_r}\left(\mathbf{x},\mathbf{y}\right)$, and then set $+1$ at position $\bar{y}_v$, and -1 at the position y_{iv}, and if it happens that some $\bar{y}_v=y_{iv}$, then set 0 at that position.

The above gradient defines The Loss Augmented Inference Layer. The gradient w.r.t weights of respond map layer is

$$\frac{\partial L_i}{\partial \mathbf{W_{hk}}}=\frac{\partial L_i}{\partial \mathbf{\Phi_r}}\frac{\partial \mathbf{\Phi_r}}{\partial \mathbf{W_{hk}}} \tag{3.7}$$

but from eq. 3.4 $\Phi_r\left(\mathbf{x},\hat{y}\right)=\mathbf{W_{hk}^T}\mathbf{x}$, therefore $\frac{\partial \mathbf{\Phi_r}}{\partial \mathbf{W_{hk}}}=\mathbf{x}$.

$$\frac{\partial L_i}{\partial \mathbf{W_{hk}}}=\mathbf{x}\otimes\Lambda^c\left(\hat{y}\right)-\mathbf{x}\otimes\Lambda^c\left(y_i\right) \tag{3.8}$$

Since we defined our multiclass SSVM as 2 layers of Neural Networks, we defined the back propagation rule with respect to the lower layer of the 2 layers. We can always back propagate to other types of Neural Network layers, and to any number of lower layers

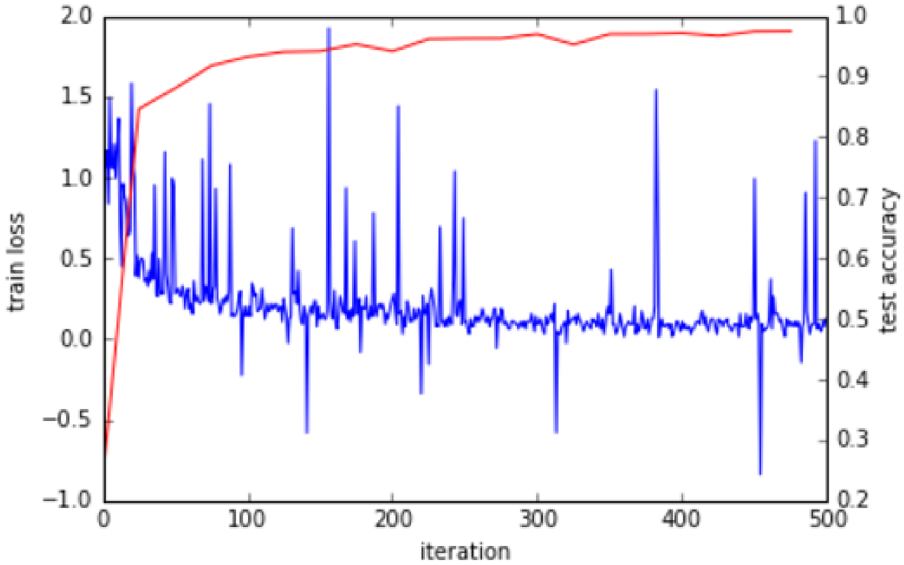


Fig. 3.1 Cost function (blue line), and test accuracy (red line) as a function of training iterations of our Multiclass SSVM Deep Convolutional Neural Network on MNIST dataset. Show first 500 iterations.

3.3 Experiment

A test is done with standard MNIST dataset[57]. Our result is comparable to previous SVM based method, and gives a better result than standard Softmax classifier. Figure 3.1 shows that the gradient of cost function is nicely reduced, as in the case of normal neural networks.

3.3.1 Neural Network Structure

Caffe library use a slightly modified Lenet[56] as its Deep Convolutional Neural Network with Softmax[8] as a multiclass classifier.The Structure of this CNN is shown in Figure 3.2. There are output neurons of kernel size 5x5 on the first Convolutional layer with stride 1 and a bias, follow by Max-pooling layer kernel size 2x2 with stride 2, follow by 50 neurons of Convolutional layer with kernel size 5x5 with stride 1 and a bias, follow by Max-pooling layer kernel size 2x2 with stride 2, follow by 500 neurons of fully

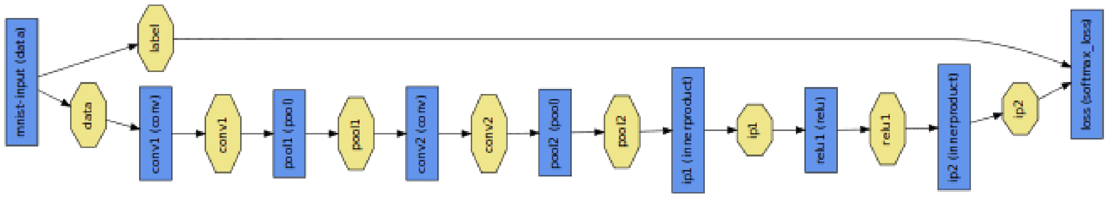


Fig. 3.2 The Structure of Convolutional Neural Network, which is slightly modified version of Lenet by Caffe library.

connect layer and a bias, follow by Rectified Linear layer, follow by Softmax classifier of 10 classes. In our setting, we used the same structure, and remove the Softmax and then add our Multiclass Structured SVM with 10 classes as the output classifier. We then compare the classification result between our setting, and Caffe library setting. MNIST dataset[57] has the image intensity value ranging from 0-255. We normalize the dataset so that the intensity range is from 0-1. Then we directly feed the image to Caffe's Lenet with SSVM multiclass classifier. We use the same learning rate for comparison. In both networks, we use momentum of 0.0005, base learning rate of 0.01, inverse alpha function learing rate with alpha of 0.0001, and power of 0.75. By this setting, we have every parameters the same value for the two comparing Convolutional Neural network, except the classifier layer. This is the equal basis for fair comparison. All weights are randomly initialized.

The MNIST dataset of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The example of MNIST dataset is shown in Figure 3.3. The goal of this dataset is to predict the class of the test set given images. The class of an image is the digit value of that image.

3.3.2 Implementation as a Caffe layer

The Loss Augmented Inference layer for multiclass classification requires forward propagation to calculate eq.3.4, and backward propagation to calculate eq.3.6. We

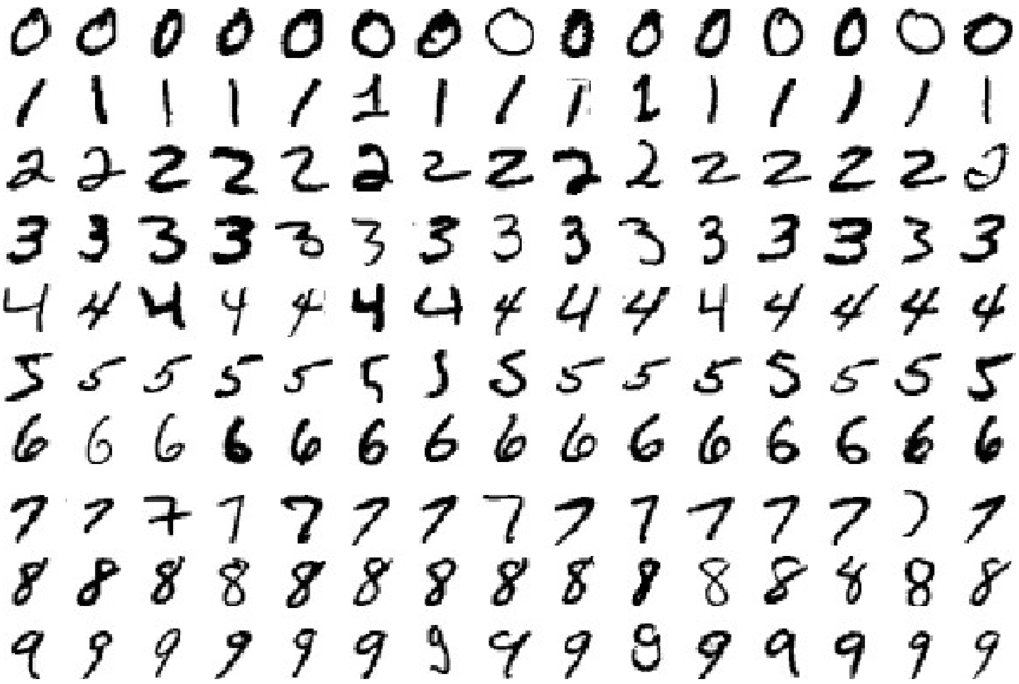


Fig. 3.3 MNIST dataset is the dataset of handwritten digits.

can use normal fully connect layer for our bottom layer calculating eq.3.4 of two layer Structured SVM. Once we back propagate the eq.3.6, lower layer back propagation is calculated as normal a modular design of Neural Network. These equations are implemented using python layer of Caffe library[49].

3.3.3 Result

Our Structured SVM Classifier has consistently higher accuracy than Softmax, using the same underlying Convolutional Neural Network structure, and learning parameters. This is shown in Table 3.1. Figure 3.4 shows the test accuracy of the two comparing networks as a function of training iterations. The red line shows the result test accuracy, the higher the better, of our Structured SVM Classifier, and the blue line shows the result test accuracy of Softmax classifier over the same underlying CNN structure. Our Structured SVM Classifier clearly consistently outperform Softmax classifier after the same number of training iterations.

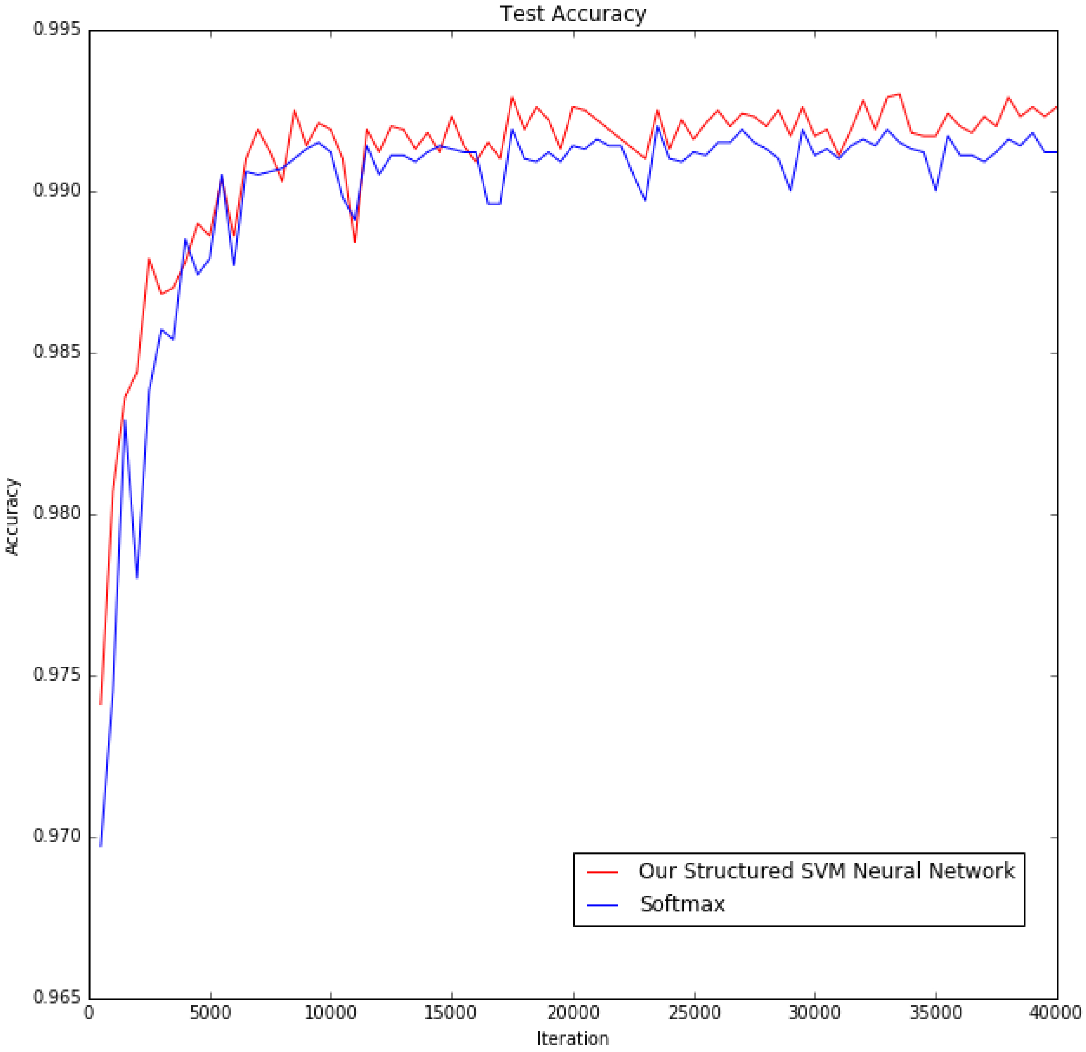


Fig. 3.4 Test Accuracy as a function of training iterations. Our Structured SVM Classifier has consistently higher accuracy than Softmax.

Table 3.1 Error of MNIST dataset classification result in percentage

Our Result	**Softmax**[49]
0.74	0.88

3.4 Conclusion

This work formulates multiclass Structured SVM as a two layer Neural Network. Our method can be used with deep learning. Our study is an extension of Tang[92]. With all parameter the same, we show that other multiclass SVM formulations also outperform Softmax by a small margin.

Chapter 4

Structured SVM as two layer neural network on Human Pose Estimation

4.1 Introduction

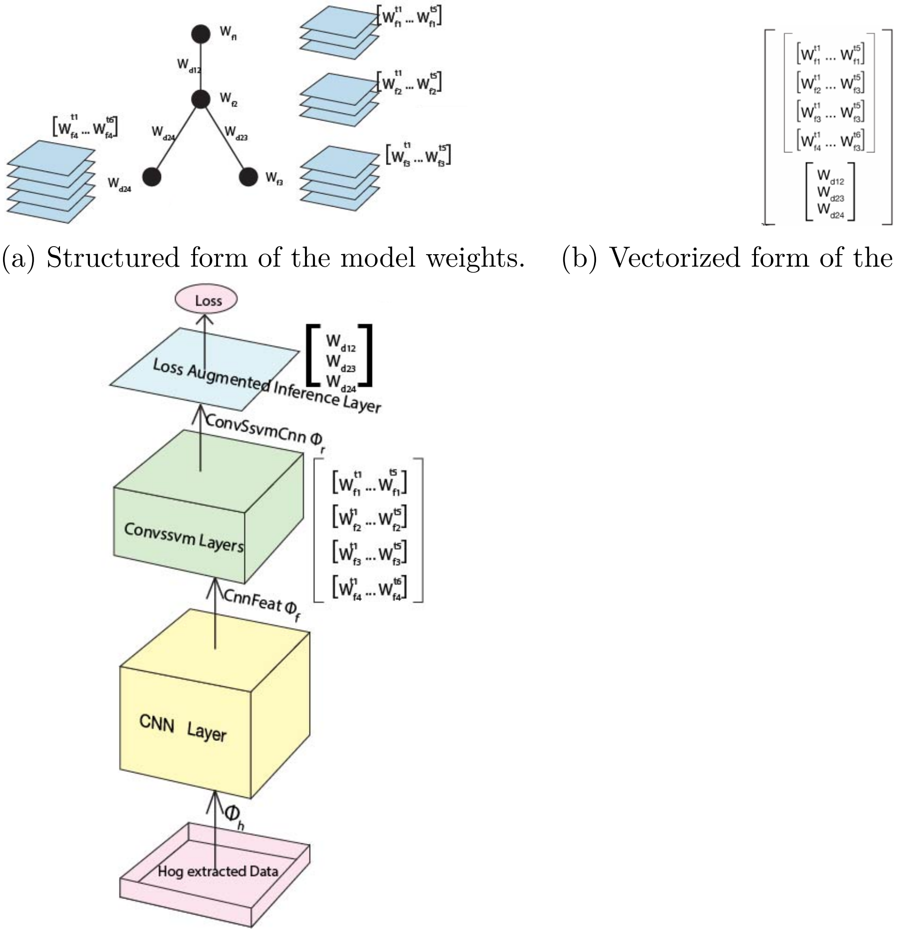
This chapter describes the main work of this thesis. A description of a DPM problem formulation in Section 4.2, and Detection Inference is in Section 4.2.2. Section 4.2.3 provides the reader with the understanding of the Subgradient method before our extension. Section 4.2.4 extends the Subgradient method so it becomes a two-layer neural network of Human Pose Estimation, which is the main contribution of this thesis.

Our proposed method jointly learns structural model and appearance model. In pictorial structure, one usually has three models combine into one single large model. They are coocurrance model, deformable model, and appearance model. The first two are called structural model in our abstract. We define bias system for coocurrance model. The deformable model refers to spring relation between position of parts. The appearance model refers to the DPM learnable filters. Our proposed method sees all model parameters as neural network parameters and jointly learns them using stochastic subgradient descent simultanously. Unlike [113] where the appearance model is learned prior to structural training, our method can learn all parameters from random initialization. We noticed that the DPM unary filters work very similarly to convolutional operation during DPM inference. Thus, we designed the DPM unary filters, which varies over human parts and human part types, as convolution layer of CNN. This DPM filters defines the appearance model's weight since they give the feature similarity score. We designed the cooccurance model's weight, bias weights, as parameters defined on Loss Augmented Inference Layer. We designed the DPM pairwise weights, the deformable weights, or simply the spring terms weights, as other type of parameters defined on Loss Augmented Inference Layer. These designs are shown in Figure 4.1. In this section, we show how Structured SVM on DPM can be implemented as two-layer neural networks, the first of which is convolutional layer, and the last of which is the Loss Augmented Inference layer.

4.2 DPM Problem formulation

Human Pose Estimation is the problem of estimating human joint positions. The joints are said to be correctly estimated if the prediction error in pixel difference is less than a certain threshold. Human Pose Estimation problem presents the following challenges. 1). Human Size can be small or large depending on how near or far from the camera the human image appears. 2) Body parts can have multiple appearances. For example, a hand can be a fist or a palm, a limb can be vertical or horizontal. 3) Human pictures are taken in 3D. Therefore, there may be many 2D pose appearances on the same 3D pose, of which there are many 3D poses. To meet these challenges, the following measures had been implemented in several previous works. 1. Multiscale human detection. This is sometime called image pyramid or feature pyramid. An image of feature $\mathbf{x}$ is scaled to all layer values $l \in \mathrm{L}$, where $\mathrm{L} = \{1, ..., l_n\}$. 2. Mixture of part types. To address different appearnaces of each human part, each human part model is designed so that they consist of multiple different part types. Body parts from training images are clustered into part type based on their relative joint positions in image coordinates with respect to its neightboring joints. The underlying assumption of this clustering is that the same group of relative joint positions with respect to its neighboring joints would have a similar appearance. 3. Co-occurance model. This model address how two neighboring parts are co-occurred with system of biases. Each type of mixtures of neighboring nodes has a bias associated with it. These measures are incoperated into a well known pictorial structure model where their edges are quantified under the assumption that the energy of placing parts is varies solely quadratically on relative distance like the energy of stretching or compressing springs from their anchor positions with respect to their parent nodes. These are models we inherited directly from [113]. Thus, we have three models combined into one single large model. They are coocurrance model, deformable model, and appearance model. The first two are called structural models in our abstract.

We propose a new method to learn the aforementioned models. We propose to learn all models, and all their parameters, the way neural network learn their parameters: by jointly learning all of them using stochastic subgradient descent. Unlike [113] where they learn the appearance model prior to structural training, our method can learn all parameters from random initialization. We noticed that the DPM unary filters worked equally to carry out a convolutional operation during DPM inference. During DPM inference, each unary filter is performing "sliding dot product" between feature matrices, and weight matrices. This operation is equal to the weights of convolutional layer of CNN acting on their input matrics. Thus, we designed the DPM unary


Loss
Loss Augmented Inference Layer
ConvSsvmCnn Φ_r
Convssvm Layers
CnnFeat Φ_f
CNN Layer
Φ_h
Hog extracted Data
W_{d12}
W_{d23}
W_{d24}
$[W_{f1}^{t1} \dots W_{f1}^{t5}]$
$[W_{f2}^{t1} \dots W_{f2}^{t5}]$
$[W_{f3}^{t1} \dots W_{f3}^{t5}]$
$[W_{f4}^{t1} \dots W_{f4}^{t6}]$
W_{f1}
W_{f2}
W_{f3}

(a) Structured form of the model weights.

(b) Vectorized form of the model weights.

(c) Neural Network form of the model weights on 2 topmost layers.

Fig. 4.1 Proposed method formulate Structured SVM two layer neural network, which are the two topmost (blue and green) layers of c). The appearance model weights are the bottom Convssvm Layer (green layer), which are a normal convolutional layer. The Loss Augmented Inference on the top layer (blue layer) has pairwise weights, the deformable model's weight, and bias weights, the cooccurance model's weight, which can be seen as structural weights. The deformable weights,$\mathbf{w_{ef_d}}$, are shown as weights between the edges in a), shown as subvector in concatenated vector of weights in b), and shown as Loss Augmented Inference weights in c). The appearance model weights,$\mathbf{w_f^t}$, are shown as weights of nodes in a), shown as subvector in concatenated vector of weights in b), and shown as Convssvm layer in c).

filters, which varies over human parts and human part types, as convolution layer of CNN. This DPM filters defines the appearance model's weight since they give the feature similarity scores. We designed the cooccurance model's weight, bias weights, as parameters defined on Loss Augmented Inference Layer. We designed the DPM pairwise weights, the deformable weights, or simple the spring terms weights, as other types of parameters defined on Loss Augmented Inference Layer. These designs are shown in Figure 4.1. In this section, we show how Structured SVM on DPM can be implemented as two-layer neural networks, the first of which is convolutional layer (convssvm layer in Figure4.1) and the last of which is the Loss Augmented Inference layer (Loss Augmented Inference layer in Figure 4.1). By transforming Structured form of the model into neural network form of the model, our proposed method can jointly learn structural model and appearance model, and can back propagate the error to further learn the underlying appearance model feature extractor learnable parameters (CNN layer in Figure4.1). In short, our proposed method translates the Structured SVM model into neural network model, thus inheriting the neural network innate ability to back propagate the error to lower layer, while precisely calculating the Structured SVM loss, and still learn Structured SVM in the original way with subgradient based method.

4.2.1 DPM Problem formulation

Our approach starts with the formulation of DPM as an instance of Structured SVM learning. As [79] shows, the SSVM can be learned by subgradient descent. Let us start with detection problem formulation

Let $\mathbf{x_i}$ represent a matrix of RGB value of its training data, and $\mathbf{y_i}$ represent the i^{th} training bounding boxes label of the form (Top Left Column, Top Left Row, Bottom Right Column, Bottom Right Row) denotes $[x_1, y_1, x_2, y_2]$. Each row of $\mathbf{y_i}$ represents each bounding box for each part similar to [113]. Therefore $\mathbf{y_i}$ is a matrix of $numparts \times 4$. The bold characters indicate that they are either vectors or matrices. We first scale $\mathbf{x_i}$ to multiple scales. To denote this scaling, let $\mathrm{L} = \{1, ..., l_n\}$ denote the set of all scaling levels. Let $l \in \mathrm{L}$ denote l^{th} layer. By scaling image x_i to multiple scales, we create an image pyramid of all $l \in$ L. The image pyramids are simply multiple RGB images, each differing by their scales. Each of these difference size images are pasted onto a zero image matrix. This creates a 4 dimensional matrix of size $width \times hight \times colorchannels \times numberoflevels$. The image pyramid, denoted as $\mathbf{x_{iL}}$, is then fed to HOG feature extraction, denoted as $\mathbf{\Phi_h}\left(\mathbf{x_{iL}}\right)$. Then the HOG-extracted feature is fed to a CNN as deep pyramid features. We call this CNN: $CnnFeat$. The result of $CnnFeat$ of $\mathbf{x_{iL}}$ is denoted as $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}\right)$

4.2.2 Model and Detection Inferences

Our model based on [113], where there are 3 major submodels. These submodels are delineated as follows.

Appearance Model

Appearance model is the individual filters for individual type of parts which model designers wish to model. For example, we need head filters, body filters, etc. Since our image representation usually has many channels, each of these models is represented by a matrix of filter size times channels. A typical value would be 5x5x32, or 5x5x64, for a filter of size 5x5, with 32 and 64 channels respectively. By doing the dot product of the filter and the feature of the same size, one obtains a particular score for such a feature. The domain of these filters are in $\mathbf{R}^{S\times C}$, where S is the filter size, and C is the number of channels. For each part and each type of part, there is an appearance model filter associated. The appearance model is the appearance filter. The similarity score created by a filter $\mathbf{w_f^t}$ is

$$score_{appearance}\left(y\right) = \mathbf{w_f^t} \cdot \mathbf{\Phi_f}\left(\mathbf{x_L}, y\right) \tag{4.1}$$

Co-occurrence Model

Suppose there are m mixture types of a part, and n mixture types of a neighbor part, the total bias among these two parts are $m \times n$. This model gives the sum of local and pairwise scores. For a parent node i, and a child node j the score of co-occurrence ij is

$$score_{cooccurance}\left(t_i, t_j\right) = b_{ij}^{t_i t_j} \tag{4.2}$$

This can be thought of as bias to favor some particular local types, and pairwise relationship among parent and child types. For example, if $b_{34}^{t_1 t_2}$ has a high value, this means that the parent part number 3 type 1 is likely to connect to child part number 4 type 2.

Deformable Model

From each parent type t_i to child part t_j, we have anchor positions of the child w.r.t parent so there is a total of $t_i \times t_j$ anchors from parent to child. The anchor positions are trained to model by simply taking the average of each of all possible types of connections. Since the anchor positions must be available prior to Structured SVM training, the part types were calculated by simple K-mean clustering of each type appearence to create different types of articulation. We employ a mixture of components to solve many types of possible part appearances as does [113]. Let $p \in \mathrm{P}$ be the p^{th} body part, where $\mathrm{P} = \{1, ..., p_n\}$ is the set of all parts. Let $k \in \mathrm{K}$ be k^{th} types of a particular part, where $\mathrm{K} = \{1, ..., k_n\}$ is the set of all types of a particular parts. Let K_p denote the total number of types K for a particular part p. We start with clustering training image parts $p \in \mathrm{P}$ into K_p clusters. We now define i^{th} sample's Structured SVM feature function. The Structured SVM has a DPM tree $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ feature. We first define the unary feature $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}, \mathbf{y_i}\right)$ to be $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}\right)$ evaluating at position $\mathbf{y_i}$.We then define the pair-wise features

$$\Psi_{ij} = -\begin{bmatrix} dx_{ij} & dy_{ij} & dx_{ij}^2 & dy_{ij}^2 \end{bmatrix}$$

where $dx_{ij} = x_{pi} - x_{pj} + anchor_x$, and $dy_{ij} = y_{pi} - y_{pj} + anchor_y$. We denote $\mathbf{\Psi}$ as $[\Psi_{ij}]$, $\forall ij \in$ E. Integrating unary and pairwise features of the tree G results in $\mathbf{\Phi_a} = [\mathbf{\Phi_f} \quad \mathbf{\Psi}]$,where subscription a denotes the word *all*. The score created by deformable model of an edge ij is

$$score_{defrom}\left(i,j\right) = \mathbf{w_{ij_d}} \cdot \Psi_{ij} \tag{4.3}$$

Putting submodel together

For each node and edge of the pictorial structure, we concatenate each and every bias weights, deformable weights, and appearance filter weights together into 2 types of data structure. The first is struct type according to their component, and the second one is vector type. With the vector type data structure, we create a large vector $\mathbf{w}$, the learnable parameters of Human Pose Estimation ready for Structured SVM learning.

$$score(t,y) = \sum_{i \in V} \mathbf{w_f^t} \cdot \mathbf{\Phi_f}\left(\mathbf{x_L}, \mathbf{y}\right) + \sum_{ij \in E} b_{ij}^{t_i t_j} + \mathbf{w_{ij_d}} \cdot \mathbf{\Psi_{ij}} \tag{4.4}$$

, or in matrix form,

$$score(t,y) = \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_L}, \mathbf{y}\right) \tag{4.5}$$

To find the location $\mathbf{y}$ whose value maximize the score, the above equation becomes:

$$\hat{\mathbf{y}} = \arg\max_{\hat{y} \in Y} \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_L}, \mathbf{y}\right) \tag{4.6}$$

This is the prediction function. The value $\hat{\mathbf{y}}$ is our prediction of part locations for an unseen test image.

4.2.3 Subgradient optimization of Structured SVM

In this subsection, we introduce the Structured SVM and a means to solve Structured SVM objective function. This section relies on the framework of [79], and [100]. Our work is to apply this framework of generic structured learning to Human Pose Estimation.

The purpose of Structural SVM is to produce Structural Prediction by learning the maximum margin classifier for each training data. Probabilistic Graphical Model such as Markov Random Fields (MRF), or Conditional Random Fields (CRF) can use Structured SVM during the learning phase to learn their weight parameters. Structured SVM is not an algorithm in which one can just plug the data and produce learning or classification like SVM, but a framework for which inference, loss, and feature modules are required to be plugged in first. For example, if a Structured SVM is to be applied with MRF, one must specify the MRF structure, which the Structured SVM will learn, the MRF inference algorithm, the MRF feature function, the loss function, and the loss augmented inference algorithm. The loss augmented inference algorithm is the inference algorithm with loss function. Then the Structured SVM learns the weights for which the prediction over training data is maximum. In our work, the purpose of SSVM is to learn the structural prediction function of the form eq.4.6

The SSVM has the optimization of the form [79] In this subsection, we describe how the work of [79] could be applied to our work. Application of their work to our current problem was the inspiration behind the formulation in the next subsection. To learn the model parameters of the eq.4.6, we define the objective function of SSVM, similar to [79].

$$\begin{aligned}
&\min_{w,\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{l}\xi_i \\
&s.t. \quad \forall i \quad , \forall y \in Y \setminus y_i : \quad \mathbf{w}\cdot\Phi_{ai}\left(\mathbf{x_i},\mathbf{y_i}\right) + \xi_i \geq \max_{y\in \mathrm{Y}} \mathbf{w}\cdot\Phi_{ai}\left(\mathbf{x_i},\mathbf{y}\right) + \Delta\left(y, y_i\right)
\end{aligned} \tag{4.7}$$

By minimizing the above objective function, we can learn the parameters $\mathbf{w}$ which maximize training accuracy for prediction function eq.4.6. The SSVM objective eq.4.7 can be solved by subgradient method as shown by [79]. In our context, the subgradient of the objective loss function is defined by

$$\frac{\partial Obj_i}{\partial \mathbf{w}} = \mathbf{\Phi_{ai}}\left(\mathbf{x_i}, \mathbf{\hat{y}_i}\right) - \mathbf{\Phi_{ai}}\left(\mathbf{x_i}, \mathbf{y_i}\right) \tag{4.8}$$

where Obj_i is the minimization objective function eq.4.7. For b^{th} batch many training data, the gradient is

$$\frac{\partial Obj_b}{\partial \mathbf{w}} = \frac{1}{b}\sum_{b} \mathbf{\Phi_{ab}}\left(\mathbf{x_b}, \mathbf{\hat{y_b}}\right) - \mathbf{\Phi_{ab}}\left(\mathbf{x_b}, \mathbf{y_b}\right) \tag{4.9}$$

,where $\mathbf{\hat{y}_b}$ is the most violated constraint from Loss Augmented Inference.

In our context, the Loss Augmented Inference is defined as

$$\mathbf{\hat{y}} = \arg\max_{y \in \mathrm{Y}} \Delta\left(\mathbf{y}, \mathbf{y_i}\right) + \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, \mathbf{y}\right) - \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, \mathbf{y_i}\right) \tag{4.10}$$

,where $\Delta\left(\mathbf{y}, \mathbf{y_i}\right) = 1 - \frac{Area(\mathbf{y} \cap \mathbf{y_i})}{Area(\mathbf{y} \cup \mathbf{y_i})}$ is standard one minus bounding box intersection over union loss. We then apply subgradient update as normal stochastic gradient descent.

4.2.4 SSVM as two-layer Neural Network

We extend the previously defined subgradient optimization of Structured SVM by back propagating further down to the lower layer of the neural network. This is because solving Structured SVM of the previous section can be implemented by a two-layer neural network. On the top is Loss Augmented Inference layer, while the lower one can be normal linear layer of Neural Network, in general. For our special case of DPM as CNN, the lower layer is a standard convolution layer.

To solve SSVM with subgradient optimization, one must calculate the Loss Augmented Inference $\mathbf{\hat{y}_i}$ so that the feature of the most violated constraint $\mathbf{\Phi_{ai}}\left(\mathbf{x_i}, \mathbf{\hat{y}_i}\right)$ of eq.4.8 can be calculated. We notice that the sliding dot product of $\mathbf{w_f^t}$ over all $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}\right)$ possible locations of $\hat{y}$ is actually a convolution operation similar to convolutional neural network. If we design this convolution operation as a layer of convolutional layer in CNN, then the DPM filters are convolutional filters of CNN. The *CnnFeat* topped with DPM filters are now called *ConvSsvmCnn*, with the DPM filters called *ConvSsvm* layer. Their corresponding feed forward is

$$\mathbf{\Phi_{rf}^t}\left(\mathbf{x}, \mathbf{y}\right) = \sum_{\mathbf{\bar{y}}} \mathbf{w_f^t}\left(\mathbf{\bar{y}}\right) \mathbf{\Phi_a}\left(\mathbf{x}, \mathbf{y} + \mathbf{\bar{y}}\right) \tag{4.11}$$

This eq.4.11 defined a lower layer of SSVM as a two-layer Neural Network. The quantity is actually the respond maps of $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}\right)$ convoluted by DPM unary filters $\mathbf{w_f^t}$. Let us denote $\mathbf{\Phi_r}$ for the concatenation of all $\mathbf{\Phi_{rf}^t}$. Having the *ConvSsvm* layer constructed, we now define the Loss Augmented Inference Layer in this two-layer Neural Network context. As the name implies, the Loss Augmented Inference layer performs Loss Augmented Inference and finds the slack loss of the objective function, and the $\mathbf{\hat{y}}$, the most violated constraint as the argument that maximizes the loss augmented inference objective. The Loss Augmented Inference objective function eq.4.10 can be rewritten using eq.4.4, eq.4.6, and eq.4.11, as follows:

$$\begin{aligned} L_i = \max_{y \in \mathrm{Y}} \Delta\left(y, y_i\right) + \sum_{v \in V} &\left\{\mathbf{\Phi_r}\left(\mathbf{x_{iL}}, y_v\right) - \mathbf{\Phi}_{\mathbf{r}v}\left(\mathbf{x_{iL}}, y_{vi}\right)\right\} \\ &+ \sum_{ef \in E} \left\{\mathbf{w_{ef_d}} \cdot \mathbf{\Psi_{ef}}\left(\mathbf{x_i}, \mathbf{y}\right) - \mathbf{w_{ef_d}} \cdot \mathbf{\Psi_{ef}}\left(\mathbf{x_i}, \mathbf{y_i}\right)\right\} \\ &\qquad + \sum_{ef \in E} \left\{\left(b_{ef}^{t_e t_f}\right)_{\mathbf{y}} - \left(b_{ef}^{t_e t_f}\right)_{\mathbf{y_i}}\right\} \end{aligned} \tag{4.12}$$

The term $\mathbf{\Phi_r}\left(\mathbf{x_{iL}}, \mathbf{y}\right) - \mathbf{\Phi_r}\left(\mathbf{x_{iL}}, \mathbf{y_i}\right)$ can be seen in the Neural Network sense as picking two scalars from the matrix of lower layer, and performing subtraction. This eq.4.12 defined the upper layer of SSVM as two-layer Neural Network. From here, one can see that $\mathbf{w}$ in eq.4.6 is now divided into two difference layers. The weights of Appearance Model eq.4.1 is on the bottom convolutional layer. The weights of co-occurrence and deformable models eq.4.2, eq.4.3 are on the top Loss Augmented Inference layer. Figure 4.1 visualizes the outcome of these two equations, eq.4.11, and eq.4.12.

The following are our back propagation rules for our two-layer Neural Network. The Gradient of Top layer, the Loss Augmented Inference Layer, is

$$\frac{\partial L_b}{\partial \mathbf{w}} = \frac{1}{b} \sum_{b} \left\{\mathbf{\Psi_b}\left(\mathbf{x_b}, \hat{\mathbf{y_b}}\right) - \mathbf{\Psi_b}\left(\mathbf{x_b}, \mathbf{y_b}\right) + \left[\delta\left(\hat{y}^{t_i t_j}\right)\right]_b - \left[\delta\left(y_i^{t_i t_j}\right)\right]_b\right\} \tag{4.13}$$

,where $[\delta(a)]$ is a vector whose elements are all zeros except the element position at a The gradient of objective function w.r.t respond map layer, $\mathbf{\Phi_r}\left(\mathbf{x_{iL}}\right)$, is

$$\frac{\partial L_i}{\partial \mathbf{\Phi_r}} = \delta\left(y_v, \hat{y}_v\right) - \delta\left(y_v, y_{iv}\right), \forall y \in \mathrm{Y}, \forall v \in \mathrm{V} \tag{4.14}$$

,where $\delta(a, b) = 1$, if $a = b$, and $\delta(a, b) = 0$, otherwise. This can be realized by creating a zero matrix of the size $\mathbf{\Phi_r}(\mathbf{x}, \mathbf{y})$, and then set $+1$ at position $\bar{y}_v$, and -1 at position y_{iv}, and if it happens that some $\bar{y}_v = y_{iv}$, then set 0 at that position.

The above two gradients define The Loss Augmented Inference Layer. For the back propagation rule of *ConvSsvm* layers, which define the appearance layer of pictorial structure, we can use a normal back propagation rule of Convolution layer of Convolutional Neural Network.

4.2.5 Solving Inferences with Max-Sum Algorithm

To solve $\hat{\mathbf{y}}$ whose configuration maximizes the eq.4.12, we can use standard Max-Sum algorithm.The max-sum algorithm seeks to find the solution of the combinatorial optimization of the form,

$$\hat{L} = \arg\max_{L} \sum_{i \in V} m_i\left(l_i\right) + \sum_{ij \in E} g\left(l_i, l_j\right) \tag{4.15}$$

,under Graph $G = \{V, E\}$. The Objective of the combinatorial optimization ,eq.4.12, is

$$\hat{\mathbf{y}} = \arg\max_{y \in \mathrm{Y}} \sum_{v \in V} \{\mathbf{\Phi_r}\left(\mathbf{x_{iL}}, y_v\right) + \Delta\left(y_v, y_{iv}\right)\} + \sum_{ef \in E} \{\mathbf{w_{ef_d}} \cdot \mathbf{\Psi_{ef}}\left(\mathbf{x_i}, \mathbf{y}\right) + \left(b_{ef}^{t_e t_f}\right)_{\mathbf{y}}\} \tag{4.16}$$

. This combinatorial objective is solved by Max-sum algorithm. Thus, the max-sum algorithm is performed on the topmost layer of our neural network. The objective of combinatorial optimization is solved by message passing. The message for v^{th} node is

$$score_v\left(y_v\right) = \mathbf{\Phi_r}\left(\mathbf{x_{iL}}, y_v\right) + \Delta\left(y_v, y_{iv}\right) + \sum_{k \in kids(v)} \mathbf{m_k}\left(y_v\right) \tag{4.17}$$

$$\mathbf{m_e}\left(t_f, l_f\right) = \max_{t_e}\left(b_{ef}^{t_e t_f} + \max_{l_e}\left[score_e\left(t_e, t_f\right) + \mathbf{w_{ef}} \cdot \psi\left(l_e - l_f\right)\right]\right) \tag{4.18}$$

Since there are a total of $L \times T$ possible prediction locations. Each message has the calculation $O\left(L^2 T^2\right)$ and for all K parts, the total calculation of the Max-Sum Algorithm is $O\left(K L^2 T^2\right)$. However, we use max-convolution distance transform [33] , the time complexity of this Max-Sum Algorithm reduces to $O\left(KLT\right)$.

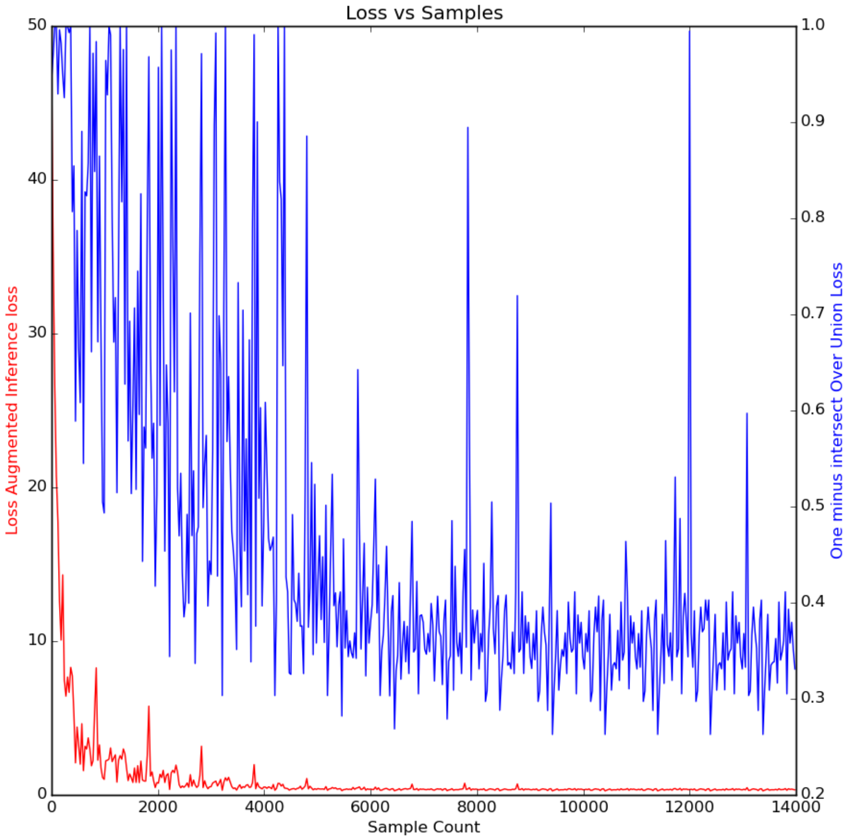


Fig. 4.2 Slack Loss, One minus Intersect over Union loss ,$\Delta\left(\hat{\mathbf{y}}, \mathbf{y_i}\right)$, versus samples.

4.3 Experiment

We train our Structured SVM the same way one trains a neural network: Stochastic Gradient Descent. Our network architecture could be a normal CNN connect with newly formulated structured SVM neural network as the last two layers. We could also simply connect the data layer with the newly formulated structured SVM neural network as the last two layers. We implement the Loss Augmented Inference as a layer of Caffe Library[49].

4.3.1 Data Preparation

PARSE dataset consists of 100 positive training samples, and 205 positive testing samples. Every image of this dataset is a full body human usually with a sports theme. Every sample has its corresponding human joint position as a label. Some of the human parts are occluded in the image, yet the human guessing joint positions are provided. There are a total of 14 joints labeled for each sample. This include head, torso, left arm, right arm, left limb, right limb. The size of the images range from 110-450 x 110-450. There are sometimes 1 or 2 full human bodies in the image. Human body size in the images also vary. The background usually contains no humans, except for small heads of audience members on a sport screen. Human Poses in the dataset vary from sitting to standing, and the limbs and arms can be doing martial arts or gymnastics.

Dataset is not yet ready for our algorithm to work with. Preprocessing is as follows. The training data is mirror-flipped, and so are the training labels, and then added to the original training data. Thus we have a double of our training data. We then find the mid point of each of the two joints, creating a total of 26 joints and mid-joint points. We change these points into boxes, with box size calculated by appropriate averaging of the length between joints among the training data. At this stage, our label to training algorithm is created as (x_1, y_1, x_2, y_2) for each box. There are 26 boxes for each training label. This is our $\mathbf{y}$ on image space.

The training image is then passed through HOG pyramid feature. In this process, a training image is resized to become multiple images of different sizes on the same picture. This is called a pyramid of images. The image pyramid is the HOG extracted to become a pyramid of features, which are then patched to a zero matrix, creating $\mathbf{\Phi_h}\left(\mathbf{x_{iL}}\right)$. Thus the matrix outside image boundaries has the value 0. This is shown in Figure 4.3. These layers represent multiscale features of the image, ranging from small to large. The brown rectangle inside the larger rectangle shows the real data from HOG features. The blue area of the larger rectangle are all zeros. Note that the brown rectangle is of a different size in each layer due to multiscale features. Here the upper left of Figure 4.3 shows a batch of pyramid of features before feeding to Neural Network. Typically, the feature may have a maximum size of about 150x150x32, which means the feature size is 150x150 with 32 channels. The smallest size is usually about 30x30x32. Typically, there are 5-20 levels of a pyramid. A batch is usually composed of 5 training images. The aforementioned feature pyramid of all levels and batch sizes are then patched to zero matrix with a size of 150x150x32x (number of levels x number of image per batch), a 4 dimensional matrix.

The label is added with feature pyramid parameters creating a practical label. This practical label, together with batch data, are then converted to Lightning Memory-Mapped Database (LMDB) so that Caffe GPU library could more efficiently do data handling.

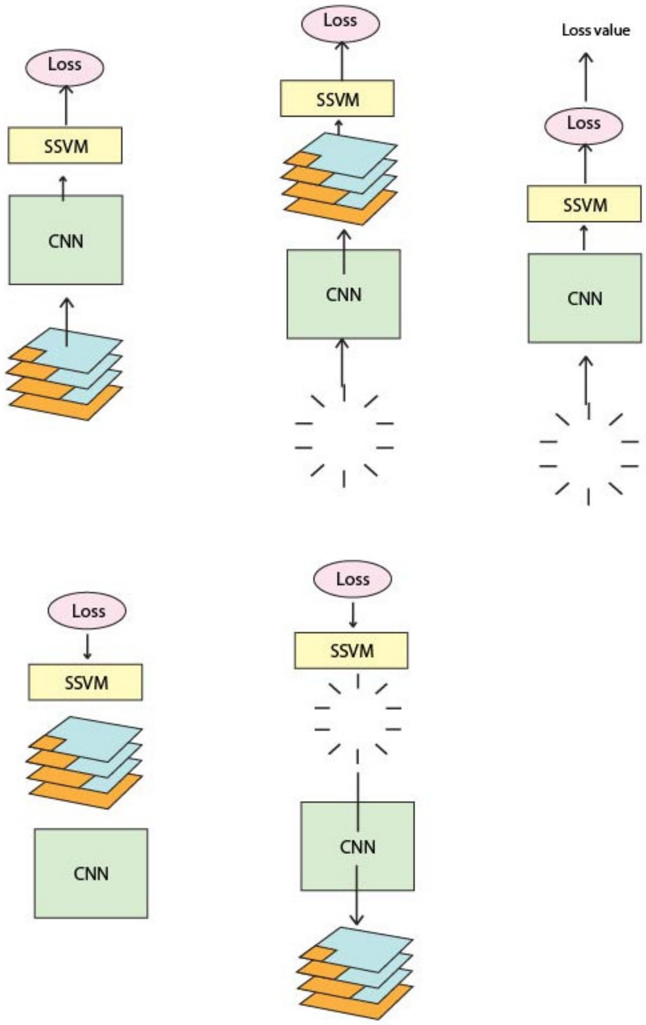


Fig. 4.3 Feature Pyramid is feeded to Structured SVM neural network as many layers.

4.3.2 Neural Network Structure

HOG-Conv-SSVM

The neural network structure is as follows: Data layer - Convolution layer $CnnFeat$ - Convolution layer $ConvSsvm$ - Loss Augmented Inference layer. The reason we put Convolutional Layer in the middle is to show that our method can learn the middle deep learning feature extractor parameters. We start our weight initialization with random initialization. Our mixture model for each node is as follows: $M_{ixture} = \{5,5,5,6,6,6,6,5,5,5,5,5,5,5,5,6,6,6,6,5,5,5,5,5,5,5\}$. This means that the first node (head node) has 5 different mixture types, the second node has 5 different mixture types, and so on. There are a total of 138 $\sum_i M_{ixture}$ mixtures. The size of $ConvSsvm$ layer is $5 \times 5 \times 256 \times 138 = 883200$, where 138 is $\sum_i M_{ixture}$. The Loss Augmented Inference has 0 weight elements of $\forall v \in \mathrm{V}, b_v^{t_v}$, and $1 + \sum_i M_{ixture}\,(i) \times M_{ixture}\,(i-1) = 702$ weight elements of $\forall vq \in \mathrm{E}, b_{vq}^{t_v t_q}$, and $133 \times 4 = 532$ weight elements of $\mathbf{w_{deform}}$. Therefore the Loss Augmented Inference layer has a total of 1234 weight elements. Since we set the size of the kernel of $CnnFeat$ to be $2 \times 2 \times 32$.The $CnnFeat$ has a total of $2 \times 2 \times 32 \times 256 = 32768$ weight elements. Therefore our system has a total weight of $883200 + 1234 + 32768 = 917202$ elements. Our batch size is set to 50, and the feature size $\mathbf{\Phi_h}$ is set to 140x140. The data layer has a size of $50 \times 32 \times 140 \times 140 = 31360000$ elements. $CnnFeat$ layer has the size of $50 \times 256 \times 139 \times 139 = 247308800$ elements. Our $ConvSsvm$ layer has the size of $50 \times 138 \times 135 \times 135 = 125752500$ elements. In total, we need 404421300 elements to store our multilayer neural network data. The total requirement for GPU memory for our system is 1284714404 bytes. Our back propagating elements is all Neural Network layers except the data layer plus all weights. There are 373,978,502 back propagating elements.

We train for 3000 iterations with learning rate of 0.005. We use the L2 regularization terms with coefficient of 0.1. We use no momentum parameter.

4.3.3 PCP evaluation

PCP measures the rate of correctly detected limbs: a limb is considered correctly detected if the distances between detected limb endpoints and ground truth limb endpoints are within half of the limb length. This was initially defined in [35]. The stricter version of PCP defined in [76] is due to a different interpretation of the definition in [35].

"according to the definition of PCP from [35] the body part is considered correct if both of its endpoints are closer to their ground truth positions than a threshold. The code in Buffy toolkit requires that the average over endpoint distances is smaller than the threshold." from [76]

There are 2 interpretations: the strict version of the intepretation as[76], and the non strict version as used in [113]. We explain only the strict version of PCP because it has been the standard of PCP for many years.

Starting with 26 bounding boxes, the PCP evaluation omitted the mid points to a total of 14 joint points, from head to both toes. The 14 joint points create a total of 9 sticks. Additional body sticks are calculated by averaging the left shoulder and right shoulder as upper body points and the left hip and right hip as lower body points. There are thus 10 sticks for evaluations. For each test image, and for each stick, the stick is said to be matched with ground truth if the ratio of difference with respect to ground truth length of both end points differs less than a certain threshold. In other words, the stick is said to be matched if $\frac{d_1}{l} \leq thresh$, and $\frac{d_2}{l} \leq thresh$. The PCP of this dataset prediction for this stick is simply the number of matches divided by the total number of sticks in the test set.

Table 4.1 Strict Percentage of Correct Point (PCP)[112] Comparison on PARSE dataset.

class	Our Result	Yang & Ramanan[113]
Head	59.0	**77.6**
Torso	75.1	**82.9**
L. arm	13.9	**35.4**
U. arm	34.6	**55.1**
L. leg	46.1	**63.9**
U. leg	55.1	**69.0**
total PCP	43.4	**60.7**

4.3.4 Implementation as a Caffe layer

Once $\mathbf{\Phi_h}\left(\mathbf{x_{iL}}\right)$ on data layer passes through the *CnnFeat* layer, the $\mathbf{\Phi_f}\left(\mathbf{x_{iL}}\right)$ is created. A further forward pass through *ConvSsvm* layer creates $\mathbf{\Phi_r}\left(\mathbf{x_{iL}}\right)$, the respond map. This process is shown in Figure 4.3. The Loss Augmented Inference layer is implemented using python layer of Caffe library[49]. The Loss Augmented Inference layer uses max-sum algorithm to calculate Objective Function value. The max-sum algorithm outputs the optimal level, the most violated constraint $\hat{\mathbf{y}}$. This output feature is the value of both $\mathbf{\Phi_a}\left(\mathbf{x_{iL}}, \hat{\mathbf{y}}\right)$, and $\mathbf{\Phi_a}\left(\mathbf{x_{iL}}, \mathbf{y_i}\right)$, The Max-marginal value which maximized eq.4.16, and the loss function $\Delta\left(\hat{\mathbf{y}}, \mathbf{y_i}\right)$. This is done through searching for each pyramid level, then finding the best max marginal score. The higher score in the new pyramid level will overwrite the previous result.

After the most violated constraint is calculated, we calculate gradient in eq. 4.13 , and eq. 4.14 exactly as written in these equations. The average gradient, and cost function over batch size is calculated as normal mini-batch training of neural network.

4.3.5 Result

In the following subsections, we train our model with PARSE dataset with HOG-Conv-SSVM architecture. We use the same model to test different datasets. During

Table 4.2 Strict Percentage of Correct Point (PCP)[112] Comparison on Fashionista Dataset.

class	Our Result	Yamagushi et al[110]
Head	56.2	**96.2**
Torso	85.4	**97.8**
L. arm	14.6	**58.6**
U. arm	35.4	**85.1**
L. leg	60.0	**87.3**
U. leg	68.1	**92.9**
total PCP	49.8	**84.1**

Table 4.3 Strict Percentage of Correct Point (PCP)[112] Comparison on LSP Dataset.

class	Our Result	Yang & Ramanan[113]
Head	50.8	**77.1**
Torso	76.1	**84.1**
L. arm	12.0	**35.9**
U. arm	25.9	**52.5**
L. leg	49.7	**65.6**
U. leg	52.6	**69.5**
total PCP	40.7	**60.8**

training on PARSE datasets, we observe the gradient descent results in Loss Augmented Inference Loss,L_i of eq.4.12, of training samples reduced as shown in Figure 4.2. The one-minus-intersect-over-union ,$\Delta(\hat{y}, y_i)$,loss is reduced to around 30%-40%. The L_i reduced nicely from value of 50 to 0.5. This shows that our gradient based method is correctly implemented.

Testing on PARRE dataset

We test our model on PARSE dataset [77].

We compare our results with the results from Yang and Ramanan [113] in Table 5.1. The total PCP (as in [112]) is used as measurement. Our results do not compete with their results. However, Figure 4.2 shows that our method is able to learn effectively.

Figure 4.4 shows the results of Pose Estimation. They are a sample of good detection over PARSE test set. The results look very promising.

Testing on Fashion dataset

The Fashionista dataset [110] is an easy dataset. There are 250 training images, and 185 testing data, each of which has 26 bounding boxes already defined by dataset. The training images are then mirror flipped to double the training data to 500 images. We use HOG-Conv-SSVM neural network with full back propagation to do the detection. We compare our results with the results from Yamaguchi et al[110] in Table 4.2.

Testing on LSP dataset

LSP dataset [50] contains 2000 posed annotated images of mostly athletes gathered from Flickr using the tags shown above. The test set is from 1001 to 2000th images. In total, there are 1000 test images. The images have been scaled such that the most prominent person is roughly 150 pixels in length. Each image has been annotated with 14 joint locations. Left and right joints are consistently labelled from a person-centric viewpoint. Forteen joint position labels are availabe with this order. Right ankle, Right knee, Right hip, Left hip, Left knee, Left ankle, Right wrist, Right elbow, Right shoulder, Left shoulder, Left elbow, Left wrist, Neck, Head top. We compare our results with the results from Yang and Ramanan [113] in Table 4.3. The total PCP (as in [112]) is used as measurement.

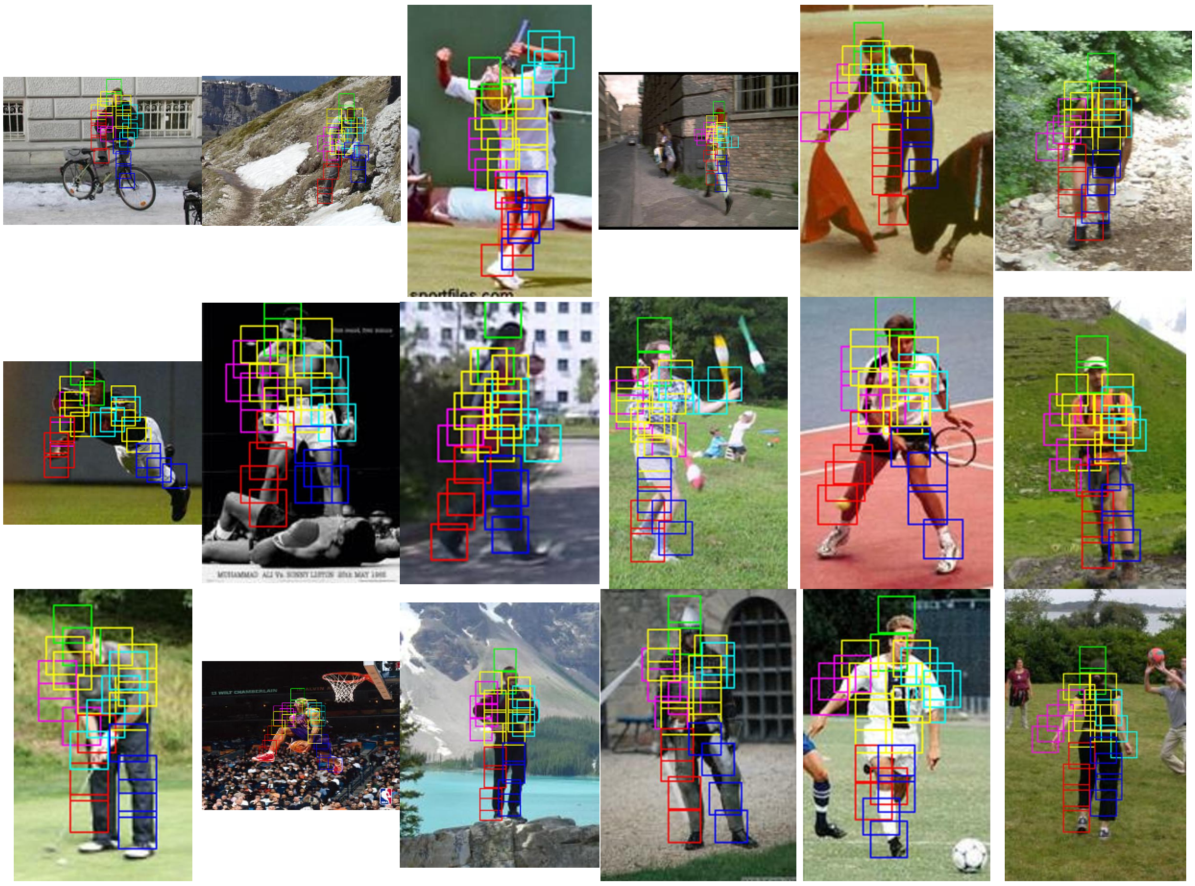


Fig. 4.4 Visualization of our result Human Pose Estimation from PARSE test dataset. The green bounding boxes are a head. The yellow boudning boxes are a torso. The cyan bounding boxes are a left arm. The blue bounding boxes are a right arm. The red bounding boxes are a left limb. The deep blue bounding boxes are a right limb.

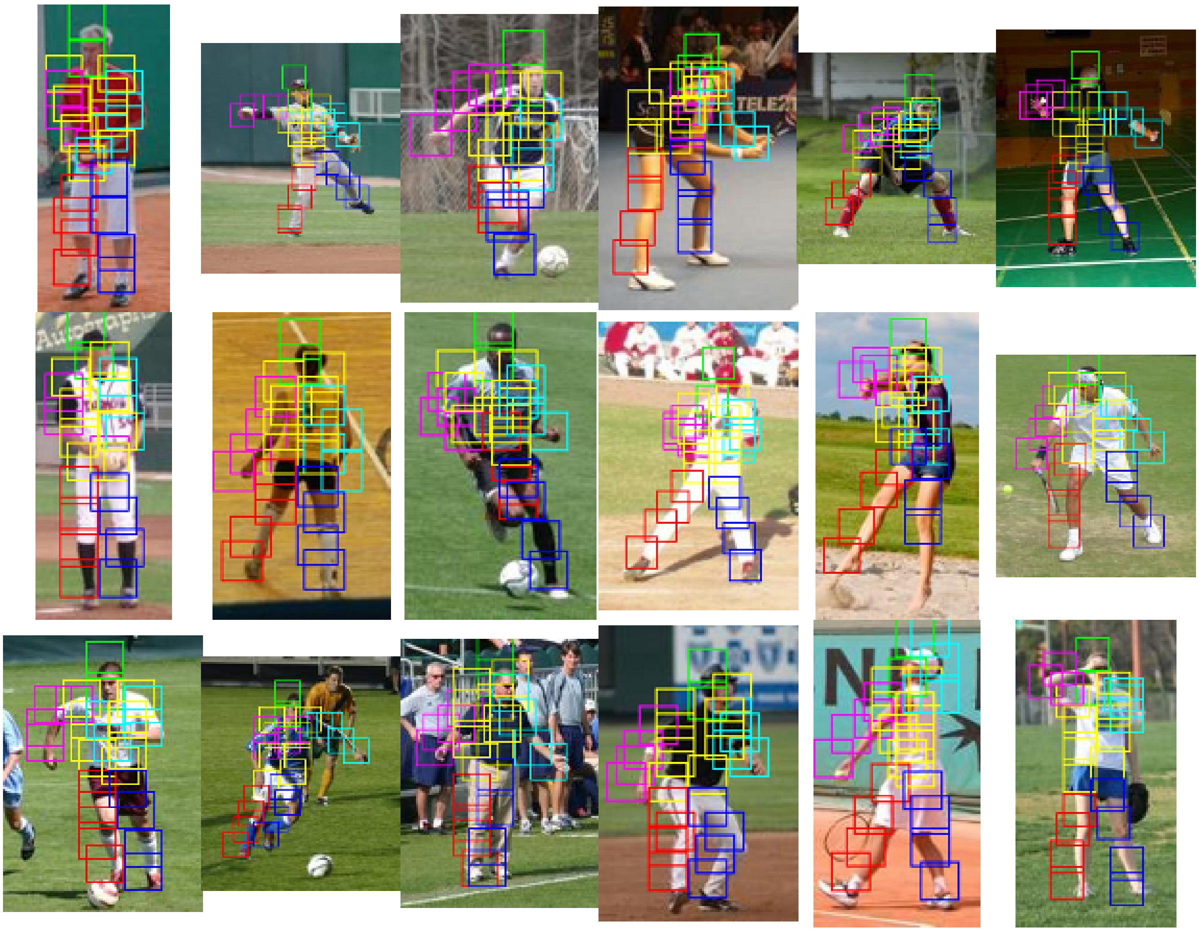


Fig. 4.5 Visualization of our result Human Pose Estimation trained with PARSE training set and tested with LSP test dataset. The green bounding boxes are a head. The yellow bounding boxes are a torso. The cyan bounding boxes are a left arm. The blue bounding boxes are a right arm. The red bounding boxes are a left limb. The deep blue bounding boxes are a right limb.

Chapter 5

Comparing different types of Structured SVM on Human Pose Estimation

5.1 Introduction

In this work, we study two types of Structured learning algorithms for Human Pose Estimation. Our results show that two types of Structured SVM learn HPE with comparable accuracy, and is comparable to standard Latent SVM accuracy. We implemented Structured SVM with matlab Quadratic Programmer. Our source code is available for download.

Chapter 4 presents a neural network based learning system invention. Chapter 4 tries to back propagate to a middle Convolutional layer of Convolutional Neural Network to show that the back propagation of Structured SVM to Neural Network can achieve Human Pose Estimation, yet the resulting PCP accuracy is not very good. Yet comparing Chapter 4's result with Ramanan is not appropriate, since the later result has pretraining of each Appearance Model filters. In this Chapter, we learn our

Structured SVM with back propagation as previously defined in Chapter 4 without having the middle Convolutional Layer. We also used pretraining of each Appearance model filter exactly the same as Yang's. We also compared the Structured SVM back propagation with our own implementation of Joachim Structured SVM with standard Matlab's Quadratic Programmer. We wish to see all Structured SVM based method to have a result PCP accuracy comparable to Yang's method.

5.2 Human Pose Estimation Part base detection

For each node and edge of the pictorial structure, we concatenate each and every bias weight, deformable weight, and appearance filter weight together into 2 types of data structure. The first one is the struct type according to their component, and the second one is the vector type. With the vector type data structure, we create a large vector **w**, the learnable parameters of Human Pose Estimation ready for Structured SVM learning.

$$score(t, y) = \sum_{i \in V} \mathbf{w_f^t} \cdot \mathbf{\Phi_f}\left(\mathbf{x_L}, \mathbf{y}\right) + \sum_{ij \in E} b_{ij}^{t_i t_j} + \mathbf{w_{ij_d}} \cdot \mathbf{\Psi_{ij}} \tag{5.1}$$

, or in matrix form,

$$score(t, y) = \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_L}, \mathbf{y}\right) \tag{5.2}$$

To find the location **y** whose value maximizes the score, the above equation becomes.

$$\hat{\mathbf{y}} = \arg\max_{\hat{y} \in Y} \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_L}, \mathbf{y}\right) \tag{5.3}$$

This formulation is suitable for Structure SVM learning.

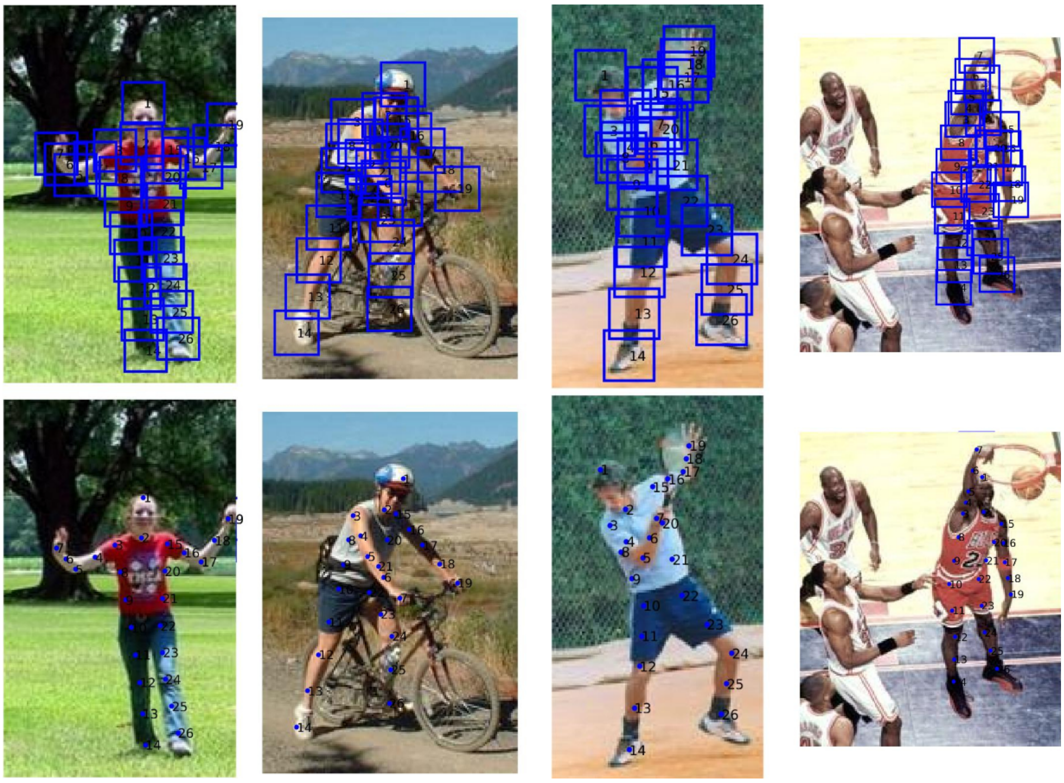


Fig. 5.1 PARSE dataset. The blue Bounding boxes are our training label $\hat{\mathbf{y}}$, which are created by their joint, mid-joint positions. There are 14 joint positions in the original PARSE dataset label. The mid-joint is calculated by finding the mid position of two joints.

5.3 Learning Human Pose Estimation with Structured SVM

Let us focus on Pose Estimation as a Structural Prediction Problem. Here we have a structure of a Human as shown in Figure 2.4. Our Structure Prediction is to find bounding box locations associated with each node such that they match well to the positions of human joint locations as shown in Figure 5.1. How can Structural Prediction be achieved? In a machine learning way, SSVM does structural learning by formulating the training data into statistical problem. In this case, how could we define the structural prediction? Under SSVM one should look to the structural prediction as the association of all structures and labels, SSVM is a supervised learning algorithm. Dataset must provide bounding boxes y for each and every positive image. Now that y are bounding boxes, they can be written (x_1, y_1, x_2, y_2), where x_1, y_1 is the matrix row and column of top-left bounding box. Suppose we focus on (x_1, y_1) only, we have an amount of pixels as a number of possible assignments for each node. We need all nodes, thus $\mathbf{y}$ is a vector of size number of nodes. To think of it as structural prediction, we think of it as sorting, the correct assignment of $\mathbf{y}$ to the training samples should be the maximum of all other possible assignments of $\mathbf{y}$.

Look at Figure 5.2. In this Figure, $(\mathbf{x_2}, \mathbf{y_2})$ is 2nd training data. Therefore, we hope that for all possible assignments of $\mathbf{y}$, $\mathbf{y} = \mathbf{y_2}$ score the highest. In other words, we want SSVM to predict $\hat{\mathbf{y}} = \arg\max_{\hat{\mathbf{y}} \in \mathbf{Y}} \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, \mathbf{y}\right)$. This is our prediction, or detection function, which we can solve with max-sum algorithm. We can think of this inner product as the score of assigning y. We can think of this prediction as finding for all space of y the largest score value: the best score. Now if we can obtain prediction equation sanctification, we can say that we correctly predict $\mathbf{y_2}$. We can say we have performed structural prediction . In this sense, structural prediction is to

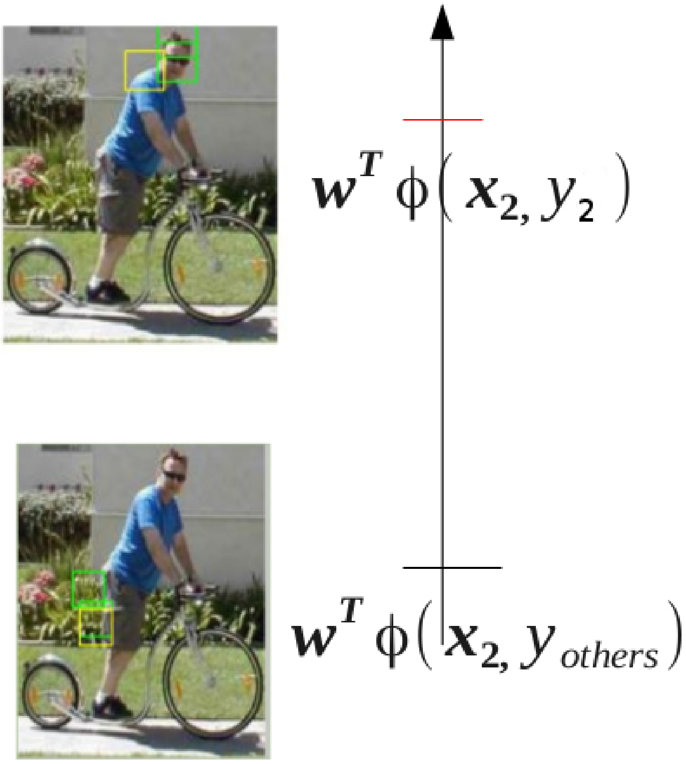


Fig. 5.2 Human Pose Estimation as SSVM sorting: This shows the idea of SSVM as sorting. The correct bounding box assignment has a higher score than the rest of other assignments.

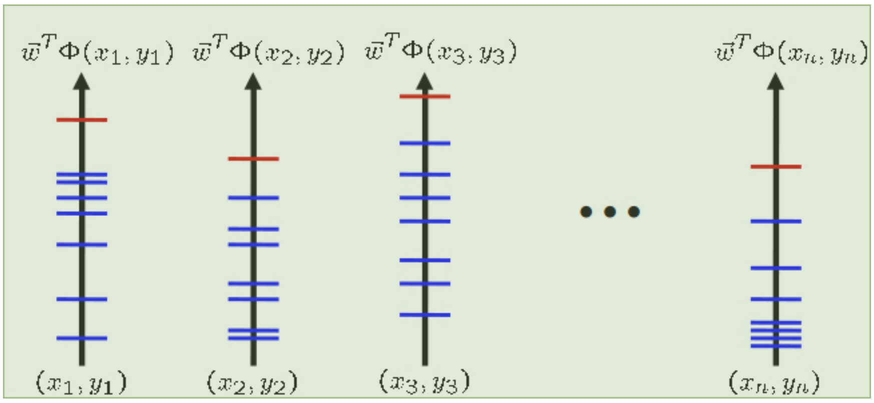

$\vec{w}^T\Phi(x_1, y_1)$
$\vec{w}^T\Phi(x_2, y_2)$
$\vec{w}^T\Phi(x_3, y_3)$
$\vec{w}^T\Phi(x_n, y_n)$
(x_1, y_1)
(x_2, y_2)
(x_3, y_3)
(x_n, y_n)

Fig. 5.3 Hard Margin constraints of Structured SVM. If we can have $\mathbf{w}$ which sorts the score $\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x_i}, \mathbf{y_i})$, to the highest of all possible ways of sorting, then we achieve 100% test accuracy by predicting the $\mathbf{y}$ with maximum score.

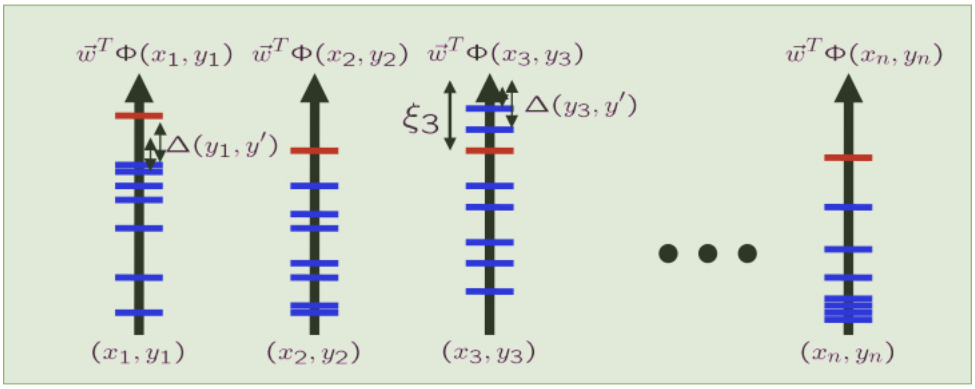


Fig. 5.4 Soft Margin constraints of Structured SVM

assign bounding box values y for all points. Our requirement for a dataset is that we want 100% training accuracy. Meaning that for each and every data $(\mathbf{x_i}, \mathbf{y_i})$ we want $\mathbf{y_i}$ to be predicted out of all possible assignment of $\mathbf{y}$ as shown in Figure . If we can do this, then we have 100% training accuracy. The Figure 5.3 shows our wish. The learning problem is how to find a correct $\mathbf{w}$ such that the training accuracy is 100% and that, as in normal SVM, its norm is the smallest. The above sentence can be written as an objective function as follows:

Hard-margin Optimization Problem:

$$\begin{aligned} &\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &s.t. \quad \forall i \quad, \forall y \in \mathbf{Y} \setminus \mathbf{y_i}: \quad \mathbf{w} \cdot \delta\Phi_i\left(\mathbf{x_i}, \mathbf{y}\right) \geq 1 \end{aligned} \tag{5.4}$$

Look at the constraints of this objective function. What it actually says is that the objective function must be minimized under the constraint that for each sample the correct assignment must have the correct assignment $\mathbf{y_i}$ the highest score compared to all other possible assignments of $\mathbf{y}$, and must be at least 1 unit score higher. This follows the idea of maximum margin classification. In this case, the margin is 1. Instead of assigning the margin value to be 1, we want this margin to change according to the difference between the correct assignment $\mathbf{y_i}$ and other possible assignments $\mathbf{y}$. Loss function $\Delta\left(\mathbf{y},\mathbf{y_i}\right)$ is introduced to measure such distance. The Loss Function is defined as

$$\Delta\left(\mathbf{y},\mathbf{y_i}\right) = 1 - \frac{Area\left(\mathbf{y}\cap\mathbf{y_i}\right)}{Area\left(\mathbf{y}\cup\mathbf{y_i}\right)} \tag{5.5}$$

. As the same as normal SVM, we can introduce Soft margin version. Altogether, the Optimization Problem becomes: Soft Margin Optimization Problem:

$$\begin{aligned} &\min_{w,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ &s.t. \quad \forall i \quad ,\forall y \in Y \setminus y_i : \quad \mathbf{w}\cdot\delta\Phi_{ai}\left(\mathbf{x_i},\mathbf{y}\right) \geq \Delta\left(y_i,y\right) - \xi_i \end{aligned} \tag{5.6}$$

Soft Margin Objective function allows ξ_i to be an error from the best solution. Then soft margin objective function minimizes for such an error. This is shown in Figure 5.4 The problem remains. Even though the structural learning problem has been formulated into a quadratic programming problem, we have a large number of constraints to feed to quadratic programmer(QP). However, if we investigate our problem, those “other $\mathbf{y}$”,i.e “other $\mathbf{y}$” means $\forall y \in Y \setminus y_i$, with low scores on current $\mathbf{w}$ is non-relevant, therefore, for each iteration of $\mathbf{w}$, we only need to optimize over those “other $\mathbf{y}$”, which has a higher score than the correct $\mathbf{y}$. This leads to cutting-plane type algorithms to solve the QP. We can see each incorrect constraints as a cutting plane. To feed a cutting plane algorithm, we find the “other $\mathbf{y}$” which has the highest violation of constraints, hence the name the most violoated constraint. The cutting plane algorithm is shown in Algorithm 1. To compute the most violated constraint of the form $\hat{y} = \arg\max_{\hat{y}\in Y}\mathbf{w}\cdot\mathbf{\Phi_a}\left(x_i,y_i\right) + \Delta\left(y,y_i\right)$ This function is called *Loss Augmented Inference.* To use Structure SVM, one must be able to find over large space of y two inferences: 1) Loss Augmented Inference,and 2) Structural Prediction Inference, also known as test inference. One must also specify loss function $\Delta\left(y,y_i\right)$ and the feature function $\mathbf{\Phi_a}\left(\mathbf{x},y\right)$.

This is how to formulate Human Pose Estimation in an SSVM learning problem.

5.3.1 Solving Structured SVM

The method explained in this section is correct for multiclass classification Structured SVM as well as general Structured SVM. Solving Human Pose Estimation problem with Structured SVM can also be done using the method described in this section.

From Multiclass SVM to Structural SVM:k class classification. Training Samples $(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)$. k class classification. Suppose we can train k weights $\mathbf{w_i}, \forall i \in \{1, ..., k\},$. Prediction is to choose i^{th} class whose test score $\hat{y} = \arg\max_{\hat{y} \in Y} \mathbf{w} \cdot \mathbf{\Phi_a}(x_i, y_i)$ is the largest. The n-slack Structured SVM with marginal rescaling [101] has the primal of the form

Before solving Structured SVM, make sure that your Loss Augmented Inference Algorithm for solving The loss augmented inference

$$\hat{y} = \arg\max_{\hat{y} \in Y} \mathbf{w} \cdot \mathbf{\Phi_a}(\mathbf{x_i}, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y_i}) \tag{5.7}$$

is prepared. As one can see both Cutting plane Algorithm and Subgradient method require this inference.

Training $SSVM_1^{\Delta m} primal$:

$$\begin{aligned} &\min_{w,\xi} \quad \|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i \\ &s.t. \quad \forall i \quad , \forall y \in Y \setminus y_i : \quad \mathbf{w} \cdot \delta\Phi_{ai}(\mathbf{x_i}, \mathbf{y}) \geq \Delta(y_i, y) - \xi_i \end{aligned} \tag{5.8}$$

where $\delta\Phi_{ai}(\mathbf{x_i}, \mathbf{y}) = \Phi_{ai}(\mathbf{x_i}, \mathbf{y}) - \Phi_{ai}(\mathbf{x_i}, \mathbf{y_{iL}})$ The constraints say that we want to find a set of vector $\mathbf{w_i}, \forall i \in \{1, ..., k\}$ such that for each training sample $(\mathbf{x_i}, y_i)$ there is a corresponding $\mathbf{w_i}$ that maximizes the score value (or similarity measure) over other $\mathbf{w_{\neq i}}$, and that for each training data, there is an error epsilon for which what said earlier is not so true, and that we want the overall smallest magnitude or smallest average error set.

The above optimization can be solved by Constraint QP solver. However, there are still exponentially many constraints since there are exponentially many members of set of possible $\mathbf{y}$. How to put this constraint optimization to QP solver?

It turns out that we do not actually need solutions for all possible constraints. The only constraints which are important are those constraints of the $(\mathbf{x_i}, y_{\neq i})$ whose score value $\Phi_{ai}(\mathbf{x_i}, \mathbf{y_{\neq i}})$ is no smaller than the wishful maximum score training label $\Phi_{ai}(\mathbf{x_i}, \mathbf{y_i})$. By including only those important constrants, we reduce the number of constraints to a pragmatic level. To find those important constraints, one must calculate the Loss Augmented Inference. [100] provides the cutting plane algorithm for feeding the QP solver.

Optimize via Cutting Plane Algorithm

Algorithm 1 Cutting Plane Algorithm for Structural SVM (N-slack formulation)[101]

```
Input: $(\mathbf{x_i}, \mathbf{y_i}) : \forall i \in \{1, ..., n\}, C, \epsilon$
Set: $S \leftarrow \emptyset, \mathbf{w} = \mathbf{0}, \epsilon = 0$
REPEAT:
    $\forall i \in \{1, ..., n\}$
        compute $\hat{y} = \arg\max_{\hat{y}\in Y} \mathbf{w} \cdot \mathbf{\Phi_a}(\mathbf{x_i}, y_i) + \Delta(y, y_i)$
        compute $\xi = \max_{y \in Y}\{0, \Delta(y_i, y) - \mathbf{w}^T(\mathbf{\Phi_a}(\mathbf{x_i}, y_i) - \mathbf{\Phi_a}(\mathbf{x_i}, y))\}\}$
        if $\Delta(y_i, y) - \mathbf{w}^T(\mathbf{\Phi_a}(\mathbf{x_i}, y_i) - \mathbf{\Phi_a}(\mathbf{x_i}, y)) > \xi_i + \epsilon$
            $S \leftarrow S \cup \{y_i\}$
            optimize

            $\min_{w,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$        (5.9)
            $s.t. \quad \forall i \quad, \forall y \in Y \setminus y_i : \quad \mathbf{w} \cdot \delta\Phi_{ai}(\mathbf{x_i}, \mathbf{y}) \geq \Delta(y_i, y) - \xi_i$

        end if
    end for
until no $S_i$ change during iteration
```

With the manageable number of constraints, we can solve QP with primal or dual objective function. The n-slack Structured SVM with marginal rescaling has the dual objective function of the form

Dual Objective

The Structured SVM primal formulation has the dual objective of the form,

$$\Theta\left(\alpha\right) = \frac{1}{2}\sum_{i,y\neq y_i}\sum_{j,\bar{y}\neq y_j}\alpha_{(iy)}\alpha_{(j\bar{y})}J_{(iy)(j\bar{y})} + \sum_{i,y\neq y_i}\alpha_{(iy)}\Delta(y_{iy},y_i) \tag{5.10}$$

where $J_{(iy)(j\bar{y})} = \delta\Phi_i\left(\mathbf{y}\right)\delta\Phi_j\left(\bar{\mathbf{y}}\right)$

dual QP objective [101]:

$$\begin{aligned} \alpha^* = \arg\max_{\alpha}\Theta\left(\alpha\right) \\ s.t. \qquad\qquad \\ \alpha \geq 0 \\ \sum_{y\neq y_i}\alpha_{iy} \leq \frac{C}{n}, \forall i = 1, ..., m \end{aligned} \tag{5.11}$$

$SSVM_1^{\Delta m}$ Dual Objective Function in matrix form.

$$\begin{aligned} \hat{\bar{\alpha}} = \arg\max_{\bar{\alpha}} -\frac{1}{2}\bar{\alpha}^T\mathbf{J}\alpha + \mathbf{h^T}\bar{\alpha} \\ s.t. \qquad\qquad \\ \bar{\alpha} \geq 0 \\ \sum_{y\neq y_i}\alpha_{iy} \leq \frac{C}{m} \end{aligned} \tag{5.12}$$

where $\bar{\alpha}$ is the vector of α_{iy}, and $\mathbf{J} = j_{(iy)(jy)}$, and $j_{(iy)(j\bar{y})} = \delta\Phi_{iy}\delta\Phi_{j\bar{y}}$, $\mathbf{h} = \Delta(y_{iy},y_i)$, all these ,$\forall iy$, and $X = \bar{\alpha}$.

$$\alpha = \begin{pmatrix} \{\alpha_{11} & \alpha_{12} & \alpha_{13} & ... & \alpha_{1n_1}\}^T \\ \{\alpha_{21} & \alpha_{22} & \alpha_{23} & ... & \alpha_{2n_1}\}^T \\ ... & & & & \\ \{\alpha_{m1} & \alpha_{m2} & \alpha_{13} & ... & \alpha_{mn_m}\}^T \end{pmatrix} \quad (5.13)$$

$$\mathbf{\Phi_{vec}} = \begin{pmatrix} \{\delta\Phi_{11} & \delta\Phi_{12} & \delta\Phi_{13} & ... & \delta\Phi_{1n_1}\}^T \\ \{\delta\Phi_{21} & \delta\Phi_{22} & \delta\Phi_{23} & ... & \delta\Phi_{2n_1}\}^T \\ ... & & & & \\ \{\delta\Phi_{m1} & \delta\Phi_{m2} & \delta\Phi_{13} & ... & \delta\Phi_{mn_m}\}^T \end{pmatrix} \quad (5.14)$$

$$\mathbf{\Delta_{vec}} = \begin{pmatrix} \{\Delta(\mathbf{y_1}, \hat{\mathbf{y}}_{11}) & \Delta(\mathbf{y_1}, \hat{\mathbf{y}}_{12}) & \Delta(\mathbf{y_1}, \hat{\mathbf{y}}_{13}) & ... & \Delta(\mathbf{y_1}, \hat{\mathbf{y}}_{1n_1})\}^T \\ \{\Delta(\mathbf{y_2}, \hat{\mathbf{y}}_{21}) & \Delta(\mathbf{y_2}, \hat{\mathbf{y}}_{22}) & \Delta(\mathbf{y_2}, \hat{\mathbf{y}}_{23}) & ... & \Delta(\mathbf{y_2}, \hat{\mathbf{y}}_{2n_1})\}^T \\ ... & & & & \\ \{\Delta(\mathbf{y_m}, \hat{\mathbf{y}}_{m1}) & \Delta(\mathbf{y_m}, \hat{\mathbf{y}}_{m2}) & \Delta(\mathbf{y_m}, \hat{\mathbf{y}}_{m3}) & ... & \Delta(\mathbf{y_m}, \hat{\mathbf{y}}_{mn_m})\}^T \end{pmatrix} \quad (5.15)$$

$$\mathbf{J} = \mathbf{\Phi_{vec}}\mathbf{\Phi_{vec}}^T \quad (5.16)$$

The standard Quadratic Programming optimization is of the form

$$\begin{aligned} \min_{X} \quad & \frac{1}{2}X\mathbf{P}X^T + \mathbf{C^T}\bar{X} \\ s.t. \quad & A_{eq}X = Beq \\ & lb \leq X \leq ub \\ & GX \leq u \end{aligned} \quad (5.17)$$

We set our A_{eq}, and B_{eq} to be empty. Our required constraints $\alpha \geq 0$ are achieved by setting $lb = 0$, and ub is empty vector. We set $P = J$. $C = -\Delta_{vec}$ because we are minimizing the negative objective of dual QP maximization.

How do we understand the subscription iy? The iy means there are many ys for each i, and there are many is. This depends on constraint insertion. Since our cutting plane algorithm guarantees that no y_i being inserted into constraints, we can simply insert the y, which violated constraint the most, into constraints, being sure that our constraints are satisfying the domain $\forall y \in Y \setminus y_i$. For example, suppose our current constraints are $\hat{y}_{11}, ..., \hat{y}_{1R_1}$ for the first training data, and $\hat{y}_{21}, ..., \hat{y}_{1R_2}$, for second training data, where R_1, R_2 are the number of constraints inserted since the algorithm run, then there are a total $R1 + R2$ number of αs. Therefore, the number of α is the number of constraints inserted. Hence $\bar{\alpha} = \{\{\alpha_{11}, \alpha_{12}, ..., \alpha_{1R_1}\}, \{\alpha_{21}, \alpha_{22}, ..., \alpha_{2R_2}\}\}^T$. The $\bar{\alpha} \geq \mathbf{0}$ can be implemented by setting the lower bound of quadratic optimization to be zero. Here we need to add inequality constraints $\sum_{y \neq y_i} \alpha_{iy} \leq \frac{C}{m}$ to quadratic programmer. This inequality meaning is "The summation over all violated constraints found at i^{th} training samples, bar y_i, must be less than or equal $\frac{C}{m}$". For example, suppose there are 2 violated constraints found for i^{th} sample, then

$$\alpha_{iy_{violate1}} + \alpha_{iy_{violate2}} \leq \frac{C}{m}$$

Therefore, there are at most m inequalities. On quadratic programmer, one can place this under $\mathbf{GX} \leq \mathbf{u}$ form of constraints. For example, if we have

$$\begin{aligned} \alpha_{1y_1} + \alpha_{1y_2} \leq & \ \frac{C}{m} \\ \alpha_{2y_1} \leq & \ \frac{C}{m} \end{aligned} \tag{5.18}$$

In this case, there are two summation inequalities. The first one is the summation of all recorded most violated constraints since the training started with the first training sample. The second one is the summation of all recorded most violated constraints since the training started with the first training sample. Note that there may be different numbers of most violated constraints found in each training sample. This is because the requirement of cutting plane algorithm is to accumulate the most violated constraints only if their values are greater than ϵ. We can then input $G = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\bar{\alpha} = \left[\alpha_{1y_1}, \alpha_{1y_2}, \alpha_{2y_1}\right]^T$, and $\mathbf{u} = \frac{C}{m}\left[1, 1\right]^T$. In general, our

$$J = \left[\delta\Phi_{ai}\left(\mathbf{x_i}, \mathbf{y_{iy}}\right)\right]\left[\delta\Phi_{ai}\left(\mathbf{x_i}, \mathbf{y_{iy}}\right)\right]^T$$

To find the optimal $\hat{\mathbf{w}}$ from the dual Lagrange multipliers α, we calculate

$$\hat{w}\left(\alpha\right) = \sum_i \sum_{y \neq y_i} \hat{\alpha}_{iy} \delta\Phi_{ai}\left(\mathbf{x_i}, \mathbf{y_{iy}}\right)$$

. Or in the matrix form

$$\hat{\mathbf{w}} = \delta\mathbf{\Phi}_{\mathbf{ai}}^{\mathbf{T}}\bar{\alpha} \tag{5.19}$$

.

Primal Objective

For primal objective quadratic programming, it is much more straight forward. We can feed 5.9 to quadratic programmer. The above optimization can be solved for multiclass SSVM with the CVX [45].

We strictly follow the formula given by Joachim. Loss Augmented Inference, and Joint Feature location function $\hat{\mathbf{y}}$,$\delta\Phi_{ai}\left(\mathbf{x_i}, \mathbf{y}\right)$. The Loss function $\Delta\left(\mathbf{y}, \mathbf{y_i}\right) = 1 - \frac{Area(\mathbf{y} \cap \mathbf{y_i})}{Area(\mathbf{y} \cup \mathbf{y_i})}$ are jointly calculated by SSVM Loss Augmented inference function. This Loss Augmented Inference function calculates "the most violoated constraint" as shown in Algorithm 1 line 6. We then strictly follow Algorithm 1 by calculating 7. If the current gap between loss augmented inference optimal value,$\mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, \hat{\mathbf{y}}\right) - \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, \mathbf{y_i}\right) + \Delta\left(\mathbf{y}, \mathbf{y_i}\right)$, and one minus intersect over union loss , $\Delta\left(\mathbf{y}, \mathbf{y_i}\right)$, is larger than ith slack variable ξ_i by at least ϵ, then the most violated constraint tuple $\mathbf{x_i}, \hat{\mathbf{y}}$ is unioned. Then the weights $\mathbf{w}$ is optimized with Structured SVM objective function (line of Algorithm 1). This is iterated until the set S has no more change, which means that all the gaps are smaller than $\xi_i + \epsilon$.

Optimization via Back Propagation Method

Algorithm 2 Back Propagation Algorithm for Structural SVM (N-slack formulation)(Chapter 4)

```
Input: $(\mathbf{x_i}, \mathbf{y_i}) : \forall i \in \{1, ..., n\}, C, \epsilon$
compute $\hat{y} = \arg\max_{\hat{y} \in Y} \mathbf{w} \cdot \mathbf{\Phi_a}\left(\mathbf{x_i}, y_i\right) + \Delta\left(y, y_i\right)$
add back propagation $g \leftarrow g + \mathbf{\Phi_a}\left(\mathbf{x_i}, \hat{y}\right) - \mathbf{\Phi_a}\left(\mathbf{x_i}, y_i\right)$
return g
```

Another method for optimizing Structured SVM is back propagation method. The back propagation optimization is similar to gradient descent in its procedure. The difference is that the subgradient method is used in the piecewise continuous objective function with respect to $\mathbf{w}$. In our risk minimization objective 5.20, our objective function is a highly piecewise continuous function. By minimizing the regularized risk $c\left(\mathbf{w}\right)$ objective[79], one can get the optimal weight $\mathbf{w}$.

$$
\begin{aligned}
&\min_{w,\xi} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{l} \xi_i \\
&s.t. \quad \forall i : \quad \mathbf{w} \cdot \Phi_{ai}\left(\mathbf{x_i}, \mathbf{y_i}\right) + \xi_i \geq \max_{y \in \mathrm{Y}} \mathbf{w} \cdot \Phi_{ai}\left(\mathbf{x_i}, \mathbf{y}\right) + \Delta\left(y, y_i\right)
\end{aligned}
\tag{5.20}
$$

When some dimension of $\mathbf{w}$ changes its value, this causes our loss augmented algorithm to compute different $\hat{y}$. Therefore, the objective value given by risk objective 5.20 abruptly changes, yet for every value $\mathbf{w}$ within the domain $\mathbf{W}$, has its functional value. If $\mathbf{w}$ changes too little, the loss augmented inference returns the same $\hat{y}$, therefore the objective function remains continuous function. The aformentioned objective function is clearly a piecewise continuous function. Thus our technique to solve the convex optimization under piecewise continuous function is by subgradient algorithm.

We use the method described in Chapter 4 to solve the eq. 5.20 with back propagation algorithm.

5.4 Experiment

Data preparation is no different than in the previous Chapter. The neural network structure is as follows. Data layer - Convolution layer *ConvSsvm* - Loss Augmented Inference layer. We start our weight initialization with pretrianed weights as in [113]. Our mixture model for each node is as follows: M_{ixture} is the same as in the previous Chapter. There are total 138 of $\sum_i M_{ixture}$ mixtures. The size of *ConvSsvm* layer is $5 \times 5 \times 32 \times 138 = 110400$, where 138 is $\sum_i M_{ixture}$. The Loss Augmented Inference has 0 weight elements of $\forall v \in \mathrm{V}, b_v^{t_v}$, and $1 + \sum_i M_{ixture}\left(i\right) \times M_{ixture}\left(i-1\right) = 702$ weight elements of $\forall vq \in \mathrm{E}, b_{vq}^{t_v t_q}$, and $133 \times 4 = 532$ weight elements of $\mathbf{w_{deform}}$. Therefore the Loss Augmented Inference layer has a total of 1234 weight elements. Therefore our system has a total weight of $110400+1234 = 111634$ elements. Figure 5.5. The majority of one-minus-intersect-over-union ,$\Delta\left(\hat{y}, y_i\right)$,loss is reduced to around 20%-30%, as opposed to the previous Chapter of 20%-40%. The L_i reduced nicely from the value of 50 to 0.3. This, perhaps, is due to the fact that we pretrained the *Convssvm* filters before training with our Back Propagation algorithm.

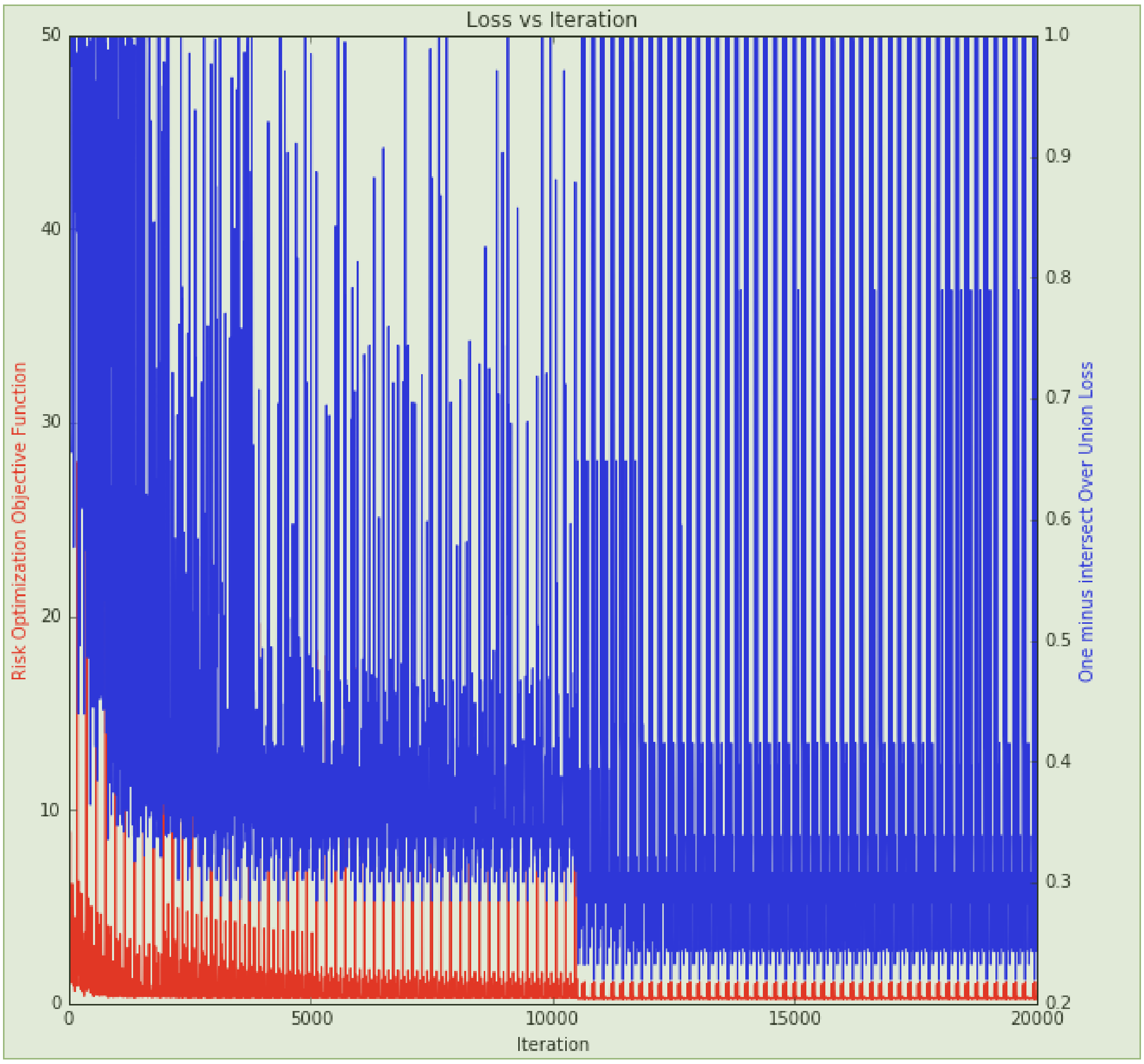


Fig. 5.5 Slack Loss, One minus Intersect over Union loss ,$\Delta\left(\hat{\mathbf{y}}, \mathbf{y_i}\right)$, versus samples

Table 5.1 Strict Percentage of Correct Point (PCP)[112] Comparison

object class	**Structured SVM Quadproc**	**Structured SVM Back Propagation**	**Yang& Ramanan [113]**
Head	**79.0**	76.1	77.6
Torso	**85.4**	56.9	82.9
L. arm	22.6	32.7	**35.4**
U. arm	48.5	47.8	**55.1**
L. leg	62.9	61.1	**63.9**
U. leg	**72.0**	65.1	69.0
total PCP	58.5	56.9	**60.7**

5.4.1 Result

Table 5.1 shows comparable PCP accuracy as we wished.

5.5 Conclusion

In this Chapter, we show that our Structured SVM back propagation, without a middle Convolutional layer, can achieve accuracy comparable to LSVM.

Chapter 6

Conclusion

Currently there are many part based detection works which rely on convolutional neural network as front end. It has been confirmed in many cases that classification and feature extraction by back propagation of the classifier into deep learning feature extractor gives better performance. We plan to use this newly invented layer, Structured SVM neural network, on Deep Convolutional Neural Network for part based image detection problems. In this work, we study the feasibility of such a plan by showing that reducing loss of Structured SVM neural network can perform part based detection. In the future, when we add this new layer to Deep Convolutional Neural network, we can create the full end-to-end Neural Network for Human Pose Estimation problem, as well as other part based detection problems.

Our work shows that the unary potential of Makov Random Fields can be learned with neural network based feature extractor, and that the whole Makrov Random Field based model can be learned with Structured SVM back propagating to Neural Networks. Markov Random Field based models are very important in solving the following problems: Image Segmentation problems, Stereo Vision problems, Image deblurring problems. Our work shows that these computer vision problems could be able to learn with Structured SVM Back propagating to Neural Network. Hence through this work's implication, we add to the methods for solving general computer vision problem.

We proposed a new learning algorithm, Structured SVM Convolutional neural network, to learn Human Pose Estimation model. We defined the Structured SVM as two-layer neural network that can further back propagate the inference error to deep learning feature extractor. Our Structured SVM within Structured SVM Convolutional Neural Network learn its weights the same way it learns it weight without back propagation. Our method works as a generic learning algorithm for broad deep learning computer vision problems. Even though our result is not yet as good as state of the art, our method can learn the model parameters of Human Pose Estimation without negative samples. We believe our method can perform much better if we use deep convolutional neural network as front end. Due to the time consuming nature of our training, we have not yet tried to use deep convolutional neural network as front end. In the future, we shall implement our method as full deep learning machine.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Anand, A., Koppula, H. S., Joachims, T., and Saxena, A. (2013). Contextually Guided Semantic Labeling and Search for 3D Point Clouds. *International Journal of Robotics Research.*

[3] and U, M., Franc, V., and V, H. (2012). Detector of Facial Landmarks Learned by the Structured Output {SVM}. In *VISAPP '12: Proceedings of the 7th International Conference on Computer Vision Theory and Applications.*

[4] Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR.*

[5] Belongie, S., Malik, J., and Puzicha, J. (2000). Shape context: A new descriptor for shape matching and object recognition. *Advances in Neural Information Processing Systems*, 546:831–837.

[6] Bertelli, L., Yu, T., Vu, D., and Gokturk, B. (2011). Kernelized structural SVM learning for supervised object segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[7] Blaschko, M. B. and Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).*

[8] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory.*

[9] Branson, S., Beijbom, O., and Belongie, S. (2013). Efficient large-scale structured learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1806–1813.

[10] Branson, S., Perona, P., and Belongie, S. (2011). Strong supervision from weak annotation: Interactive training of deformable part models. In *Proceedings of the IEEE International Conference on Computer Vision*.

[11] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[12] Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2015). Human Pose Estimation with Iterative Error Feedback. *Nips'15*, pages 1–12.

[13] Charles, J., Pfister, T., Magee, D., Hogg, D., and Zisserman, A. (2014). Upper Body Pose Estimation with Temporal Sequential Forests. In *Proceedings of the British Machine Vision Conference 2014*.

[14] Chen, K., Gong, S., and Xiang, T. (2011). Human pose estimation using structural support vector machines. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 846–851.

[15] Chen, X. and Yuille, A. (2015). Parsing occluded people by flexible compositions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 3945–3954. IEEE Computer Society.

[16] Chen, X. and Yuille, A. L. (2014). Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations. *Advances in Neural Information Processing Systems (NIPS)*, pages 1736–1744.

[17] Cherian, A., Mairal, J., and Alahari, K. (2014a). Mixing Body-Part Sequences for Human Pose Estimation. *CVPR 2014-IEEE ...*.

[18] Cherian, A., Mairal, J., Alahari, K., and Schmid, C. (2014b). Mixing Body-Part Sequences for Human Pose Estimation. *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*.

[19] Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE.

[20] Cires, D. and Meier, U. (2012). Multi-column Deep Neural Networks for Image Classification. *Applied Sciences*.

[21] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

[22] Crammer, K. and Singer, Y. (2001). On The Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research (JMLR)*.

[23] Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 886–893. IEEE Computer Society.

[24] Dann, C., Gehler, P., Roth, S., and Nowozin, S. (2012). Pottics–the potts topic model for semantic image segmentation. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 397–407. Springer.

[25] Dantone, M., Gall, J., Leistner, C., and Van Gool, L. (2014). Body Parts Dependent Joint Regressors for Human Pose Estimation in Still Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2131–2143.

[26] Desai, C., Ramanan, D., and Fowlkes, C. C. (2011). Discriminative models for multi-class object layout. *International Journal of Computer Vision*.

[27] Dong, J., Chen, Q., Shen, X., Yang, J., and Yan, S. (2014). Towards unified human parsing and pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[28] Dong, J., Chen, Q., Xia, W., Huang, Z., and Yan, S. (2013). A deformable mixture parsing model with parselets. In *Proceedings of the IEEE International Conference on Computer Vision*.

[29] Eichner, M. and Ferrari, V. (2012). Human pose co-estimation and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2282–2288.

[30] Eichner, M., Marin-Jimenez, M., Zisserman, A., and Ferrari, V. (2012). 2D articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision*, 99:190–214.

[31] Felzenszwalb, P. F., Girshick, R. B., Mcallester, D., and Ramanan, D. (2009). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20.

[32] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004a). Distance transforms of sampled functions. *Cornell Computing and Information Science Technical Report TR20041963*, 4:1–15.

[33] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004b). Distance transforms of sampled functions. *Cornell Computing and Information Science Technical Report TR20041963*.

[34] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79.

[35] Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive Search Space Reduction for Human Pose Estimation. *IEEE Conference on Computer Vision and Pattern Recognition (2008)*, 2:1–8.

[36] Feyereisl, J., Kwak, S., Son, J., and Han, B. (2014). Object Localization based on Structural SVM using Privileged Information. *Nips*.

[37] Fidler, S., Mottaghi, R., Yuille, A., and Urtasun, R. (2013). Bottom-up segmentation for top-down detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[38] Fischler, M. A. and Elschlager, R. A. (1973). The Representation and Matching of Pictorial Structures Representation. *IEEE Transactions on Computers*.

[39] Forney G.D., J. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):302–309.

[40] Freund, Y. and Schapire, R. E. (1997). A decision theoretic generalization of on-line learning and an application to boosting. *Computer Systems Science*, 57:119–139.

[41] Gilks, W. R. (2005). Markov chain monte carlo. *Encyclopedia of Biostatistics*.

[42] Girshick, R., Iandola, F., Darrell, T., and Malik, J. (2014). Deformable Part Models are Convolutional Neural Networks. *arXiv:1409.5403*, page 8.

[43] Girshick, R. B., Felzenszwalb, P. F., and Mcallester, D. (2011). Object detection with grammar models. *Advances in Neural*.

[44] Gkioxari, G., Hariharan, B., Girshick, R., and Malik, J. (2014). R-CNNs for Pose Estimation and Action Detection. *arXiv preprint arXiv:1406.5212*.

[45] Grant, M. and Boyd, S. (2008). CVX: Matlab software for disciplined convex programming.

[46] He, K., Sigal, L., and Sclaroff, S. (2014). Parameterizing object detectors in the continuous pose space. In *European Conference on Computer Vision*, pages 450–465. Springer.

[47] Jain, A., Tompson, J., Andriluka, M., Taylor, G. W., and Bregler, C. (2013). Learning Human Pose Estimation Features with Convolutional Networks. *arXiv preprint arXiv: . . .*, pages 1–10.

[48] Jain, A., Tompson, J., LeCun, Y., and Bregler, C. (2015). MoDeep: A Deep Learning Framework Using Motion Features for Human Pose Estimation. In Cremers, D., Reid, I., Saito, H., and Yang, M.-H., editors, *Computer Vision – ACCV 2014 SE - 21*, volume 9004 of *Lecture Notes in Computer Science*, pages 302–315. Springer International Publishing.

[49] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

[50] Johnson, S. and Everingham, M. (2010). Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. *Procedings of the British Machine Vision Conference 2010*, (i):12.1—-12.11.

[51] Johnson, S. and Everingham, M. (2011). Learning effective human pose estimation from inaccurate annotation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1465–1472.

[52] Kai, W., Babenko, B., and Belongie, S. (2011). End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*.

[53] Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2012). Block-coordinate frank-wolfe optimization for structural svms. *arXiv preprint arXiv:1207.4747*.

[54] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition.

[55] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323.

[56] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[57] LeCun Yann, Cortes Corinna, and Burges Christopher (1998). THE MNIST DATABASE of handwritten digits. *The Courant Institute of Mathematical Sciences*.

[58] Li, B., Hu, W., Wu, T., and Zhu, S. C. (2013). Modeling occlusion by discriminative AND-OR structures. In *Proceedings of the IEEE International Conference on Computer Vision*.

[59] Li, H., Lin, Z., Shen, X., Brandt, J., and Hua, G. (2015). A convolutional neural network cascade for face detection.

[60] Li, S. Z. (2009). *Markov random field modeling in image analysis*. Springer Science & Business Media.

[61] Lin, D., Fidler, S., Kong, C., and Urtasun, R. (2014). Visual semantic search: Retrieving videos via complex textual queries. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[62] Lin, L., Wang, X., Yang, W., and Lai, J. H. (2015). Discriminatively trained and-Or graph models for object shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[63] Lin, L., Wu, T., Porway, J., and Xu, Z. (2009). A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*.

[64] Liu, F., Lin, G., and Shen, C. (2015). CRF learning with CNN features for image segmentation. *Pattern Recognition*.

[65] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation preprint. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[66] Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2.

[67] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[68] Lu, H., Shao, X., and Xiao, Y. (2013). Pose estimation with segmentation consistency. *IEEE Transactions on Image Processing.*

[69] Mottaghi, R., Chen, X., Liu, X., Cho, N. G., Lee, S. W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[70] Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience, New York, NY, USA.

[71] Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

[72] Ouyang, W., Chu, X., and Wang, X. (2014). Multi-source Deep Learning for Human Pose Estimation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2344.

[73] Özuysal, M., Fua, P., and Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[74] Pishchulin, L., Andriluka, M., Gehler, P., and Schiele, B. (2013a). Poselet conditioned pictorial structures. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 588–595.

[75] Pishchulin, L., Andriluka, M., Gehler, P., and Schiele, B. (2013b). Strong Appearance and Expressive Spatial Models for Human Pose Estimation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3487–3494.

[76] Pishchulin, L., Jain, A., Andriluka, M., Thormahlen, T., and Schiele, B. (2012). Articulated people detection and pose estimation: Reshaping the future. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3178–3185.

[77] Ramanan, D. (2007). Learning to parse images of articulated bodies. *Advances in Neural Information Processing Systems*, 19:1129–1136.

[78] Rantalankila, P., Kannala, J., and Rahtu, E. (2014). Generating Object Segmentation Proposals Using Global and Local Search. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on.*

[79] Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. a. (2006). (Online) Subgradient Methods for Structured Prediction. *Artificial Intelligence and Statistics*, 2007.

[80] Rothrock, B., Park, S., and Zhu, S. C. (2013). Integrating grammar and segmentation for human pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[81] Sabzmeydani, P. and Mori, G. (2007). Detecting pedestrians by learning shapelet features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[82] Sapp, B. and Taskar, B. (2013). MODEC: Multimodal decomposable models for human pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3674–3681.

[83] Schwing, A., Hazan, T., Pollefeys, M., and Urtasun, R. (2011). Distributed message passing for large scale graphical models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[84] Sermanet, P., Kavukcuoglu, K., Chintala, S., and Lecun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3626–3633.

[85] Shen, J., Liu, G., Chen, J., Fang, Y., Xie, J., Yu, Y., and Yan, S. (2014). Unified structured learning for simultaneous human pose estimation and garment attribute classification. *IEEE Transactions on Image Processing*.

[86] Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S., and Zhang, Z. (2013). Scene text recognition using part-based tree-structured character detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[87] Shi-Xiong Zhang , Chaojun Liu, K. Y. and Gong, Y. (2015). DEEP NEURAL SUPPORT VECTOR MACHINES FOR SPEECH RECOGNITION. *Icassp*, (1):4275–4279.

[88] Sigal, L. and Black, M. J. (2006). Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2041–2048.

[89] Sun, J. and Ponce, J. (2013). Learning discriminative part detectors for image classification and cosegmentation. In *Proceedings of the IEEE International Conference on Computer Vision*.

[90] Sun, Y., Wang, X., and Tang, X. (2013). Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3476–3483.

[91] Szegedy, C., Toshev, a., and Erhan, D. (2013). Deep Neural Networks for Object Detection. *Advances in Neural Information . . .*, pages 1–9.

[92] Tang, Y. (2013). Deep Learning using Linear Support Vector Machines. *Deeplearning.Net*.

[93] Telaprolu, M. and Savarese, S. (2012). An efficient branch-and-bound algorithm for optimal human pose estimation. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1616–1623.

[94] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[95] Tian, T. P. and Sclaroff, S. (2010). Fast globally optimal 2D human detection with loopy graph models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 81–88.

[96] Tian, Y., Zitnick, C. L., and Narasimhan, S. G. (2012). Exploring the Spatial Hierarchy of Mixture Models for Human Pose Estimation. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, volume 7576 of *Lecture Notes in Computer Science*, pages 256–269. Springer.

[97] Tompson, J., Goroshin, R., Jain, A., Lecun, Y., and Bregler, C. (2015). Efficient Object Localization Using Convolutional Networks. *Cvpr*, page 2014.

[98] Tompson, J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. *Advances in neural information processing systems*, pages 1799—-1807.

[99] Toshev, A. and Szegedy, C. (2013). DeepPose: Human Pose Estimation via Deep Neural Networks. *Arxiv preprint arXiv*.

[100] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *ICML*, page 104. ACM Press.

[101] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484.

[102] Tuzel, O., Porikli, F., and Meer, P. (2007). Human Detection via Classification on Riemannian Manifolds. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 30(10):1–8.

[103] Vapnik, V. (2000). SVM method of estimating density, conditional probability, and conditional density. *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, 2.

[104] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1.

[105] Wan, L., Eigen, D., and Fergus, R. (2015). End-to-end integration of a Convolutional Network, Deformable Parts Model and non-maximum suppression. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[106] Wang, F. and Li, Y. (2013). Beyond Physical Connections: Tree Models in Human Pose Estimation. *arXiv.org*, cs.CV.

[107] Wang, X., Han, T. X., and Yan, S. (2009). An HOG-LBP human detector with partial occlusion handling. *Computer Vision, 2009 IEEE 12th International Conference on.*

[108] Wang, Y., Tran, D., and Liao, Z. (2011). Learning hierarchical poselets for human parsing. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[109] Xu, J., Schwing, A. G., and Urtasun, R. (2014). Tell Me What You See and I Will Show You Where It Is. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on.*

[110] Yamaguchi, K., Kiapour, M. H., Ortiz, L. E., and Berg, T. L. (2012). Parsing clothing in fashion photographs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[111] Yang, J. and Yang, M. H. (2012). Top-down visual saliency via joint CRF and dictionary learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[112] Yang, W., Ouyang, W., Li, H., and Wang, X. (2016). End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. CVPR.

[113] Yang, Y. and Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1385–1392.

[114] Yao, B., Liu, Z., Nie, X., and Zhu, S.-C. (2013a). Animated Pose Templates for Modelling and Detecting Human Actions. *IEEE transactions on pattern analysis and machine intelligence.*

[115] Yao, R., Shi, Q., Shen, C., Zhang, Y., and Van Den Hengel, A. (2013b). Part-based visual tracking with online latent structural learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[116] Zhang, L. and Van Der Maaten, L. (2013). Structure preserving object tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

[117] Zhang, Y., Sohn, K., Villegas, R., Pan, G., and Lee, H. (2015). Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. *IEEE Conference on Computer Vision and Pattern Recognition.*

[118] Zhao, Y. (2011). Image Parsing via Stochastic Scene Grammar. *Advances in Neural Information Processing Systems (NIPS).*

[119] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. S. (2015). Conditional Random Fields as Recurrent Neural Networks. *ICCV.*

[120] Zhu, L., Chen, Y., Yuille, A., and Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[121] Zhu, X. and Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[122] Zuffi, S., Romero, J., Schmid, C., and Black, M. J. (2013). Estimating human pose with flowing puppets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3312–3319.