ABSTRACT

Title of Dissertation: TOWARDS A FAST AND ACCURATE FACE RECOGNITION SYSTEM FROM DEEP REPRESENTATIONS

Rajeev Ranjan
Doctor of Philosophy, 2019

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

The key components of a machine perception algorithm are feature extraction followed by classification or regression. The features representing the input data should have the following desirable properties: 1) they should contain the discriminative information required for accurate classification, 2) they should be robust and adaptive to several variations in the input data due to illumination, translation/rotation, resolution, and input noise, 3) they should lie on a simple manifold for easy classification or regression. Over the years, researchers have come up with various hand crafted techniques to extract meaningful features. However, these features do not perform well for data collected in unconstrained settings due to large variations in appearance and other nuisance factors.

Recent developments in deep convolutional neural networks (DCNNs) have shown impressive performance improvements on various machine perception tasks such as object detection and recognition. DCNNs are highly non-linear regressors because of the presence of hierarchical convolutional layers with non-linear activation. Unlike the hand crafted features, DCNNs learn the feature extraction and feature classification/regression modules from the data itself in an end-to-end fashion. This enables the DCNNs to be robust to variations present in the data and at the same time improve their discriminative ability. Ever-increasing computation power and availability of large datasets have led to significant performance gains from DCNNs. However, these developments in deep *learning are not directly applicable to the face analysis tasks due to large variations in illumination, resolution, viewpoint, and attributes of faces acquired in unconstrained settings. In this dissertation, we address this issue by developing efficient DCNN architectures and loss functions for multiple face analysis tasks such as face detection, pose estimation, landmarks localization, and face recognition from unconstrained images and videos.*

In the first part of this dissertation, we present two face detection algorithms based on deep pyramidal features. The first face detector, called DP2MFD, utilizes the concepts of deformable parts model (DPM) in the context of deep learning. It is able to detect faces of various sizes and poses in unconstrained conditions. It reduces the gap in training and testing of DPM on deep features by adding a normalization layer to the DCNN. The second face detector, called Deep Pyramid Single Shot Face Detector (DPSSD), is fast and capable of detecting faces with large scale variations (especially tiny faces). It makes use of the inbuilt pyramidal hierarchy present in a DCNN, instead of creating an image pyramid. Extensive experiments on publicly available unconstrained face detection datasets show that both these face detectors are able to capture the meaningful structure of faces and perform significantly better than many traditional face detection algorithms.

In the second part of this dissertation, we present two algorithms for simultaneous face detection, landmarks localization, pose estimation and gender recognition using DCNNs. The first method called, HyperFace, fuses the intermediate layers of a deep CNN using a separate CNN followed by a multi-task learning algorithm that operates on the fused features. The second approach extends HyperFace to incorporate additional tasks of face verification, age estimation and smile detection, in All-In-One Face. HyperFace and All-In-One Face exploit the synergy among the tasks which improves individual performances.

In the third part of this dissertation, we focus on improving the task of face verification by designing a novel loss function that maximizes the inter-class distance and minimizes the intra-class distance in the feature space. We propose a new loss function, called Crystal Loss, that adds an L_2-constraint to the feature descriptors which restricts *them to lie on a hypersphere of a fixed radius. This module can be easily implemented using existing deep learning frameworks. We show that integrating this simple step in the training pipeline significantly boosts the performance of face verification. We additionally describe a deep learning pipeline for unconstrained face identification and verification which achieves state-of-the-art performance on several benchmark datasets. We provide the design details of the various modules involved in automatic face recognition: face detection, landmark localization and alignment, and face identification/verification. We present experimental results for end-to-end face verification and identification on IARPA Janus Benchmarks A, B and C (IJB-A, IJB-B, IJB-C), and the Janus Challenge Set 5 (CS5).*

Though DCNNs have surpassed human-level performance on tasks such as object classification and face verification, they can easily be fooled by adversarial attacks. These attacks add a small perturbation to the input image that causes the network to mis-classify the sample. In the final part of this dissertation, we focus on safeguarding the DCNNs and neutralizing adversarial attacks by compact feature learning. In particular, we show that learning features in a closed and bounded space improves the robustness of the network. We explore the effect of Crystal Loss, that enforces compactness in the learned features, thus resulting in enhanced robustness to adversarial perturbations. Additionally, we propose compact convolution, a novel method of convolution that when incorporated in conventional CNNs improves their robustness. Compact convolution ensures feature compactness at every layer such that they are bounded and close to each other. Extensive experiments show that Compact Convolutional Networks (CCNs) neutralize multiple types of attacks, and perform better than existing methods in defending adversarial attacks, without incurring any additional training overhead compared to CNNs.

Towards a Fast and Accurate Face Recognition System from Deep Representations

by

Rajeev Ranjan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Dedication

To the memory of my undergraduate advisor Professor Somnath Sengupta, who kindled my interest in scientific research.

Acknowledgements

First of all I would like to thank my advisor Professor Chellappa for providing me valuable guidance and advice throughout the years of my study. He is the funniest advisor and one of the smartest persons I know. He has always been supportive and has given me the freedom to pursue and work on my own ideas. He provided me the opportunity to work on major IARPA funded projects that enhanced my research experience. Brainstorming sessions with him have helped me expand my knowledge of computer vision and machine learning. I feel blessed to have learned from such a great researcher and a wonderful person.

It is an honor to have Professor David Jacobs, Professor Behtash Babadi, Professor Abhinav Shrivastava and Dr. Carlos Castillo in my dissertation committee. I am thankful to them for serving in my committee and providing insightful and diverse suggestions to improve this dissertation.

I am thankful to Professor Vishal Patel, Dr. Carlos Castillo, Dr. Jun-Cheng Chen, and Dr. Swami Sankaranarayanan for intense and fruitful research discussions that led to a good number of publications. I would like to thank my research colleagues at UMIACS for making my PhD years memorable.

I especially thank my mother, my father, and my brother for their unconditional love and support. My hard-working parents have sacrificed their lives to ensure proper education for my brother and myself. I would not have made it this far without them.

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute

Table of Contents

Chapter 1: Introduction

1.1 Motivation

Facial analytics is a challenging problem in computer vision and has been actively researched for over two decades [221]. The goal is to extract as much information as possible, such as key-point locations, pose, gender, ID, age, emotion, etc. from a face. Applications of this technology include detecting and identifying a person of interest from surveillance videos, active authentication of users cell phones, payment transactions using face biometrics, smart car, etc. In addition, there has been a growing interest in face recognition and verification from unconstrained images and videos which also involve subtasks, such as face detection, facial landmark localization, etc.

Face identification and verification systems typically have three modules. First, a face detector is needed for detecting and localizing faces in an image. Desirable properties of a face detector are robustness to variations in pose, illumination, and scale. Also, a good face detector should have consistent output and well localized bounding boxes. The second module localizes facial landmarks such as eye centers, tip of the nose, corners of the mouth, tips of ear lobes, etc. These landmarks are used to align faces which mitigates the effects of in-plane rotation and scaling. Third, a feature extractor encodes the identity information in a high-dimension descriptor. These descriptors are then used to compute a similarity score between two faces. An effective feature extractor needs to be robust to errors introduced by previous steps in the pipeline: face detection, landmark localization, and face alignment.

DCNNs have been shown to be very effective for several computer vision tasks like image classification [88, 171, 62], and object detection [48, 148, 108]. Deep CNNs (DCNNs) are highly non-linear regressors because of the presence of hierarchical convolutional layers with non-linear activations. DCNNs have been used as building blocks for all three modules of automatic face recognition: face detection [126, 145, 141, 143], facial keypoint localization [90, 145, 143], and face verification/identification [22, 7]. Ever-increasing computation power and availability of large datasets like CASIA-WebFace [209], UMDFaces [5, 6], MegaFace [81, 127], MS-Celeb-1M [57], VGGFace [136, 9], and WIDER Face [206] have led to significant performance gains from DCNNs. This is because of the large variations in pose, illumination, and scale of faces present in these datasets.

Existing approaches for the tasks of face detection, key-points localization and face verification/recognition have their own limitations. Most of the available face detectors are unable to detect tiny faces (face size less than 5% of the image size). The key challenge in unconstrained face detection using hand crafted features like Haar wavelets and HOG is that they do not capture the salient facial information at different resolution, poses and illumination conditions. Although DCNNs overcome most of these limitations, they are still below par in detecting very small faces which is crucial for an accurate face recognition system.

Existing methods for facial key-points localization task have focused primarily on detecting essential landmarks for frontal faces (pose yaw angles in between -60° and 60°). Most of these methods fail to correctly localize key-points for off-frontal or profile faces which occur frequently in images collected in unconstrained settings. Moreover, manually annotating facial key-points locations is a tedious task and hence it is very difficult to collect large number of training samples to train a DCNN for this task.

Face verification in unconstrained settings is a challenging problem. A typical pipeline for face verification based on deep learning includes training a deep network for subject classification with softmax loss, using the penultimate layer output as the feature descriptor, and generating a cosine similarity score given a pair of face images (see Fig. 1.1). Despite the excellent performance of recent face verification systems on datasets like Labeled Faces in the Wild (LFW) [70], it is still difficult to achieve similar accuracy on faces with extreme variations in viewpoints, resolution, occlusion and image quality. This is evident from the performance of traditional algorithms on the publicly available IJB-A [84] dataset. Data quality imbalance in the training set is one of the reasons for this performance gap. Existing face recognition training datasets contain large amount of high quality and frontal faces, whereas the unconstrained and difficult faces occur rarely. Most of the DCNN-based methods trained with softmax loss for classification tend to over-fit to the high quality data and fail to correctly classify faces acquired in challenging conditions.

Using the softmax loss function for training a face verification system has its own advantages and disadvantages. On the one hand, it can be easily implemented using inbuilt functions from the publicly available deep leaning toolboxes such as Caffe [75], Torch [27] and TensorFlow [1]. Unlike triplet loss [157], it does not have any restrictions

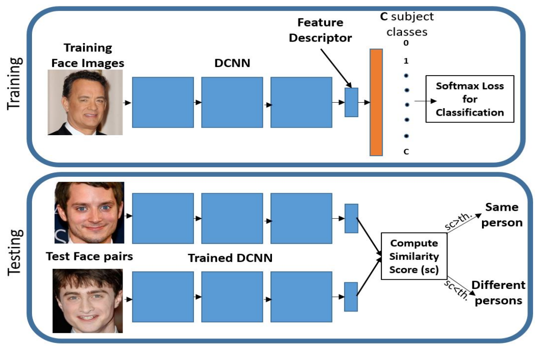


Figure 1.1: A general pipeline for training and testing a face verification algorithm using DCNN.

on the input batch size and converges quickly. The learned features are discriminative enough for efficient face verification without any metric learning. On the other hand, the softmax loss is biased to the sample distribution. Unlike contrastive loss [168] and triplet loss [157] which specifically attend to hard samples, the softmax loss maximizes the conditional probability of all the samples in a given mini-batch. Hence, it is suited to handle high quality faces, ignoring the rare difficult faces in a training mini-batch. We observe that the L_2-norm of features learned using softmax loss is informative of the quality of the face [135]. Features for good quality frontal faces have a high L_2-norm while blurry faces with extreme pose have low L_2-norm. Moreover, the softmax loss does not optimize the verification requirement of keeping positive pairs closer and negative pairs far from each other. In order to address this limitation, many methods either apply metric learning on top of softmax features [154, 21, 136, 22] or train an auxiliary loss [182, 168, 183] along with the softmax loss to achieve enhanced verification performance.

In recent years, it has been shown that DCNNs are vulnerable to small adversarial perturbations which, when added to the input image, can cause the network to mis-classify with high confidence [172, 53, 123, 122]. Adversarial images thus generated are often visually indistinguishable from the original images. Adversarial attacks have emerged as a potential threat to DCNN-based systems. Adversarial images can be used by a suspect to fool a face verification system, by letting the person go unidentified. These attacks can also cause self-driving cars to mis-classify scene objects such as a stop sign leading to adverse effects when these systems are deployed in real time. As networks move from the research labs to the field, they need to be designed in a way that they are not only accurate, but also robust to adversarial perturbations.

1.2 Proposed Approaches and Contributions

To solve the above mentioned issues present in the existing face recognition systems, and to safeguard them from adversarial attacks, we propose novel algorithms for face detection, face key-points and attributes detection, face verification/recognition and adversarial defense.

In the first part of this dissertation, we present two face detection algorithms based on deep pyramidal features. The first method, called Deep Pyramid Deformable Parts Model for Face Detection (DP2MFD), detects faces at multiple scales, poses and occlusion by efficiently integrating deep pyramid features [49] with Deformable Parts Model (DPM) [45]. It consists of two modules. The first one generates a seven level normalized deep feature pyramid for any input image of arbitrary size. Fixed-length features from each location in the pyramid are extracted using the sliding window approach. The second module is a linear SVM which takes these features as input to classify each location as face or non-face, based on their scores. The second face detector, called Deep Pyramid Single Shot Face Detector (DPSSD), improves upon DP2MFD and is faster and more accurate. Similar to DP2MFD it provides the output in a single pass of the network. In order to detect faces at different scales, we make use of the inbuilt pyramidal hierarchy present in a DCNN, instead of creating an image pyramid. This further reduces the processing time. We develop specific anchor filters for detecting tiny faces. We apply the bottom-up approach to incorporate contextual information, by adding features from deeper layers to the features from shallower layers. Our proposed face detectors are able to capture the meaningful structure of faces and perform significantly better than many competitive face detection algorithms.

In the second part of this dissertation, we present two multi-task learning (MTL) DCNN frameworks for face analytics. The first one, called HyperFace, performs face detection, landmarks localization, pose estimation and gender recognition by fusing the intermediate layers of the network. The second one, called All-In-One Face, builds upon HyperFace and performs additional tasks of smile detection, age estimation and face identification/verification. The MTL framework regularizes the shared parameter space using multiple loss functions and training domains, which improves the individual performance of each tasks altogether. We primarily use the key-points locations and the pose estimates obtained from All-in-One Face network in our end-to-end face recognition pipeline. The other attributes obtained from the network can be used for subsequent analysis.

In the third part of this dissertation, we explicitly focus on improving the tasks of face verification and identification. A typical pipeline for face verification includes training a deep network for subject classification with softmax loss, using the penultimate layer output as the feature descriptor, and generating a cosine similarity score given a pair of face images. The softmax loss function does not optimize the features to have higher similarity score for positive pairs and lower similarity score for negative pairs, which leads to a performance gap. We provide a symptomatic treatment to issues associated with the softmax loss. We propose Crystal Loss that adds a constraint on the features during training such that their L_2-norm remain constant. In other words, we restrict the features to lie on a hypersphere of a fixed radius. The proposed Crystal loss has two advantages. Firstly, it provides equal attention to both good and bad quality faces since all the features have the same L_2-norm now, which is essential for improved performance in unconstrained settings. Secondly, it strengthens the verification features by forcing the features from the same subject to be closer and features from different subjects to be far from each other in the normalized space. Therefore, it maximizes the margin for the normalized L_2 distance or cosine similarity score between negative and positive pairs. In this way, the proposed Crystal loss overcomes the limitations of the regular softmax loss. We also provide the design details for an end-to-end face recognition system containing three major modules. First, our proposed DPSSD face detector is used for detecting and localizing faces in an image. The second module localizes facial landmarks such using All-In-One Face algorithm. These landmarks are used to align faces which mitigates the effects of in-plane rotation and scaling. Third, a feature extractor encodes the identity information in a high-dimension descriptor. These descriptors are then used to compute a similarity score between two faces.

Additionally, we provide a comprehensive comparison of humans and computer for face identification in forensic applications. We show that the latest face verification/identification network trained with Crystal Loss scores better than median of the forensic facial examiners. We also show that fusing the judgments of single forensic facial examiners with the latest face recognition algorithm is more accurate than the combination of two examiners. Therefore, collaboration between humans and machines offers tangible benefits to face identification accuracy in important applications.

In the last part of this dissertation, we impose constraints on the sensitivity of the learned feature space and using our insight, propose a modified convolution operation that can desensitize the learned mapping in the direction of adversarial perturbations. We employ the property of *compactness* in the context of feature learning that would enhance a network's robustness to adversarial attacks. *Compactness* enforces the features to be bounded and lie in a closed space. It reduces the degree of freedom for the features to be learned. This restricts the extent to which a feature for perturbed image can move, making it less likely to cross the class boundary. To enforce *compactness* in the feature space, we explore the L_2-Softmax Loss proposed by Ranjan et al. [142]. The L_2-Softmax Loss establishes *compactness* by constraining the features to lie on a hypersphere of fixed radius, before applying the softmax loss. It brings the intra-class features close to each other and separates the inter-class features far apart. In this way, features from the original and the adversarial image are closer to each other using L_2-Softmax Loss, compared to training with regular softmax loss. Using these insights, we propose a novel convolution method, called *compact convolution*, that significantly enhances a network's robustness by ensuring compact feature learning at every layer of the network. A compact convolution module applies the L_2-normalization step and scaling operations to every input patch before applying the convolutional kernel in a sliding window fashion. Compact Convolutional Networks (CCNs), built using these modules, are highly robust compared to a typical CNN.

1.3 Organization

Chapter 2 discusses our DP2MFD and DPSSD for face detection algorithms in detail. Chapter 3 presents our multi-task approaches, namely HyperFace and All-In-One Face, for different face analytics tasks. In chapter 4, we introduce the Crystal loss function that significantly improves the performance of face verification task. We also discuss the end-to-end face verification/identification pipeline in this chapter. We present Compact Convolutional Network for adversarial defense in chapter 5 , and conclude this dissertation in chapter 6.

Chapter 2: Face Detection

Face detection is the first step in any face recognition/verification pipeline. It is a challenging problem that has been actively researched for over two decades [220], [213]. A face detection algorithm outputs the locations of all faces in a given input image, usually in the form of bounding boxes. A face detector needs to be robust to variations in pose, illumination, view-point, expression, scale, skin-color, some occlusions, disguises, make-up, etc. Current methods work well on images that are captured under user controlled conditions. However, their performance degrades significantly on images that have cluttered backgrounds and have large variations in face viewpoint, expression, skin color, occlusions and cosmetics. Most recent DCNN-based face detectors are inspired by general object detection approaches.

2.1 Previous Work

The seminal work of Viola and Jones [178] made face detection feasible in real world applications. They used cascaded classifiers on Haar-like features to detect faces. The cascade structure has been the subject of extensive research since then. Cascade detectors work well on frontal faces, however, sometimes they fail to detect profile or partially occluded faces. A recently developed joint cascade-based method [17] yields improved detection performance by incorporating a face alignment step in the cascade structure. Headhunter [116] uses rigid templates along similar lines. The method based on Aggregate Channel Features (ACF) [202] deploys a cascade of channel features while Pixel Intensity Comparisons Organized (Pico) [114] uses a cascade of rejectors for improved face detection.

Most of the hand-crafted face detectors are based on the DPM structure [46] where a face is defined as a collection of parts. These parts are trained side-by-side with the face using a spring-like constraint. They are fine-tuned to work efficiently with the HOG [30] features. A unified approach for face detection, pose estimation and landmark localization using the DPM framework was recently proposed in [225]. This approach defined a “part” at each facial landmark and used mixture of tree-structured models resilient to viewpoint changes. A properly trained simple DPM is shown to yield significant improvement for face detection in [116].

The limitation of traditional face detectors is more due to the features used than the classifiers. However, with recent advances in deep learning techniques and the availability of GPUs, it is becoming possible to use DCNNs for feature extraction. In has been shown in [88] that a DCNN pretrained with a large generic dataset such as Imagenet [31], can be used as a meaningful feature extractor. The deep features thus obtained have been used extensively for object detection. For instance, Regions with CNN (R-CNN) [48] computes region-based deep features and achieves state-of-art result on the Imagenet challenge. Methods like Overfeat [159] and Densenet [71] adopt a sliding window approach to detect objects from the $pool_5$ features. Deep Pyramid [49] and Spatial Pyramid [63] remove the fixed-scale input dependency from DCNNs which makes them attractive to be integrated with DPMs. Although, a lot of research on deep learning has focused on object detection and classification, very few have used deep features for face detection which is equally challenging because of large variations in pose, ethnicity, occlusions, etc. It was shown in [43] that DCNN features fine-tuned on faces are informative enough for face detection, and hence do not require an SVM classifier. They detect faces based on the heat map score obtained directly from the fifth convolutional layer. Although they report competitive results, detection performance for faces of various sizes and occlusions needs improvement. Most recent DCNN-based face detectors are inspired by general object detection approaches. They can be divided into two sub-categories: 1) region-based, and 2) sliding window-based.

Region-based approaches first generate a set of object-proposals and use a DCNN classifier to classify each proposal as a face or non-face. The first step is usually an off-the-shelf proposal generator like selective search [176]. Some recent detectors which use this approach are HyperFace [143], and All-in-One Face [145]. Instead of generating object proposals by a generic method, Faster R-CNN [148] uses a Region Proposal Network (RPN). Jiang and Learned-Miller use a Faster R-CNN network to detect faces in [76]. Similarly, [103] proposes a multi-task face detector based on the Faster-RCNN framework. Chen *et al.* [16] train a multi-task RPN for face detection and facial keypoint localization. This allows them to reduce the number of redundant face proposals and improve their quality. The Single Stage Headless face detector [126] is also based on an RPN.

Sliding window-based methods output face detections at every location in a feature map at a given scale. These detections are composed of a face detection score and a bounding box. This approach does not rely on a separate proposal generation step and is, thus, much faster than region-based approaches. In some methods [43], multi-scale detection is accomplished by creating an image pyramid at multiple scales. Similarly, Li *et al.* [98] use a cascade architecture for multiple resolutions. The Single Shot Detector (SSD) [108] is also a multi-scale sliding-window based object detector. However, instead of using an object pyramid for multi-scale processing, it utilizes the hierarchal nature of DCNNs. Methods like ScaleFace [207], and S3FD [217] use similar techniques for face detection.

In addition to the development of improved detection algorithms, rapid progress in face detection performance has been spurred by the availability of large annotated datasets. FDDB [74] consists of $2,845$ images containing a total of $5,171$ faces. Similar in scale is the MALF [203] dataset which contains $5,250$ images with $11,931$ faces. A much larger dataset is WIDER Face [206]. It contains over $32,000$ images containing faces with large variations in expression, scale, pose, illumination, etc. Most state-of-the-art face detectors are trained on the WIDER Face dataset. This dataset contains many tiny faces. Several of the above mentioned face detectors still struggle with detecting these small faces in images. Hu *et al.* [68] show that context is important for detecting such faces.

2.2 Deep Pyramid Deformable Parts Model for Face Detection

Here, we propose a face detector which detects faces at multiple scales, poses and occlusion by efficiently integrating deep pyramid features [49] with DPMs. We propose

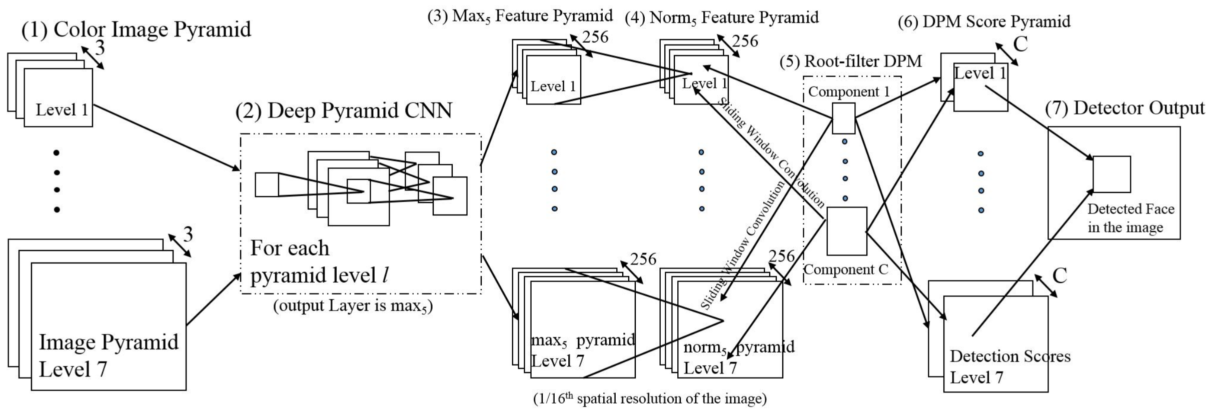


Figure 2.1: Overview of our approach. (1) An image pyramid is built from a color input image with level 1 being the lowest size. (2) Each pyramid level is forward propagated through a deep pyramid CNN [49] that ends at max variant of convolutional layer 5 (*max*$_5$). (3) The result is a pyramid of *max*$_5$ feature maps, each at 1/16th the spatial resolution of its corresponding image pyramid level. (4) Each *max*$_5$ level features is normalized using z-score to form *norm*$_5$ feature pyramid. (5) Each *norm*$_5$ feature level gets convoluted with every root-filter of a C-component DPM to generate a pyramid of DPM score (6). The detector outputs a bounding box for face location (7) in the image after non-maximum suppression and bounding box regression.

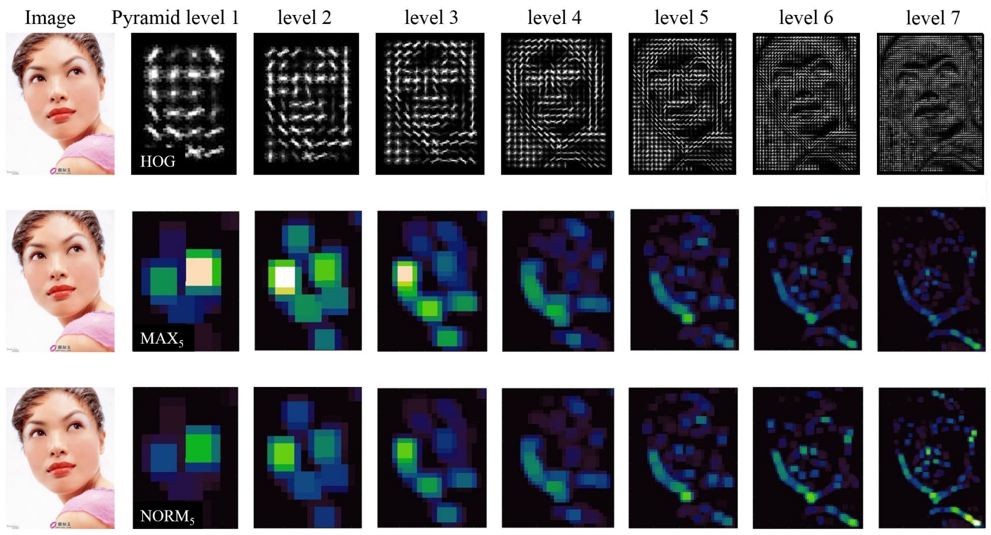


Figure 2.2: Comparison between HOG, *max*$_5$ and *norm*$_5$ feature pyramids. In contrast to *max*$_5$ features which are scale selective, *norm*$_5$ features have almost uniform activation intensities across all the levels.

adding a normalization layer to the deep CNN to reduce the bias in face sizes. Extensive experiments show that we outperform traditional face detectors on four challenging face detection datasets. Our proposed face detector, called Deep Pyramid Deformable Parts Model for Face Detection (DP2MFD), consists of two modules. The first one generates a seven level normalized deep feature pyramid for any input image of arbitrary size. Fixed-length features from each location in the pyramid are extracted using the sliding window approach. The second module is a linear SVM which takes these features as input to classify each location as face or non-face, based on their scores. In this section, we provide the design details of our face detector and describe its training and testing processes.

2.2.1 DPM Compatible Deep Feature Pyramid

We build our model using the feature pyramid network implementation provided in [49]. It takes an input image of variable size and constructs an image pyramid with seven levels. Each level is embedded in the upper left corner of a large (1713×1713 pixels) image and maintains a scale factor of $\sqrt{2}$ with its next lower level in the hierarchy. Using this image pyramid, the network generates a pyramid of 256 feature maps at the fifth convolution layer ($conv_5$). A 3×3 max filter is applied to the feature pyramid at a stride of one to obtain the max_5 layer which essentially incorporates the $conv_5$ "parts" information. Hence, it suffices to train a root-only DPM on the max_5 feature maps without explicitly training on DPM parts. A cell at location (j,k) in the max_5 layer corresponds to the pixel $(16j,16k)$ in the input image, with a highly overlapping receptive field of size 163×163 pixels. Despite having a large receptive field , the features are well localized to be effective for sliding window detectors.

It has been suggested in [49] that deep feature pyramids can be used as a replacement for HOG Pyramid in DPM implementation. However, this is not entirely obvious as deep features are different than HOG features in many aspects. Firstly, the deep features from max_5 layer have a receptive field of size 163×163 pixels, unlike HOG where the receptive region is localized to a bin of 8×8 pixels. As a result, max_5 features at face locations in the test images would be substantially different from that of a cropped face. This prohibits us from using the deep features of cropped faces as positive training samples, which is usually the first step in training a HOG-based DPM. Hence, we take a different approach of collecting positive and negative training samples from the deep feature pyramid itself. This procedure is described in detail in subsection 3.2.2.

Secondly, the deep pyramid features lack the normalization attribute associated with HOG. The feature activations vary widely in magnitude across the seven pyramid levels as shown in Figure 2.2. Typically, the activation magnitude for a face region decreases with the size of pyramid level. As a result, a large face detected by a fixed-size sliding window at a lower pyramid level will have a high detection score compared to a small face getting detected at a higher pyramid level. In order to reduce this bias to face size, we apply a z-score normalization step on the max_5 features at each level. For a 256-dimensional feature vector $x_{i,j,k}$ at the pyramid level i and location (j,k), the normalized feature $\hat{x}_{i,j,k}$ is computed as:

$$\hat{x}_{i,j,k} = \frac{x_{i,j,k} - \mu_i}{\sigma_i}, \tag{2.1}$$

where μ_i is the mean feature vector, and σ_i is the standard deviation for the pyramid level i. We refer to the normalized max_5 features as "$norm_5$". A root-only DPM is trained on the $norm_5$ feature pyramid using a linear SVM. Figure 2.1 shows the complete overview of our model.

2.2.2 Testing

At test time, each image is fed to the model described above to obtain the $norm_5$ feature pyramid. They are convolved with the fixed size root-filters for each component of DPM in a sliding window fashion, to generate a detection score at every location of the pyramid. Locations having scores above a certain threshold are mapped to their corresponding regions in the image. These regions undergo a greedy non-maximum suppression to prune low scoring detection regions with Intersection-Over-Union (IOU) overlap above 0.3. In order to localize the face as accurately as possible, the selected boxes undergo bounding box regression. Owing to the subsampling factor of 16 between the input image and $norm_5$ layer, the total number of sliding windows account to approximately 25k compared to approximately 250k for the HOG pyramid, which reduces the effective test-time.

2.2.3 Training

For training, both positive and negative faces are sampled directly from the $norm_5$ feature pyramid. The dimensions of root filters for DPM are decided by the aspect ratio distribution for faces in the dataset. The root-filter sizes are scaled down by a factor of 8 to match the face size in the feature pyramid. Since, a given training face maps its bounding box at each pyramid level, we choose the optimal level l for the corresponding positive sample by minimizing the sum of absolute difference between the dimensions of bounding box and the root filter at each level. For a root-filter of dimension (h,w) and bounding box dimension of (b_i^y, b_i^x) for the pyramid level i, l is given by

$$l = \arg\min_i |b_i^y - h| + |b_i^x - w|. \tag{2.2}$$

The ground truth bounding box at level l is then resized to fit the DPM root-filter dimensions. We finally extract the "$norm_5$" feature of dimension $h \times w \times 256$ from the shifted ground truth position in the level l as a positive sample for training.

The negative samples are collected by randomly choosing root-filter sized boxes from the normalized feature pyramid. Only those boxes having IOU less than 0.3 with the ground truth face at the particular level are considered as negative samples for training.

Once the training features are extracted, we optimize a linear SVM for each component of the root-only DPM. Since the training data is large to fit in the memory, we adopt the standard hard negative mining method [169, 46] to train the SVM. We also train a bounding box regressor to localize the detected face accurately. The procedure is similar to the bounding box regression used in R-CNN [48] , the only difference being our bounding box regressor is trained on the $norm_5$ features.

2.3 Deep Pyramid Single Shot Face Detector

We propose a novel DCNN-based face detector, called Deep Pyramid Single Shot Face Detector (DPSSD), that is fast and capable of detecting faces at a large variety of scales. It is especially good at detecting tiny faces (face size less than 5% of image size). Since face detection is a special case of generic object detection, many researchers have used an off-the-shelf object detector and fine-tuned it for the task of face detection [76]. However, in order to design an efficient face detector, it is crucial to address the following differences between the tasks of face and object detection. First, faces can occur at a much lower scale/size in an image compared to a general object. Typically, object detectors are not designed to detect at such a low resolution which is required for the task of face detection. Second, variations in the aspect ratio of faces are much less compared to those in a typical object. As faces incur less structural deformations compared to objects, they do not need any additional processing incorporated in object detection algorithms to handle multiple aspect ratios. We design our face detector to address these points.

We start with the Single Shot Detector (SSD) [108] trained on the truncated VGG-16 [163] network for the task of object detection. SSD [108] has a speed advantage over other object detectors like Faster R-CNN [148] since it is single stage and does not use proposals. The SSD approach is fully convolutional and generates a fixed number of bounding boxes and scores for inputs of fixed size. Additional convolutional layers are added to the end of the truncated VGG-16 [163] to detect objects at multiple scales. The objects are detected from multiple feature layers using different convolutional models for each layer. We modify the SSD [108] architecture and the approach in such a way that it is able to detect tiny faces efficiently. Fig. 2.3 shows the overall architecture of the proposed DPSSD face detector.

2.3.1 Anchor pyramid with fixed aspect-ratio

In order to detect faces at multiple scales, we leverage the feature pyramid structure inbuilt in the DCNN. We resize the input image such that the side with minimum length has a dimension of 512. After every convolutional block, max pooling is performed which reduces the dimension of feature maps by half and doubles the stride. For instance, the feature maps at conv3_3 layer have a minimum spatial dimension of 128. Additionally, a unit stride in this layer corresponds to 4 pixels stride in the original image. As shown in Table 2.1, initial layers of a DCNN have low stride in feature maps, which is beneficial for detecting tiny faces since small size anchors can be matched with high Jaccard overlap of 0.5. However, features from these layers have less discriminative ability. On the other hand, features from deeper layers are semantically stronger, but do not provide good spatial localization because of the large stride value. Hence, we choose the anchor sizes and the corresponding feature maps for generating detections with an optimal combination of low spatial stride and highly discriminative features.

Layer	Stride (pixels)	Anchor-Sizes (pixels)	#boxes
conv3_3	4	$16/\sqrt{2}$, 16	32768
conv4_3	8	$32/\sqrt{2}$, 32	8192
fc7	16	$64/\sqrt{2}$, 64	2048
conv6_2	32	$128/\sqrt{2}$, 128	512
conv7_2	64	$256/\sqrt{2}$, 256	128
conv8_2	128	$512/\sqrt{2}$, 512	32

Table 2.1: Statistics for different layers of DPSSD. The sizes of the two anchors and the stride are measured in pixels.

We choose 12 anchor boxes, each at a different scale. The largest anchor box has a size of 512. Each anchor box maintains a scale factor of $\sqrt{2}$ with its next lower level in the hierarchy. We apply these anchor boxes to generate detections from 6 different feature maps (see Table 2.1). Small-sized anchor boxes are applied to shallower feature maps while large-sized anchor boxes are applied to deeper feature maps. Unlike SSD [108], we make use of the conv3_3 layer for generating the detections since it has a high spatial resolution of 128. This helps us in detecting tiny faces of size as low as 8 pixels.

We fix the aspect ratio of every anchor box to the mean aspect-ratio for face (0.8). We compute this value from the WIDER Face [206] training dataset. For a given anchor size a, the anchor box $m \times n$ is calculated as:

$$m = \frac{a}{\sqrt{0.8}}, \quad n = a \times \sqrt{0.8}, \tag{2.3}$$

where m is the width and n is the height of the anchor box. Detection scores and bounding box offsets are provided at each location of the feature map for a given anchor box. Feature maps with larger spatial resolution result in more detection boxes. The number of detection boxes generated by every anchor layer for an image of size 512×512, is also provided in Table 2.1. The conv3_3 layer outputs the largest number of boxes since it has a spatial resolution of 128×128. All of these generated boxes are passed through the classifier network at the time of training.

2.3.2 Contextual Features from upsampling layers

It has been established that contextual information is useful for detecting tiny faces [68]. Although features from the conv3_3 layer have appropriate spatial resolution for tiny face detection, they are neither semantically strong nor they contain contextual information. In order to provide contextual information, we add a stack of bilinear upsampling and convolution layer at the end of the SSD [108] network. The 6 chosen layers (Table 2.1) are then added element-wise to these upsampled layers (see Fig. 2.3). Thus, the features become rich in both semantics and localization. The final detection boxes are generated from these upsampled layers using the anchor box matching technique.

Every output level generates two sets of detections, one for each anchor box cor-

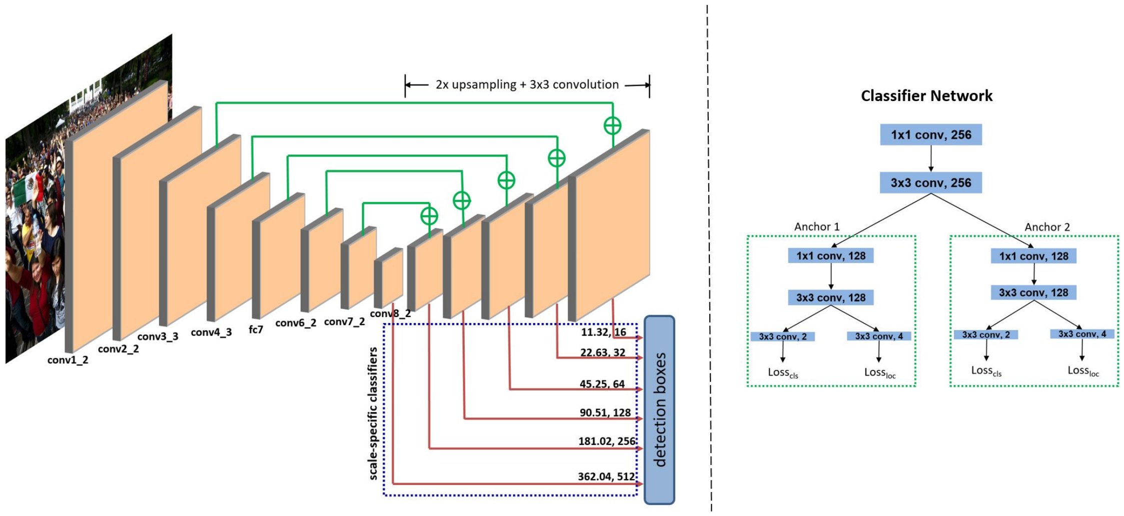


Figure 2.3: The network architecture for the proposed Deep Pyramid Single Shot Face Detector (DPSSD). Starting from the base SSD [108] network, we add upsampling layers to generate rich contextual features for face detection. The faces are pooled from six different layers of the network, with 2 scales at each layer. The numbers on the red arrows denote the anchor sizes for a given layer. The classifier network generates the face detection probability scores as well as the normalized bounding box offsets for every anchor (shown on right).

responding to the given layer. A classifier network (see Fig. 2.3 right) is attached to all the 6 output feature maps, that provides the classification probabilities and bounding box offsets corresponding to each of the 12 anchor boxes. The classifier network is branched into two to handle each anchor box separately. These branches are further subdivided into classification and regression subnetworks.

2.3.3 Training

We use the training set of WIDER Face [206] dataset to train our face detector. The network is initialized with the SSD [108] model for object detection. The new layers that are added are initialized randomly. We use a batch size of 8. The initial learning rate is set to 0.001 which is decreased by 0.5 after 30k, 50k and 60k iterations. Training is carried out till 70k iterations. The matching strategy is similar to SSD [108]. A location in the predictor feature map is labeled as positive class ($y_c = 1$) if the anchor box for that location has an Intersection-over-Union (IoU) overlap of 0.5 or more with any ground truth face. All the other locations are labeled as negative class ($y_c = 0$). For all the positive classes, we also perform bounding box regression. We use the binary cross-entropy loss for face classification and smooth-L1 loss for bounding box regression. The overall loss (L) is a weighted sum of classification loss (L_{cls}) and regression loss (L_{loc}) as shown in (2.4), (2.5) and (2.6). We use Caffe [75] library to train our network.

$$L_{cls} = -y_c \cdot \log(p_c) - (1 - y_c) \cdot \log(1 - p_c), \tag{2.4}$$

$$L_{loc} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L1}\ (t_i - v_i), \tag{2.5}$$

$$L = L_{cls} + \lambda \cdot y_c \cdot L_{loc}, \tag{2.6}$$

where y_c is the class label, p_c is the softmax probability obtained from the network, $v = \{v_x, v_y, v_w, v_h\}$ denote the ground-truth normalized bounding box regression targets while $t = \{t_x, t_y, t_w, t_h\}$ are the bounding box offsets predicted by the network. The value of λ is chosen to be 1. The smooth$_{L1}$ loss is defined in (2.7).

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{2.7}$$

The total number of detection boxes generated from an image is $43,680$. Out of these, only a few boxes (around 10-50) correspond to the positive class while others form the negative class. To avoid this large class imbalance we select only a few negative boxes such that the ratio of positive to negative class is $1:3$. We use hard negative mining to select these negative boxes as proposed in [108]. We use the data augmentation technique proposed in [108] to make the detector more robust to various face sizes.

2.3.4 Testing

At test time, the input image is resized such that the minimum side has the dimension of 512 pixels. The aspect ratio of the image is not changed. The image is then passed through the trained DPSSD face detector to get the detection scores and bounding box co-ordinates for different locations in the image. Non-maximum suppression (NMS) with threshold of 0.6 is used to filter out the redundant boxes. Since the outputs are generated in a single pass of the network, the total processing time is very low (100*ms*). To further improve the detection performance, we construct the image pyramid as discussed in HR [68] face detector. Performance evaluations on different face detection datasets are discussed in the experiments section.

2.4 Experimental Results

2.4.1 Datasets

We evaluate the proposed DP2MFD face detector on the following unconstrained face detection datasets - Annotated Face in-the-Wild (AFW) [225], Face Detection Dataset and Benchmark (FDDB) [74], Multi-Attribute Labelled Faces (MALF) [203] and the IARPA Janus Benchmark A (IJB-A) [84], [23] dataset. We train our DP2MFD face detector on the FDDB images using Caffe [75] for both 1-component (DP2MFD-1c) and 2-components (DP2MFD-2c) DPM. We evaluate the proposed DPSSD face detector on the following unconstrained face detection datasets - WIDER Face [206], Unconstrained Face Detection Dataset (UFDD) [125], Face Detection Dataset and Benchmark (FDDB) [74] and Pascal Faces [200]. We train our DPSSD face detector on the WIDER Face training set.

2.4.1.1 AFW Dataset

The AFW dataset [225] contains 205 images with 468 faces collected from Flickr. Images in this dataset contain cluttered backgrounds with large variations in both face viewpoint and appearance.

2.4.1.2 FDDB Dataset

The FDDB dataset [74] is the most widely used benchmark for unconstrained face detection. It consists of 2,845 images containing a total of 5,171 faces collected from news articles on the Yahoo website. All images were manually localized for generating the ground truth. The FDDB dataset has two evaluation protocols - discrete and continuous which essentially correspond to coarse match and precise match between the detection and the ground truth, respectively.

2.4.1.3 MALF Dataset

The MALF dataset [203] consists of 5,250 high-resolution images containing a total of 11,931 faces. The images were collected from Flickr and image search service provided by Baidu Inc. The average image size in this dataset is 573×638. On average, each image contains 2.27 faces with 46.97% of the images contain one face, 43.41% contain 2 to 4 faces, 8.30% contain 5 to 9 faces and 1.31% images contain more than 10 faces. Since this dataset comes with multiple annotated facial attributes, evaluations on attribute-specific subsets are proposed. Different subsets are defined corresponding to different combinations of attribute labels. In particular, ‘easy’ subset contains faces without any large pose, occluded or exaggerated expression variations and are larger than 60×60 in size and ‘hard’ subset contains faces that are larger than 60×60 in size with one of extreme pose or expression or occlusion variations. Furthermore, scale-specific evaluations are also proposed in which algorithms are evaluated on two subsets - ‘small’ and ‘large’. The ‘small’ subset contains images that have size smaller than 60×60 and the ‘’large’ subset contains images that have size larger than 90×90.

2.4.1.4 IJB-A Dataset

The IJB-A dataset contains images and videos from 500 subjects collected from online media [84], [23]. In total, there are 67,183 faces of which 13,741 are from images and the remaining are from videos. The locations of all faces in the IJB-A dataset were manually ground truthed by human annotators. The subjects were captured so that the dataset contains wide geographic distribution. All face bounding boxes are about 36 pixels or larger.

2.4.1.5 WIDER Face Dataset

The dataset contains $32,203$ images with $393,703$ face annotations, out of which 40% images are used for training, 10% for validation, and remaining 50% for test. It contains rich annotations, including occlusions, poses, event categories, and face bounding boxes. The faces possess large variations in scale, pose and occlusion. The dataset is extremely challenging for the task of tiny face detection, since the face width can be as low as 4 pixels.

2.4.1.6 UFDD Dataset

UFDD is a recent face detection dataset that captures several realistic issues not present in any existing dataset. It contains face images with weather-based degradations (rain, snow and haze), motion blur, focus blur, etc. Additionally, it contains distractor images that either contain non-human faces such as animal faces or no faces at all, which makes this dataset extremely challenging. It contains a total of $6,425$ images with $10,897$ face annotations.

2.4.1.7 PASCAL Faces Dataset

The PASCAL faces [200] dataset was collected from the test set of the person layout dataset which is a subset of PASCAL VOC [41]. The dataset contains $1,335$ faces from 851 images with large variations in appearance and pose.

2.4.2 DP2MFD Results

2.4.2.1 AFW Dataset Results

The precision-recall curves [*] of different academic as well as commercial methods on the AFW dataset are shown in Figure 2.4. Some of the academic face detection methods compared in Figure 2.4 include OpenCV implementations of the 2-view Viola-Jones algorithm, DPM [46], mixture of trees (Zhu et al.) [225], boosted multi-view face detector (Kalal et al.) [78], boosted exemplar [100] and the joint cascade methods [17]. As can

[*]The results of the methods other than our DP2MFD methods compared in Figure 2.4 were provided by the authors of [225], [17] and [100].

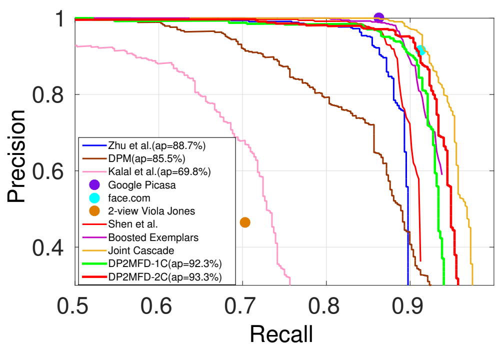
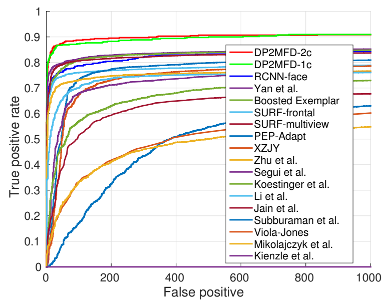


Figure 2.4: Performance evaluation on the AFW dataset.

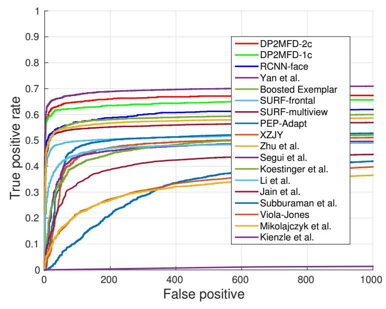
be seen from this figure, our method outperforms most of the academic detectors and performs comparably to a recently introduced joint cascade-based method [17] and the best commercial face detector Google Picassa. Note that the joint cascade-based method [17] uses face alignment to make the detection better and trains the model on 20,000 images. In contrast, we do not use any alignment procedure in our detection algorithm and train on only 2,500 images.

2.4.2.2 FDDB Dataset Results

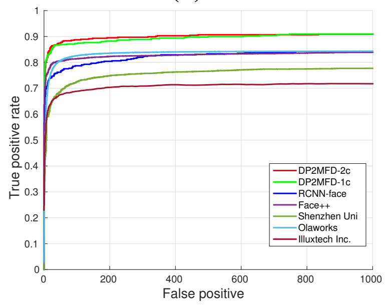
Figure 2.5 compares the performance of different academic and commercial detectors using the Receiver Operating Characteristic (ROC) curves on the FDDB dataset. The academic algorithms compared in Figure 2.5(a)-(b) include Yan et al. [198], boosted exemplar [100], SURF frontal and multi-view [102], PEP adapt [99], XZJY [162], Zhu et al. [225], Segui et al. [158], Koestinger et al. [87], Li et al. [101], Jain et al. [73], Subburaman et al. [165], Viola-Jones [178], Mikolajczyk et al. [120], Kienzle et al. [82] and the commercial algorithms compared in Figure 2.5(c)-(d) include Face++, the Olaworks face



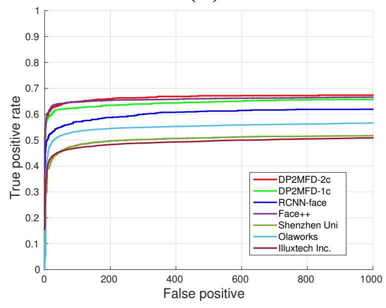
(a)



(b)



(c)



(d)

Figure 2.5: Performance evaluation on the FDDB dataset. (a) and (b) compare our method with previously published methods under the discrete and continuous protocols, respectively. Similarly, (c) and (d) compare our method with commercial systems under the discrete and continuous protocols, respectively.

detector, the IlluxTech frontal face detector and the Shenzhen University face detector [†].

As can be seen from this figure, our method significantly outperforms all previous academic and commercial detectors under the discrete protocol and performs comparably to the previous state-of-the-art detectors under the continuous protocol. A decrease in performance for the continuous case is mainly because of low IOU score obtained in matching our detectors' rectangular bounding box with elliptical ground truth mask for the FDDB dataset.

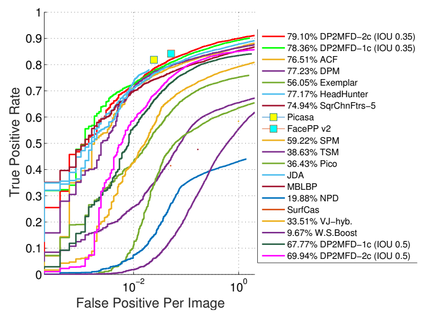
We also implemented an R-CNN method for face detection and evaluated it on the FDDB dataset. The R-CNN method basically selects face independent candidate regions from the input image and computes a 4096 dimensional fc_7 feature vector for each of them. An SVM trained on fc_7 features classifies each region as face or non-face based on the detection score. The method represented by "RCNN-face" performs better than most of the academic face detectors [225, 102, 99]. This shows the dominance of deep CNN features over HOG, SURF. However, RCNN-Face's performance is inferior to the DP2MFD method as the region selection process might miss a face from the image.

2.4.2.3 MALF Dataset Results

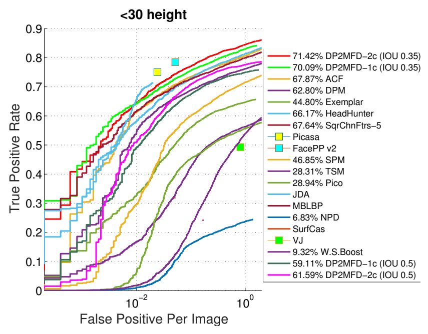
The performance of different algorithms, both from academia and industry, are compared in Figure 2.6 by plotting the True Positive Rate vs. False Positive Per Images curves [‡]. Some of the academic methods compared in Figure 2.6 include ACF [202],

[†]http://vis-www.cs.umass.edu/fddb/results.html

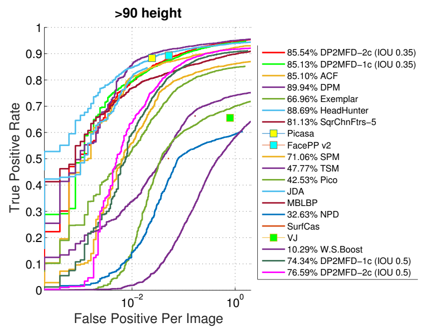
[‡]The results of the methods other than our DP2MFD methods compared in Figure 2.6 were provided by the authors of [203].



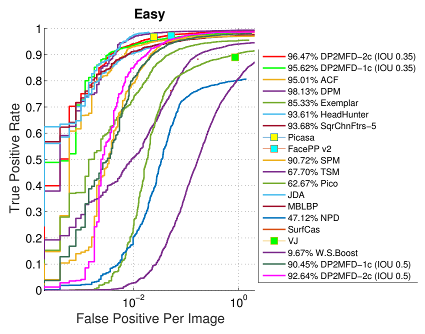
(a)



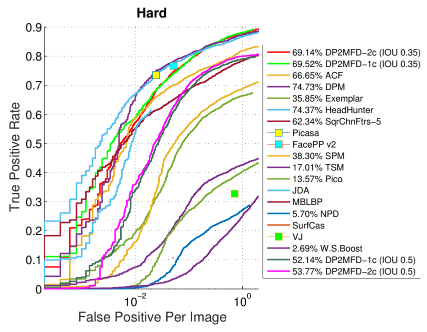
(b)



(c)



(d)



(e)

Figure 2.6: Fine-grained performance evaluation on the MALF dataset. (a) on the whole test set, (b) on the small faces sub-set, (c) on the large faces sub-set, (d) on the 'easy' faces sub-set and (e) on the 'hard' faces sub-set.

DPM [116], Exemplar method [100], Headhunter [116], TSM [225], Pico [114], NPD [105] and W. S. Boost [78]. From Figure 2.6(a), we see that overall the performance of our DP2MFD method is the best among the academic algorithms and is comparable to the best commercial algorithms FacePP-v2 and Picasa.

In the ‘small’ subset, denoted by < 30 height in Figure 2.6(b), the performance of all algorithms drop a little but our DP2MFD method still performs the best among the other academic methods. On the ’large’, ’easy, and ’hard’ subsets, the DPM method [116] performs the best and our DP2MFD method performs the second best as shown in Figure 2.6(c), (d) and (e), respectively. The DPM and Headhunter [116] are better as they train multiple models to fully capture faces in all orientations, apart from training on more than 20,000 samples.

We provide the results of our method for the IOU of 0.35 as well as 0.5 in Figure 2.6. Since the non-maximum suppression ensures that no two detections can have IOU> 0.3, the decrease in performance for IOU of 0.5 is mainly due to improper bounding box localization. One of the contributing factors might be the localization limitation of CNNs due to high amount of sub-sampling. In future, we plan to analyze this issue in detail.

2.4.2.4 IJB-A Dataset Results

Nine different face detection algorithms were evaluated on this dataset in [23]. Some of the algorithms compared in [23] include one commercial off the shelf (COTS) algorithm, three government off the shelf (GOTS) algorithms, two open source face detection algorithms (OpenCV’s Viola Jones and the detector provided in the Dlib library), and PittPat ver 4 and 5. In Figure 2.7 (a) and (b) we show the prevision vs. recall curves and the ROC curves, respectively corresponding to our method and one of the best reported methods in [23]. As can be seen from this figure, our method outperforms the best performing method reported in [23] by a large margin.

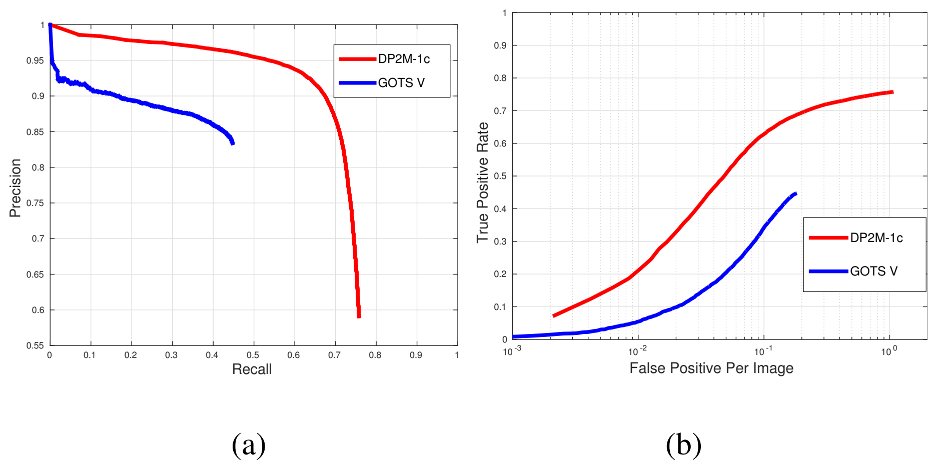
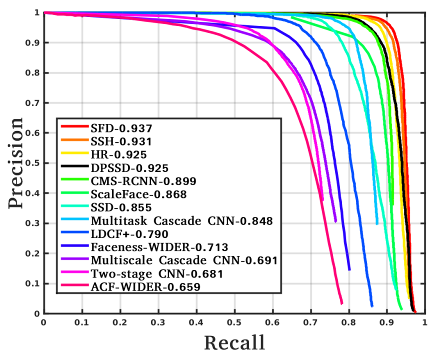


Figure 2.7: Performance evaluation on the IJB-A dataset. (a) Precision vs. recall curves. (b) ROC curves.

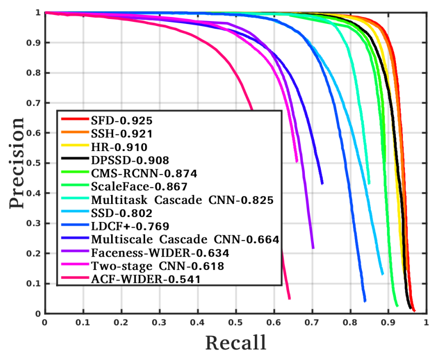
2.4.3 DPSSD Results

2.4.3.1 WIDER Face Dataset Results

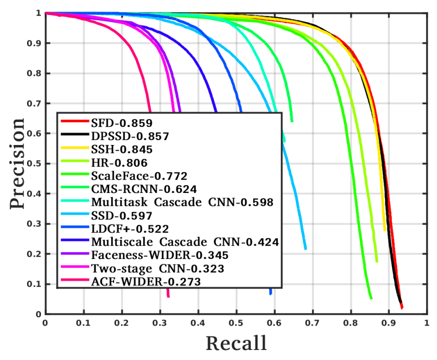
Fig. 2.8 provides the comparison of recently published face detection algorithms with the proposed DPSSD. We compare the performance of DPSSD with S^3FD [217], SSH [126], HR [68], CMS-RCNN [222], ScaleFace [207], Multitask Cascade [215], LDCF+ [130], Faceness [205], Multiscale Cascade [206], Two-stage CNN [206], and ACF [202]. We observe that DPSSD achieves competitive performance with state-of-the-art methods (S^3FD [217], SSH [126], and HR [68]). It achieves a mean average precision



(a) Easy



(b) Medium



(c) Hard

Figure 2.8: Performance evaluation on the WIDER Face [206] validation dataset for (a) Easy, (b) Medium, and (c) Hard faces. The numbers in the legend represent the mean average precision (mAP) for the corresponding method.

(mAP) of 0.925 and 0.908 on easy and medium difficulty set, respectively. On the hard set, it performs very close to the best performing method (S^3FD [217]) with the mAP of 0.857.

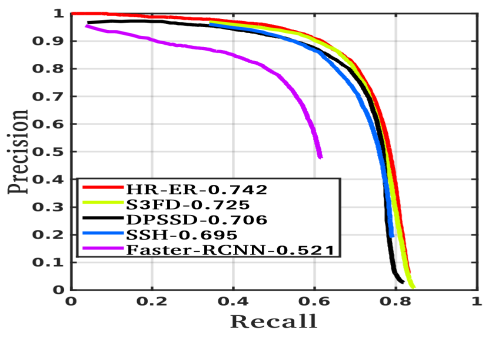
We also compare our method with the baseline face detector trained by fine-tuning SSD [20]. We outperform SSD [20] on easy, medium as well as hard set. Particularly on the hard set, DPSSD improves the mAP by a factor of 44% over the SSD [20]. It shows that redesigning anchor pyramid with fixed aspect ratio, and adding the upsampling layers helps tremendously in boosting the performance of face detection.

2.4.3.2 UFDD Dataset Results

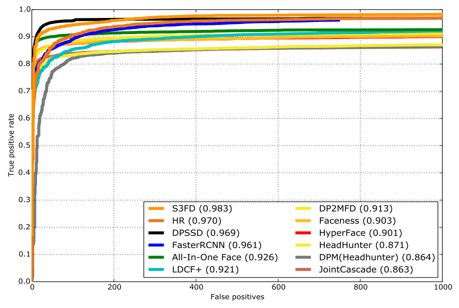
We compare our proposed method with S^3FD [217], SSH [126], HR [68], and Faster-RCNN [76] (see Fig. 2.9 (a)). Similar to WIDER Face [206] dataset, we achieve competitive results with a mAP of 0.706. Note that our algorithm was not fine-tuned on the UFDD dataset.

2.4.3.3 FDDB Dataset Results

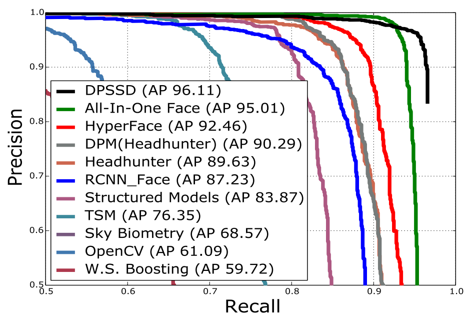
We evaluate the performance of our method on the discrete protocol using the Receiver Operating Characteristic (ROC) curves, as shown in Fig. 2.9 (b). As can be seen from the figure, our method exhibits competitive performance with state-of-the-art methods (S^3FD [217] and HR [68]) and achieves a mAP of 0.969. It should be noted that our method does not use any fine-tuning or bounding box regression specific to the FDDB dataset.



(a) UFDD



(b) FDDB



(c) PASCAL Faces

Figure 2.9: Performance evaluation (a) UFDD dataset [125], (b) FDDB dataset [74], and (c) PASCAL Faces dataset [200]. The numbers in the legend represent the mean average precision (mAP) for the corresponding dataset.

2.4.3.4 PASCAL Faces Dataset Results

Fig. 2.9 (c) compares the performance of different face detectors on this dataset. From the figure, we observe that our proposed DPSSD face detector achieves the best mAP of 96.11% on this dataset.

2.5 Discussion and Runtime

Its clear from these results that our DP2MFD-2c method performs slightly better than the DP2MFD-1c method. The DPSSD algorithm is better than DP2MFD because of the following reasons: 1) DPSSD is trained with more number of samples and much harder dataset compared to DP2MFD, 2) It is trained in an end-to-end manner while DP2MFD uses the features from a pretrained network, and 3) DPSSD employs bottom up feature aggregation which provides context information to detect tiny faces. Fig. 2.10 shows several detection results on the four datasets for DP2MFD. Fig. 2.11 shows a sample face detection output using DPSSD algorithm. It can be seen from this figures, that both the methods are able to detect profile faces as well as faces with large scale variations in images with cluttered background.

Our face detectors were tested on a machine with 4 cores, 12GB RAM, 1.6GHz processing speed. We used GTX 1080 GPU for processing. DP2MFD takes 500*ms* to generate the output while DPSSD takes less than 100*ms* for the same. This shows that DPSSD is not only superior in accuracy but also in speed. Hence, we use DPSSD face detector in our end-to-end face recognition pipeline.

2.6 Conclusions

In this chapter, we presented two algorithms for unconstrained face detection. The DP2MFD algorithm trains DPM for faces on deep feature pyramid. We add a normalization layer to the deep CNN architecture which reduces the bias in the face sizes. The DPSSD algorithm uses the inbuilt feature pyramid present in a DCNN to detect faces at multiple scales. Bottom-up feature aggregation provides the context information to accurately detect tiny faces. Extensive experiments on publicly available unconstrained face detection datasets demonstrate the effectiveness of our proposed approaches.

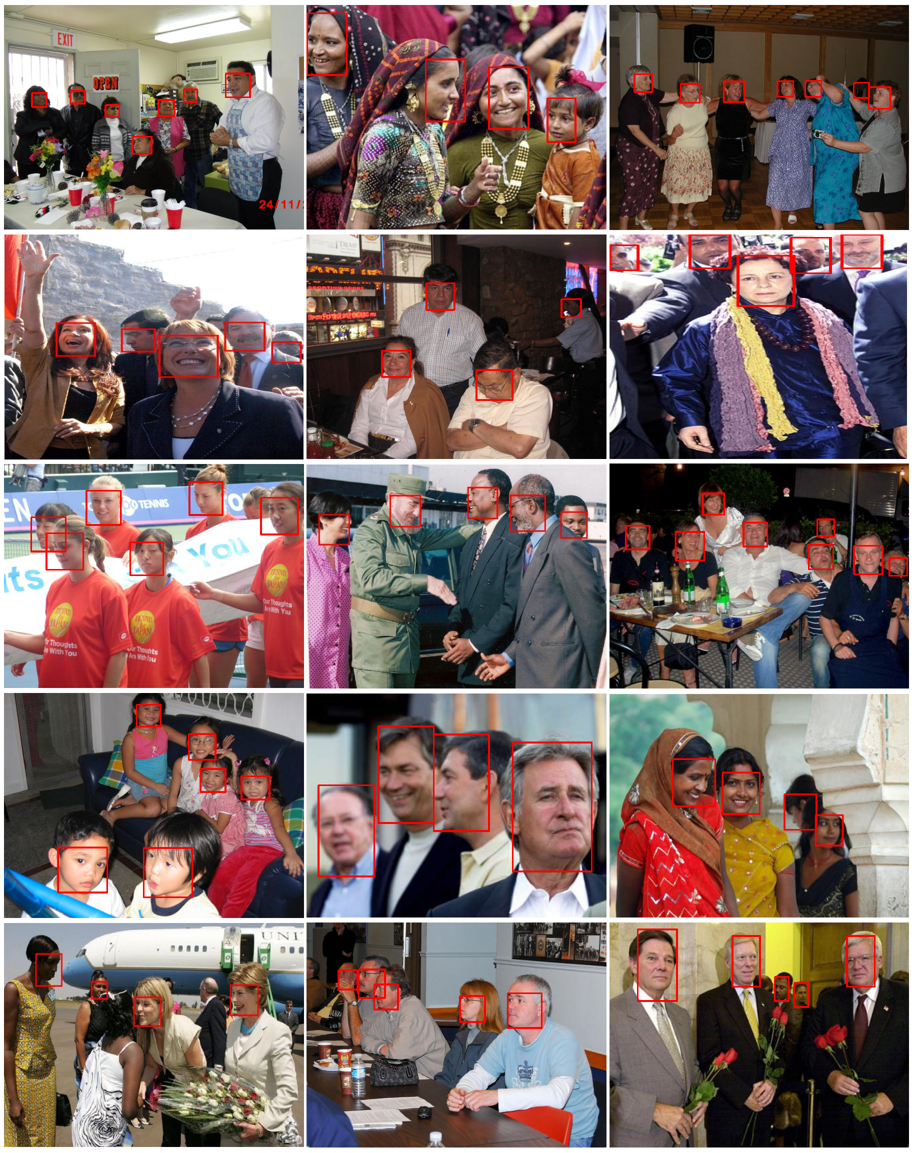


Figure 2.10: Qualitative results of DP2MFD face detector

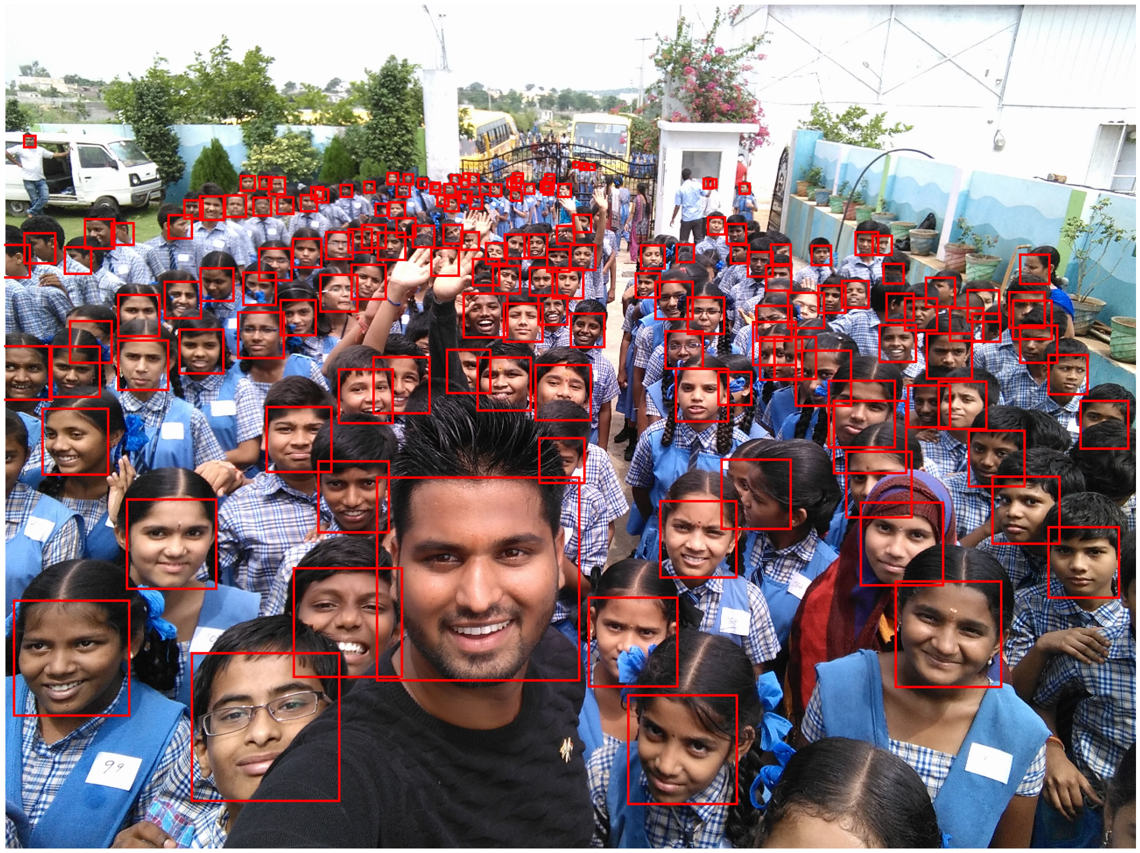
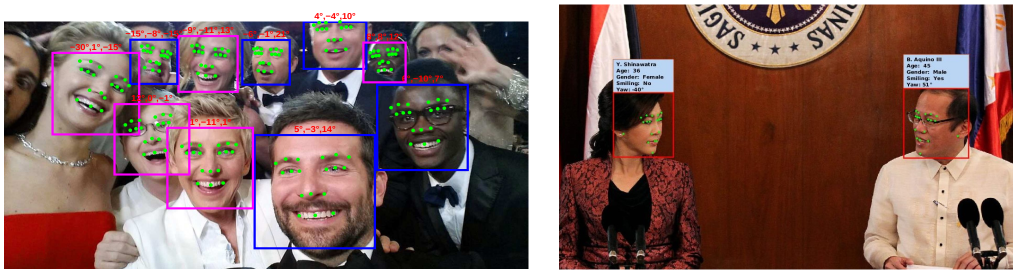


Figure 2.11: A sample output of DPSSD face detector.

Chapter 3: Multi-Task Learning for Face Analytics

Face analysis is a challenging and actively researched problem with applications in face recognition, emotion analysis, biometrics security, etc. Though the performance of few challenging face analysis tasks such as unconstrained face detection and face verification have greatly improved when DCNN-based methods are used, other tasks such as face alignment, head-pose estimation, gender and smile recognition are still challenging due to lack of large publicly available relevant training data. Furthermore, all these tasks have been approached as separate problems, which makes their integration into end-to-end systems inefficient. For example, a typical face recognition system needs to detect and align a face from the given image before determining the identity. This results in error accumulation across different modules. Even though the above mentioned tasks are correlated, existing methods do not leverage the synergy among them. It has been shown recently that jointly learning correlated tasks can boost the performance of individual tasks [225, 17].

In this chapter, we present two novel DCNN models for multi-task face analytics. The first one, called HyperFace, performs simultaneous face detection, facial landmarks localization, head pose estimation and gender recognition from a given image (see Figure 3.1(a)). The second one, called All-In-One Face, simultaneously solves the tasks of



(a) (b)

Figure 3.1: Sample outputs from (a) HyperFace, and (b) All-In-One Face. HyperFace can simultaneously detect faces, predict their landmarks locations, pose angles and gender from any unconstrained face image. All-In-One Face can additionally predict smile expression, age as well as the identity.

face detection, landmark localization, pose estimation, gender recognition, smile detection, age estimation and face verification and recognition (see Fig. 3.1(b)). We choose this set of tasks since they span a wide range of applications. We train a DCNN jointly in a multi-task learning (MTL) framework (Caruana [13]), such that parameters from shallower layers of DCNN are shared among all the tasks. In this way, the shallower layers learn a general representation common to all the tasks, whereas the deeper layers are more specific to the given task, which reduces over-fitting in the shared layers. Employing multiple tasks enables the network to learn the correlations between data from different distributions in an effective way. This approach saves both time and memory in an end-to-end system, since it can simultaneously solve the tasks and requires the storage of a single DCNN model instead of separate DCNN for each task. To the best of our knowledge, this is the first work which simultaneously solves a diverse set of face analysis tasks using a single DCNN in an end-to-end manner.

In HyperFace algorithm, we design a DCNN architecture to learn common features for multiple tasks and exploit the synergy among them. We exploit the fact that information contained in features is hierarchically distributed throughout the network as demonstrated in [212]. Lower layers respond to edges and corners, and hence contain better localization properties. They are more suitable for learning landmarks localization and pose estimation tasks. On the other hand, deeper layers are class-specific and suitable for learning complex tasks such as face detection and gender recognition. It is evident that we need to make use of all the intermediate layers of a DCNN in order to train different tasks under consideration. We refer the set of intermediate layer features as *hyperfeatures*. We borrow this term from [3] which uses it to denote a stack of local histograms for multilevel image coding.

Since a DCNN architecture contains multiple layers with hundreds of feature maps in each layer, the overall dimension of hyperfeatures is too large to be efficient for learning multiple tasks. Moreover, the hyperfeatures must be associated in a way that they efficiently encode the features common to multiple tasks. This can be handled using feature fusion techniques. Features fusion aims to transform the features to a common subspace where they can be combined linearly or non-linearly. Recent advances in deep learning have shown that DCNNs are capable of estimating an arbitrary complex function. Hence, we add a fusion sub-network to fuse the hyperfeatures. In order to learn the tasks, we train the feature sub-network and the fusion sub-network simultaneously in an end-to-end fashion using multiple loss functions. In this way, the features get better at understanding faces, which leads to improvements in the performances of individual tasks.

In All-In-One Face algorithm, we initialize our network with the DCNN model trained for face recognition task by Sankaranarayanan et al. [154]. We argue that a network pre-trained on face recognition task possesses fine-grained information of a face which can be used to train other face-related tasks efficiently. Task-specific sub-networks are branched out from different layers of this network depending on whether they rely on local or global information of the face. The complete network, when trained end-to-end, significantly improves the face recognition performance as well as other face analysis tasks.

3.1 Previous Work

Multi-Task Learning: Multi-task learning was first analyzed in detail by Caruana [13]. Since then, several approaches have adopted MTL for solving different problems in computer vision. One of the earlier approaches for jointly addressing the tasks of face detection, pose estimation, and landmark localization was proposed in [225] and later extended in [226]. This method is based on a mixture of trees with a shared pool of parts in the sense that every facial landmark is modeled as a part and uses global mixtures to capture the topological changes due to viewpoint variations. A joint cascade-based method was recently proposed in [17] for simultaneously detecting faces and landmark points on a given image. This method yields improved detection performance by incorporating a face alignment step in the cascade structure.

Multi-task learning using DCNNs has also been studied recently. Eigen and Fergus [37] proposed a multi-scale DCNN for simultaneously predicting depth, surface normals and semantic labels from an image. They apply DCNNs at three different scales where the output of the smaller scale network is fed as input to the larger one. Uber-

Net [85] adopts a similar concept of simultaneously training low-, mid- and high-level vision tasks. It fuses all the intermediate layers of a DCNN at three different scales of the image pyramid for multi-task training on diverse sets. Gkioxari et al. [50] train a DCNN for person pose estimation and action detection, using features only from the last layer. The use of MTL for face analysis is somewhat limited. Zhang et al. [219] used MTL-based DCNN for facial landmark detection along with the tasks of discrete head yaw estimation, gender recognition, smile and eye glass detection. In their method, the predictions for all theses tasks were pooled from the same feature space. Instead, we strategically design the network architecture such that the tasks exploit low level as well as high level features of the network. We also jointly predict the task of face detection and landmark localization. These two tasks always go hand-in-hand and are used in most end-to-end face analysis systems.

Feature Fusion: Fusing intermediate layers from a DCNN to bring both geometry and semantically rich features together has been used by quite a few methods. Hariharan et al. [59] proposed Hypercolumns to fuse pool2, conv4 and fc7 layers of AlexNet [88] for image segmentation. Yang and Ramanan [208] proposed DAG-CNNs, which extract features from multiple layers to reason about high, mid and low-level features for image classification. Sermanet et al. [160] merge the 1st stage output of DCNN to the classifier input after sub-sampling, for the application of pedestrian detection.

Face detection: Viola-Jones detector [178] is a classic method which uses cascaded classifiers on Haar-like features to detect faces. This method provides realtime face detection, but works best for full, frontal, and well lit faces. Deformable Parts Model [46]-based face detection methods have also been proposed in the literature where a face is essentially defined as a collection of parts [225], [116]. It has been shown that in unconstrained face detection, features like HOG or Haar wavelets do not capture the discriminative facial information at different illumination variations or poses. To overcome these limitations, various DCNN-based face detection methods have been proposed in the literature [141], [98], [205], [43], [201]. These methods have produced state-of-the-art results on many challenging publicly available face detection datasets. Some of the other recent face detection methods include NPDFaces [105], PEP-Adapt [99], and [17].

Landmarks localization: Fiducial points extraction or landmarks localization is one of the most important steps in face recognition. Several approaches have been proposed in the literature. These include both regression-based [10], [199], [197], [195], [80], [175] and model-based [28], [117], [104] methods. While the former learns the shape increment given a mean initial shape, the latter trains an appearance model to predict the keypoint locations. DCNN-based landmark localization methods have also been proposed in recent years [166], [219],[90] and have achieved remarkable performance.

Although much work has been done for localizing landmarks for frontal faces, limited attention has been given to profile faces which occur more often in real world scenarios. Jourabloo and Liu recently proposed PIFA [77] that estimates 3D landmarks for large pose face alignment by integrating a 3D point distribution model with a cascaded coupled-regressor. Similarly, 3DDFA [227] fits a dense 3D model by estimating its parameters using a DCNN. Zhu et al. [223] proposed a cascaded compositional learning approach that combines shape prediction from multiple domain specific regressors.

Pose estimation: The task of head pose estimation is to infer the orientation of person's head relative to the camera view. It is useful in face verification for matching face similarity across different orientations. Non-linear manifold-based methods have been proposed in [4], [67], [164] to classify face images based on pose. A survey of various head pose estimation methods is provided in [124].

Gender and Smile Classification: The tasks of gender and smile classification from unconstrained images have been considered as a part of facial attribute inference. Recently, Liu et al. [111] released CelebA dataset containing about $200,000$ near-frontal images with 40 attributes including gender and smile, which accelerated the research in this field [180, 216, 36]. Faces of the world [40] challenge dataset further advanced the research on these tasks for faces with varying scale, illumination and pose [97, 177, 214].

Age Estimation: Age Estimation is the task of estimating the real or apparent age of a person based on their face image. Few methods have already surpassed human error for the apparent age estimation challenge [39] using DCNNs [151, 19].

Face Verification: is the task of predicting whether a pair of faces belong to the same person. Recent methods such as DeepFace [173], Facenet [157], VGG-Face [136] have significantly improved the verification accuracy on the LFW [70] dataset by training DCNN models on millions of annotated data. However, it is still a challenging problem for unconstrained faces with large variations in viewpoint and illumination (IJB-A [84] dataset). We address this issue by regularizing the DCNN parameters using the MTL framework, with only half-a-million samples (CASIA [209]) for training.

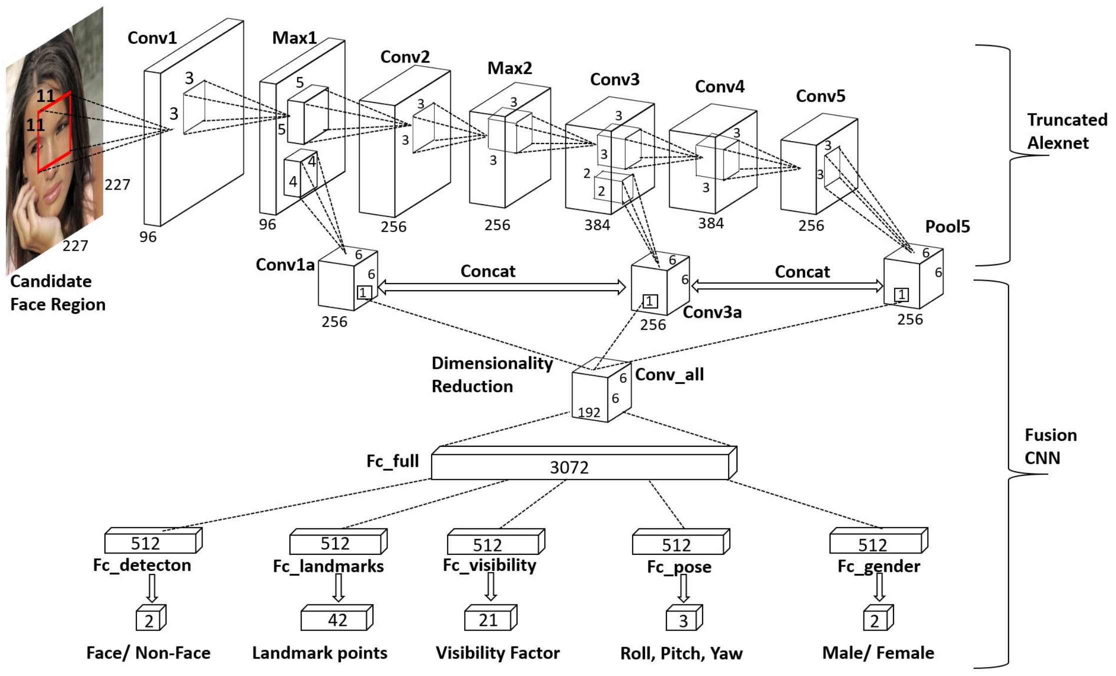


Figure 3.2: The architecture of the proposed HyperFace. The network is able to classify a given image region as face or non-face, estimate the head pose, locate face landmarks and recognize gender.

3.2 Proposed Method - HyperFace

In this section, we propose a single DCNN model for simultaneous face detection, landmark localization, pose estimation and gender classification. The network architecture is deep in both vertical and horizontal directions, i.e., it has both top-down and lateral connections, as shown in Figure 4.4. In this section, we provide a brief overview of the system and then discuss the different components in detail.

The proposed algorithm called *HyperFace* consists of three modules. The first one generates class independent region-proposals from the given image and scales them to 227×227 pixels. The second module is a DCNN which takes in the resized candidate regions and classifies them as face or non-face. If a region gets classified as a face, the network additionally provides facial landmarks locations, estimated head pose and gender information. The third module is a post-processing step which involves Iterative Region Proposals (IRP) and Landmarks-based Non-Maximum Suppression (L-NMS) to boost the face detection score and improve the performance of individual tasks.

3.2.1 HyperFace Architecture

We start with Alexnet [88] for image classification. The network consists of five convolutional layers along with three fully connected layers. We initialize the network with the weights of R-CNN_Face network trained for face detection task as described in Section 3.3. All the fully connected layers are removed as they encode image-classification specific information, which is not needed for pose estimation and landmarks extraction. We exploit the following two observations to create our network. 1) The features in

DCNN are distributed hierarchically in the network. While the lower layer features are effective for landmarks localization and pose estimation, the higher layer features are suitable for more complex tasks such as detection or classification[212]. 2) Learning multiple correlated tasks simultaneously builds a synergy and improves the performance of individual tasks as shown in [17, 219]. Hence, in order to simultaneously learn face detection, landmarks, pose and gender, we need to fuse the features from the intermediate layers of the network (hyperfeatures), and learn multiple tasks on top of it. Since the adjacent layers are highly correlated, we do not consider all the intermediate layers for fusion.

We fuse the max_1, $conv_3$ and $pool_5$ layers of Alexnet, using a separate network. A naive way for fusion is directly concatenating the features. Since the feature maps for these layers have different dimensions $27 \times 27 \times 96$, $13 \times 13 \times 384$, $6 \times 6 \times 256$, respectively, they cannot be easily concatenated. We therefore add $conv_{1a}$ and $conv_{3a}$ convolutional layers to $pool_1$, $conv_3$ layers to obtain consistent feature maps of dimensions $6 \times 6 \times 256$ at the output. We then concatenate the output of these layers along with $pool_5$ to form a $6 \times 6 \times 768$ dimensional feature map. The dimension is still quite high to train a multi-task framework. Hence, a 1×1 kernel convolution layer ($conv_{all}$) is added to reduce the dimensions [171] to $6 \times 6 \times 192$. We add a fully connected layer (fc_{all}) to $conv_{all}$, which outputs a 3072 dimensional feature vector. At this point, we split the network into five separate branches corresponding to the different tasks. We add $fc_{detection}$, $fc_{landmarks}$, $fc_{visibility}$, fc_{pose} and fc_{gender} fully connected layers, each of dimension 512, to fc_{all}. Finally, a fully connected layer is added to each of the branch to predict the individual task labels. After every convolution or a fully connected layer, we deploy the Rectified Linear Unit (ReLU). We do not include any pooling operation in the fusion network as it provides local invariance which is not desired for the face landmark localization task. Task-specific loss functions are then used to learn the weights of the network.

3.2.2 Training

We use the AFLW[86] dataset for training the HyperFace network. It contains $25,993$ faces in $21,997$ real-world images with full pose, expression, ethnicity, age and gender variations. It provides annotations for 21 landmark points per face, along with the face bounding-box, face pose (yaw, pitch and roll) and gender information. We randomly selected 1000 images for testing, and used the rest for training the network. Different loss functions are used for training the tasks of face detection, landmark localization, pose estimation and gender classification.

Face Detection: We use the Selective Search [176] algorithm in R-CNN [48] to generate region proposals for faces in an image. A region having an Intersection over Union (IOU) overlap of more than 0.5 with the ground truth bounding box is considered a positive sample ($l = 1$). The candidate regions with IOU overlap less than 0.35 are treated as negative instances ($l = 0$). All the other regions are ignored. We use the softmax loss function given by (3.1) for training the face detection task.

$$loss_D = -(1-l)\cdot\log(1-p) - l\cdot\log(p), \tag{3.1}$$

where p is the probability that the candidate region is a face. The probability values p and $1-p$ are obtained from the last fully connected layer for the detection task.

Landmarks Localization: We use 21 point markups for face landmarks locations as provided in the AFLW[86] dataset. Since the faces have full pose variations, some of the landmark points are invisible. The dataset provides the annotations for the visible landmarks. We consider bounding-box regions with IOU overlap greater than 0.35 with the ground truth for learning this task, while ignoring the rest. A region can be characterized by $\{x, y, w, h\}$ where (x, y) are the co-ordinates of the center of the region and w,h are the width and height of the region respectively. Each visible landmark point is shifted with respect to the region center (x, y), and normalized by (w, h) as given by (3.2)

$$(a_i, b_i) = \left(\frac{x_i - x}{w}, \frac{y_i - y}{h} \right). \tag{3.2}$$

where (x_i, y_i)'s are the given ground truth fiducial co-ordinates. The (a_i, b_i)'s are treated as labels for training the landmark localization task using the Euclidean loss weighted by the visibility factor. The loss in predicting the landmark location is computed from (3.3)

$$loss_L = \frac{1}{2N} \sum_{i=1}^{N} v_i((\hat{x}_i - a_i)^2 + ((\hat{y}_i - b_i)^2), \tag{3.3}$$

where $(\hat{x}_i, \hat{y}_i)$ is the i^{th} landmark location predicted by the network, relative to a given region, N is the total number of landmark points (21 for AFLW[86]). The visibility factor v_i is 1 if the i^{th} landmark is visible in the candidate region, else it is 0. This implies that there is no loss corresponding to invisible points and hence they do not take part during back-propagation.

Learning Visibility: We also learn the visibility factor in order to test the presence of the predicted landmark. For a given region with overlap higher than 0.35, we use a simple Euclidean loss to train the visibility as shown in (3.4)

$$loss_V = \frac{1}{N} \sum_{i=1}^{N} (\hat{v}_i - v_i)^2, \tag{3.4}$$

where $\hat{v}_i$ is the predicted visibility of i^{th} landmark. The true visibility v_i is 1 if the i^{th} landmark is visible in the candidate region, else it is 0.

Pose Estimation: We use the Euclidean loss to train the head pose estimates of roll (p_1), pitch (p_2) and yaw (p_3). We compute the loss for a candidate region having an overlap more than 0.5 with the ground truth, from (3.5)

$$loss_P = \frac{(\hat{p_1} - p_1)^2 + (\hat{p_2} - p_2)^2 + (\hat{p_3} - p_3)^2}{3}, \tag{3.5}$$

where $(\hat{p_1}, \hat{p_2}, \hat{p_3})$ are the estimated pose labels.

Gender Recognition: Predicting gender is a two class problem similar to face detection. For a candidate region with overlap of 0.5 with the ground truth, we compute the softmax loss given in (3.6)

$$loss_G = -(1-g) \cdot \log(1-p_g) - g \cdot \log(p_g), \tag{3.6}$$

where $g = 0$ if the gender is male, or else $g = 1$. Here, (p_0, p_1) is the two dimensional probability vector computed from the network.

The total loss is computed as the weighted sum of the five individual losses as shown in (3.7)

$$loss_{full} = \sum_{i=1}^{i=5} \lambda_{t_i} loss_{t_i}, \tag{3.7}$$

where t_i is the i^{th} element from the set of tasks $T = \{D, L, V, P, G\}$. The weight parameter λ_{t_i} is decided based on the importance of the task in the overall loss. We choose ($\lambda_D = 1, \lambda_L = 5, \lambda_V = 0.5, \lambda_P = 5, \lambda_G = 2$) for our experiments. Higher weights are assigned to landmark localization and pose estimation tasks as they need spatial accuracy.

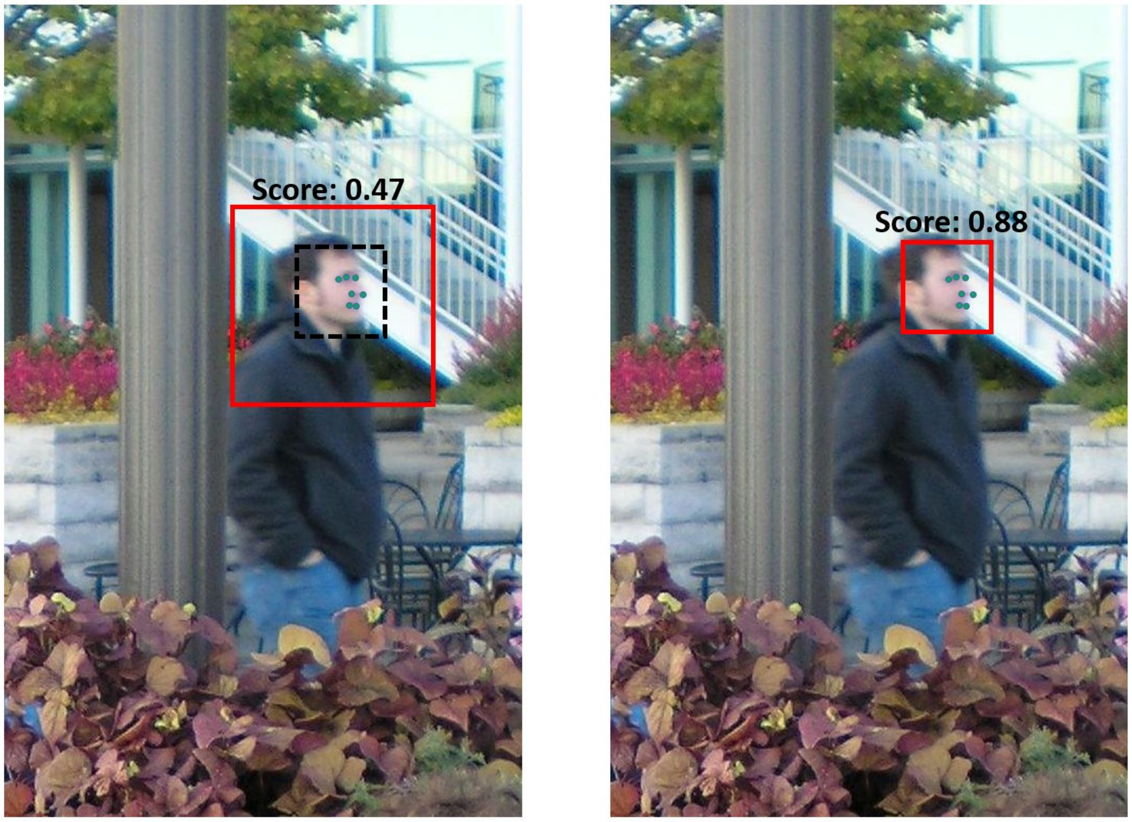


Figure 3.3: Candidate face region (red box on left) obtained using Selective Search gives a low score for face detection, while landmarks are correctly localized. We generate a new face region (red box on right) using the landmarks information and feed it through the network to increase the detection score.

3.2.3 Testing

From a given test image, we first extract the candidate region proposals using[176]. For each region, we predict the task labels by a forward-pass through the HyperFace network. Only those regions, whose detection scores are above a certain threshold, are classified as face and processed for subsequent tasks. The predicted landmark points are scaled and shifted to the image co-ordinates using (3.8)

$$(x_i, y_i) = (\hat{x}_i w + x, \hat{y}_i h + y), \tag{3.8}$$

where $(\hat{x}_i, \hat{y}_i)$ are the predicted locations of the i^{th} landmark from the network, and $\{x, y, w, h\}$ are the region parameters defined in (3.2). Points obtained with predicted visibility less than a certain threshold are marked invisible. The pose labels obtained from the network are the estimated roll, pitch and yaw for the face region. The gender is assigned according to the label with maximum predicted probability.

There are two major issues while using the proposal-based face detector. First, the proposals might not be able to capture small and difficult faces, hence reducing the overall recall of the system. Second, the proposal boxes might not be well localized with the actual face region. It is a common practice to use bounding-box regression [48] as a post processing step to improve the localization of the detected face box. This adds an additional burden of training regressors to learn the transformation from the detected candidate box to the annotated face box. Moreover, the localization is still weak since the regressors are usually linear. Recently, Gidaris and Komodakis proposed LocNet [47] which tries to solve these limitations by refining the detection bounding box. Given a set of initial bounding box proposals, it generates new sets of bounding boxes that maximize the likelihood of each row and column within the box. It allows an accurate inference of bounding box under a simple probabilistic framework.

Instead of using the probabilistic framework [47], we solve the above mentioned issues in an iterative way using the predicted landmarks. The fact that we obtain landmark locations along with the detections, enables us to improve the post-processing step so that all the tasks benefit from it. We propose two novel methods: Iterative Region Proposals (IRP) and Landmarks-based Non-Maximum Suppression (L-NMS) to improve the performance. IRP improves the recall by generating more candidate proposals by using the predicted landmarks information from the initial set of region proposals. On the other hand, L-NMS improves the localization by re-adjusting the detected bounding boxes according to the predicted landmarks and performing NMS on top of them. No additional training is required for these methods.

Iterative Region Proposals (IRP): We use a fast version of Selective Search[176] which extracts around 2000 regions from an image. We call this version *Fast_SS*. It is quite possible that some faces with poor illumination or small size fail to get captured by any candidate region with a high overlap. The network would fail to detect that face due to low score. In these situations, it is desirable to have a candidate box which precisely captures the face. Hence, we generate a new candidate bounding box from the predicted landmark points using the FaceRectCalculator provided by [86], and pass it again through the network. The new region, being more localized yields a higher detection score and improves the corresponding tasks output, thus increasing the recall. This procedure can be repeated (say T time), so that boxes at a given step will be more localized to faces as compared to the previous step. From our experiments, we found that the localization component saturates in just one step ($T = 1$), which shows the strength of the predicted landmarks. The pseudo-code of IRP is presented in Algorithm 1. The usefulness of IRP can be seen in Figure 3.3, which shows a low-resolution face region cropped from the top-right image in Figure 3.19.

Algorithm 1 Iterative Region Proposals

```
1: $\mathbf{boxes} \leftarrow selective\_search(\mathbf{image})$
2: $\mathbf{scores} \leftarrow get\_hyperface\_scores(\mathbf{boxes})$
3: $\mathbf{detected\_boxes} \leftarrow \mathbf{boxes}(scores \geq threshold)$
4: $\mathbf{new\_boxes} \leftarrow \mathbf{detected\_boxes}$
5: for stage = 1 to T do
6:     $\mathbf{fids} \leftarrow get\_hyperface\_fiducials(\mathbf{new\_boxes})$
7:     $\mathbf{new\_boxes} \leftarrow FaceRectCalculator(\mathbf{fids})$
8:     $\mathbf{deteced\_boxes} \leftarrow [\mathbf{deteced\_boxes} | \mathbf{new\_boxes}]$
9: end
10: $\mathbf{final\_scores} \leftarrow get\_hyperface\_scores(\mathbf{detected\_boxes})$
```

Landmarks-based Non-Maximum Suppression (L-NMS): The traditional approach of non-maximum suppression involves selecting the top scoring region and discarding all the other regions with overlap more than a certain threshold. This method can fail in the following two scenarios: 1) If a region corresponding to the same detected face has less overlap with the highest scoring region, it can be detected as a separate face. 2) The highest scoring region might not always be localized well for the face, which can create some discrepancy if two faces are close together. To overcome these issues, we perform NMS on a new region whose bounding box is defined by the boundary coordinates as $[\min_i x_i, \min_i y_i, \max_i x_i, \max_i y_i]$ of the landmarks for the given region. In this way, the candidate regions would get close to each other, thus decreasing the ambiguity of the overlap and improving the localization.

Algorithm 2 Landmarks-based NMS

```
1: Get detected_boxes from Algorithm 1
2: fids ← $get\_hyperface\_fiducials(\textbf{detected\_boxes})$
3: precise_boxes ← $[min_x, min_y, max_x, max_y](\textbf{fids})$
4: faces ← $nms(\textbf{precise\_boxes}, overlap)$
5: for each face in faces do
6:     top-k_boxes ← Get top-k scoring boxes
7:     final_fids ← $median(fids(\textbf{top-k\_boxes}))$
8:     final_pose ← $median(pose(\textbf{top-k\_boxes}))$
9:     final_gender ← $median(gender(\textbf{top-k\_boxes}))$
10:    final_visibility ← $median(visibility(\textbf{top-k\_boxes}))$
11:    final_bounding_box ← $FaceRectCalculator(\textbf{final\_fids})$
12: end
```

We apply landmarks-based NMS to keep the top-k boxes, based on the detection scores. The detected face corresponds to the region with maximum score. The landmark points, pose estimates and gender classification scores are decided by the median of the top k boxes obtained. Hence, the predictions do not rely only on one face region, but considers the votes from top-k regions for generating the final output. From our experiments, we found that the best results are obtained with the value of k being 5. The pseudo-code for L-NMS is given in Algorithm 2.

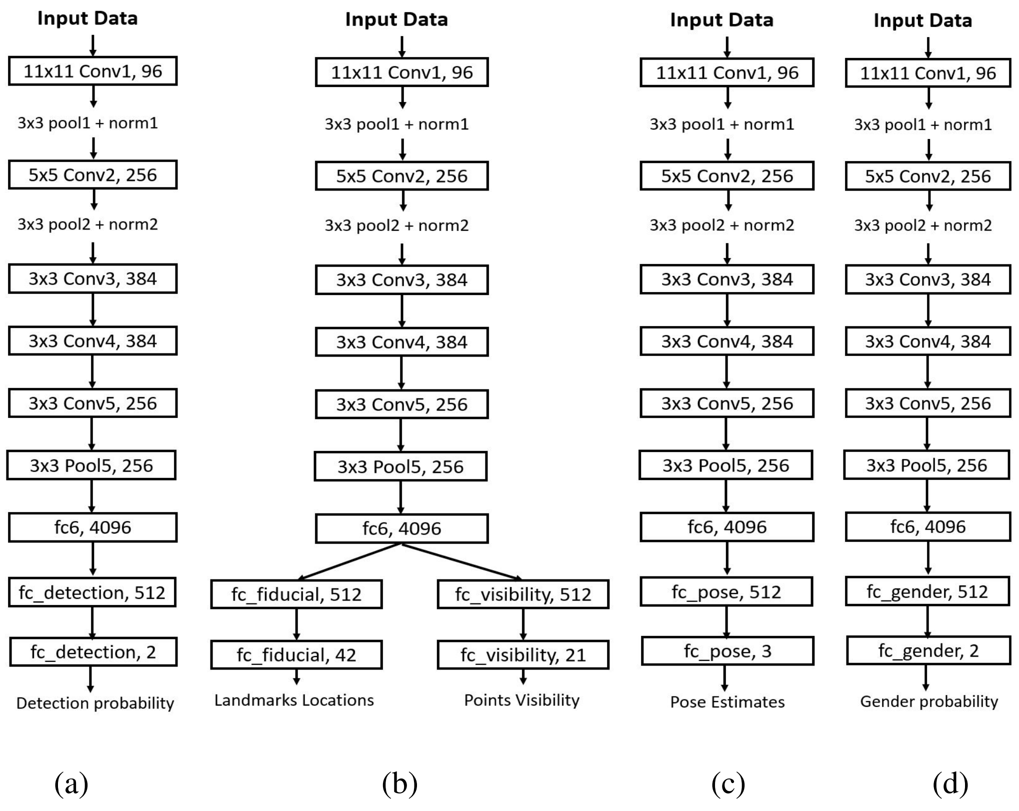


Figure 3.4: R-CNN-based network architectures for (a) Face Detection (R-CNN_Face), (b) Landmark Localization (R-CNN_Fiducial), (c) Pose Estimation (R-CNN_Pose), and (d) Gender Recognition (R-CNN_Gender). The numbers on the left denote the kernel size and the numbers on the right denote the cardinality of feature maps for a given layer.

3.3 Network Architectures

To emphasize the importance of multitask approach and fusion of the intermediate layers of DCNN, we study the performance of simpler DCNNs devoid of such features. We evaluate four R-CNN-based models, one for each task of face detection, landmark localization, pose estimation and gender recognition. We also build a separate Multitask_Face model which performs multitask learning just like HyperFace, but does not fuse the information from the intermediate layers. These models are described as follows:

R-CNN_Face: This model is used for face detection task. The network architecture is shown in Figure 3.4(a). For training R-CNN_Face, we use the region proposals from AFLW[86] training set, each associated with a face label based on the overlap with the ground truth. The loss is computed as per (3.1). The model parameters are initialized using the Alexnet [88] weights trained on the Imagenet dataset [31]. Once trained, the learned parameters from this network are used to initialize other models including Multitask_Face and HyperFace as the standard Imagenet initialization doesn't converge well. We also perform a linear bounding box regression to localize the face co-ordinates.

R-CNN_Fiducial: This model is used for locating the facial landmarks. The network architecture is shown in Figure 3.4(b). It simultaneously learns the visibility of the points to account for the invisible points at test time, and thus can be used as a standalone fiducial extractor. The loss functions for landmarks localization and visibility of points are computed using (3.3) and (3.4), respectively. Only region proposals which have an overlap>0.5 with the ground truth bounding box are used for training. The model parameters are initialized from R-CNN_Face.

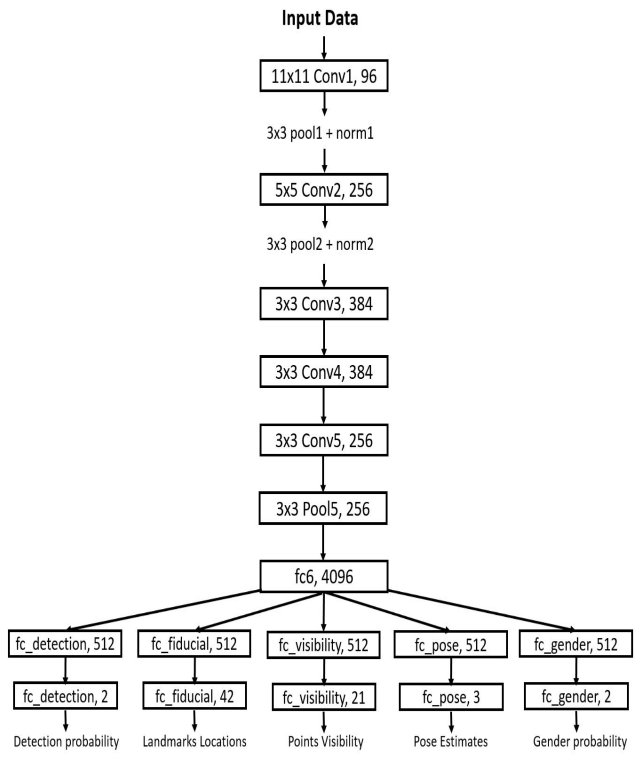


Figure 3.5: Network Architecture of Multitask_Face. The numbers on the left denote the kernel size and the numbers on the right denote the cardinality of feature maps for a given layer.

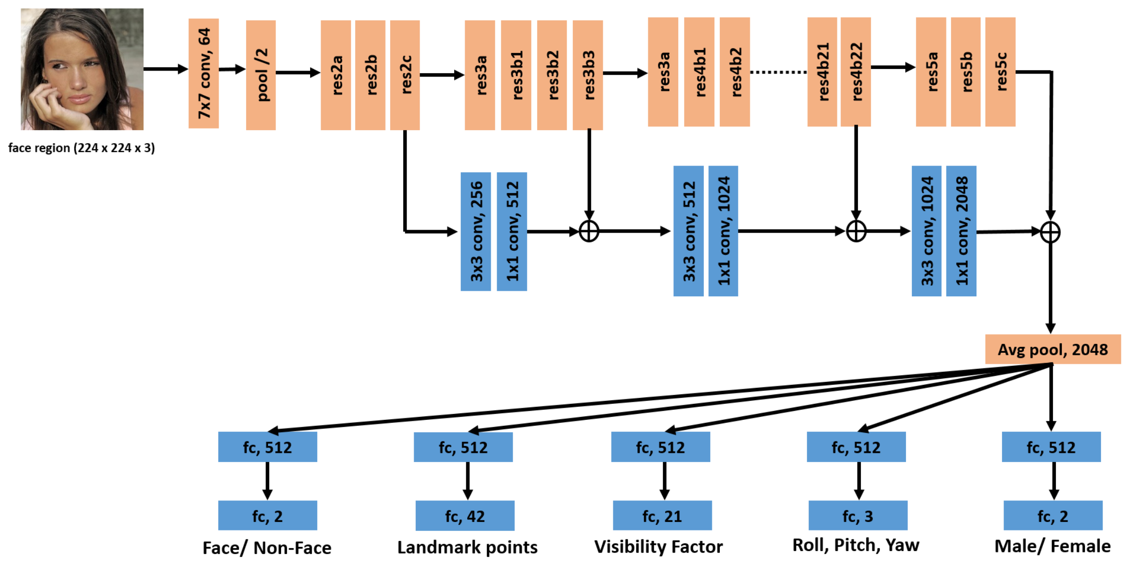


Figure 3.6: The architecture of the proposed HyperFace-Resnet (HF-ResNet). ResNet-101 model is used as the backbone network, represented in color orange. The new layers added are represented in color blue. The network is able to classify a given image region as face or non-face, estimate the head pose, locate face landmarks and recognize gender.

R-CNN_Pose: This model is used for head pose estimation task. The outputs of the network are roll, pitch and yaw of the face. Figure 3.4(c) presents the network architecture. Similar to R-CNN_Fiducial, only region proposals with overlap>0.5 with the ground truth bounding box are used for training. The training loss is computed using (3.5).

R-CNN_Gender: This model is used for face gender recognition task. The network architecture is shown in Figure 3.4(d). It has the same training set as R-CNN_Fiducial and R-CNN_Pose. The training loss is computed using (3.6).

Multitask_Face: Similar to HyperFace, this model is used to simultaneously detect face, localize landmarks, estimate pose and predict its gender. The only difference between Multitask_Face and HyperFace is that HyperFace fuses the intermediate layers of the network whereas Multitask_Face combines the tasks using the common fully connected layer at the end of the network as shown in Figure 3.5. Since it provides the landmarks and face score, it leverages iterative region proposals and landmark-based NMS post-processing algorithms during evaluation.

3.3.1 HyperFace-ResNet

The DCNN architectures have improved a lot over the years, mainly due to an increase in number of layers [62], effective convolution kernel size [163], batch normalization [72] and skip connections. Recently, He et al. [62] proposed a deep residual network architecture with more than 100 layers, that achieves state-of-the-art results on the ImageNet challenge [31]. Hence, we propose a variant of HyperFace that is built using the ResNet-101 [62] model instead of AlexNet [88]. The proposed network called HyperFace-ResNet (HF-ResNet) significantly improves upon its AlexNet baseline for all the tasks of face detection, landmarks localization, pose estimation and gender recognition. Figure 3.6 shows the network architecture for HF-ResNet.

Similar to HyperFace, we fuse the geometrically rich features from the lower layers and semantically strong features from the deeper layers of ResNet, such that multi-task learning can leverage from their synergy. Taking inspiration from [68], we fuse the features using hierarchical element-wise addition. Starting with ‘res2c’ features, we first reduce its resolution using a 3×3 convolution kernel with stride of 2. It is then passed through the a 1×1 convolution layer that increases the number of channels to match the next level features (‘res3b3’ in this case). Element-wise addition is applied between the two to generate a new set of fused features. The same operation is applied in a cascaded manner to fuse ‘res4b22’ and ‘res5c’ features of the ResNet-101 model. Finally, average pooling is carried out to generate 2048-dimensional feature vector that is shared among all the tasks. Task-specific sub-networks are branched out separately in a similar way as HyperFace. Each convolution layer is followed by a Batch-Norm+Scale [72] layer and ReLU activation unit. We do not use dropout in HF-ResNet. The training loss functions are the same as described in Section 3.2.2.

HF-ResNet is slower than HyperFace since it performs more convolutions. This makes it difficult to be used with Selective Search [176] algorithm which generates more than 2000 region proposals to be processed. Hence, we use a faster version of region proposals using high recall SSD [108] face detector. It produces 200 proposals, needing just 0.05s. This considerably reduces the total runtime for HF-ResNet to less than 1s. The fast version of HyperFace is discussed in Section 3.5.6.

3.4 Proposed Method - All-In-One Face

We propose a multi-purpose DCNN which can simultaneously detect faces, extract key-points and pose angles, determine smile expression and gender and estimate age from any unconstrained image of a face. Additionally, it assigns an identity descriptor to each face which can be used for face recognition and verification. The proposed algorithm is trained in a MTL framework which builds a synergy among different face related tasks improving the performance for each of them. In this section we discuss the advantages of MTL in the context of face analysis and provide the details of network design, training and testing procedures.

3.4.1 Multi-task Learning

Typically, a face analysis task requires a cropped face region as the input. The DCNN processes the face to obtain a representation and extract meaningful information related to the task. According to [212], lower layers of DCNN learn features common to a general set of face analysis tasks whereas upper layers are more specific to individual tasks. Therefore, we share the parameters of lower layers of DCNN among different tasks to produce a generic face representation which is subsequently processed by the task-specific layers to generate the required outputs (Fig. 3.7). Goodfellow et al. [51] interprets MTL as a regularization methodology for DCNNs. The MTL approach used in our framework can be explained by following two types of regularization.

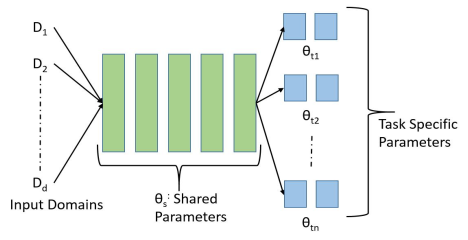


Figure 3.7: A general multitask learning framework for DCNN architecture. The lower layers are shared among all tasks and input domains.

3.4.1.1 Task-based Regularization

Let the cost function for a given task t_i with shared parameters θ_s and task-specific parameters θ_{t_i} be $J_i(\theta_s, \theta_{t_i}; D)$, where D is the input data. For isolated learning, the optimum network parameters $(\theta_s^*, \theta_{t_i}^*)$ can be computed using (3.9)

$$(\theta_s^*, \theta_{t_i}^*) = \underset{(\theta_s, \theta_{t_i})}{\arg\min} J_i(\theta_s, \theta_{t_i}; D) \tag{3.9}$$

For MTL, the optimal parameters for the task t_i can be obtained by minimizing the weighted sum of loss functions for each task, as shown in (3.10). The loss weight for task t_i is denoted by α_i.

$$(\theta_s^*, \theta_{t_i}^*) = \underset{(\theta_s, \theta_{t_i})}{\arg\min} \alpha_i J_i(\theta_s, \theta_{t_i}; D) + \sum_{j \neq i}^{n} \alpha_j J_j(\theta_s, \theta_{t_j}; D) \tag{3.10}$$

Since other tasks contribute only to the learning of shared parameters, they can be interpreted as a regularizer R_i on θ_s with respect to the given task t_i, as shown in (3.11). Thus, MTL shrinks the solution space of θ_s such that the learned parameter vector is in consensus with all the tasks, thus reducing over-fitting and enabling the optimization procedure to find a more robust solution.

$$(\theta_s^*, \theta_{t_i}^*) = \underset{(\theta_s, \theta_{t_i})}{\arg\min} J_i(\theta_s, \theta_{t_i}; D) + \lambda R_i(\theta_s; D) \tag{3.11}$$

Table 3.1: Datasets used for training

Dataset	Face Analysis Tasks	# training samples
CASIA [209]	Identification, Gender	490,356
MORPH [150]	Age, Gender	55,608
IMDB+WIKI [151]	Age, Gender	224,840
Adience [96]	Age	19,370
CelebA [111]	Smile, Gender	182,637
AFLW [86]	Detection, Pose, Fiducials	20,342
Total		**993,153**

3.4.1.2 Domain-based Regularization

For face analysis tasks, we do not have a large dataset with annotations for fiducial points, pose, gender, age, smile and identity information available. Hence, we adopt the approach of training multiple DCNNs with respect to task-related datasets D_i, and share the parameters among them. In this way, the shared parameter θ_s adapts to the complete set of domains $(D_1, D_2, ...D_d)$ instead of fitting to a task-specific domain. Additionally, the total number of training samples increases to roughly one-million, which is advantageous for training DCNNs. Table 3.1 lists the different datasets used for training our all-in-one DCNN, along with their respective tasks and sample sizes.

3.4.2 Network Architecture

The all-in-one DCNN architecture is shown in Fig. 4.7. We start with the pre-trained face identification network from Sankaranarayanan et al. [154]. The network consists of seven convolutional layers followed by three fully connected layers. We use it as a backbone network for training the face identification task and sharing the parameters from its first six convolution layers with other face-related tasks. Parametric Rectifier Linear units (PReLUs) are used as the activation function. We argue that a DCNN pre-trained on face identification task provides a better initialization for a generic face analysis task, since the filters retain discriminative face information.

We divide the tasks into two groups: 1) subject-independent tasks which include face detection, keypoints localization and visibility, pose estimation and smile prediction, and 2) subject-dependent tasks which include age estimation, gender prediction and face recognition. Similar to HyperFace [143] we fuse the first, third and fifth convolutional layers for training the subject-independent tasks, as they rely more on local information available from the lower layers of the network. We attach two convolution layers and a pooling layer respectively to these layers, to obtain a consistent feature map size of 6×6. A dimensionality reduction layer is added to reduce the number of feature maps to 256. It is followed by a fully connected layer of dimension 2048, which forms a generic representation for subject-independent tasks. At this point, the specific tasks are branched into fully connected layers of dimension 512 each, which are followed by the output layers respectively.

The subject-dependent tasks of age estimation and gender classification are branched out from the sixth convolutional layer of the backbone network after performing the max pooling operation. The global features thus obtained are fed to a 3-layered fully connected network for each of these tasks. We keep the seventh convolutional layer unshared to adapt it specifically to the face recognition task. Task-specific loss functions are used to train the complete network end-to-end.

3.4.3 Training

The training DCNN model contains five sub-networks with parameters shared among them. The tasks of face detection, key-points localization and visibility, and pose estimation are trained in a single sub-network, since all of them use a common dataset (AFLW [86]) for training. The remaining tasks of smile detection, gender recognition, age estimation and face recognition are trained using separate sub-networks. At test time, these sub-networks are fused together into a single all-in-one DCNN (Fig. 4.7). All tasks are simultaneously trained end-to-end using Caffe [75]. Here, we discuss the loss functions and training dataset for each of them.

3.4.3.1 Face Detection, Key-points Localization and Pose Estimation

These tasks are trained in a similar manner as HyperFace [143], using AFLW [86] dataset. We randomly select 1000 images from the dataset for testing, and use the remaining images for training. We use the Selective Search [176] algorithm to generate region proposals for faces from an image. Regions with Intersection-Over-Union (IOU) overlap of more than 0.5 with the ground truth bounding-box are considered positive examples

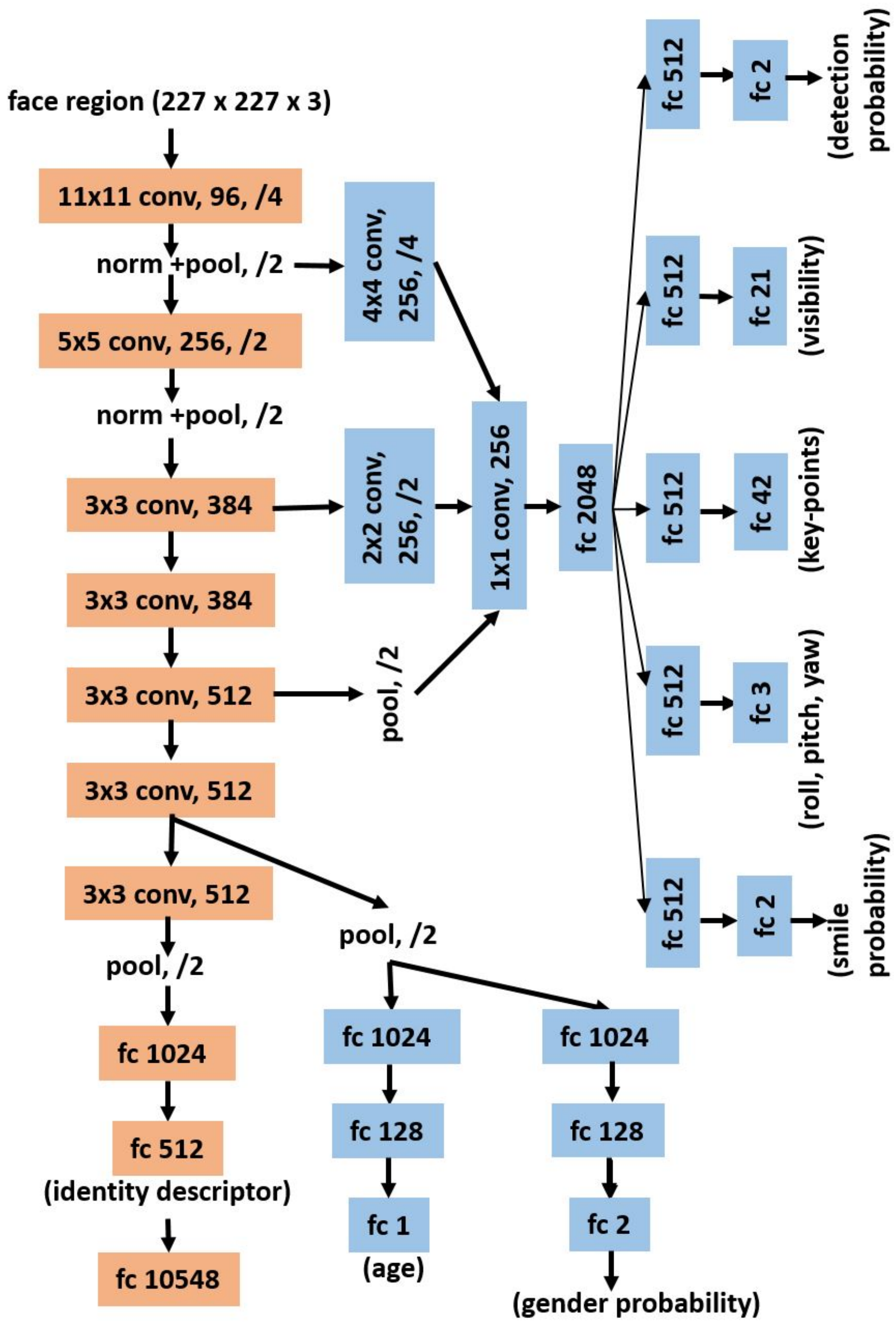


Figure 3.8: DCNN Architecture for the proposed method. Each layer is represented by filter kernel size, type of layer, number of feature maps and the filter stride. Orange represents the pre-trained network from Sankaranarayanan et al. [154], while blue represents added layers for MTL.

whereas regions with IOU<0.35 are chosen as negative examples for training the detection task using a softmax loss function. Facial landmarks, key-points visibility and pose estimation tasks are treated as regression problems and trained with the Euclidean loss. Only regions with IOU>0.35 contribute to back-propagation during their training.

3.4.3.2 Gender Recognition

It is a binary classification problem similar to face detection. The datasets used for training gender are listed in Table 3.1. The training images are first aligned using facial key-points which are either provided by the dataset or computed using HyperFace [143]. A cross-entropy loss L_G is used for training as shown in (3.12)

$$L_G = -(1-g)\cdot\log(1-p_g) - g\cdot\log(p_g), \tag{3.12}$$

where $g = 0$ for male and 1 for female. p_g is the predicted probability that the input face is a female.

3.4.3.3 Smile Detection

The smile attribute is trained to make the network robust to expression variations for face recognition. We use CelebA [111] dataset for training. Similar to the gender classification task, the the images are aligned before passing them through the network. The loss function L_S is given by (3.13)

$$L_S = -(1-s)\cdot\log(1-p_s) - s\cdot\log(p_s), \tag{3.13}$$

where $s = 1$ for a smiling face and 0 otherwise. p_s is the predicted probability that the input face is smiling.

3.4.3.4 Age Estimation

We formulate the age estimation task as a regression problem in which the network learns to predict the age from a face image. We use IMDB+WIKI [151], Adience [96] and MORPH [150] datasets for training. It has been shown by Ranjan et. al. [146] that the Gaussian loss works better than the Euclidean loss for apparent age estimation when the standard deviation of age is given. However, the gradient of Gaussian loss is close to zero when the predicted age is far from the true age (Fig. 3.9), which slows the training process. Hence, we use a linear combination of these two loss functions weighted by λ as shown in (3.14)

$$L_A = (1-\lambda)\frac{1}{2}(y-a)^2 + \lambda\left(1 - exp(-\frac{(y-a)^2}{2\sigma^2})\right), \tag{3.14}$$

where L_A is the age loss, y is the predicted age, a is the ground-truth age and σ is the standard deviation of the annotated age value. λ is initialized with 0 at the start of the training, and increased to 1 subsequently. In our implementation, we keep $\lambda = 0$ initially and switch it to 1 after $20k$ iterations. σ is fixed at 3 if not provided by the training set.

3.4.3.5 Face Recognition

We use $10,548$ subjects from CASIA [209] dataset to train the network for the task of face identification. The images are aligned using HyperFace [143] before passing them

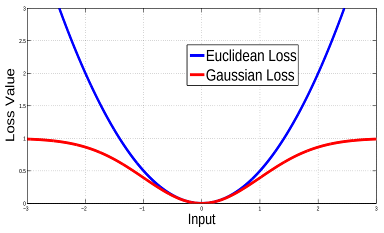


Figure 3.9: Euclidean and Gaussian loss functions.

through the network. We deploy a multi-class cross-entropy loss function L_R for training as shown in (3.15)

$$L_R = \sum_{c=0}^{10547} -y_c \cdot \log(p_c), \tag{3.15}$$

where $y_c = 1$ if the sample belongs to class c, otherwise 0. The predicted probability that a sample belongs to class c is given by p_c.

The final overall loss L is the weighted sum of individual loss functions, given in (3.16)

$$L = \sum_{t=1}^{t=8} \lambda_t L_t, \tag{3.16}$$

where L_t is the loss and λ_t is the loss-weight corresponding to task t. The loss-weights are chosen empirically. We assign a higher weight to regression tasks as they tend to have lower loss magnitude than classification tasks.

3.4.4 Testing

We deploy a two-stage process during test time as shown in Fig. 4.11. In the first stage, we use the Selective Search [176] to generate region proposals from a test image, which are passed through our all-in-one network to obtain the detection scores, pose estimates, fiducial points and their visibility. We use Iterative Region Proposals and Landmarks-based NMS [143] to filter out non-faces and improve fiducials and pose estimates.

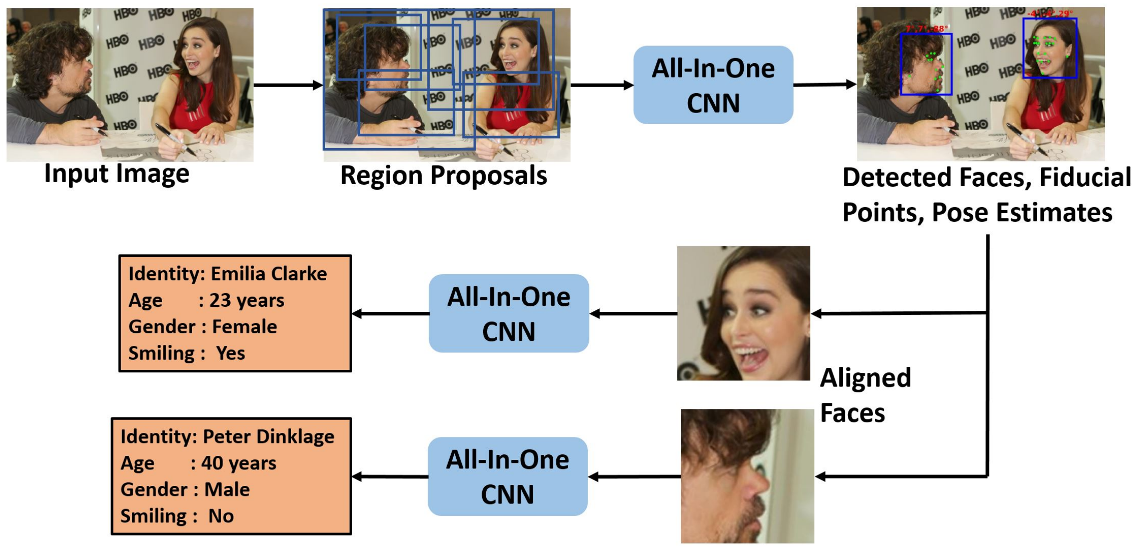


Figure 3.10: The end-to-end pipeline for the proposed method during test time.

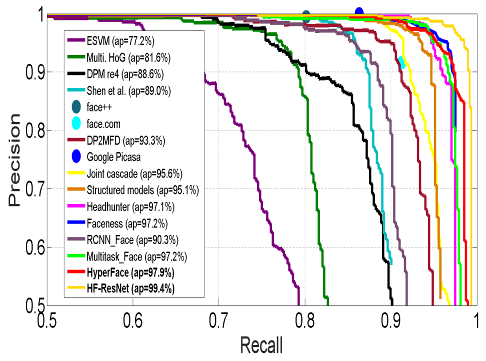
For the second stage, we use the extracted fiducial points to align each detected face to a canonical view using similarity transform. The aligned faces, along with their flipped versions are passed again through the network to get the smile, gender, age and identity information. We use the 512-dimensional feature from the penultimate fully connected layer of the identification network as the identity descriptor.

3.5 Experimental Results - HyperFace

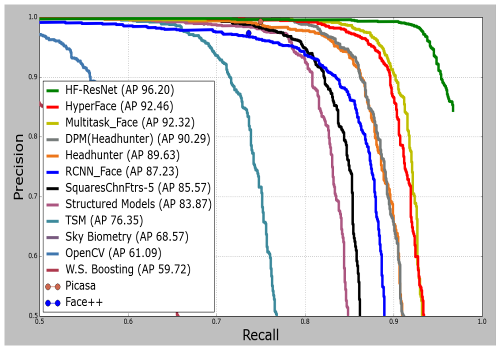
We evaluated the proposed HyperFace method, along with HF-ResNet, Multask_Face, R-CNN_Face, R-CNN_Fiducial, R-CNN_Pose and R-CNN_Gender on six challenging datasets:

- Annotated Face in-the-Wild (AFW) [225] for evaluating face detection, landmarks localization, and pose estimation tasks
- 300-W Faces in-the-wild (IBUG) [153] for evaluating 68-point landmarks localization.
- Annotated Facial Landmarks in the Wild (AFLW) [86] for evaluating landmarks localization and pose estimation tasks
- Face Detection Dataset and Benchmark (FDDB) [74] and PASCAL faces [200] for evaluating the face detection results
- Large-scale CelebFaces Attributes (CelebA) [111] and LFWA [70] for evaluating gender recognition results.

Our method was trained on randomly selected 20,997 images from the AFLW dataset using Caffe [75]. The remaining 1000 images were used for testing.



(a)



(b)

Figure 3.11: **Face Detection** performance evaluation on (a) the AFW dataset, (b) the PASCAL faces dataset. The numbers in the legend are the mean average precision (mAP) for the corresponding datasets.

3.5.1 Face Detection

We present face detection results for AFW, PASCAL and FDDB datasets. The AFW dataset [225] was collected from Flickr and the images in this dataset contain large variations in appearance and viewpoint. In total there are 205 images with 468 faces in this dataset. The FDDB dataset [74] consists of 2,845 images containing 5,171 faces collected from news articles on the Yahoo website. This dataset is the most widely used benchmark for unconstrained face detection. The PASCAL faces dataset [200] was collected from the test set of PASCAL person layout dataset, which is a subset from PASCAL VOC [41]. This dataset contains 1335 faces from 851 images with large appearance variations. For improved face detection performance, we learn a SVM classifier on top of $fc_{detection}$ features using the training splits from the FDDB dataset.

Some of the recent published methods compared in our evaluations include DP2MFD [141], Faceness [205], HeadHunter [116], JointCascade [17], CCF [201], SquaresChnFtrs-5 [116], CascadeCNN [98], Structured Models [200], DDFD [43], NPDFace [105], PEP-Adapt [99], TSM [225], as well as three commercial systems Face++, Picasa and Face.com.

The precision-recall curves of different detectors corresponding to AFW and PASCAL faces datasets are shown in Figures 3.20 (a) and (b), respectively. Figure 3.12 compares the performance of different detectors using the Receiver Operating Characteristic (ROC) curves on the FDDB dataset. As can be seen from these figures, both HyperFace and HF-ResNet outperform all the reported academic and commercial detectors on the AFW and PASCAL datasets. HyperFace achieves a high mean average precision (*mAP*) of 97.9% and 92.46%, for AFW and PASCAL datasets respectively. HF-ResNet further

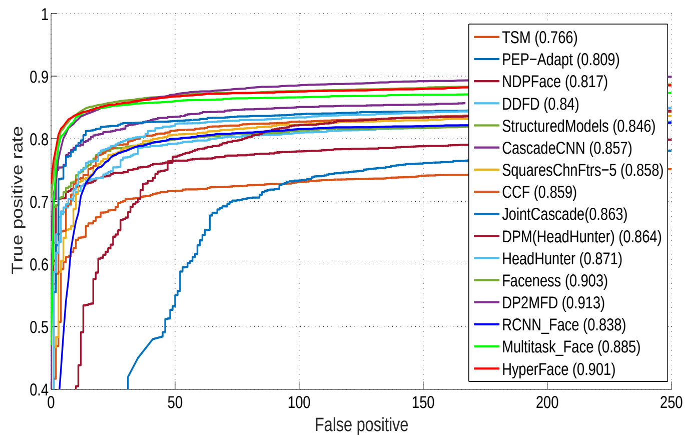


Figure 3.12: **Face Detection** performance evaluation on the FDDB dataset. The numbers in the legend are the mean average precision.

improves the mAP to 99.4% and 96.2% respectively.

The FDDB dataset is very challenging for HyperFace and any other R-CNN-based face detection methods, as the dataset contains many small and blurred faces. First, some of these faces do not get included in the region proposals from selective search. Second, re-sizing small faces to the input size of 227×227 adds distortion to the face resulting in low detection score. In spite of these issues, HyperFace performance is comparable to recently published deep learning-based face detection methods such as DP2MFD [141] and Faceness [205] on the FDDB dataset [*] with *mAP* of 90.1%.

It is interesting to note the performance differences between R-CNN_Face, Multitask_Face and HyperFace for the face detection tasks. Figures 3.20, and 3.12 clearly show that multitask DCNNs (Multitask_Face and HyperFace) outperform R-CNN_Face by a wide margin. The boost in the performance gain is mainly due to the following two reasons. First, multitask learning approach helps the network to learn improved features

[*]http://vis-www.cs.umass.edu/fddb/results.html

for face detection which is evident from their *mAP* values on the AFW dataset. Using just the linear bounding box regression and traditional NMS, the HyperFace obtains a *mAP* of 94% (Figure 3.17) while R-CNN_Face achieves a *mAP* of 90.3%. Second, having landmark information associated with detection boxes makes it easier to localize the bounding box to a face, by using IRP and L-NMS algorithms. On the other hand, HyperFace and Multi-task_Face perform comparable to each other for all the face detection datasets which suggests that the network does not gain much by fusing intermediate layers for the face detection task.

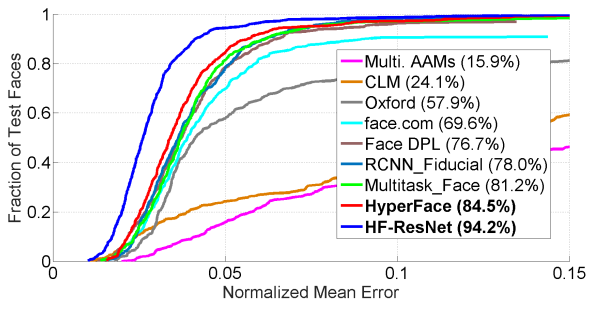


Figure 3.13: **Landmarks Localization** cumulative error distribution curves on the AFW dataset. The numbers in the legend are the fraction of testing faces that have average error below (5%) of the face size.

3.5.2 Landmarks Localization

We evaluate the performance of different landmarks localization algorithms on AFW [225] and AFLW [86] datasets. Both of these datasets contain faces with full pose variations. Some of the methods compared include Multiview Active Appearance Model-based method (Multi. AAM) [225], Constrained Local Model (CLM) [156], Oxford facial landmark detector [42], Zhu [225], FaceDPL [226], JointCascade [17], CDM [210], RCPR [8], ESR [10], SDM [197] and 3DDFA [227]. Although both of these datasets provide ground truth bounding boxes, we do not use them for evaluating on HyperFace, HF-ResNet, Multitask_Face and R-CNN_Fiducial. Instead we use the respective algorithms to detect both the face and its fiducial points. Since, the R-CNN_Fiducial cannot detect faces, we provide it with the detections from the HyperFace.

Figure 3.13 compares the performance of different landmark localization methods on the AFW dataset using the protocol defined in [226]. In this figure, (*) indicates that models that are evaluated on near frontal faces or use hand-initialization [225]. The dataset provides six keypoints for each face which are: left_eye_center, right_eye_center, nose_tip, mouth_left, mouth_center and mouth_right. We compute the error as the mean distance between the predicted and ground truth keypoints, normalized by the face size. The plots for comparison were obtained from [226].

For the AFLW dataset, we calculate the error using all the visible keypoints. We adopt the same protocol as defined in [227]. The only difference is that our AFLW testset consists of only 1000 images with 1132 face samples, since we use the rest of the images for training. To be consistent with the protocol, we randomly create a subset of 450 samples from our testset whose absolute yaw angles within $[0°, 30°]$, $[30°, 60°]$ and $[60°, 90°]$ are $1/3$ each. Figure 3.14 compares the performance of different landmark localization methods. We obtain the comparison plots from [227] where the evaluations for RCPR, ESR and SDM are carried out after adapting the algorithms to face profiling. Table 3.2 provides the Normalized Mean Error (NME) for AFLW dataset, for each of the pose group.

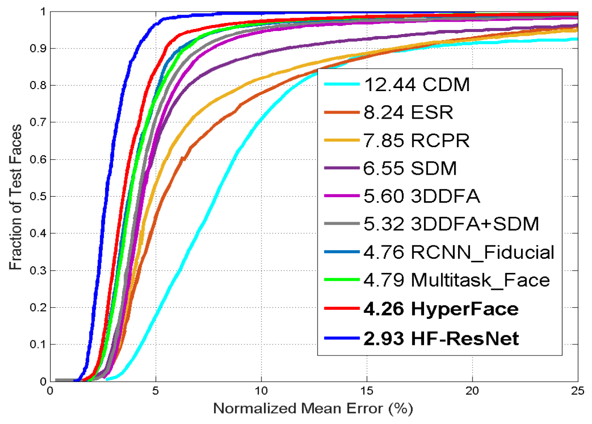
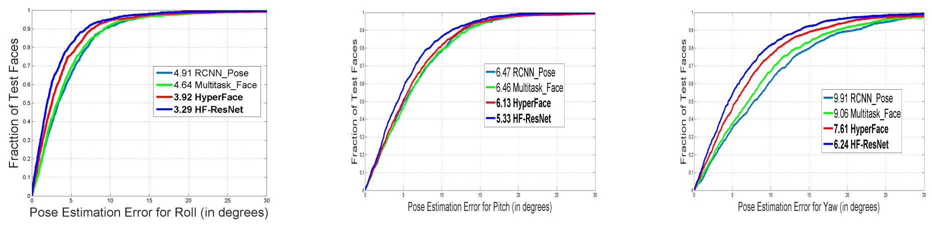


Figure 3.14: **Landmarks Localization** cumulative error distribution curves on the AFLW dataset. The numbers in the legend are the average NME for the test images. The test samples are chosen such that samples with absolute yaw angles between [0°,30°], [30°,60°] and [60°,90°] are 1/3 each.



(a) (b) (c)

Figure 3.15: **Pose Estimation** performance evaluation on AFLW dataset for (a) roll (b) pitch and (c) yaw angles. The numbers in the legend are the mean error in degrees for the respective pose angles.

Table 3.2: The NME(%) of face alignment results on AFLW test set with the best results highlighted.

	AFLW Dataset (21 pts)				
Method	[0, 30]	[30, 60]	[60, 90]	mean	std
CDM [210]	8.15	13.02	16.17	12.44	4.04
RCPR [8]	5.43	6.58	11.53	7.85	3.24
ESR [10]	5.66	7.12	11.94	8.24	3.29
SDM [197]	4.75	5.55	9.34	6.55	2.45
3DDFA [227]	5.00	5.06	6.74	5.60	0.99
3DDFA [227]+SDM	4.75	4.83	6.38	5.32	0.92
R-CNN_Fiducial	4.49	4.70	5.09	4.76	0.30
Multitask_Face	4.20	4.93	5.23	4.79	0.53
HyperFace	**3.93**	**4.14**	**4.71**	**4.26**	**0.41**
HF-ResNet	**2.71**	**2.88**	**3.19**	**2.93**	**0.25**

As can be seen from the figures, R-CNN_Fiducial, Multitask_Face, HyperFace and HF-ResNet outperform many recent state-of-the-art landmark localization methods including FaceDPL [226], 3DDFA [227] and SDM [197]. Table 3.2 shows that HyperFace performs consistently accurate over all pose angles. This clearly suggests that while most of the methods work well on frontal faces, HyperFace is able to predict landmarks for faces with full pose variations. Moreover, we find that R-CNN_Fiducial and Multitask_Face attain similar performance. The HyperFace has an advantage over them as it uses the intermediate layers for fusion. The local information is contained well in the lower layers of DCNN and becomes invariant as depth increases. Fusing the layers brings out that hidden information which boosts the performance for the landmark localization task. Additionally, we observe that HF-ResNet significantly improves the performance over HyperFace for both AFW and AFLW datasets. The large margin in performance can be attributed to the larger depth for the HF-ResNet model.

We also evaluate our models on the challenging subset of the 300-W [153] landmarks localization dataset (IBUG). The dataset contains 135 test images with wide variations in expression and illumination. The head-pose angle varies from -60° to 60° in yaw. Since the dataset contains 68 landmarks points instead of 21 used in AFLW [86] training, the model cannot be directly applied for evaluating IBUG. We retrain the network for predicting 68 facial key-points as a separate task in conjunction with the proposed tasks in hand. We implement it by adding two fully-connected layers in a cascade manner to the shared feature space (fc-full), having dimensions 512 and 136, respectively.

Following the protocol described in [149], we use $3,148$ faces with 68-point annotations for training. The network is trained end-to-end for the localization of 68-points landmarks along with the other tasks mentioned in Section 3.2.2. We use the standard Euclidean loss function for training. For evaluation, we compute the average error of all 68 landmarks normalized by the inter-pupil distance. Table 3.3 compares the Normalized Mean Error (NME) obtained by HyperFace and HF-ResNet with other recently published methods. We observe that HyperFace achieves a comparable NME of 10.88, while HF-ResNet achieves the state-of-the-art result on IBUG [153] with NME of 8.18. This shows the effectiveness of the proposed models for 68-point landmarks localization.

Table 3.3: Normalized Mean Error (in %) of 68-point landmarks localization on IBUG [153] dataset.

Method	Normalized Mean Error
CDM [210]	19.54
RCPR [8]	17.26
ESR [10]	17.00
SDM [197]	15.40
LBF [149]	11.98
LDDR [90]	11.49
CFSS [224]	9.98
3DDFA [227]	10.59
TCDCN [218]	8.60
HyperFace	10.88
HF-ResNet	**8.18**

3.5.3 Pose Estimation

We evaluate R-CNN_Pose, Multitask_Face and HyperFace on the AFW [225] and AFLW [86] datasets for the pose estimation task. The detection boxes used for evaluating the landmark localization task are used here as well for initialization. For the AFW dataset, we compare our approach with Multi. AAM [225], Multiview HoG [225], FaceDPL[†] [226] and face.com. Note that multiview AAMs are initialized using the ground truth bounding boxes (denoted by *). Figure 3.16 shows the cumulative error distribution curves on AFW dataset. The curve provides the fraction of faces for which the estimated pose is within some error tolerance. As can be seen from the figure, both HyperFace and HF-ResNet outperform existing methods by a large margin. For the AFLW dataset, we do not have pose estimation evaluation for any previous method. Hence, we show the performance of our method for different pose angles: roll, pitch and yaw in Figure 3.15 (a), (b) and (c) respectively. It can be seen that the network is able to learn roll, and pitch information better than yaw.

The performance traits of R-CNN_Pose, Multitask_Face, HyperFace and HF-ResNet for pose estimation task are similar to that of the landmarks localization task. R-CNN_Pose and Multitask_Face perform comparable to each other whereas HyperFace achieves a boosted performance due to the intermediate layers fusion. It shows that tasks which rely on the structure and orientation of the face work well with features from lower layers of the DCNN. HF-ResNet further improves the performance for roll, pitch as well as yaw.

[†]Available at: `http://www.ics.uci.edu/~dramanan/software/face/face_journal.pdf`

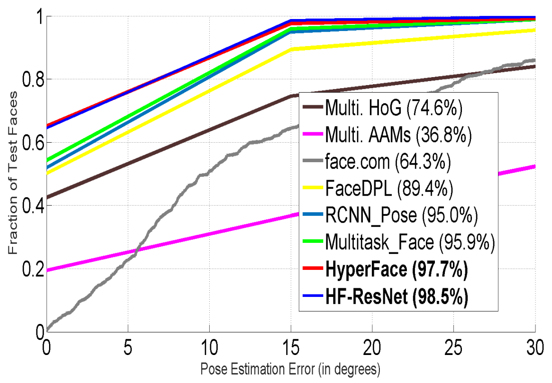


Figure 3.16: **Pose Estimation** cumulative error distribution curves on AFW dataset. The numbers in the legend are the percentage of faces that are labeled within $\pm 15^\circ$ error tolerance.

3.5.4 Gender Recognition

We present the gender recognition performance on CelebA [111] and LFWA [70] datasets since these datasets come with gender information. The CelebA and LFWA datasets contain labeled images selected from the CelebFaces [167] and LFW [70] datasets, respectively [111]. The CelebA dataset contains 10,000 identities and there are 200,000 images in total. The LFWA dataset has 13,233 images of 5,749 identities. We compare our approach with FaceTracer [91], PANDA-w [216], PANDA-1 [216], [102] with ANet and [111]. The gender recognition performance of different methods is reported in Table 3.4. On the LFWA dataset, our method outperforms PANDA [216] and FaceTracer [91], and is equal to [111]. On the CelebA dataset, our method performs comparably to [111]. Unlike [111] which uses $180,000$ images for training and validation, we only use $20,000$ images from validation set of CelebA to fine-tune the network.

Similar to the face detection task, we find that gender recognition performs better

Table 3.4: Performance comparison (in %) of **gender recognition** on CelebA and LFWA datasets.

Method	CelebA	LFWA
FaceTracer [91]	91	84
PANDA-w [216]	93	86
PANDA-1 [216]	97	92
[102]+ANet	95	91
LNets+ANet [111]	**98**	**94**
R-CNN_Gender	95	91
Multitask_Face	97	93
HyperFace	97	**94**
HF-ResNet	**98**	**94**

for HyperFace and Multitask_Face as compared to R-CNN_Gender proving that learning related tasks together improves the discriminating capability of the individual tasks. Again, we do not see much difference in the performance of Multitask_Face and HyperFace suggesting intermediate layers do not contribute much for the gender recognition task. HF-ResNet achieves state-of-the-art results on both CelebA [111] and LFWA [70] datasets.

3.5.5 Effect of Post-Processing

Figure 3.17 provides an experimental analysis of the post-processing methods: IRP and L-NMS, for face detection task on the AFW dataset. *Fast SS* denotes the quick version of selective search which produces around 2000 region proposals and takes $2s$ per image to compute. On the other hand, *Quality SS* refers to its slow version which outputs more than $10,000$ region proposals consuming more than $10s$ for one image. The HyperFace with a linear bounding box regression and traditional NMS achieves a *mAP* of 94%. Just by replacing them with L-NMS provides a boost of 1.2%. In this case, bounding-box is constructed using the landmarks information rather linear regression. Additionaly, we can see from the figure that although *Quality SS* generates more region proposals, it performs worse than *Fast SS* with ierative region proposals. IRP adds 300 new regions for a typical image consuming less than $0.5s$ which makes it highly efficient as compared to *Quality SS*.

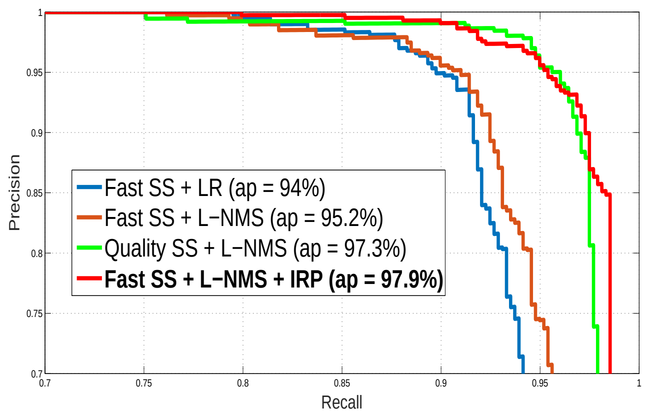


Figure 3.17: Variations in performance of HyperFace with respect to the Iterative Region Proposals and Landmarks-based NMS. The numbers in the legend are the mean average precision.

3.5.6 Fast-HyperFace

The Hyperface method is tested on a machine with 8 cores and GTX TITAN-X GPU. The overall time taken to perform all the four tasks is 3s per image. The limitation is not because of DCNN, but due to Selective Search [176] algorithm which takes approximately 2s to generate candidate region proposals. One forward pass through the HyperFace network for 200 proposals takes merely 0.1s.

We also propose a fast version of HyperFace which uses a high recall fast face detector instead of Selective Search [176] to generate candidate region proposals. We implement a face detector using Single Shot Detector (SSD) [108] framework. The SSD-based face detector takes a 512×512 dimensional input image and generates face boxes in less than 0.05s, with confidence probability scores ranging from 0 to 1. We use a probability threshold of 0.01 to select high recall detection boxes. Unlike traditional SSD, we do not use non-maximum suppression on the detector output, so that we have more

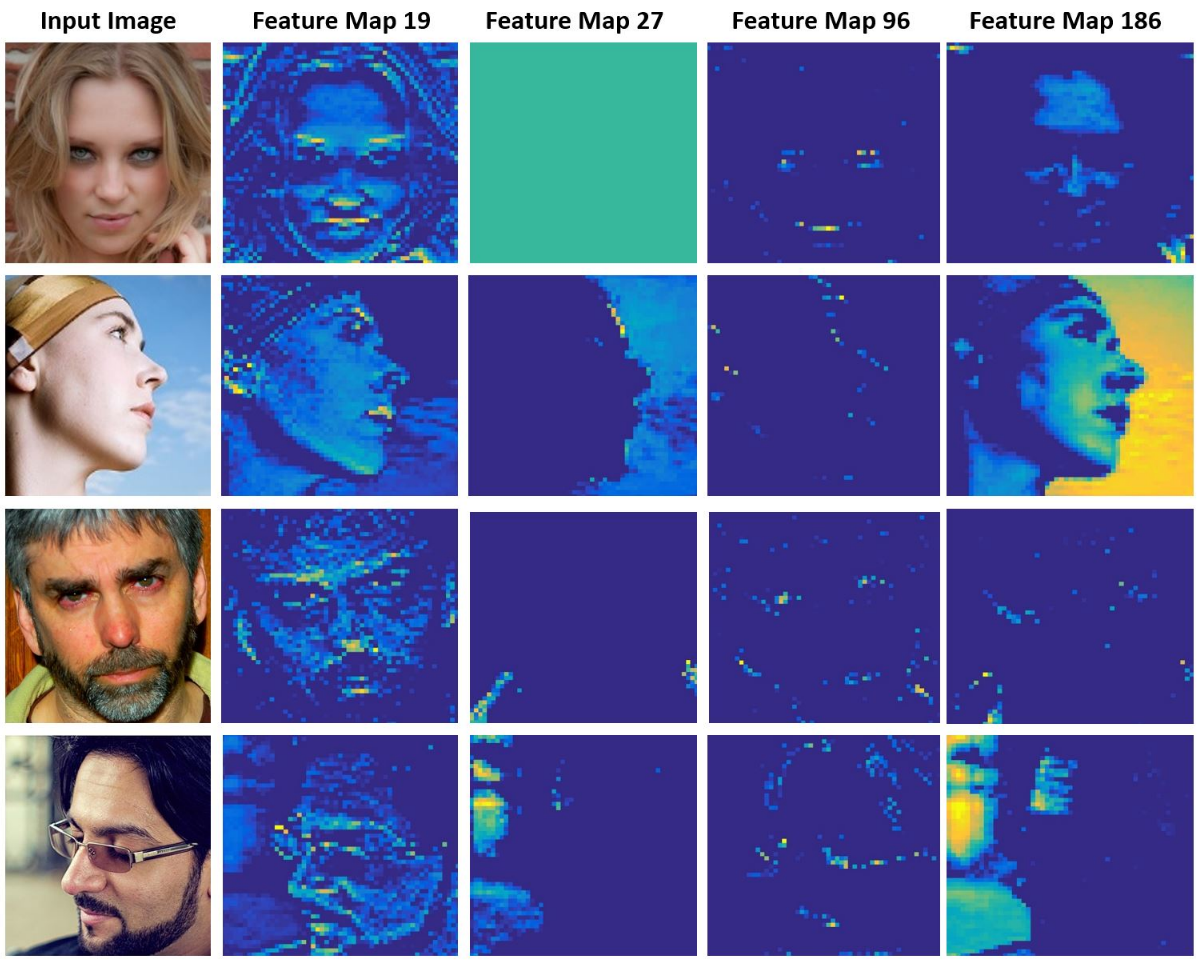


Figure 3.18: Activations of selected feature maps from **conv_all** layer of the HyperFace architecture. Green and yellow colors denote high activation whereas blue denotes low activation units. These features depict the distinguishable face traits for the tasks of face detection, landmarks localization, pose estimation and gender recognition.

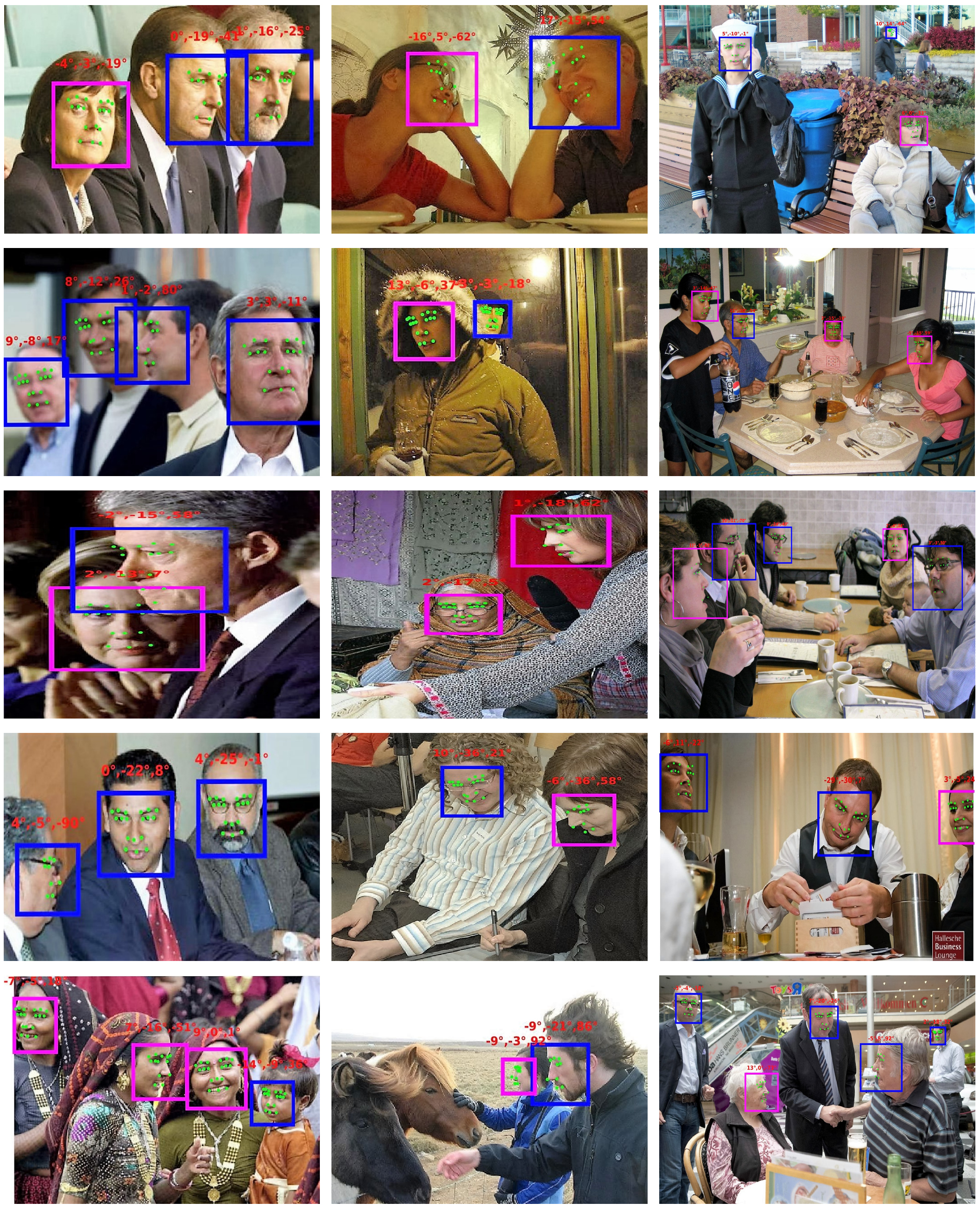


Figure 3.19: Qualitative results of our method. The blue boxes denote detected male faces, while pink boxes denote female faces. The green dots provide the landmark locations. Pose estimates for each face are shown on top of the boxes in the order of roll, pitch and yaw.

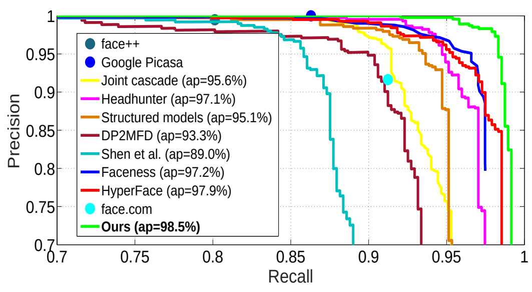
number of region proposals. Typically, the SSD face detector generates 200 proposals per image. These proposals are directly passed through HyperFace to generate face detection scores, localize face landmarks, estimate pose and recognize gender for every face in the image. Fast-HyperFace consumes a total time of 0.15s (0.05s for SSD face detector, and 0.1s for HyperFace) on a GTX TITAN X GPU. The Fast-HyperFace achieves a mAP of 97.6% on AFW face detection task, which is comparable to the HyperFace mAP of 97.9%. Thus, Fast-HyperFace improves the speed by a factor of 12 with negligible degradation in performance.

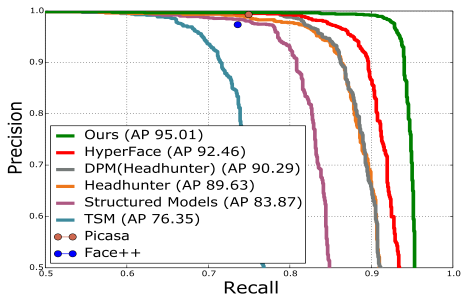
3.6 Experimental Results - All-In-One Face

The proposed method is evaluated for all the tasks on which it was trained except the key-points visibility, due to the lack of a proper evaluation protocol. We select HyperFace [143] as a comparison baseline for the tasks of face detection, pose estimation, landmarks localization and gender recognition. For face recognition task, the method from Sankaranarayanan et. al. [154], which is used as the initialization, is used as the baseline.

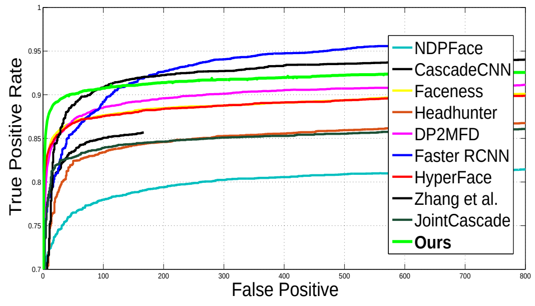
3.6.1 Face Detection

We evaluate the face detection task on Annotated Face in-the-Wild (AFW) [225], PASCAL faces [200] and Face Detection Dataset and Benchmark (FDDB) [74] datasets. All these datasets contain faces with wide variations in appearance, scale, viewpoint and illumination. To evaluate on AFW and PASCAL datasets, we finetune the face detection





(a) (b)



(c)

Figure 3.20: Performance evaluation on (a) the AFW dataset, (b) the PASCAL faces dataset and (c) FDDB dataset. The numbers in the legend are the mean average precision for the corresponding datasets.

branch of the network on FDDB. To evaluate on the FDDB dataset, we finetune according to the 10-fold cross validation experiments [74].

The precision-recall curves for AFW and PASCAL datasets, and the Receiver Operating Characteristic (ROC) curve for FDDB dataset are shown in Fig. 3.20. It can be seen from the figures that our method achieves state-of-the-art performance on AFW and PASCAL dataset with mean average precision (mAP) of 98.5% and 95.01% respectively. On the FDDB dataset, our method performs better than most of the reported algorithms. It gets lower recall than Faster-RCNN [76] and Zhang et al. [215], since small faces of FDDB fail to get captured in any of the region proposals. Other recently published methods compared in our detection evaluations include DP2MFD [141], Faceness [205], Headhunter [116], Joint Cascade [17], Structured Models [200], Cascade CNN [98], NDPFace [105], TSM [225], as well as three commercial systems Face++, Picasa and Face.com.

3.6.2 Landmarks Localization

We evaluate our performance on AFW [225] and AFLW [86] datasets as they contain large variations in viewpoints of faces. The landmarks location is computed as the mean of the predicted landmarks corresponding to region proposals having IOU$>$0.5 with the test face. For AFLW [86] evaluation, we follow the protocol given in [224]. We randomly create a subset of 450 samples from our test set such that the absolute yaw angles within $[0^\circ, 30^\circ]$, $[30^\circ, 60^\circ]$ and $[60^\circ, 90^\circ]$ are $1/3$ each. Table 3.5 compares the Normalized Mean Error (NME) for our method with recent face alignment method adapted to face profoling [224], for each of the yaw bins. Our method significantly outperforms the previous best HyperFace [143], reducing the error by more than 30%. A low standard deviation of 0.13 suggests that landmarks prediction is consistent as pose angles vary.

Table 3.5: The NME(%) of face alignment results on AFLW test set.

	AFLW Dataset (21 pts)				
Method	[0, 30]	[30, 60]	[60, 90]	mean	std
RCPR [8]	5.43	6.58	11.53	7.85	3.24
ESR [10]	5.66	7.12	11.94	8.24	3.29
SDM [197]	4.75	5.55	9.34	6.55	2.45
3DDFA [224]	5.00	5.06	6.74	5.60	0.99
3DDFA+SDM	4.75	4.83	6.38	5.32	0.92
HyperFace [143]	3.93	4.14	4.71	4.26	0.41
Ours	**2.84**	**2.94**	**3.09**	**2.96**	**0.13**

For AFW [225] evaluation, we follow the protocol described in [223]. Fig. 3.21(a) shows comparisons with recently published methods such as CCL [223], HyperFace [143], LBF [149], SDM [197], ERT [80] and RCPR [8]. It is evident that the proposed algorithm performs better than existing methods on unconstrained and profile faces since it predicts landmarks with less than 5% NME on more than 95.5% of test faces. However, it lacks in pixel-accurate precise localization of key-points for easy faces, which can be inferred from the lower end of the curve. Most of these algorithms use cascade stage-wise regression to improve the localization, which is slower compared to a single forward pass of the network.

3.6.3 Pose Estimation

We evaluate our method on AFW [225] dataset for the pose estimation task. According to the protocol defined in [225], we compute the absolute error only for the yaw angles. Since, the ground-truth yaw angles are provided in multiples of 15°, we round-off our predicted yaw to the nearest 15° for evaluation. Fig. 3.21(b) shows the comparison of our method with HyperFace [143], Face DPL [226], Multiview HoG [225] and face.com. It is clear that the proposed algorithm performs better than competing methods and is able to predict the yaw in the range of ±15° for more than 99% of the faces.

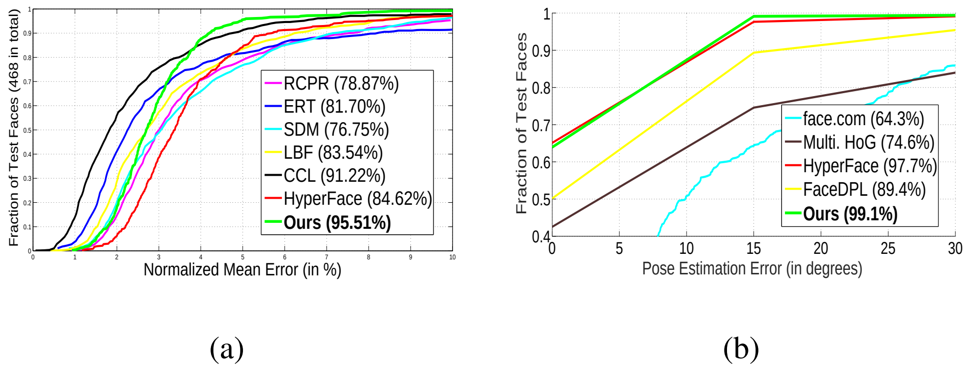


Figure 3.21: Performance evaluation on AFW dataset for (a) landmarks localization task, (b) pose estimation task. The numbers in the legend are the percentage of test faces with (a) NME less than 5%, (b) absolute yaw error less than or equal to 15°

3.6.4 Gender and Smile Recognition

We evaluate the smile and gender recognition performance on Large-scale CelebFaces Attributes (CelebA) [111] and ChaLearn Faces of the World [40] datasets. While

CelebA [111] has wide variety of subjects, they mostly contain frontal faces. Faces of the World [40] has wide variations in scale and viewpoints of faces. We take the mean of the predicted scores obtained from region proposals having IOU>0.5 with the given face, as our final score for smile and gender attributes. Table 3.6 compares the gender and smile accuracy with recently published methods. On CelebA [111], we outperform all the methods for gender accuracy. Our smile accuracy is lower only to Walk and Learn [180] which uses other contextual information to improve prediction. The gender and smile branches of the network were finetuned on the training set of Faces of the World [40] before its evaluation. We achieve state-of-the-art performance for both gender and smile classification on their validation set.

Table 3.6: Accuray (%) for Gender and Smile classification on CelebA [111] (left) and Faces of the World [40] (right)

Method	Gender	Smile
PANDA-1 [216]	97	92
MT-RBM [36]	90	88
LNets+ANet [111]	98	92
HyperFace [143]	97	-
Walk & Learn [180]	96	**98**
Ours	**99**	93

Method	Gender	Smile
MT-RBM [36]	71.7	80.8
CMP+ETH [177]	89.15	79.03
DeepBE [97]	90.44	88.43
SIAT_MMLAB [214]	91.66	89.34
Ours	**93.12**	**90.83**

3.6.5 Age Estimation

We use Chalearn LAP2015 [39] apparent age estimation challenge dataset and FG-NET [58] Aging Database for evaluating our age estimation task. We fine-tune the age-task branch of the network on the training set of the challenge dataset, and show the results on the validation set. The error is computed according to the protocol described in [39]. For FG-Net [58], we follow the standard Leave-One-Out-Protocol (LOPO). Table 3.7 lists the evaluation error for both these datasets. We surpass human error of 0.34 and perform comparable to state-of-the-art methods, obtaining an error of 0.293 on Chalearn LAP2015 [39] dataset. On FG-Net [58], we significantly outperform other methods, achieving an average error of 2 years.

Table 3.7: Age Estimation error on LAP2015(left) and FG-NET(right)

Method	Error
UMD [146]	0.359
Human	0.34
CascadeAge [19]	0.297
CVL_ETHZ [151]	**0.278**
ICT-VIPL [110]	0.292
Ours	0.293

Method	Error
Han2013 [58]	4.6
Chao2013 [15]	4.38
Hong2013 [65]	4.18
El Dib2010 [38]	3.17
CascadeAge [19]	3.49
Ours	**2.00**

Table 3.8: Face Identification and Verification Evaluation on IJB-A dataset

	IJB-A Verification (TAR@FAR)			IJB-A Identification			
Method	0.001	0.01	0.1	FPIR=0.01	FPIR=0.1	Rank=1	Rank=10
GOTS [84]	0.2(0.008)	0.41(0.014)	0.63(0.023)	0.047(0.02)	0.235(0.03)	0.443(0.02)	-
VGG-Face [136]	0.604(0.06)	0.805(0.03)	0.937(0.01)	0.46(0.07)	0.67(0.03)	0.913(0.01)	0.981(0.005)
Chen et al. [21]	-	0.838(0.042)	0.967(0.009)	-	-	0.903(0.012)	0.977(0.007)
Masi et al. [115]	0.725	0.886	-	-	-	0.906	0.977
NAN [204]	0.785(0.03)	0.897(0.01)	0.959(0.005)	-	-	-	-
Sankaranarayanan et al. [154] w/o TPE	0.766(0.02)	0.871(0.01)	0.952(0.005)	0.67(0.05)	0.82(0.013)	0.925(0.01)	0.978(0.005)
Sankaranarayanan et al. [154]	0.813(0.02)	0.90(0.01)	0.964(0.005)	0.753(0.03)	0.863(0.014)	0.932(0.01)	0.977(0.005)
Crosswhite et al. [29]	-	**0.939(0.013)**	-	0.774(0.049)	0.882(0.016)	0.928(0.01)	0.986(0.003)
Ours	0.787(0.04)	0.893(0.01)	0.968(0.006)	0.704(0.04)	0.836(0.014)	0.941(0.008)	0.988(0.003)
Ours + TPE [154]	**0.823(0.02)**	0.922(0.01)	**0.976(0.004)**	**0.792(0.02)**	**0.887(0.014)**	**0.947(0.008)**	**0.988(0.003)**

Table 3.9: Comparison of End-to-End face recognition systems on IJB-A

Face Detection	Face Alignment	Identity Descriptor	Metric Learning	Verif @FAR=0.01	Ident Rank=1
DP2MFD [141]	LDDR [90]	Chen et al. [18]	Joint Bayesian [18]	0.776(0.033)	0.834(0.017)
HyperFace [143]		Sankaranarayanan et al [154]	cosine	0.871(0.01)	0.925(0.01)
HyperFace [143]		Sankaranarayanan et al [154]	TPE [154]	0.90(0.01)	0.932(0.01)
HyperFace [143]		**Ours**	cosine	0.889(0.01)	0.939(0.01)
Ours			cosine	0.893(0.01)	0.941(0.008)
Ours			TPE [154]	**0.922(0.01)**	**0.947(0.008)**

3.6.6 Face Identification/Verification

We evaluate the tasks of face recognition and verification on the IARPA Janus Benchmark-A (IJB-A) [84] dataset. The dataset contains 500 subjects with a total of $25,813$ images including $5,399$ still images and $20,414$ video frames. It contains faces with extreme viewpoints, resolution and illumination which makes it more challenging than the commonly used LFW [70] dataset.

For IJB-A dataset, given a template containing multiple faces, we generate a common vector representation by media pooling the individual face descriptors, as explained in [154]. A naive way to measure the similarity of a template pair, is by taking the cosine distance between their descriptors. A better way is to learn an embedding space where features corresponding to similar pairs are close to each other while dissimilar pairs are far away. We train a Triplet Probabilistic Embedding (TPE) [154] using the training splits provided by the dataset. Table 3.8 compares with recently published methods on IJB-A. We achieve state-of-the-results for the face identification task. Although we perform comparable to template-adaptaion learning (Crosswhite et al. [29]) on verification task, we achieve a significantly faster query time ($0.1s$ after face detection per image pair). We get a consistent improvement of 2% to 3% over the baseline network [154] for all metrics.

We also compare our performance with end-to-end face recognition methods in Table 3.9. Our method outperforms existing end-to-end systems which shows that training all the tasks in the pipeline simultaneously, reduces the error. We see a two-fold improvement, i.e., about 80% performance gain is a result of improved identity descriptor and 20% gain is due to improved face alignment.

3.7 Conclusions

We present some observations based on our experiments. First, all the face related tasks benefit from using the multi-task learning framework. The gain is mainly due to the network's ability to learn more discriminative features, and post-processing methods which can be leveraged by having landmarks as well as detection scores for a region. Secondly, fusing intermediate layers improves the performance for structure dependent tasks of pose estimation and landmarks localization, as the features become invariant to geometry in deeper layers of DCNN. The HyperFace exploits these observations to improve the performance for all the four tasks.

We also visualize the features learned by the HyperFace network. Figure 3.18 shows the network activation for a few selected feature maps out of 192 from the $conv_{all}$ layer. It can be seen that some feature maps are dedicated solely for a single task while others can be used to predict different tasks. For example, feature map 27 and 186 can be used for face detection and gender recognition, respectively. The former distinguishes the face and non-face regions whereas the latter outputs high activation for the female faces. Similarly, feature map 19 shows high activation near eyes and mouth regions, while feature map 96 gives a rough contour of the face orientation. These features can be used for landmark localization and pose estimation tasks.

Additionally, we presented a multi-task DCNN-based method for simultaneous face detection, face alignment, pose estimation, gender and smile classification, age estimation and face verification and recognition. Our method performs significantly better than HyperFace, even though both of them use the MTL framework. This work demonstrates that subject-independent tasks benefit from domain-based regularization and network initialization from face recognition task. Also, the improvement in face verification and recognition performance compared to [154] clearly suggests that MTL helps in learning robust feature descriptors.

Chapter 4: Crystal Loss for Discriminative Face Verification

In this chapter, we provide a symptomatic treatment of issues associated with using softmax loss for the task of unconstrained face verification. We propose the Crystal loss function that adds a constraint on the features during training such that their L_2-norm remains constant. In other words, we restrict the features to lie on a hypersphere of a fixed radius. The proposed Crystal loss has two advantages. Firstly, it provides equal attention to both good and bad quality faces since all the features have the same L_2-norm now, which is essential for improved performance in unconstrained settings. Secondly, it strengthens the verification features by forcing the same subject features to be closer and features from different subjects to be far from each other in the normalized space. Therefore, it maximizes the margin for the normalized L_2 distance or cosine similarity score between negative and positive pairs. In this way, the proposed Crystal loss overcomes the limitations of the regular softmax loss.

The Crystal loss also retains the advantages of the regular softmax loss. Similar to the softmax loss, it is a one network, one loss system. It doesn't necessarily require any joint supervision as used by many recent methods [182, 136, 183, 168]. It can be easily implemented using inbuilt functions from Caffe [75], Torch [27] and TensorFlow [1], and converges very fast. It introduces just a single scaling parameter to the network.

Compared to the regular softmax loss, the Crystal loss gains a significant improvement in the performance. It achieves new state-of-the-art results on IJB-A, IJB-B, IJB-C and LFW datasets, and competitive results on YouTube Face datasets. It surpasses the performance of several state-of-the-art systems, which use multiple networks or multiple loss functions or both. Moreover, the gains from Crystal Loss are complementary to metric learning (eg: TPE [154], joint-Bayes [21]) or auxiliary loss functions (eg: center loss [182], contrastive loss [168]). We show that applying these techniques on top of the Crystal Loss can further improve the verification performance. Combining with TPE [154], Crystal loss achieves a record True Accept Rate (TAR) of 0.921 at False Accept Rate (FAR) of 0.0001 on the challenging IJB-A [84] dataset, and a TAR of 0.606 at FAR of 1e-8 on the challenging IJB-C [118] dataset.

4.1 Related Work

In recent years, there has been significant improvements in the accuracy of face verification using deep learning methods [157, 173, 136, 154, 196, 168, 182, 144]. Most of these methods have even surpassed human performance on the LFW [70] dataset. Although these methods use DCNNs, they differ from each other by the type of loss function used for training. For face verification, it is essential for the features of positive subject pair to be closer and features of negative subject pair to be far apart. To solve this problem, researchers have adopted two major approaches.

In the first approach, pairs of face images are input to the training algorithm to learn a feature embedding where positive pairs are closer and negative pairs are far apart. In this direction, Chopra et al. [24] proposed siamese networks with contrastive loss for training. Hu et al. [66] designed a discriminative deep metric with a margin between positive and negative face pairs. FaceNet [157] introduced triplet loss to learn the metric using hard triplet face samples.

In the second approach, the face images along with their subject labels are used to learn discriminative identification features in a classification framework. Most of the recent methods [168, 173, 136, 204, 144] train a DCNN with softmax loss to learn these features which are later used either to directly compute the similarity score for a pair of faces or to train a discriminative metric embedding [154, 21]. Another strategy is to train the network for joint identification-verification task [168, 183, 182]. Xiong et al. [194] proposed transferred deep feature fusion (TDFF) which involves two-stage fusion of features trained with different networks and datasets. Template adaptation [29] is applied to further boost the performance.

A recent approach [182] introduced center loss to learn face embeddings which have better discriminative ability. Our proposed method is different from the center loss in the following aspects. First, we use one loss function (i.e., Crystal Loss) whereas [182] uses center loss jointly with the softmax loss during training. Second, center loss introduces $C \times D$ additional parameters during training where C is the number of classes and D is the feature dimension. On the other hand, the Crystal Loss introduces just a single parameter that defines the fixed L_2-norm of the features. Moreover, the center loss can also be used in conjunction with Crystal Loss, which performs better than center loss trained with regular softmax loss (see Section 4.4.1.4).

Recently, some algorithms have used feature normalization during training to improve performance. Hasnat et al. [60] uses class-conditional von Mises-Fisher distribution to model the feature representation. SphereFace [109] proposes angular softmax (A-softmax) loss that enables DCNNs to learn angularly discriminative features. Another method called DeepVisage [61] uses a special case of batch normalization [72] technique to normalize the feature descriptor before applying softmax loss. Our proposed method is different as it applies an L_2-constraint on the feature descriptors enforcing them to lie on a hypersphere of a given radius.

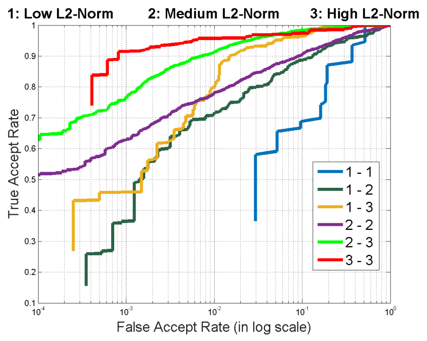
4.2 Motivation

We first summarize the general pipeline for training a face verification system using DCNN as shown in Figure 4.11. Given a training dataset with face images and corresponding identity labels, a DCNN is trained as a classification task where the network learns to classify a given face image to its correct identity label. A softmax loss function is used for training the network, given by (4.1)

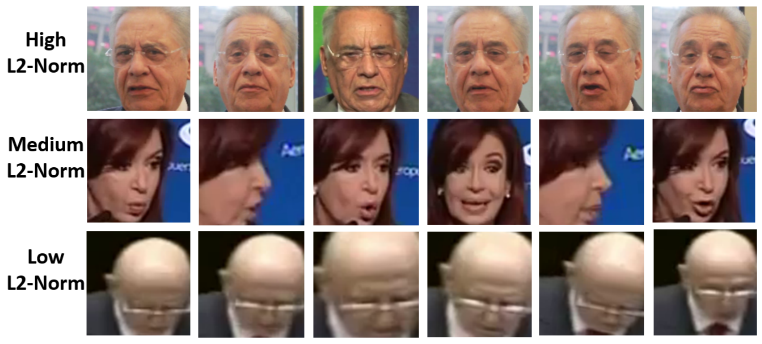
$$L_S = -\frac{1}{M}\sum_{i=1}^{M}\log\frac{e^{W_{y_i}^T f(\mathbf{x}_i)+b_{y_i}}}{\sum_{j=1}^{C} e^{W_j^T f(\mathbf{x}_i)+b_j}}, \tag{4.1}$$

where M is the training batch size, $\mathbf{x}_i$ is the i^{th} input face image in the batch, $f(\mathbf{x}_i)$ is the corresponding output of the penultimate layer of the DCNN, y_i is the corresponding class label, and W and b are the weights and bias for the last layer of the network which acts as a classifier.

At test time, feature descriptors $f(\mathbf{x}_g)$ and $f(\mathbf{x}_p)$ are extracted for the pair of test face images $\mathbf{x}_g$ and $\mathbf{x}_p$ respectively using the trained DCNN, and normalized to unit



(a)



(b)

Figure 4.1: (a) Face Verification Performance on the IJB-A dataset. The templates are divided into 3 sets based on their L_2-norm. '1' denotes the set with low L_2-norm while '3' represents high L_2-norm. The legend 'x-y' denote the evaluation pairs where one template is from set 'x' and another from set 'y'. (b) Sample template images from IJB-A dataset with high, medium and low L2-norm

length. Then, a similarity score is computed on the feature vectors which provides a measure of distance or how close the features lie in the embedded space. If the similarity score is greater than a threshold, the face pairs are decided to be of the same person. Usually, the similarity score is computed as the L_2-distance between the normalized features [157, 136] or by using the cosine similarity score s, as given by (4.2) [182, 21, 145, 154]. Both these similarity measures are equivalent and produce same results.

$$s = \frac{f(\mathbf{x}_g)^T f(\mathbf{x}_p)}{\|f(\mathbf{x}_g)\|_2 \|f(\mathbf{x}_p)\|_2} \tag{4.2}$$

There are two major issues with this pipeline. First, the training and testing steps for face verification task are decoupled. Training with softmax loss doesn't necessarily ensure the positive pairs to be closer and the negative pairs to be far apart in the normalized or angular space.

Secondly, the softmax classifier is weak in modeling difficult or extreme samples. In a typical training batch with data quality imbalance, the softmax loss gets minimized by increasing the L_2-norm of the features for easy samples, and ignoring the hard samples. The network thus learns to respond to the quality of the face by the L_2-norm of its feature descriptor. To validate this claim, we perform a simple experiment on the IJB-A [84] dataset where we divide the templates (groups of images/frames of the same subject) into three different sets based on the L_2-norm of their feature descriptors. The features were computed using Face-Resnet [182] trained with regular softmax loss. Templates with descriptors' L_2-norm <90 are assigned to set1. Templates with L_2-norm >90 but

<150 are assigned to set2, while templates with L_2-norm >150 are assigned to set3. In total, they form six sets of evaluation pairs. Figure 4.1(a) shows the performance of the these six different sets for the IJB-A face verification protocol. It can be clearly seen that pairs having low L_2-norm for both templates perform very poorly, while pairs with high L_2-norm perform the best. The difference in performance between each set is quite significant. Figure 4.1(b) shows some sample templates from set1, set2 and set3 which confirms that the L_2-norm of the feature descriptor is informative of its quality.

To solve these issues, we enforce the L_2-norm of the features to be fixed for every face image. Specifically, we add an L_2-constraint to the feature descriptor such that it lies on a hypersphere of a fixed radius. This approach has two advantages. Firstly, on a hypersphere, minimizing the softmax loss is equivalent to maximizing the cosine similarity for the positive pairs and minimizing it for the negative pairs, which strengthens the verification signal of the features. Secondly, the softmax loss is able to model the extreme and difficult faces better, since all the face features have the same L_2-norm.

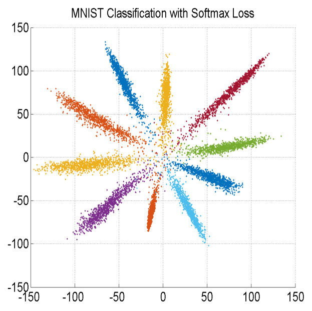
4.3 Proposed Method

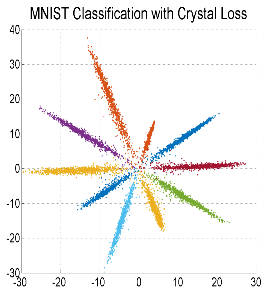
The proposed Crystal Loss is given by (4.3)

$$\begin{aligned} &\text{minimize} \quad -\frac{1}{M}\sum_{i=1}^{M}\log\frac{e^{W_{y_i}^T f(\mathbf{x}_i)+b_{y_i}}}{\sum_{j=1}^{C} e^{W_j^T f(\mathbf{x}_i)+b_j}} \\ &\text{subject to} \quad \|f(\mathbf{x}_i)\|_2 = \alpha, \ \ \forall i = 1,2,...M, \end{aligned} \tag{4.3}$$

where $\mathbf{x}_i$ is the input image in a mini-batch of size M, y_i is the corresponding class label, $f(\mathbf{x}_i)$ is the feature descriptor obtained from the penultimate layer of DCNN, C is the number of subject classes, and W and b are the weights and bias for the last layer of the network which acts as a classifier. Equation (4.3) adds an additional L_2-constraint to the softmax loss defined in (4.1). We show the effectiveness of this constraint using MNIST [95] data.

4.3.1 MNIST Example





(a) (b)

Figure 4.2: Vizualization of 2-dimensional features for MNIST digit classification test set using (a) Softmax Loss. (b) Crystal Loss

We study the effect of Crystal Loss on the MNIST dataset [95]. We use a deeper and wider version of LeNet mentioned in [182], where the last hidden layer output is restricted to 2-dimensional space for easy visualization. For the first setup, we train the network end-to-end using the regular softmax loss for digit classification with the number of classes equal to 10. For the second setup, we add an L_2-normalize layer and a scale layer to the 2-dimensional features which enforces the L_2-constraint described in (4.3) (seen Section 4.3.2 for details). Figure 4.2 depicts the 2-D features for different classes for the MNIST test set containing $10,000$ digit images. Each of the lobes shown in the figure represents 2-D features of unique digits classes. The features for the second setup were obtained before the L_2-normalization layer.

Table 4.1: Accuracy on MNIST test set in (%)

	Softmax Loss	Crystal Loss
Accuracy	98.88	99.05

We find two clear differences between the features learned using the two setups discussed above. First, the intra-class angular variance is large when using the regular softmax loss, which can be estimated by the average width of the lobes for each class. On the other hand, the features obtained with crystal loss have lower intra-class angular variability, and are represented by thinner lobes. Second, the magnitudes of the features are much higher with the softmax loss (ranging upto 150), since larger feature norms result in a higher probability for a correctly classified class. In contrast, the feature norm has minimal effect on the crystal loss since every feature is normalized to a circle of fixed radius before computing the loss. Hence, the network focuses on bringing the features from the same class closer to each other and separating the features from different classes in the normalized or angular space. Table 4.1 lists the accuracy obtained with the two setups on MNIST test set. Crystal loss achieves a higher performance, reducing the error by more than 15%. Note that these accuracy numbers are lower compared to a typical DCNN since we are using only 2-dimensional features for classification.

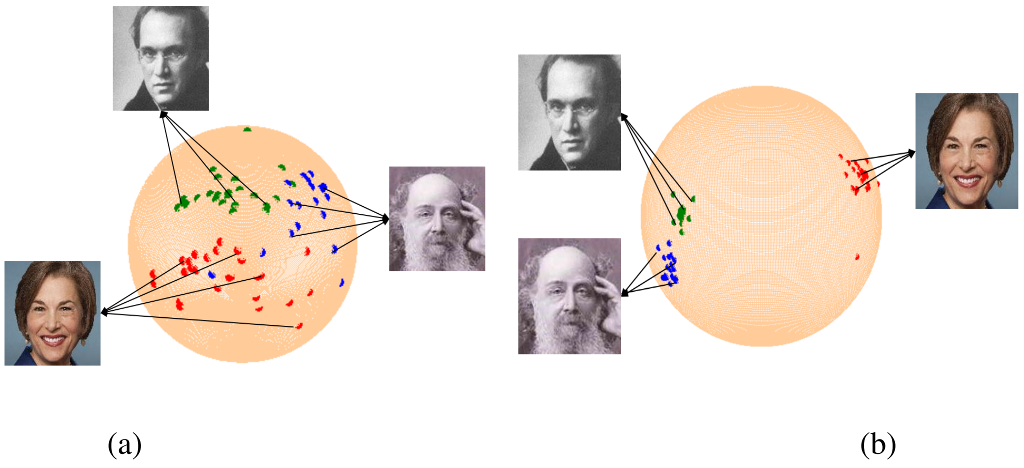


Figure 4.3: Three-dimensional normalized features for three different identities, obtained from (a) network trained with Softmax Loss. (b) network trained with Crystal Loss. The intra-class cosine distance reduces while the inter-class cosine distance increases by using the Crystal Loss.

4.3.2 Implementation Details

Here, we provide the details of implementing the L_2-constraint described in (4.3) in the framework of DCNNs. The constraint is enforced by adding an L_2-normalization layer followed by a scale layer as shown in Figure 4.4.

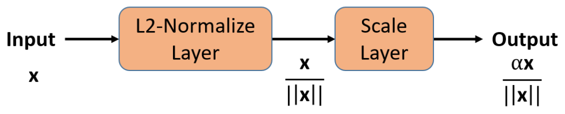


Figure 4.4: We add an L_2-normalization layer and a scale layer to constrain the feature descriptor to lie on a hypersphere of radius α.

This module is added just after the penultimate layer of DCNN which acts as a feature descriptor. The L_2-normalization layer normalizes the input feature $\mathbf{x}$ to a unit vector given by (4.4). The scale layer scales the input unit vector to a fixed radius given by the parameter α (4.5). In total, we just introduce one scalar parameter (α) which can be trained along with the other parameters of the network.

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \tag{4.4}$$

$$\mathbf{z} = \alpha \cdot \mathbf{y} \tag{4.5}$$

The module is fully differentiable and can be used in the end-to-end training of the network. At test time, the proposed module is redundant, since the features are eventually normalized to unit length while computing the cosine similarity. At training time, we backpropagate the gradients through the L2-normalization and the scale layer, as well as compute the gradients with respect to the scaling parameter α using the chain rule as given below.

$$\begin{aligned}
\frac{\partial l}{\partial y_i} &= \frac{\partial l}{\partial z_i} \cdot \alpha \\
\frac{\partial l}{\partial \alpha} &= \sum_{j=1}^{D} \frac{\partial l}{\partial z_j} \cdot y_j \\
\frac{\partial l}{\partial x_i} &= \sum_{j=1}^{D} \frac{\partial l}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \\
\frac{\partial y_i}{\partial x_i} &= \frac{\|\mathbf{x}\|_2^2 - x_i^2}{\|\mathbf{x}\|_2^3} \\
\frac{\partial y_j}{\partial x_i} &= \frac{-x_i \cdot x_j}{\|\mathbf{x}\|_2^3}
\end{aligned} \tag{4.6}$$

The features learned using Softmax Loss and Crystal Loss are shown in Figure 4.3. We train two networks, one with Softmax Loss and another with Crystal Loss, using

100 training identities. We restrict the feature dimension to three for better visualization on a sphere. The blue, green and red points depict the L_2-normalized features for three different identities. It is clear from the figure that Crystal Loss forces the features to have a low intra-class angular variability and higher inter-class angular variability, which improves the face verification accuracy.

4.3.3 Bounds on Parameter α

The scaling parameter α plays a crucial role in deciding the performance of L_2-softmax loss. There are two ways to enforce the L_2-constraint: 1) by keeping α fixed throughout the training, and 2) by letting the network to learn the parameter α. The second way is elegant and always improves over the regular softmax loss. But, the α parameter learned by the network is high which results in a relaxed L_2-constraint. The softmax classifier aimed at increasing the feature norm for minimizing the overall loss, increases the α parameter instead, allowing it more freedom to fit to the easy samples. Hence, the α learned by the network forms an upper bound for the parameter. Improved performance is obtained by fixing α to a lower constant value.

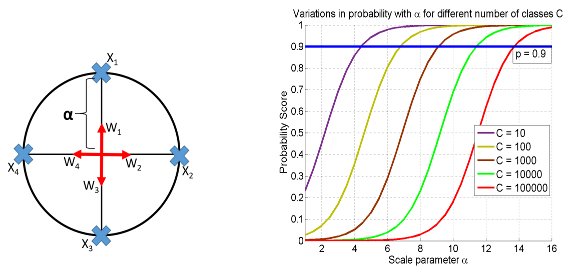
On the other hand, with a very low value of α, the training algorithm does not converge. For instance, $\alpha = 1$ performs poorly on the LFW [70] dataset, achieving an accuracy of 86.37% (see Figure 4.8). The reason being that a hypersphere with small radius (α) has limited surface area for embedding features from the same class together and those from different classes far from each other.

Here, we formulate a theoretical lower bound on α. Assuming the number of classes C to be lower than twice the feature dimension D, we can distribute the classes on a hypersphere of dimension D such that the centers of any two classes are at least 90° apart. Figure 4.5(a) represents this case for $C = 4$ class centers distributed on a circle of radius α. We assume the classifier weights (W_i) to be a unit vector pointing in the direction of their respective class centers. We ignore the bias term. The average softmax probability p for correctly classifying a feature is given by (4.7)

$$\begin{aligned} p &= \frac{e^{W_i^T X_i}}{\sum_{j=1}^{4} e^{W_j^T X_i}} \\ &= \frac{e^{\alpha}}{e^{\alpha} + 2 + e^{-\alpha}} \end{aligned} \tag{4.7}$$

Ignoring the term $e^{-\alpha}$ and generalizing it for C classes, the average probability becomes:

$$p = \frac{e^{\alpha}}{e^{\alpha} + C - 2} \tag{4.8}$$



(a) (b)

Figure 4.5: (a) 2-D vizualization of the assumed distribution of features (b) Variation in Softmax probability with respect to α for different number of classes C

Figure 4.5(b) plots the probability score as a function of the parameter α for various number of classes C. We can infer that to achieve a given classification probability (say $p = 0.9$), we need to have a higher α for larger C. Given the number of classes C for a dataset, we can obtain the lower bound on α to achieve a probability score of p by using (4.9).

$$\alpha_{low} = \log \frac{p(C-2)}{1-p} \tag{4.9}$$

4.3.4 Relation to von Mises-Fisher Distribution

The distribution of features learned using Crystal Loss can be characterized as a special case of von Mises-Fisher distribution [60]. In directional statistics, von Mises-Fisher distribution is a probability distribution on a hypersphere, whose probability density function is represented using (4.10)

$$f_p(\mathbf{x}, \mu, \kappa) = C_p \exp(\kappa \mu^T \mathbf{x}), \tag{4.10}$$

where $\kappa \geq 0$ is the concentration parameter, $\|\mu\|_2 = 1$, $\|\mathbf{x}\|_2 = 1$, and C_p is the normalization constant dependent on κ and the feature dimension p. Keeping the concentration parameter κ same for all the C classes, the log maximum a posteriori estimate for the parameters of von Mises-Fisher distribution results in the formulation of Crystal Loss

(**L**) as shown in (4.11)

$$\begin{aligned}\mathbf{L} = &\text{ maximize } \log \frac{f_p(\mathbf{x}_i, \mu_i, \kappa)}{\sum_{j=1}^{C} f_p(\mathbf{x}_j, \mu_j, \kappa)} \\ = &\text{ minimize } -\log \frac{\exp(\kappa \mu_i^T \mathbf{x}_i)}{\sum_{j=1}^{C} \exp(\kappa \mu_j^T \mathbf{x}_j)}\end{aligned} \tag{4.11}$$

The concentration parameter κ corresponds to the scale factor in the Crystal Loss. The κ value decides the spread of the features on the hypersphere, as shown in Figure 4.6 [*]. A low value of κ results in high intra-class angular variability, while a high value of κ decreases the inter-class angular distance. Hence, an optimal value of κ or the scale factor for Crystal Loss is required (see Section 4.3.3) so that features from same class are close together and features from different classes are far from each other in angular space. We do not normalize the classifier weight vectors since it significantly slows down the training process for large number of classes.

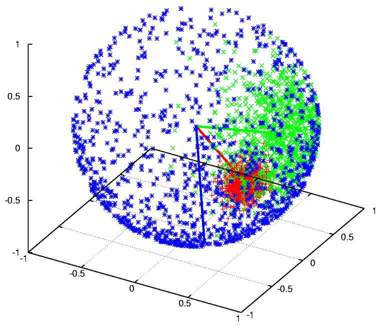


Figure 4.6: Visualization of features on a sphere sampled from von Mises-Fisher distribution. The blue, green and red color represents features for different concentration parameters $\kappa = 1$, $\kappa = 10$ and $\kappa = 100$ respectively.

[*]https://en.wikipedia.org/wiki/Von_MisesFisher_distribution

4.4 Results

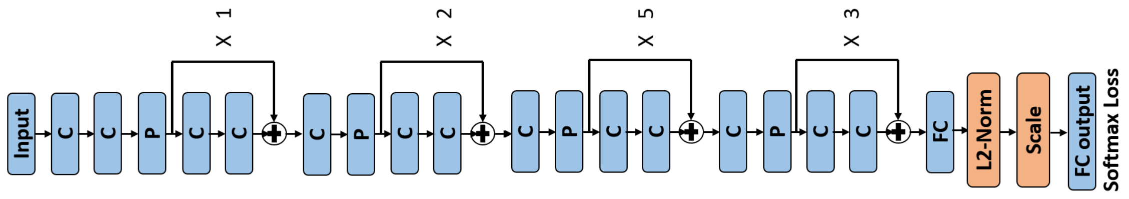


Figure 4.7: The Face-Resnet architecture [182] used for the experiments. **C** denotes Convolution Layer followed by PReLU [64] while **P** denotes Max Pooling Layer. Each pooling layer is followed by a set of residual connections, the count for which is denoted alongside. After the fully-connected layer (**FC**), we add an L_2-Normalize layer and Scale Layer which is followed by the softmax loss.

We use the publicly available Face-Resnet [182] DCNN for our experiments. Figure 4.7 shows the architecture of the network. It contains 27 convolutional layers and 2 fully-connected layers. The dimension of the feature descriptor is 512. It utilizes the widely used residual skip-connections [62]. We add an L_2-normalization layer and a scale layer after the fully-connected layer to enforce the L_2-constraint on the descriptor. All our experiments are carried out in Caffe [75].

4.4.1 Baseline experiments

In this subsection, we experimentally validate the usefulness of the L_2-softmax loss for face verification. We form two subsets of training dataset from the MS-Celeb-1M [57] dataset: 1) MS-small containing 0.5 million face images with the number of subjects being 13403, and 2) MS-large containing 3.7 million images of 58207 subjects. The dataset was cleaned using the clustering algorithm presented in [106]. We train the Face-Resnet network with softmax loss as well as Crystal loss for various α. While training with MS-small, we start with a base learning rate of 0.1 and decrease it by $1/10^{th}$ after $16K$ and $24K$ iterations, upto a maximum of $28K$ iterations. For training on MS-large, we use the same learning rate but decrease it after $50K$ and $80K$ iterations upto a maximum of $100K$ iterations. A training batch size of 256 was used. Both softmax and Crystal loss functions consume the same amount of training time which is around 9 hours for MS-small and 32 hours for MS-large training set respectively, on two TITAN X GPUs. We set the learning multiplier and decay multiplier for the scale layer to 1 for trainable α, and 0 for fixed α during the network training. We evaluate our baselines on the widely used LFW dataset [70] for the unrestricted setting, and the challenging IJB-A dataset [84] for the 1:1 face verification protocol. The faces were cropped and aligned to the size of 128×128 in both training and testing phases by implementing the face detection and alignment algorithm presented in [145] .

4.4.1.1 Experiment with small training set

Here, we compare the network trained on MS-small dataset using the proposed Crystal loss, against the one trained with regular softmax loss. Figure 4.8 shows that the softmax loss attains an accuracy of 98.1% whereas the proposed Crystal loss achieves the best accuracy of 99.28%, thereby reducing the error by more than 62%. It also shows the variations in performance with the scale parameter α. The performance is poor when α is below a certain threshold and stable with α higher than the threshold. This behavior is consistent with the theoretical analysis presented in Section 4.3.3. From the figure, the performance of Crystal loss is better for α >12 which is close to its lower bound computed using equation 4.9 for $C = 13403$ with a probability score of 0.9.

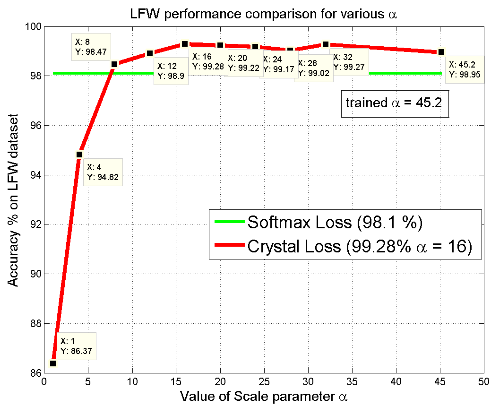


Figure 4.8: The red curve shows the variations in LFW accuracy with the parameter α for Crystal loss. The green line is the accuracy using softmax loss.

A similar trend is observed for 1:1 verification protocol on IJB-A [84] as shown in Table 4.2, where the numbers denote True Accept Rate (TAR) at False Accept Rates (FAR) of 0.0001, 0.001, 0.01 and 0.1. Our proposed approach improves the TAR@FAR=0.0001 by 19% compared to the baseline softmax loss. The performance is consistent with α ranging between 16 to 32. Another point to note is that by allowing the network to learn the scale parameter α by itself results in a slight decrease in performance, which shows that having a tighter constraint is a better choice.

4.4.1.2 Experiment with large training set

We train the network on the MS-large dataset for this experiment. Figure 4.9 shows the performance on the LFW dataset. Similar to the small training set, the Crystal loss

Table 4.2: TAR on IJB-A 1:1 Verification Protocol @FAR

	0.0001	0.001	0.01	0.1
Softmax Loss	0.553	0.730	0.881	0.957
Crystal Loss (α=8)	0.257	0.433	0.746	0.953
Crystal Loss (α=12)	0.620	0.721	0.875	0.970
Crystal Loss (α=16)	0.734	0.834	**0.924**	0.974
Crystal Loss (α=20)	0.740	0.820	0.922	0.973
Crystal Loss (α=24)	**0.744**	0.831	0.912	0.974
Crystal Loss (α=28)	0.740	**0.834**	0.922	**0.975**
Crystal Loss (α=32)	0.727	0.831	0.921	0.972
Crystal Loss (α trained)	0.698	0.817	0.914	0.971

significantly improves over the baseline, reducing the error by 60% and achieving an accuracy of 99.6%. Similarly, it improves the TAR@FAR=0.0001 on IJB-A by more than 10% (Table 4.3). The performance of Crystal loss is consistent with α in the range 40 and beyond. Unlike, the small set training, the self-trained α performs equally good compared to fixed α of 40 and 50. The theoretical lower bound on α is not of much use in this case since improved performance is achieved for $\alpha >$30. We can deduce that as the number of subjects increases, the lower bound on α is less reliable, and the self-trained α is more reliable with performance. This experiment clearly suggests that the proposed Crystal loss is consistent across the training and testing datasets.

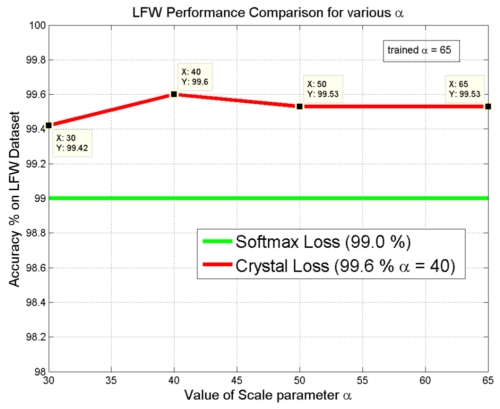


Figure 4.9: The red curve shows the variations in LFW accuracy with the parameter α for Crystal loss. The green line is the accuracy using the Softmax loss.

4.4.1.3 Experiment with a different DCNN

To check the consistency of our proposed Crystal loss, we apply it on the All-In-One Face [145] instead of the Face-Resnet. We use the recognition branch of the All-In-One Face to fine-tune on the MS-small training set. The recognition branch of All-In-One

Table 4.3: TAR on IJB-A 1:1 Verification Protocol @FAR

	0.0001	0.001	0.01	0.1
Softmax Loss	0.730	0.851	0.926	0.972
Crystal Loss (α=30)	0.775	0.871	0.938	0.978
Crystal Loss (α=40)	0.827	0.900	0.951	**0.982**
Crystal Loss (α=50)	**0.832**	**0.906**	**0.952**	0.981
Crystal Loss (α trained)	**0.832**	0.903	0.950	0.980

Face consists of 7 convolution layers followed by 3 fully-connected layers and a softmax loss. We add an L_2-normalize and a scale layer after the 512 dimension feature descriptor. Figure 4.10 shows the comparison of Crystal loss and the Softmax loss on LFW dataset. Similar to the Face-Resnet, All-In-One Face with Crystal loss improves over the Softmax performance, reducing the error by 40%, and achieving an accuracy of 98.82%. The improvement obtained by using All-In-One Face is smaller compared to the Face-Resnet. This shows that residual connections and depth of the network generate better feature embedding on a hypersphere. The performance variation with scaling parameter α is similar to that of Face-Resnet, indicating that the optimal scale parameter does not depend on the choice of the network.

4.4.1.4 Experiment with auxiliary loss

Similar to softmax loss, the Crystal loss can be coupled with auxiliary losses such as center loss, contrastive loss, triplet loss, etc. to further improve the performance. Here

Table 4.4: Accuracy on LFW (%)

Softmax loss	98.10
Center loss [182] + Softmax loss	99.23
Crystal loss	99.28
Center loss [182] + Crystal loss	**99.33**

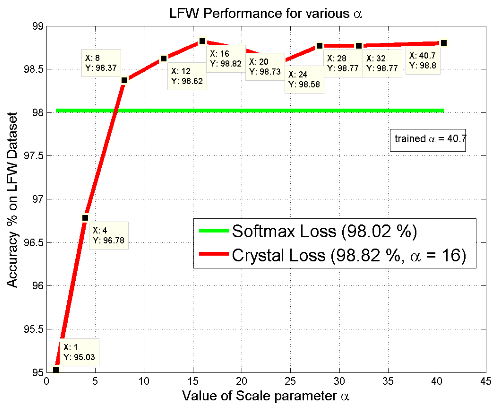


Figure 4.10: The red curve shows the variations in LFW accuracy with the parameter α for Crystal loss. The green line is the accuracy using the Softmax loss.

we study the performance variation of Crystal loss when coupled with the center loss. We use the MS-small dataset for training the networks. Table 4.4 lists the accuracy obtained on the LFW dataset by different loss functions. The softmax loss performs the worst. The center loss improves the performance significantly when trained in conjunction with the softmax loss, and is comparable to the Crystal loss. Training center loss with Crystal loss gives the best performance of 99.33% accuracy. This shows that Crystal loss is as versatile as the softmax loss and can be used efficiently with other auxiliary loss functions.

4.4.2 Experiments on LFW and YTF Datasets

We compare our algorithm with recently reported face verification methods on LFW [70] and YouTube Face [189] datasets. We crop and align the images for all these datasets by implementing the algorithm mentioned in [145]. We train the Face-Resnet (FR) with Crystal loss (CrL) as well as Softmax loss using the MS-large training set. Additionally, we train ResNet-101(R101) [62] and ResNeXt-101(RX101) [191] deep networks for face recognition using MS-large training set with Crystal loss. Both R101 and RX101 models were initialized with parameters pre-trained on ImageNet [152] dataset. A fully-connected layer of dimension 512 was added before the Crystal loss classifier. The scaling parameter was kept fixed with a value of $\alpha = 50$. Experimental results on different datasets show that Crystal loss works efficiently with deeper models.

The LFW dataset [70] contains $13,233$ web-collected images from 5749 different identities. We evaluate our model following the standard protocol of unrestricted with labeled outside data. We test on 6,000 face pairs and report the experiment results in

Table 4.5: Verification accuracy (in %) of different methods on LFW and YTF datasets.

Method	Images	#nets	One loss	LFW	YTF
Deep Face [173]	4M	3	No	97.35	91.4
DeepID-2+ [168]	-	25	No	99.47	93.2
FaceNet [157]	200M	1	Yes	99.63	95.12
VGG Face [136]	2.6M	1	No	98.95	**97.3**
Baidu [107]	1.3M	1	No	99.13	-
Wen et al. [182]	0.7M	1	No	99.28	94.9
NAN [204]	3M	1	No	–	95.72
DeepVisage [61]	4.48M	1	Yes	99.62	**96.24**
SphereFace [109]	0.5M	1	Yes	99.42	95.0
Softmax(FR)	3.7M	1	Yes	99.0	93.82
CrL (FR)	3.7M	1	Yes	99.60	95.54
CrL (R101)	3.7M	1	Yes	99.67	96.02
CrL (RX101)	3.7M	1	Yes	**99.78**	**96.08**

Table 4.5. Along with the accuracy values, we also compare with the number of images, networks and loss functions used by the methods for their overall training. The proposed method attains state-of-the-art performance with the RX101 model, achieving an accuracy of 99.78%. Unlike other methods which use auxiliary loss functions such as center loss and contrastive loss along with the primary softmax loss, our method uses a single loss training paradigm which makes it easier and faster to train.

YouTube Face (YTF) [189] dataset contains 3425 videos of 1595 different people, with an average length of 181.3 frames per video. It contains 10 folds of 500 video pairs. We follow the standard verification protocol and report the average accuracy on splits with cross-validation in Table 4.5. We achieve the accuracy of 96.08% using Crystal loss with RX101 network. Our method outperforms many recent algorithms and is only behind DeepVisage [61] which uses larger number of training samples, and VGG Face [136] which further uses a discriminative metric learning on YTF.

4.5 A State-of-the-art Face Verification and Recognition Pipeline

In this section, we discuss a state-of-the-art end-to-end pipeline for face identification and verification, built by authors over the last eighteen months. An overview of the pipeline is given in Fig. 4.11. Given an image, we first detect all the faces using the DPSSD face detector described in chapter 2. Then, we crop out all the detected faces from the image and pass them through the All-In-One face [145] network to extract facial key-points (described in chapter 3). These key-points are used to align the corresponding faces in canonical coordinates. The aligned faces are then passed through face DCNNs,

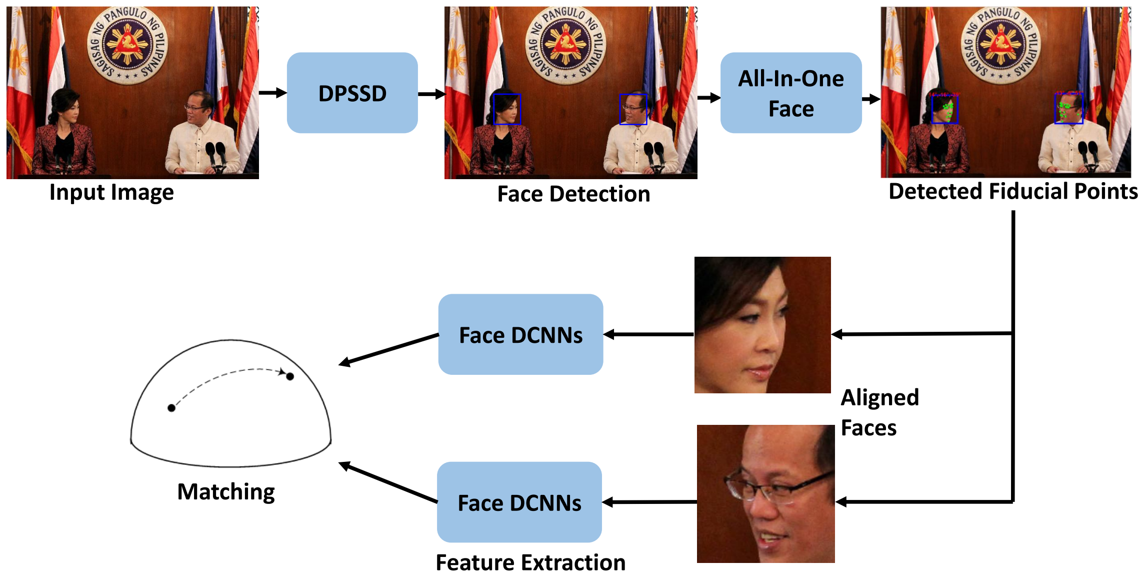


Figure 4.11: Our face recognition pipeline. We detect faces using our proposed DPSSD face detector (chapter 2). These detections are passed to the All-in-One Face network [145] which outputs facial keypoints for each face. These are used to align faces to canonical views. We pass these aligned faces through our face representation networks and obtain the similarity between two faces using cosine similarity.

trained using Crystal Loss, to generate feature descriptors which are later used for verifying or identifying a face.

4.5.1 Training Datasets

We use the Universe face dataset from [7] for training our face representation networks. This is a combination of UMDFaces images [6], UMDFaces video frames [5], and curated MS-Celeb-1M [57]. The Universe dataset contains about 5.6 million images of about 58,000 identities. This includes about 3.5 million images from MS-Celeb-1M, 1.8 million video frames from UMDFaces videos, and 300,000 images from UMDFaces. This dataset has the advantage of being a combination of different datasets which makes networks trained using this dataset generalize better. Another advantage is that it contains both still images and video frames which makes the networks more robust to test datasets that contain both images and videos.

4.5.2 Face Representation

We use two networks for feature representation and perform fusion by averaging the similarity scores obtained from each of them. Using an ensemble of networks leads to more robust representations and better performance. We next describe the two networks along with their respective training details. These two networks are based on a ResNet-101 [62], and Inception ResNet-v2 [170]. For pre-processing the detected faces, we crop and resize the aligned faces to each network's input dimensions. For data augmentation, we apply random horizontal flips to the input images.

ResNet-101 (R) : We train a ResNet-101 deep convolutional neural network with PReLU activations after every convolution layer. Since we use the Universe dataset for training the network, we use a $58,000$-way classification layer with crystal loss. For this network, we set the α parameter to 50 and the batch size at 128. The learning rate is set to 0.1 and is reduced by a factor of 0.2 after every $50k$ iterations. The network is trained for a total of $250k$ iterations. We use a 512-D layer as the feature layer and use TPE [154] to find a 128-D embedding which was trained with the UMDFaces dataset.

Inception ResNet-v2 (A) : The Inception ResNet-v2 network is also trained with the Universe dataset. This network has 244 convolution layers. We add a 512-D feature layer after these and then a final classification layer. We again use crystal loss with $\alpha = 40$. The initial learning rate is set to 0.1 and is reduced by a factor of 0.2 after every $50k$ iterations. We train the network for $120k$ iterations with a batch-size of 120 on 8 NVIDIA Quadro P6000 GPUs. We resize the inputs to 299×299. Similar to the ResNet, we use UMDFaces to train a final 128-D embedding with TPE.

4.5.3 Feature Fusion

Template Feature: For both face verification and identification, we need to compare template features. To obtain feature vectors for a template, we first average all the features for a media in the template. We further average these media-averaged features to get the final template feature.

Score-level Fusion: To get the similarity between two templates, we average the similarities obtained by our two networks.

4.5.4 Experimental Results

In this section, ROC curves are used to measure the performance of face verification (1:1 matching) methods and CMC scores are used for evaluating face identification (1:N search). The IJB-A [84], IJB-B [187], IJB-C [118], and CS5 datasets contain a gallery and a probe which leads to evaluation using all positive and negative pairs. This is different from LFW [70] and YTF [189] where only a few negative pairs are used to evaluate verification performance. Another difference between LFW/YTF and the evaluation datasets here is the inclusion of templates instead of only single images. A template is a collection of images and video frames of a subject. These datasets are much more challenging than older datasets due to extreme variations in pose, illumination, and expression. We evaluate our models for the following four protocols on the IJB and CS5 datasets:

1:1 Verification: Verify if the given pair of templates belong to the same subject. Templates are comprised of mixed media (frames and stills).

1:N Mixed Search: Open set identification protocol using mixed media (frames and stills) as probe and galleries.

Wild Probe Search: Identify subjects of interest from a collection of faces detected from still images and frames.

1:N end-to-end Still Image: Identify identity clusters of interest from a collection of still images.

4.5.4.1 IJB-A

The IJB-A dataset contains 500 subjects with 5,397 images and 2,042 videos split into 20,412 frames. This dataset is a very difficult dataset owing to the presence of extreme pose, viewpoint, resolution, and illumination variations. Additionally, mixing still images and video frames causes difficulties for models trained with only one of these modalities due to domain shift. An identity in this dataset is represented as a template. Also note that each subject can have multiple templates in the dataset. The evaluation protocol for this dataset contains for 1:1 verification and 1:N mixed search. The dataset is divided into 10 splits, each with 333 randomly selected subjects for training and 167 subjects for testing. We generate a common template representation by fusing features of all the faces in the template. We compute the similarity scores using the two networks (**R** and **A**) and then do a score-level fusion as described in 4.5.3. Table 4.6 provides the results from our system for the verification task and Table 4.7 provides the results for 1:N mixed search for the IJB-A dataset. We achieve the state-of-the-art results for every setting.

4.5.4.2 IJB-B

The IJB-B dataset [187], which extends IJB-A, contains about $22,000$ still images and $55,000$ video frames spread over $1,845$ subjects. Evaluation is done for the same tasks as IJB-A, *viz.*, 1:1 verification, and 1:N identification. The IJB-B verification protocol consists of $8,010,270$ pairs between templates in the galleries (G1 and G2) and the probe templates. Out of these, 8 million are impostor pairs and the rest $10,270$ are

	True Accept Rate (%) @ False Accept Rate			
Method	0.0001	0.001	0.01	0.1
Casia [179]	-	51.4	73.2	89.5
Pose [2]	-	-	78.7	91.1
NAN [204]	-	88.1	94.1	97.8
3D [115]	-	72.5	88.6	-
DCNN$_{fusion}$ [22]	-	76.0	88.9	96.8
DCNN$_{tpe}$ [154]	-	81.3	90.0	96.4
DCNN$_{all}$ [145]	-	78.7	89.3	96.8
All + TPE [145]	-	82.3	92.2	97.6
TP [29]	-	-	93.9	-
RX101$_{l2+tpe}$ [142]	90.9	94.3	97.0	98.4
Ours$_A$	<u>91.7</u>	**95.3**	96.8	98.3
Ours$_R$	91.4	94.8	**97.1**	**98.5**
Fusion (Ours)	**92.1**	<u>95.2</u>	<u>96.9</u>	<u>98.4</u>

Table 4.6: IJB-A Verification. Ours$_A$ is the Inception ResNet-v2 model and Ours$_R$ is the ResNet-101 model. The best results are in bold and the second best results are underlined.

	TPIR (%) @ FPIR		Retrieval Rate (%)		
Method	0.01	0.1	Rank=1	Rank=5	Rank=10
Casia [179]	38.3	61.3	82.0	92.9	-
Pose [2]	52.0	75.0	84.6	92.7	94.7
BL [25]	-	-	89.5	96.3	-
NAN [204]	81.7	91.7	95.8	98	98.6
3D [115]	-	-	90.6	96.2	97.7
DCNN$_{fusion}$ [22]	65.4	83.6	94.2	98.0	98.8
DCNN$_{tpe}$ [154]	75.3	83.6	93.2	-	97.7
DCNN$_{all}$ [145]	70.4	83.6	94.1	-	98.8
ALL + TPE [145]	79.2	88.7	94.7	-	98.8
TP [29]	77.4	88.2	92.8	-	98.6
RX101$_{l2+tpe}$ [142]	91.5	95.6	97.3	-	98.8
Ours$_A$	91.4	$\underline{96.1}$	97.3	98.2	98.5
Ours$_R$	$\underline{91.6}$	96.0	$\underline{97.4}$	$\underline{98.5}$	$\underline{98.9}$
Fusion (Ours)	**92.0**	**96.2**	**97.5**	**98.6**	**98.9**

Table 4.7: IJB-A 1:N Mixed Search. Ours$_A$ is the Inception ResNet-v2 model and Ours$_R$ is the ResNet-101 model. The best results are in bold and the second best results are underlined.

genuine comparisons. Table 4.8 and Table 4.9 provide the verification and identification results respectively.

4.5.4.3 IJB-C

The IJB-C evaluation dataset [118] further extends IJB-B. It contains $31,334$ still images and $117,542$ video frames of $3,531$ subjects. In addition to the evaluations from IJB-B, this dataset evaluates end-to-end recognition. There are about $20,000$ genuine comparisons, and about 15.6 million impostor pairs in the verification protocol. For the 1:N mixed search protocol, there are about $20,000$ probe templates. In Table 4.10 we list the results of our system for 1:1 verification. Similarly, in Table 4.11 we give results for 1:N mixed search. We also report the 1:N wild probe search results in Table 4.12.

4.5.4.4 CS5

We evaluate on the (as-yet-unreleased to public) JANUS Challenge Set 5 dataset as well. This dataset consists of $2,875,950$ still images. It also provides a training set consisting of $235,616$ identity clusters and $981,753$ images. Note that we did not use this training set to train our networks. The still image verification protocol contains $547,131$ templates with $332,574$ genuine matches, and $822,354,805$ imposter matches. For the 1:N identification task, there are $332,574$ probe templates. Gallery, G1 has $1,106,778$ identity clusters and G2 has $1,107,779$ identity clusters. The major differences between CS5 and IJB datasets are: 1) CS5 contains much more templates and verification pairs compared to IJB datasets, 2) CS5 gallery contains 1 million distractor faces

	True Accept Rate (%) @ False Accept Rate					
Method	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
GOTS [187]	-	-	16.0	33.0	60.0	-
VGGFaces [136]	-	-	55.0	72.0	86.0	-
FPN [14]	-	-	83.2	91.6	96.5	-
Light CNN-29 [190]	-	-	87.7	92.0	95.3	-
VGGFace2 [9]	-	70.5	83.1	90.8	95.6	-
Center Loss [182]	31.0	63.6	80.7	90.0	95.1	98.4
MN-vc [193]	-	-	83.1	90.9	95.8	98.5
SENet50+DCN [192]	-	-	84.9	93.7	**97.5**	**99.7**
ArcFace [32]	37.5	**89.0**	**94.2**	**96.0**	<u>97.5</u>	98.4
Ours$_A$	27.7	61.6	89.1	94.3	97.0	98.7
Ours$_R$	**48.4**	<u>80.4</u>	89.8	94.4	97.2	98.9
Fusion (Ours)	<u>45.6</u>	77.8	<u>90.3</u>	<u>94.6</u>	97.3	<u>98.9</u>

Table 4.8: IJB-B Verification. Ours$_A$, Ours$_R$, and Fusion are the Inception ResNet-v2, ResNet-101, and Fused features respectively. The best results are in bold and the second best results are underlined.

	TPIR (%) @ FPIR (For G1, G2)		Retrieval Rate (%) (For G1, G2)		
Method	0.01	0.1	Rank=1	Rank=5	Rank=10
GOTS [187]	-	-	42.0	-	62.0
VGGFace [136]	-	-	78.0	-	89.0
FPN [14]	-	-	91.1	-	96.5
Light CNN-29 [190]	-	-	91.9	94.8	-
VGGFace2 [9]	74.3	86.3	90.2	94.6	95.9
Center Loss [182]	75.5, 67.7	87.5, 82.8	92.2, 86.0	95.4, 92.5	96.2, 94.4
Ours$_A$	83.1, 75.5	93.6, <u>89.3</u>	95.5, 90.8	97.5, 94.2	98.0, 95.8
Ours$_R$	<u>86.9</u>, <u>78.6</u>	<u>94.0</u>, 89.1	<u>95.6</u>, <u>91.5</u>	**97.7, 95.4**	<u>98.0</u>, **96.5**
Fusion (Ours)	**88.2, 79.4**	**94.3, 89.7**	**95.8, 91.8**	<u>97.7</u>, <u>95.2</u>	**98.1**, <u>96.4</u>

Table 4.9: IJB-B 1:N Mixed Search. Ours$_A$ and Ours$_R$ are our Inception ResNet-v2 and ResNet-101 models respectively. Note that the retrieval rates for some past methods are average over G1 and G2. The best results are in bold and the second best results are underlined.

	True Accept Rate (%) @ False Accept Rate							
Method	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
Center Loss [182]	36.0	37.6	66.1	78.1	85.3	91.2	95.3	98.2
MN-vc [193]	-	-	-	-	86.2	92.7	96.8	98.9
SENet50+DCN [192]	-	-	-	-	88.5	94.7	**98.3**	**99.8**
ArcFace [32]	-	-	**85.4**	**92.8**	**95.6**	**97.2**	<u>98.0</u>	98.8
Ours$_A$	16.5	19.5	43.6	77.6	91.9	95.6	97.8	99.0
Ours$_R$	**60.6**	**67.4**	<u>76.4</u>	86.2	91.9	95.7	97.9	99.2
Fusion (Ours)	<u>54.1</u>	<u>55.9</u>	69.5	<u>86.9</u>	<u>92.5</u>	<u>95.9</u>	97.9	<u>99.2</u>

Table 4.10: IJB-C Verification. Ours$_A$ is the Inception ResNet-v2 model and Ours$_R$ is the ResNet-101 model. Fusion is the fusion of the two features. The best results are in bold and the second best results are underlined.

	TPIR (%) @ FPIR (For G1, G2)		Retrieval Rate (%) (For G1, G2)		
Method	0.01	0.1	Rank=1	Rank=5	Rank=10
Center Loss [182]	79.1, 75.3	86.4, 84.2	91.7, 89.8	94.6, 93.6	95.6, 94.9
Ours$_A$	87.7, 82.4	93.5, 91.0	95.7, 92.8	97.4, 95.4	97.9, 96.4
Ours$_R$	88.0, 84.2	93.2, 90.6	95.9, 93.2	97.6, 96.1	98.1, **97.0**
Fusion (Ours)	**89.6, 85.0**	**93.8, 91.3**	**96.2, 93.6**	**97.7, 96.2**	**98.2**, 96.9

Table 4.11: IJB-C 1:N Mixed Search. Ours$_A$ and Ours$_R$ are the models described in section 4.5.2.

	Retrieval Rate (%) (For G1, G2)					
Method	Rank=1	Rank=2	Rank=5	Rank=10	Rank=20	Rank=50
Ours$_A$	91.1, 86.9	93.0, 89.0	94.8, 91.1	95.8, 92.5	96.5, 93.8	97.4, 95.3
Ours$_R$	90.8, 86.3	93.0, 88.8	95.0, 91.1	96.0, 92.6	96.7, 93.9	97.5, 95.5
Fusion (Ours)	**91.8, 87.5**	**93.6, 89.7**	**95.3, 91.6**	**96.3, 93.0**	**97.0, 94.4**	**97.7, 95.8**

Table 4.12: IJB-C Wild Probe Search. Our models and fusion method are described in sections 4.5.2 and 4.5.3.

	True Accept Rate (%) @ False Accept Rate							
Method	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
Ours$_A$	52.44	78.99	94.88	97.34	98.18	98.74	99.28	99.75
Ours$_R$	71.52	89.68	95.20	97.28	98.19	98.79	99.36	99.78
Fusion (Ours)	70.72	90.74	95.80	97.49	98.25	98.80	99.35	99.78

Table 4.13: CS5 1:1 Verification. Ours$_A$ and Ours$_R$ are the Inception ResNet-v2 and ResNet-101 models respectively.

from MegaFace dataset [81] which increases the difficulty level for the open-set 1:N face identification task. Tables 4.13, 4.14, and 4.15 give results for 1:1 verification, 1:N identification, and 1:N end-to-end identification respectively.

	TPIR (%) @ FPIR (Average for G1 and G2)				Retrieval Rate (%) (Average for G1 and G2)			
Method	0.0001	0.001	0.01	0.1	Rank=1	Rank=5	Rank=10	Rank=20
Fusion (Ours)	29.96	75.90	86.76	95.59	96.99	97.76	97.92	98.06

Table 4.14: CS5 1:N Identification

	Retrieval Rate (%) (Average for G1 and G2)					
Method	Rank=1	Rank=2	Rank=5	Rank=10	Rank=20	Rank=50
Fusion (Ours)	97.18	97.65	97.90	98.04	98.16	98.31

Table 4.15: CS5 1:N end-to-end still image identification

4.5.5 Timing

The proposed end-to-end face recognition system can detect, align, and extract identity feature descriptors at a rate of 4 frames/second using NVIDIA K40 GPU. The DPSSD face detector takes about 100ms to detect faces with an input image size of 512 pixels. The face alignment part is the fastest step and takes only 20ms. The two face DCNNs (**R** and **A**) take 50ms and 85ms respectively. Since the proposed system is modular, we can improve its speed at the cost of minor reduction in accuracy by just replacing the algorithms for each modules independently. For example, if the dataset contains a single large face per image (LFW [70]), we can replace DPSSD with SSD [20] face detector. Similarly, to improve the speed, we can just use a single face DCNN model **R** to extract feature descriptor. In this scenario, our system can attain a speed of 11 frames/second.

4.6 Face recognition accuracy of examiners and face recognition algorithms

4.6.1 Introduction

[*]Societies rely on the expertise and training of professional forensic facial examiners, because decisions by professionals are thought to assure the highest possible level of face identification accuracy. If accuracy is the goal, however, the scientific literature in psychology and computer vision points to three additional approaches that merit consideration. First, untrained superrecognizers from the general public perform surprisingly well on laboratory-based face recognition studies [129]. Second, wisdom-of-crowds effects for face recognition, implemented by averaging individuals judgments, can boost performance substantially over the performance of a person working alone [184, 186, 34, 131]. Third, computer-based face recognition algorithms over the last decade have steadily closed the gap between human and machine performance on increasingly challenging face recognition tasks [140, 138].

[*]This section contains an extract of a co-authored paper entitled “Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms” that was published in *Proceedings of the National Academy of Sciences*. The co-authors are: P Jonathon Phillips, Amy N Yates, Ying Hu, Carina A Hahn, Eilidh Noyes, Kelsey Jackson, Jacqueline G Cavazos, Graldine Jeckeln, Swami Sankaranarayanan, Jun-Cheng Chen, Carlos D Castillo, Rama Chellappa, David White, and Alice J OToole. The study uses the features from the All-In-One Face model (chapter 3) and the ResNet model trained with Crystal Loss (chapter 4).

Computer-based face recognition systems now assist forensic face examiners by searching databases of images to generate potential identity matches for human review [185]. Direct comparisons between human and machine accuracy have been based on algorithms developed before 2013. At that time, algorithms performed well with high-quality frontal images of faces with minimal changes in illumination and expression. Since then, deep learning and deep convolutional neural networks (DCNNs) have become the state of the art for face recognition [136, 21, 145, 142, 173]. DCNNs can recognize faces from highly variable, low-quality images. These algorithms are often trained with millions of face images of thousands of people.

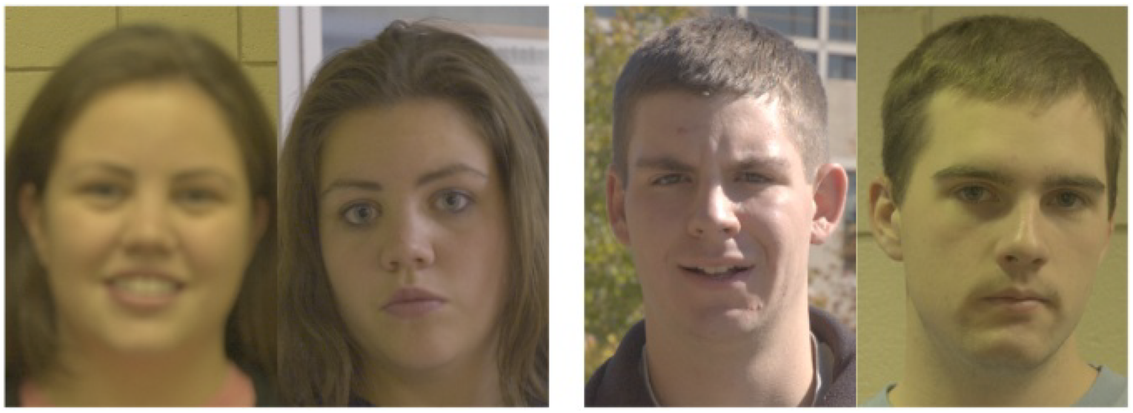


Figure 4.12: Examples highlighting the face region in the images used in this study. (Left) This pair is a same identity pair, and (Right) this pair shows a different identity pair.

Our goal was to achieve the most accurate face identification using people and/or machines working alone or in collaboration. The task was to determine whether pairs of face images showed the same person or different people. Image pairs were prescreened to be highly challenging based on data from humans and computer algorithms. Images were taken with limited control of illumination, expression, and appearance. Fig. 4.12

shows two example pairs. To provide a comprehensive assessment of human accuracy, we tested three face specialist groups (forensic facial examiners, forensic facial reviewers, and superrecognizers) and two control groups (fingerprint examiners and undergraduate students). Humans responded on a 7-point scale that varied from high confidence that the pair showed the same person (+3) to high confidence that the pair showed different people (-3). We also tested four face recognition algorithms based on DCNNs developed between 2015 and 2017. Algorithm responses were real-valued similarity scores indicating the likelihood that the images showed the same person. The five subject groups and four algorithms were tested on the same image pairs. Facial examiners, reviewers, superrecognizers, and fingerprint examiners had 3 months to complete the test. Students took the test in a single session.

To compare humans with face recognition algorithms, four DCNNs were tested on the same stimuli judged by humans. We refer to the algorithms as A2015 [136], A2016 [21], A2017a [145], and A2017b [142]. A2017a is the All-In-One Face algorithm described in chapter 3 while A2017b is a ResNet-based DCNN trained using Crystal Loss. The inclusion of multiple algorithms provides a robust sample of the state of the art for automatic face recognition. To make the test comparable with humans as an "unfamiliar" face matching test, we verified that none of the algorithms had been trained on images from the dataset used for the human test. Note that A2015 can be downloaded from the web and therefore, provides a public benchmark algorithm.

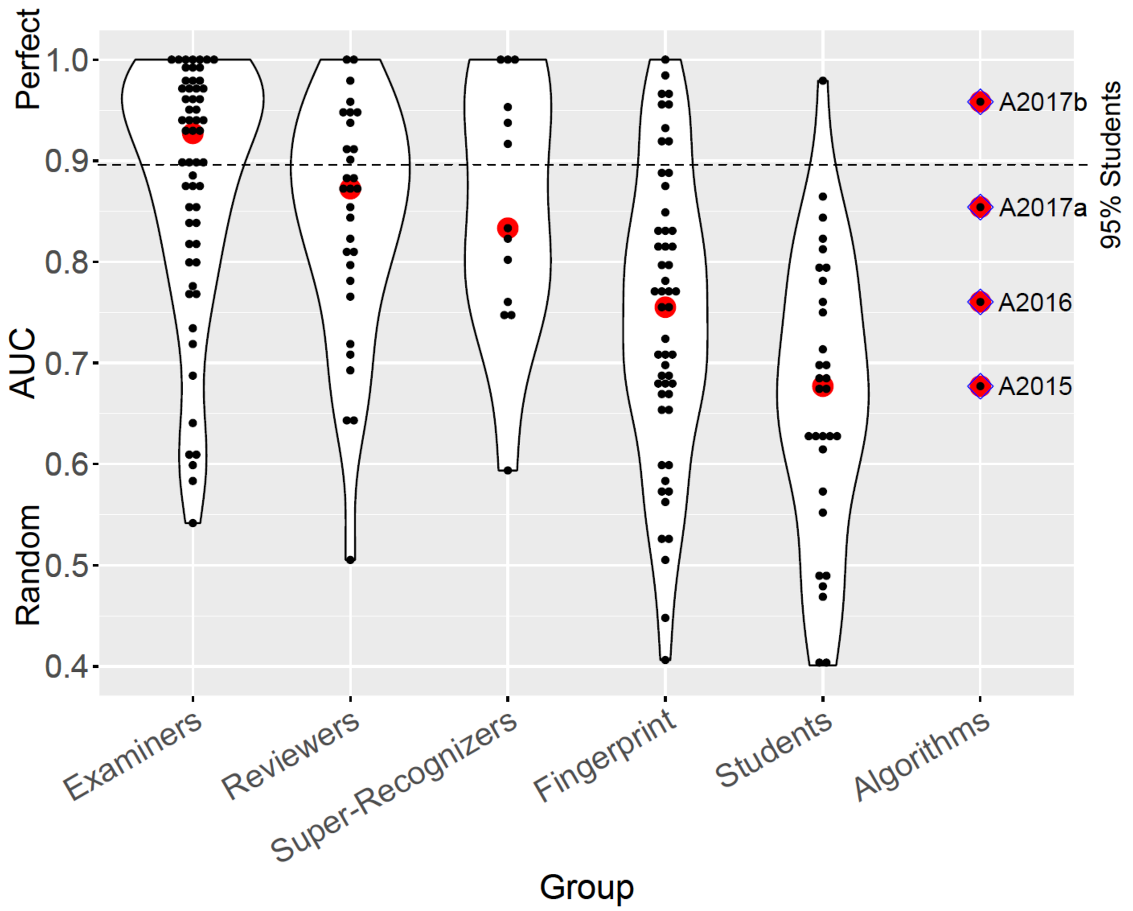


Figure 4.13: Human and machine accuracy. Black dots indicate AUCs of individual participants; red dots are group medians. In the algorithms column, red dots indicate algorithm accuracy. Face specialists (facial examiners, facial reviewers, and superrecognizers) surpassed fingerprint examiners, who surpassed the students. The violin plot outlines are estimates of the density for the AUC distribution for the subject groups. The dashed horizontal line marks the accuracy of a 95th percentile student. All algorithms perform in the range of human performance. The best algorithm places slightly above the forensic examiners median.

4.6.2 Results

4.6.2.1 Accuracy

Fig. 4.13 shows performance of the subject groups and algorithms using the area under the receiver operating characteristic curve (AUC) as a measure of accuracy. The groups are ordered by AUC median from the most to least accurate: facial examiners (0.93), facial reviewers (0.87), superrecognizers (0.83), fingerprint examiners (0.76), and students (0.68). Algorithm performance increased monotonically from the oldest algorithm (A2015) to the newest algorithm (A2017b). Comparing the algorithms with the human groups, the publicly available algorithm (A2015) performed at a level similar to the students (0.68). Algorithm A2016 performed at the level of fingerprint examiners (0.76). Algorithm A2017a performed at a level (0.85) comparable with the superrecognizers (0.83) and reviewers (0.87). The performance of A2017b (0.96) was slightly higher than the median of the facial examiners (0.93).

4.6.2.2 Performance Distributions

For the algorithms, the accuracy of A2017b was higher than the majority (73%) of participants in the face specialist groups. Conversely, 35% of examiners, 13% of reviewers, and 23% of superrecognizers were more accurate than A2017b. Compared with students, the accuracy of A2017b was equivalent to a student at the 98th percentile (z score = 2.090), A2017a was at the 91st percentile (z score = 1.346), A2016 was at the 76th percentile (z score = 0.676), and A2015 was at the 53rd percentile (z score = 0.082).

These results show a steady increase in algorithm accuracy from a level comparable with students in 2015 to a level comparable with the forensic facial examiners in 2017.

4.6.2.3 Fusing Humans and Machines

We examined the effectiveness of combining examiners, reviewers, and superrecognizers with algorithms. Human judgments were fused with each of the four algorithms as follows. For each face image pair, an algorithm returned a similarity score that is an estimate of how likely it is that the images show the same person. Because the similarity score scales differ across algorithms, we rescaled the scores to the range of human ratings. For each face pair, the human rating and scaled algorithm score were averaged, and the AUC was computed for each participantalgorithm fusion.

Fig. 4.14 shows the results of fusing humans and algorithms. The most effective fusion was the fusion of individual facial examiners with algorithm A2017b, which yielded a median AUC score of 1.0. This score was superior to the combination of two facial examiners (MannWhitney U test = $2:82 \times 10^4$, n1 =1,596, n2 =57, P=8.37×10^{-7}). Fusing individual examiners with A2017a and A2016 yielded performance equivalent to the fusion of two examiners (MannWhitney U test = 4.53×10^4, n1 = 1,596, n2 =57, P=0:956; MannWhitney U test = 4.33×10^4, n1 =1,596, n2 =57, P=0:526, respectively). Fusing one examiner with A2015 did not improve accuracy over a single examiner (MannWhitney U test=1,592, n1 =57, n2 =57, P=0:86). Fusing one examiner with A2017b proved more accurate than fusing one examiner with either A2017a or A2016 (Mann Whitney U test=1,054, n1 =57, n2 =57, P=7.92×10^{-4}; MannWhitney U test=942, n1 =57, n2

=57, P=7.28×10^{-5}, respectively). Finally, fusing one examiner with both A2017b and A2017a did not improved accuracy over fusing one examiner with A2017b (MannWhitney U test=1,414, n1 =57, n2 =57, P=0:21). This analysis was repeated for fusing algorithms and facial reviewers and for fusing algorithms and superrecognizers. Similar results were found for both groups.

4.6.3 Discussion

The results of the study point to tangible ways to maximize face identification accuracy by exploiting the strengths of humans and machines working collaboratively. First, to optimize the accuracy of face identification, the best approach is to combine human and machine expertise. Fusing the most accurate machine with individual forensic facial examiners produced decisions that were more accurate than those arrived at by any pair of human and/or machine judges. This human-machine combination yielded higher accuracy than the fusion of two individual forensic facial examiners. Computational theory indicates that fusing systems works best when their decision strategies differ [83, 69]. Therefore, the superiority of human-machine fusion over human-human fusion suggests that humans and machines have different strengths and weaknesses that can be exploited/mitigated by cross-fusion.

Second, the results indicate the potential for machines to contribute beneficially to the forensic process. Accuracy of the publicly available algorithm that we tested (A2015) was at the level of median accuracy of the studentsmodestly above chance. The other algorithms follow a rapid upward performance trajectory: from parity with a median fin-

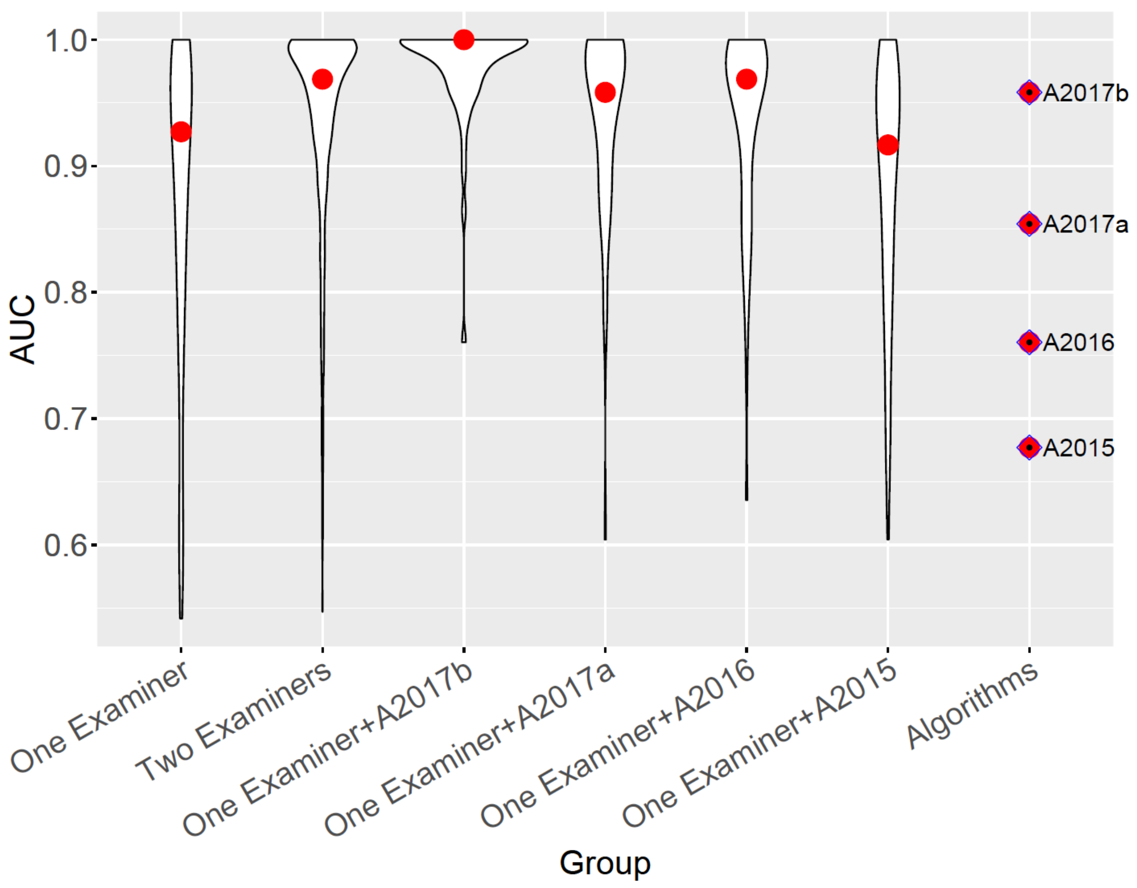


Figure 4.14: Fusion of examiners and algorithms. Violin plots show the distribution of AUCs for each fusion test. Red dots indicate median AUCs. The distribution of individual examiners and the fusion of two examiners appear in columns 1 and 2. Also, algorithm performance appears in column 7. In between, plots show the forensic facial examiners fused with each of the four algorithms. Fusing one examiner and A2017b is more accurate than fusing two examiners, fusing examiners and A2017a or A2016 is equivalent to fusing two examiners, and fusing examiners with A2015 does not improve accuracy over a single examiner.

gerprint examiner (A2016) to parity with a median superrecognizer (A2017a) and finally, to parity with median forensic facial examiners (A2017b). There is now a decade-long effort to compare the accuracy of face recognition algorithms with humans [140]. In the earliest tests [139], the face matching tasks presented relatively controlled images. As these tests progressed, algorithms and humans were compared on progressively more challenging image pairs. In this study, image pairs were selected to be extremely challenging based on both human and algorithm performance. The difficulty of these items for humans was supported by the accuracy of students, who represent a general population of untrained humans. Students performed poorly on these challenging image pairs. All four of the algorithms performed at or above median student performance. Two algorithms performed in the range of the facial specialists, and one algorithm matched the performance of forensic facial examiners.

4.7 Conclusions

In this chapter, we proposed Crystal Loss that adds a simple, yet effective, L_2-constraint to the regular softmax loss for training a face verification system. The constraint enforces the features to lie on a hypersphere of a fixed radius characterized by parameter α. We also provided bounds on the value of α for achieving a consistent performance. We also presented an overview of modern face recognition systems based on DCNNs. We discussed the major components of our end-to-end face recognition pipeline. Face detection is carried out by the proposed DPSSD face detector while keypoint localization is done using the All-in-One CNN. We also presented the details of our face verification/identification module which uses an ensemble of two networks for feature representation. We discussed training and datasets details for our system and how it relates to existing works on face recognition. We presented the results of our system for four challenging datasets, *viz.*, IJB-A, IJB-B, IJB-C, and IARPA Janus Challenge Set 5 (CS5). We show that our ensemble-based system achieves near state-of-the-art results. Our face verification accuracy for IJB-C is lower only to ArcFace [32] at FAR of 1e-6 to 1e-1. ArcFace uses a larger training dataset (5.8 million images with 85k identities) and a more sophisticated loss function which builds on Crystal Loss function. However, ArcFace does not report results for FARs of 10-7 or 10-8. Our pipeline produces reasonable numbers even at these extreme FARs. In conclusion, Crystal loss is a valuable replacement for the existing softmax loss, for the task of face recognition. In the future, we would further explore the possibility of exploiting the geometric structure of the feature encoding using manifold-based metric learning.

Additionally, we performed the most comprehensive examination to date of face identification performance across groups of humans with variable levels of training, experience, talent, and motivation. We compared the accuracy of state-of-the-art face recognition algorithms with humans and show the benefits of a collaborative effort that combines the judgments of humans and machines. The work draws on previous cornerstone findings on human expertise and talent with faces, strategies for fusing human judgments, and computational advances in face recognition. The study provides an evidence-based roadmap for achieving highly accurate face identification. These methods should be extended in future work to test humans and machines on a wider range of face recognition tasks, including recognition across viewpoint and with low-quality images and video as well as recognition of faces from diverse demographic categories.

Chapter 5: Compact Convolutional Networks for Adversarial Defense

5.1 Introduction

Adversarial attacks have emerged as a potential threat to CNN-based systems. Adversarial images can be used by a suspect to fool a face verification system, by letting the person go unidentified. These attacks can also cause self-driving cars to mis-classify scene objects such as a stop sign leading to adverse effects when these systems are deployed in real time. As networks move from the research labs to the field, they need to be designed in a way that they are not only accurate, but also robust to adversarial perturbations. Several recent works have been proposed to improve robustness, such as adversarial training [53, 155], gradient masking [174], etc. In this work, we impose constraints on the sensitivity of the learned feature space and propose a modified convolution operation that can desensitize the learned mapping in the direction of adversarial perturbations.

It has been hypothesized that CNNs learn a highly non-linear manifold on which the images of same class lie together, while images from different class are separated. Hence, the original image and the adversarial image lying close to each other in Euclidean space, are far separated on the manifold or in feature space. When designing a robust classifier, we would like to address the following question: *Can we bring the original and perturbed images closer in the feature space of a learned mapping to improve its robustness?* To

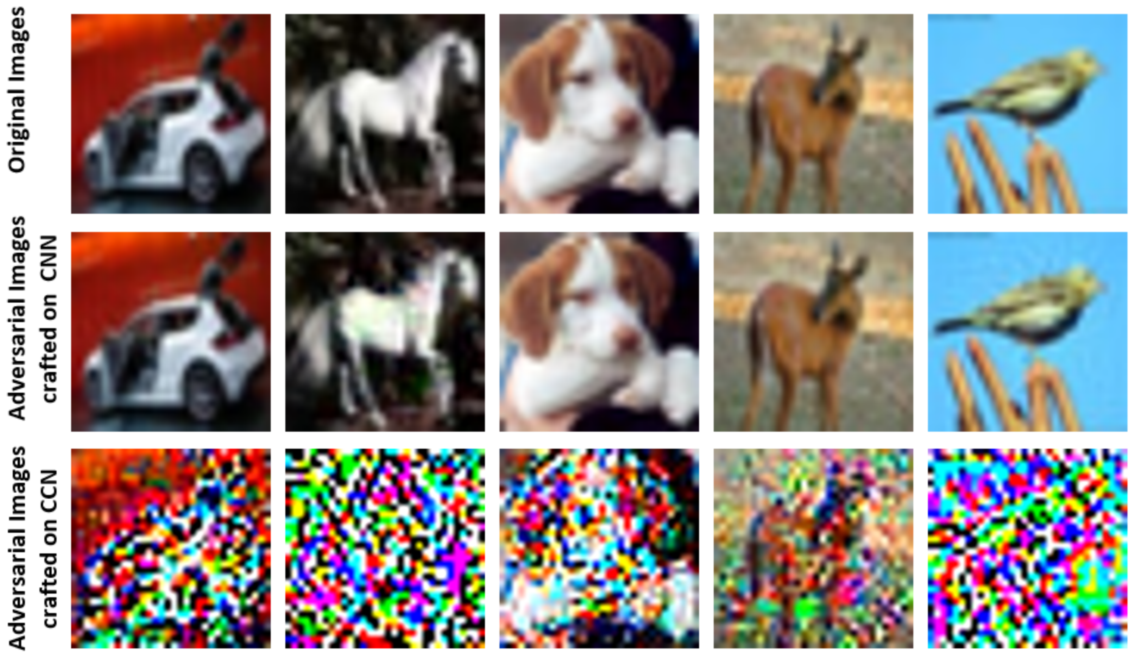

Original Images
Adversarial Images crafted on CNN
Adversarial Images crafted on CCN

Figure 5.1: Five test samples from CIFAR10 [89] dataset (top) that are correctly classified by both CNN and CCN. Corresponding adversarial images crafted using DeepFool [123] attack and misclassified by CNN (middle) and CCN (bottom). The adversarial attacks on CCN can be detected trivially by a human observer.

address this question, we employ the property of *compactness* in the context of feature learning that would enhance a network's robustness to adversarial attacks. *Compactness* enforces the features to be bounded and lie in a closed space. It reduces the degree of freedom for the features to be learned. This restricts the extent to which a feature for perturbed image can move, making it less likely to cross the class boundary.

To enforce *compactness* in the feature space, we explore the L_2-Softmax Loss proposed in [142]. The L_2-Softmax Loss establishes *compactness* by constraining the features to lie on a hypersphere of fixed radius, before applying the softmax loss. It brings the intra-class features close to each other and separates the inter-class features far apart.

In this way, features from the original and the adversarial image are closer to each other using L_2-Softmax Loss, compared to training with regular softmax loss (see Fig. 5.2).

Using these insights, we propose a novel convolution method, called *compact convolution*, that significantly enhances a network's robustness by ensuring compact feature learning at every layer of the network. A compact convolution module applies the L_2-normalization step and scaling operations to every input patch before applying the convolutional kernel in a sliding window fashion. Compact Convolutional Networks (CCNs), built using these modules, are highly robust compared to a typical CNN. Fig. 5.1 shows some sample images and corresponding adversarial attacks generated using DeepFool [123] to fool a CNN and a CCN. The adversarial samples for CCN can easily be distinguished from the original samples by a human observer. The figure shows that CCNs are robust to small adversarial perturbations such that to fool a CCN the magnitude of perturbations required is much higher, which completely distorts the image and can be detected easily.

5.2 Related Works

A lot of research has gone into generating adversarial perturbations to fool a deep network. Szegedy et al. [172] first showed the existence of adversarial perturbations in CNNs and proposed a L-BFGS based optimization scheme to generate the same. Later, Goodfellow et al. [53] proposed Fast Gradient Sign Method (FGSM) to generate adversarial samples. DeepFool [123] attack iteratively finds a minimal perturbation required to cause a network to mis-classify. Other recently proposed adversarial attacks include

Jacobian-based Saliency Map Approach (JSMA) [133], Carlini-Wagner (CW) attack [12], Universal Perturbations [122], etc.

To safeguard the network from adversarial attacks, researchers have focused on two approaches: 1) *Adversarial Detection*, and 2) *Adversarial Defense*. Methods based on adversarial detection [112, 119, 54, 44] attempt to detect an adversarial sample before passing it through the network for inference. These methods put an extra effort in designing a separate adversarial detector which itself has the risk of being fooled by the attacker. Recently, Carlini and Wagner [11] showed that most of the adversarial detectors are ineffective and can be fooled.

Methods based on adversarial defense aim at improving the network's robustness to classify adversarial samples correctly. One way to achieve robustness is by simultaneously training the network with clean and adversarial samples [53, 121, 161, 92]. These methods are stable to the attack on which they are trained, but ineffective against a different attack. Preprocessing the input to nullify the adversarial effect is another way to defend the network [35, 56]. Few methods have focused on modifying the network topology or optimization procedure for adversarial defense. Gu and Rigazio [55] proposed Deep Contractive Network that adds a smoothness penalty on the partial derivatives at every layer. Cisse et al. [26] proposed Parseval Networks that improves robustness by enforcing the weight matrices of convolutional and linear layers to be Parseval tight frames. Papernot et al. [134] showed that knowledge distillation with high temperature parameter can be used as defense against adversarial samples. Warde et al. [181] showed that a similar robustness as defensive distillation can be obtained by training the network with smooth labels. Zantedeschi et al. [211] used Bounded ReLU activations to enhance network's robustness to adversarial perturbations.

While these methods have focused on improving defense to adversarial attacks in general, most of them have focused on white box attacks and incur additional computational overhead during training. In this work, we propose an approach to achieve adversarial defense in CNNs using compact convolutions which can be seamlessly integrated into any existing deep network architecture. We further demonstrate its effectiveness against both white box and black box adversarial attacks.

5.3 Compact Learning for Adversarial Defense

5.3.1 Compactness

Compactness is a property associated with a subset of Euclidean space which is closed and bounded. A space is closed when it contains all its limiting points. A space is bounded when all its points lie within a fixed distance of each other. Euclidean space in itself is not *compact*, since it is not bounded.

The features obtained from a typical CNN are not *compact*, since the softmax loss does not constrain them to lie in a closed or bounded space. It distributes the features in the Euclidean space such that the overall training loss is minimized. In a way, it overfits to the training domain. Thus an adversarially perturbed image, although close to the original image in input space, can lie very far away in the Euclidean feature space. On the other hand, features learned on a compact space have restricted degrees of freedom which does not allow the adversarial features to move far away from the original features. A given perturbed image would lie farther from the original image in the Euclidean space compared to a compact space. Thus compact feature learning helps in improving the robustness of the network.

One way to enforce compactness on the CNN features is to restrict them to lie on a compact space during training. A simple example of a compact space is a hypersphere manifold, which is both closed and bounded. L_2-Softmax Loss [142] (discussed in Section 5.3.2), performs compact feature learning by constraining the features to lie on a hypersphere of a fixed radius.

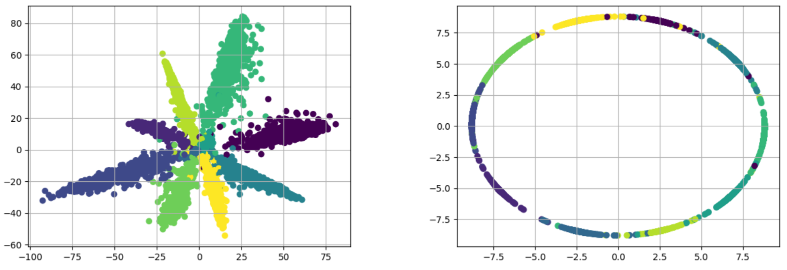
5.3.2 L_2-Softmax Loss

L_2-Softmax loss was recently proposed in [142] for improving the task of face verification. The loss imposes a constraint on the deep features to lie on a hypersphere of a fixed radius, before applying the softmax penalty. The loss is defined as:

$$L_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T(\frac{\alpha \mathbf{x}_i}{\|\mathbf{x}_i\|_2})+b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T(\frac{\alpha \mathbf{x}_i}{\|\mathbf{x}_i\|_2})+b_j}}, \tag{5.1}$$

where $\mathbf{x}_i$ is the i^{th} deep feature for the class label y_i, W_j is the weight and b_j is the bias corresponding to the class j, α is a positive scalar parameter, and m, n are the batch-size and number of classes respectively. The features are first normalized to unit length and then scaled by α before passing it through the softmax classifier. Constraining the features to lie on a hypersphere reduces the intra-class variations and enhances the inter-class separability.

From Fig. 5.2(a), we can see that the features trained using softmax loss do not satisfy the *compactness* property since the feature space is not closed or bounded. The



(a) (c)

Figure 5.2: Feature vizualization from the last layer (dimension=2) of CNN trained on MNIST [95] with (a) Softmax Loss, (b) L_2-Softmax Loss [142].

L_2-Softmax Loss constrains the features to lie on a hypersphere manifold which is a closed space. Also, the L_2-norm of the feature vectors is always constant and equal to α, which makes the feature space bounded. Hence, the L_2-Softmax Loss obeys the *compactness* property (Fig. 5.2(b)). Experimental analysis (see section 5.5) shows that L_2-Softmax Loss is more robust to adversarial attacks than softmax loss. To the best of our knowledge, L_2-Softmax loss has not been used for adversarial defense before.

5.3.3 Sub-linearity of Adversarial Examples with Compact Learning

In this section, we show that compact feature learning restricts the change in activation due to perturbation η to grow sub-linearly with the input dimension. Consider a linear model with weight vector w and input vector x with dimension n. Let the adversarial example be represented as $\tilde{x} = x + \eta$. The activation is given by the dot product:

$$w^T \tilde{x} = w^T x + w^T \eta \tag{5.2}$$

It was shown by Goodfellow et al. [53] that the change in activation $w^T\eta$ will grow with εmn, where m is the average magnitude of the weight vector, and $\|\eta\|_\infty < \varepsilon$. Thus, the effect of adversarial perturbation increases linearly with the input dimension. Now, with compact feature learning using L_2-Softmax loss with $\alpha = 1$, (5.2) can be rewritten as:

$$w^T \frac{\tilde{x}}{\|\tilde{x}\|_2} = w^T \frac{x}{\|x+\eta\|_2} + w^T \frac{\eta}{\|x+\eta\|_2}. \tag{5.3}$$

The second term in (5.3) is the change in activation due to adversarial perturbation. The denominator in this term, which is the L_2-norm of the perturbed signal ($\|x+\eta\|_2$), changes proportional to the square root of the dimension of x. This can be intuitively thought of as follows. Let the average absolute activation of x be p. Then the L_2-norm of the signal x can be well approximated by $p\sqrt{n}$. While the scaling factor (p) depends on the actual variations in the feature value, the dependency on the dimension is clear. Applying this insight to (5.3), we see that the change in the activation now grows at the rate of $\sqrt{n}$ instead of n, due to the normalizing factor in the denominator. Thus, for a given input dimension, the change in activation due to adversarial perturbation would be smaller for compact features compared to the typical CNN features, which makes it ideal for training a robust network.

5.4 Compact Convolution

The L_2-Softmax loss [142] enforces *compactness* only to deep features from the last layer of CNN. It was motivated by efficient representation of normalized feature at the output space, whereas in this paper we want to reduce the sensitivity of the activations at each layer to spurious perturbations. Hence, we propose to extend the *compactness* property to features from the intermediate layers of CNNs as well. A typical CNN is a hierarchy of convolutional and fully-connected layers stacked with non-linear activation functions after every layer. A discrete convolution is a linear function applied on a patch of a signal in a sliding window fashion. Let W be the convolution kernel of size $2k+1$, $\mathbf{x}_{n,k}$ be an input patch defined as:

$$\mathbf{x}_{n,k} = [x(n-k), x(n-k+1), ..., x(n+k)], \tag{5.4}$$

where $x(n)$ is the n^{th} element of input vector $\mathbf{x}$. The convolution operation is represented as:

$$y(n) = W^T \mathbf{x}_{n,k}, \tag{5.5}$$

where $y(n)$ is the n^{th} element of the output vector $\mathbf{y}$. To enforce *compactness* in convolutional layers, we need to ensure that every input patch at a given location is first L_2-normalized and scaled before multiplying with the convolutional kernel W. Formally, we want the convolution output ($\tilde{y}(n)$) at position n to be:

$$\tilde{y}(n) = W^T \frac{(\alpha \mathbf{x}_{n,k})}{||\mathbf{x}_{n,k}||_2 + \delta}, \tag{5.6}$$

where δ is a small constant added to avoid division by zero. We call this new method of patch-normalized convolution as *compact convolution*. A toy example depicting the difference between typical convolution and compact convolution is shown in Fig. 5.3.

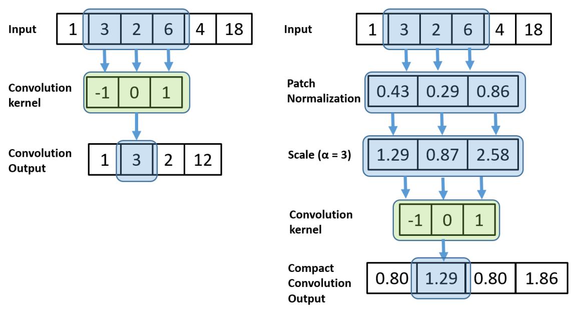


Figure 5.3: A toy example for convolution (left) and compact convolution (right).

In a deep CNN, the convolution kernel is typically applied to high dimensional feature maps. Normalizing every feature patch before multiplying with the convolutional kernel is computationally expensive and redundant, since the patches are overlapping. To implement compact convolution efficiently in a deep network, we propose a compact convolution module (shown in Fig. 5.4). We split the input feature map into two branches. The first branch carries out the traditional convolution operation with parameters of size $k \times k$, without bias addition. The second branch first computes the sum of squares along the channel dimension of the input. Subsequently, it is convolved with a $k \times k$ kernel containing fixed value of all ones. This step provides the squared L_2-Norm of sliding-window patches for every output location in a feature map. We perform element-wise square-root on top of it and add a small constant $\delta = 0.01$. Lastly, each channel of the convolutional output from the first branch is divided element-wise with the output from the second branch. We then scale the final output with a learnable scalar parameter α and add the bias term. The compact convolution module uses just one extra learnable parameter (α) compared to the traditional convolutional layer.

Since the linear operation in fully-connected layers is a special case of convolution,

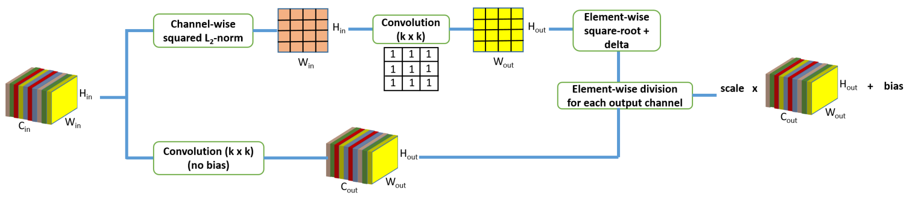

Channel-wise squared L2-norm
Hin
Win
Convolution (k x k)
1 1 1
1 1 1
1 1 1
Hout
Wout
Element-wise square-root + delta
Element-wise division for each output channel
scale x
Hout + bias
Cout
Wout
Cin
Convolution (k x k) (no bias)

Figure 5.4: Block diagram of a compact convolution module.

the compact convolution operation for these layers results in L_2-normalization and scaling of the feature vectors. It constrains the features to lie on a hypersphere of fixed radius (α), before applying the dot product with the layer parameters. We call the deep networks with compact convolution modules as Compact Convolutional Networks (CCNs). They follow the *compactness* property at every layer of the network, which greatly enhances their robustness against multiple kinds of adversarial attacks. The L_2-Softmax Loss [142] inherently gets applied in CCNs.

5.5 Experiments

We evaluate the effectiveness of the proposed defense methods on MNIST [95], CIFAR10 [89] and ImageNet [152] datasets. The MNIST [95] dataset contains $60,000$ training and $10,000$ test images of handwritten digits. The images are 28×28 dimensional with values in $[0,1]$. CIFAR10 [89] dataset contains $50,000$ training and $10,000$ test images of 10 classes. The images are $32 \times 32 \times 3$ dimensional, with values scaled between 0 and 1.

We use two well-known methods for crafting adversarial attacks: Fast Gradient Sign Method [53] (FGSM) and DeepFool [123]. The FGSM attack adds the sign of the gradient to the input, scaled by the factor ε as shown in (5.7)

$$\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon\, \mathrm{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)), \tag{5.7}$$

where $\mathbf{x}$ is the input image, $\tilde{\mathbf{x}}$ is the crafted adversarial image, $\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$ is the gradient of the cost function with respect to the input. This method is very fast, since it uses a single backward step to generate the attack. On the other hand, DeepFool [123] iteratively finds the minimal perturbation required to mis-classify the input in the direction of the nearest class boundary. Though slower than FGSM [53], DeepFool [123] can generate adversarial images with smaller magnitude of perturbations, which are indistinguishable to human observer. We use the Foolbox [147] library to generate these attacks.

Table 5.1 provides the network architectures used for training. For training on MNIST [95], we use the architecture proposed by Papernot et al. [134]. The learning rate is set to 0.1 for the first thirty epochs, and decreased by a factor of 0.1 after every ten epochs. We train the network for fifty epochs. For training on CIFAR10 [89], we use the standard VGG11 [163] network. The convolutional layers use 3×3 kernels with padding of one. We start with a learning rate of 0.1 which is decreased by a factor of 0.2 after 60, 120 and 160 epochs. We train the network for 200 epochs. We use SGD with momentum (0.9) and weight decay (5×10^{-4}) for all our training. We use mean subtraction of 0.5 as a pre-processing step.

We compare and evaluate the following defense methods against adversarial attacks:

• **SM** - The baseline model trained using the typical Softmax loss function.

Table 5.1: Overview of network architectures for MNIST [95] and CIFAR10 [89] datasets

Layer	MNIST	CIFAR10
Conv+ReLU	32 filters (3x3)	64 filters (3x3)
Conv+ReLU	32 filters (3x3)	128 filters (3x3)
MaxPool	2x2	2x2
Conv+ReLU	64 filters (3x3)	256 filters (3x3)
Conv+ReLU	64 filters (3x3)	256 filters (3x3)
MaxPool	2x2	2x2
Conv+ReLU	-	512 filters (3x3)
Conv+ReLU	-	512 filters (3x3)
MaxPool	-	2x2
Conv+ReLU	-	512 filters (3x3)
Conv+ReLU	-	512 filters (3x3)
MaxPool	-	2x2
FC+ReLU	200 units	512 units
FC+ReLU	200 units	512 units
Softmax	10 units	10 units

• **LS** (*Label Smoothing*) - The model trained using Softmax Loss with soft labels $\{\frac{1}{90}, 0.9\}$ [181] instead of discrete labels $\{0,1\}$

• **BReLU** (*Bounded ReLU*) - The model trained using bounded ReLU [211] activation function instead of ReLU. The activations are clipped between $[0,1]$.

• **L2SM** (L_2-*Softmax*) - The model trained using L_2-Softmax Loss [142] as discussed in Section 5.3.2.

• **CCN** (*Compact Convolutional Network*) - The model trained with compact convolution modules instead of traditional convolutional and fully-connected layers, as discussed in Section 5.4.

• **CCN+LS** (*Compact Convolutional Network* with *Label Smoothing*) - A CCN trained using soft labels $\{\frac{1}{90}, 0.9\}$

The experiments are organized as follows. Section 5.5.1 evaluates the proposed models on MNIST [95], CIFAR10 [89] and ImageNet [152] datasets, against FGSM and DeepFool attacks in a white-box setting. It also analyzes the effect of adversarial training on different models. Section 5.5.2 evaluates the robustness in a black-box setting, and discusses the transferability of various attacks. Section 5.5.3 compares the robustness using other feature normalization methods such as Local Response Normalization [88] and batch-normalization [72].

5.5.1 White-Box Attacks

In a white-box attack, the attacker has full access to the network to be attacked. For each of the defense methods, we generate FGSM [53] and DeepFool [123] attack for MNIST [95] and CIFAR10 [89] testset.

Table 5.2 provides the classification accuracy of various defense methods on adversarial examples crafted using FGSM [53] for MNIST [95] and CIFAR10 [89] testset. We perform evaluations for four different ε values $\{0.1, 0.2, 0.3, 0.4\}$. Higher values of ε lead to larger perturbation, thus decreasing accuracy. Note that $\varepsilon = 0$ corresponds to the clean test samples without any adversarial perturbation. From the table, we find that both *CCN* and *CCN+LS* are highly robust to FGSM attack, with minimal degradation in accuracy. Specifically, for $\varepsilon = 0.3$, *CCN* achieves an accuracy of 81.38% on MNIST [95] which is more than $2\times$ factor improvement over the baseline model with accuracy 31.76%. Label Smoothing along with CCN (*CCN+LS*) further enhances the robustness to achieve an accuracy of 89.73%. The *L2SM* model shows significant improvement over the baseline, which establishes its robustness. A similar trend is observed with FGSM [53] attack on CIFAR10 [89] testset. Since CIFAR10 is a harder dataset, we use the ε values of $\{\frac{2}{255}, \frac{4}{255}, \frac{8}{255}, \frac{16}{255}\}$ to craft the FGSM attack. For $\varepsilon = \frac{8}{255}$, we observe $5\times$ improvement using *CCN* and *CCN+LS*, compared to the baseline model.

Table 5.3 provides the classification accuracies of different defense methods against DeepFool [123] attack, on MNIST and CIFAR10 datasets. Since, DeepFool is an iterative attack, it will mostly find a perturbation to fool the network. To evaluate using DeepFool, the iterations are carried out until the adversarial perturbation (η) causes the network to

Table 5.2: Accuracy (%) on MNIST [95] and CIFAR10 [89] for FGSM [53] attack. The best accuracy is shown in bold and the second-best accuracy is underlined

	MNIST					CIFAR10				
Method	ε=0	ε=0.1	ε=0.2	ε=0.3	ε=0.4	ε=0	ε=2	ε=4	ε=8	ε=16
SM	99.50	92.61	63.71	31.76	19.39	<u>91.14</u>	60.82	34.62	14.30	8.57
LS [181]	<u>99.53</u>	95.62	85.13	44.25	15.57	91.03	66.7	58.4	54.07	51.05
BReLU [211]	**99.54**	95.29	74.83	42.51	19.41	90.87	61.49	35.02	18.01	13.73
L2SM	99.48	94.51	79.68	60.32	45.16	**91.37**	70.58	65.39	63.71	62.59
CCN	99.50	<u>96.99</u>	<u>91.64</u>	<u>81.38</u>	<u>60.65</u>	90.54	<u>73.72</u>	<u>71.47</u>	<u>69.65</u>	<u>66.23</u>
CCN+LS	99.43	**97.26**	**93.68**	**89.73**	**75.95**	90.51	**80.93**	**79.2**	**76.8**	**71.38**

mis-classify, or the ratio of L_2-norm of the perturbation (η) and the input ($\mathbf{x}$) reaches the *max-residue-ratio* (d), as given in (5.8).

$$\frac{\|\eta\|_2}{\|\mathbf{x}\|_2} \leq d \tag{5.8}$$

The evaluations are carried out with different values of $d = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. From Table 5.3, we find that *LS* and *CCN* perform comparably against DeepFool attack on MNIST. The model *CCN+LS* achieves the best performance since it leverages the qualities of both *LS* and *CCN* models. On CIFAR10 dataset, *CCN* performs better than *LS* against DeepFool attack. The effect of label smoothing is less for CIFAR10, since the probability scores are lower owing to the difficulty of the dataset. The models *CCN* and *CCN+LS* significantly outperform other defense methods against DeepFool attack.

We also provide a quantitative measure of model's robustness against DeepFool [123]

Table 5.3: Accuracy (%) for DeepFool [123] attack (with different max-residue-ratio d) on MNIST [95] and CIFAR10 [89] testset. The best accuracy is shown in bold and the second-best accuracy is underlined

	MNIST					CIFAR10				
Method	d=0.1	d=0.2	d=0.3	d=0.4	d=0.5	d=0.1	d=0.2	d=0.3	d=0.4	d=0.5
SM	93.24	63.22	18.73	1.89	0.53	5.73	5.72	5.72	5.72	5.72
LS [181]	96.85	<u>92.2</u>	<u>88.12</u>	<u>82.05</u>	<u>70.08</u>	57.33	43.93	31.77	21.89	14.6
BReLU [211]	96.15	84.33	70.13	57.25	45.5	22.44	15.49	10.48	7.29	6.37
L2SM	96.5	88.22	75.94	59.47	40.31	66.63	56.93	47.89	38.52	28.85
CCN	<u>97.06</u>	92.18	83.46	69.25	48.54	<u>73.17</u>	<u>71.78</u>	<u>70.24</u>	<u>68.13</u>	<u>64.02</u>
CCN+LS	**97.76**	**95.42**	**92.43**	**88.65**	**84.01**	**80.8**	**79.38**	**78.08**	**76.43**	**74.22**

attack in Table 5.4. The robustness $\rho_{adv}(\hat{k})$ is computed using (5.9).

$$\rho_{adv}(\hat{k}) = \mathbb{E}_{\mathbf{x}} \frac{\|\mathbf{r}_{\mathbf{x},\hat{k}}\|_2}{\|\mathbf{x}\|_2}, \tag{5.9}$$

where $\mathbf{r}_{\mathbf{x},\hat{k}}$ is the generated perturbation, $\mathbf{x}$ is the input image, and $\mathbb{E}_{\mathbf{x}}$ is the expectation over the entire test data. Similar to the classification accuracy, the robustness of *LS* is higher than *CCN* for MNIST but lower for CIFAR10. *CCN+LS* is the most robust model for both the datasets.

5.5.1.1 Adversarial Training

Goodfellow et al. [53] proposed to train the network simultaneously with original and crafted adversarial images to improve it's stability. However, adversarial training is sensitive to the value of ϵ that is used to train the network on. We train the models on real

Table 5.4: Robustness of defense methods against DeepFool [123] attack for MNIST [95] (left) and CIFAR10 [89] (right). The best robustness is shown in bold and the second-best robustness is underlined

Method	Robustness
SM	0.225
LS [181]	<u>0.571</u>
BReLU [211]	0.493
L2SM	0.441
CCN	0.479
CCN+LS	**0.827**

Method	Robustness
SM	0.014
LS [181]	0.185
BReLU [211]	0.057
L2SM	0.285
CCN	<u>0.549</u>
CCN+LS	**0.625**

and adversarial images crafted from FGSM attack with $\varepsilon = 0.3$ for MNIST, and $\varepsilon = \frac{8}{255}$ for CIFAR10 dataset. The classification accuracies of adversarially trained models are reported in Table 5.5 for MNIST and CIFAR10 datasets. The results show that adversarial training improves the robustness for all the models. The performances of the models on MNIST dataset are comparable to each other with accuracy values above 95%. CIFAR10 results show better distinct between performance of various models. Most of the the models perform well for the ε value with which they were trained, but the performance degrades on other ε values. The models *CCN* and *CCN+LS* are are less sensitive to the training perturbation amount, and consistently stable across all the ε values.

Table 5.5: Accuracy (%) with adversarial training against FGSM [53] attack on MNIST [95] and CIFAR10 [89] testset. The best accuracy is shown in bold and the second-best accuracy is underlined

	MNIST					CIFAR10				
Method	ε=0	ε=0.1	ε=0.2	ε=0.3	ε=0.4	ε=0	ε=2	ε=4	ε=8	ε=16
SM	**99.49**	96.51	98.67	99.15	96.06	83.8	71.56	67.18	71.46	43.09
LS [181]	99.45	98.4	98.77	99.31	**98.79**	83.28	**77.68**	69.07	52.13	34.04
BReLU [211]	99.43	96.79	**98.93**	**99.42**	97.12	85.37	72.93	67.08	**80.73**	31.01
L2SM	99.47	96.88	98.7	99.32	84.03	84.8	73.08	68.31	74.95	49.64
CCN	99.44	98.18	98.71	99.09	96.9	87.08	73.27	75.8	76.78	61.97
CCN+LS	99.48	**98.65**	98.8	98.93	94.13	**87.17**	76.68	**77.53**	77.66	**70.17**

5.5.1.2 Evaluation on ImageNet dataset

We also analyze the adversarial robustness of the proposed methods on ILSVRC [152] object classification dataset. The dataset contains $1.2M$ training images and $50k$ validation images with 1000 class labels. The dataset is more challenging than MNIST and CIFAR10 as it contains large images in real-world settings. We use off-the-shelf VGG11 [163] network for training the models. Table 5.6 provides the classification accuracy of various defense methods on adversarial examples crafted using FGSM [53] and DeepFool [123] attacks. We use the ε values of $\{0.1, 0.3, 0.5, 1.0\}$ normalized by 255 to craft the FGSM attack. From the table, we find that *LS* and *CCN+LS* are more robust to FGSM attack. On DeepFool [123] attack, *CCN+LS* is the most robust model followed by *LS*.

Additionally, we see a significant improvement in top-1 accuracy(%) by using compact learning framework. *L2SM* achieves a top-1 accuracy of 68.78% while *CCN* achieves the accuracy of 68%, which is 8% gain over the baseline *SM* model. Having large number of classes improves the discriminative capacity of compact learning leading to improved accuracy. We intend to analyze the effect of *compactness* for large number of classes in future work.

Table 5.6: Accuracy (%) on ILSVRC [152] validation set for FGSM [53] attack (left). Robustness of defense methods against DeepFool [123] attack (right). The best accuracy is shown in bold and the second-best accuracy is underlined

Method	$\varepsilon = 0$	$\varepsilon = 0.1$	$\varepsilon = 0.3$	$\varepsilon = 0.5$	$\varepsilon = 1.0$
SM	60.52	47.19	25.72	17.02	10.14
LS	63.80	**58.84**	**43.17**	**32.55**	<u>20.21</u>
BReLU	67.08	53.05	27.35	17.50	10.75
L2SM	**68.78**	<u>56.66</u>	33.74	22.95	13.92
CCN	<u>68.00</u>	51.43	28.57	19.54	12.68
CCN+LS	66.64	53.22	<u>36.63</u>	<u>29.90</u>	**23.53**

Method	Robustness (in %)
SM	0.1325
LS	<u>0.2951</u>
BReLU	0.1463
L2SM	0.1911
CCN	0.1439
CCN+LS	**0.4027**

5.5.2 Black-Box Attacks

In a typical black-box attack, the attacker has no information about the network architecture, its parameters or the training dataset. The attacker can query the network and can get the output class label for a given input. We use the black-box attack proposed by Papernot et al. [132] to evaluate the proposed models. The model on which the attack has to be applied is the defense model, and the model learned to replicate the behavior of defense model is the substitute model. We treat our defense models as oracle, and train substitute models using LeNet [94] architecture as described in [132]. Table 5.7 reports the accuracy of the defense models against FGSM attack generated using the corresponding substitute models. We observe that *CCN* and *CCN+LS* models are most robust to practical black-box attacks, and are consistent across different ε values. Hence, the results show that the proposed models are robust to both white box as well as black box attacks.

Table 5.7: Accuracy (%) against practical black-box FGSM attack (with different ε) on MNIST [95] testset. The best accuracy is shown in bold and the second-best accuracy is underlined

Method	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$
SM	98.49	87.44	55.41	32.84
LS [181]	98.89	91.12	52.53	29.28
BReLU [211]	98.90	91.96	66.45	**39.44**
L2SM	98.99	93.37	56.67	26.63
CCN	<u>99.15</u>	<u>96.76</u>	<u>77.27</u>	28.19
CCN+LS	**99.24**	**97.33**	**84.14**	<u>32.94</u>

5.5.2.1 Transferability of Adversarial Samples

It has been shown in [132] that adversarial examples generated by one type of network can be used to fool a different type of network. This makes it easier for the attackers to generate adversarial samples using their independently trained models. Our defense model should be immune to the attacks generated by itself, as well to attacks generated from a different network.

Tables 5.8 and 5.9 report the accuracies on transfered attacks between different defense models. (*) in the tables indicates that the adversarial attacks were crafted and tested on the same network, causing maximum impact. We find that the networks are more vulnerable to the transfered attacks generated using the baseline model *SM*. Among all models, *CCN* and *CCN+LS* are most robust to transferred attacks. Also, the attacks generated using these models are less likely to fool other models. This shows that *CCN* and *CCN+LS* provide a two-way defense. Firstly, these models would be less vulnerable to any unknown adversarial attacks. Secondly, the attacks generated using these models would be less harmful for any unknown network.

Table 5.8: Accuracy (%) on MNIST [95] against transfer attacks crafted using FGSM ($\varepsilon = 0.3$). The best accuracy is shown in bold and the second-best accuracy is underlined

	Attack crafted on					
Attack tested on	SM	LS [181]	BReLU [211]	L2SM	CCN	CCN+LS
SM	31.76*	61.41	54.22	77.88	<u>91.22</u>	<u>93.84</u>
LS [181]	49.48	44.25*	53.63	80.47	**91.76**	**94.69**
BReLU [211]	52.62	66.43	42.51*	79.69	90.82	93.77
L2SM	52.01	64.93	58.91	60.32*	90.65	93.59
CCN	<u>64.15</u>	<u>75.7</u>	<u>66.55</u>	<u>82.7</u>	81.38*	93.53
CCN+LS	**70.18**	**80.83**	**71.49**	**85.29**	90.49	89.73*

Table 5.9: Accuracy (%) on CIFAR10 [89] against transfer attacks crafted using FGSM ($\varepsilon = \frac{8}{255}$).

The best accuracy is shown in bold and the second-best accuracy is underlined

	Attack crafted on					
Attack tested on	SM	LS [181]	BReLU [211]	L2SM	CCN	CCN+LS
SM	14.30*	59.14	33.63	67.13	77.77	82.91
LS [181]	31.16	54.07*	33.28	67.26	<u>77.84</u>	**83.11**
BReLU [211]	33.63	59.65	18.01*	67.96	**77.85**	<u>82.95</u>
L2SM	32.59	59.95	35.01	63.71*	77.76	82.78
CCN	<u>41.33</u>	**63.6**	**42.37**	<u>69.63</u>	69.65*	80.53
CCN+LS	**41.48**	<u>63.48</u>	<u>42.29</u>	**69.98**	73.74	76.8*

5.5.3 Effect of Feature Normalization Methods

In this experiment, we analyze the effect of Local Response Normalization (LRN) [88] and batch-normalization [72] on the robustness of the network and compare it to the proposed approach. Although both these methods normalizes the features based on local activations or input batch statistics, they do not ensure *compactness*. LRN layer implements a form of lateral inhibition given by (5.10)

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}, \tag{5.10}$$

where $a^i_{x,y}$ is the input activation at location (x,y) and channel i, $b^i_{x,y}$ is the corresponding output activation. The values of constants k, n, α and β are are set as provided in [88]. On the other hand, batch-normalization [72] reduces the *internal covariate shift*, by normalizing the features across the input batch. We compare the performance of the models trained with *LRN* and batch-normalization (*BN*) with the baseline model *SM* in Table 5.10. We find from our experiments on MNIST and CIFAR-10 datasets that LRN or batch-normalization either improves marginally or weakens the robustness of the network. Our proposed models *CCN* and *CCN+LS* significantly outperforms *LRN* and *BN* models for both MNIST [95] and CIFAR10 [89] datasets.

Table 5.10: Accuracy (%) on MNIST [95] and CIFAR10 [89] for FGSM [53] attack. The best accuracy is shown in bold and the second-best accuracy is underlined

	MNIST					CIFAR10				
Method	ε=0	ε=0.1	ε=0.2	ε=0.3	ε=0.4	ε=0	ε=2	ε=4	ε=8	ε=16
SM	99.50	92.61	63.71	31.76	19.39	91.14	60.82	34.62	14.30	8.57
BN [72]	**99.58**	88.48	30.72	11.20	9.24	**92.34**	59.47	40.63	26.7	16.58
LRN [88]	99.28	88.59	52.52	25.61	13.21	<u>91.33</u>	60.68	34.82	14.99	9.08
CCN	<u>99.50</u>	<u>96.99</u>	<u>91.64</u>	<u>81.38</u>	<u>60.65</u>	90.54	<u>73.72</u>	<u>71.47</u>	<u>69.65</u>	<u>66.23</u>
CCN+LS	99.43	**97.26**	**93.68**	**89.73**	**75.95**	90.51	**80.93**	**79.2**	**76.8**	**71.38**

5.5.4 C&W Attack

We evaluate different network defense methods against the attack proposed by Carlini and Wagner [12] (C&W Attack). L_2-distance metric of the adversarial perturbation is optimized to generate the attack, as given by:

$$\mathit{minimize}\ \|\frac{1}{2}(\tanh(\mathbf{w})+1)-\mathbf{x}\|_2^2 + c\ f(\frac{1}{2}(\tanh(\mathbf{w})+1)), \qquad (5.11)$$

where $\mathbf{x}$ is the input image, and $\mathbf{w}$ is the variable to be optimized. For the target class t, the function f corresponding to input $\tilde{\mathbf{x}}$ is given by:

$$f(\tilde{\mathbf{x}}) = \max(\max\{Z(\tilde{\mathbf{x}})_i : i \neq t\} - Z(\tilde{\mathbf{x}})_t, -\kappa), \tag{5.12}$$

where $Z(\tilde{\mathbf{x}})_i$ is the logit value for the i^{th} class, and κ is the confidence with which the misclassification occurs. In our experiments, we set as the target class t to the class with second-highest classification score. The confidence κ is set to zero. The maximum number of iterations is set to 1000, and the search step is limited to 6. We observe that C&W attack is approximately 100$\times$ slower than FGSM attack [53].

Table 5.11: Mean L_2-distance between the input and perturbed image on MNIST dataset, along with the success probability of generating the C&W attack. The best result is shown in bold and the second-best result is underlined

Method	Mean Distance	Success Prob.
SM	1.393	100
LS [181]	<u>1.626</u>	100
BReLU [211]	1.473	100
L2SM	1.453	100
CCN	1.449	100
CCN+LS	**2.079**	**96.54**

Tables 5.11 and 5.12 show the performance of difference defense methods against C&W attack [12], on MNIST and CIFAR10 datasets respectively. We report the mean L_2-distance between the original and the adversarial samples, along with the success

Table 5.12: Mean L_2-distance between the input and perturbed image on CIFAR10 dataset, along with the success probability of generating the C&W attack. The best result is shown in bold and the second-best result is underlined

Method	Mean Distance	Success Prob.
SM	0.277	100
LS [181]	0.413	100
BReLU [211]	0.312	99.13
L2SM	**0.717**	**94.26**
CCN	0.352	99.95
CCN+LS	<u>0.471</u>	<u>99.08</u>

probability of generating the C&W attack. A higher mean distance signifies that the defense method is more robust. For MNIST dataset, we observe a behavior similar to the DeepFool [123] attack. The *CCN+LS* model significantly outperforms the other defense approaches, followed by *LS* model. For CIFAR10 dataset, we find that L_2-softmax loss (*L2SM*) achieves the best performance with a mean distance of 0.717. It is followed by *CCN+LS* with a mean distance of 0.471.

5.5.5 Ablation Study

5.5.5.1 Robustness analysis of *compactness* for each layer

In this section, we analyze the layer-wise effect of *compactness* on the robustness of a network. We replace one layer from a CNN with the proposed compact convolutional module, and train the entire network on MNIST [95] dataset. We use the same network architecture as given in Table 5.1.

Table 5.13 provides the accuracy of different networks, where only one layer is implemented using compact convolution, against FGSM [53] attack ($\varepsilon = 0.3$) on MNIST [95] testset. We see that the effect of *compactness* on the network's robustness decreases with the depth of the network till a certain layer, after which it increases consistently. The network without any compact layer is equivalent to the *SM* model, and achieves the accuracy of 31.76%. Applying compact module only to the *conv1* layer improves the accuracy to 44.38%. Each of the *conv2*, *conv3* and *conv4* layers decreases the accuracy marginally. The behavior gets reversed in fully connected layers, for which the accuracy increases when compact module is applied to a deeper layer. Applying compact module only to the last fully connected layer (fc3) is equivalent to the *L2SM* model, and achieves the best accuracy of 60.32%. This shows that *compactness* has moderate effect in the shallower layers of the network, and has maximum effect in the deepest layer of the network.

5.5.5.2 t-SNE visualization of adversarial features

In this subsection, we analyze the effect of *compactness* on the separability of features. Fig. 5.5 provides a two dimensional t-SNE [113] visualization of the features obtained after *conv1*, *fc1* and *fc2* layers of the network, for *SM*, *L2SM* and *CCN* models respectively. The features are computed for the adversarial images generated using FGSM attack ($\varepsilon = 0.3$), for two randomly chosen classes from MNIST [95]. From the figure, it is evident that the features from *CCN* and *L2SM* are more separable compared to features from *SM*. The difference in separability is minimum for the initial layers of the network

Table 5.13: Accuracy (%) against FGSM attack (with $\varepsilon = 0$ and $\varepsilon = 0.3$) on MNIST [95] testset.

The best accuracy is shown in bold and the second-best accuracy is underlined

Compact Layer	$\varepsilon = 0$	$\varepsilon = 0.3$
None	99.50	31.76
conv1	99.50	44.38
conv2	99.51	42.34
conv3	99.50	39.80
conv4	99.54	39.04
fc1	**99.58**	44.19
fc2	<u>99.57</u>	<u>58.71</u>
fc3	99.48	**60.32**

(such as conv1), and dominant for the deeper layers (such as fc1 and fc2).

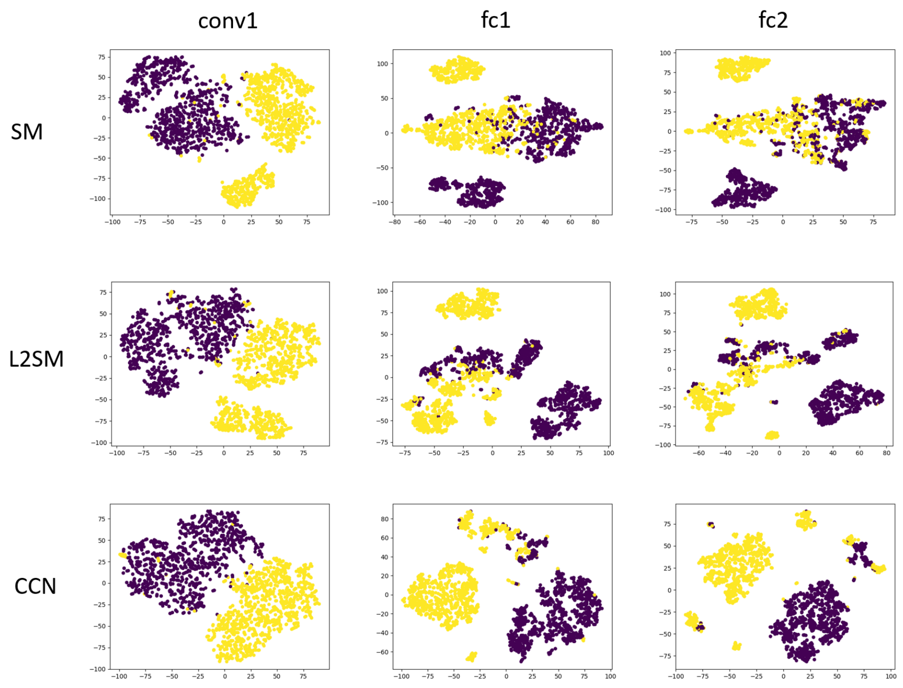


Figure 5.5: t-SNE plots for the features of *conv1*, *fc1* and *fc2* for models *SM*, *L2SM* and *CCN* respectively. The features are computed for the adversarial images generated using FGSM attack ($\epsilon = 0.3$), for two randomly chosen classes from MNIST [95]

5.5.5.3 Analysis of learned network filters

In this subsection, we analyze and compare the characteristics of the weight filters learned for *SM*, *L2SM* and *CCN* models. We compute the top 50 singular values (SVs) for the weight matrix of *conv1*, *conv2*, *conv3*, *conv4*, *fc1* and *fc2* layers, and plot their magnitudes in Fig. 5.6. From the figure, we observe that the *fc2* layer of *CCN* and *L2SM* models has fewer dominant singular values that decay very rapidly compared to SVs of *SM* model. This suggests that *CCN* and *L2SM* models have a strong suppression for the trailing dimensions, which makes them stable to the adversarial variations in the data. For the initial layers of the networks, the SVs for *CCN*, *L2SM* and *SM* models are dominant throughout, since they are less invariant to input deformations.

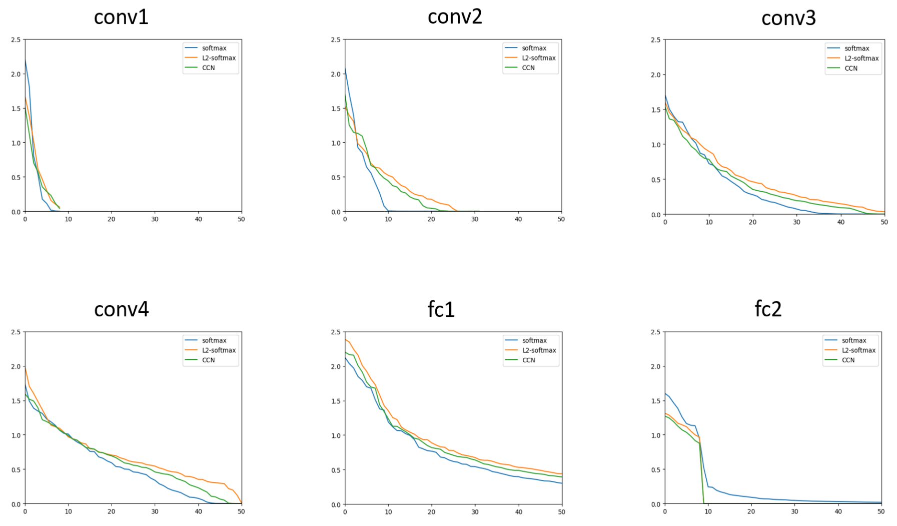


Figure 5.6: Average singular value (SV) spectrum showing top 50 SVs of the weight matrix for each layer of the network.

5.6 Conclusion

In this chapter, we show that learning features by imposing *compactness* constraints can improve a network's robustness against adversarial attacks. The L_2-Softmax Loss, that ensures feature *compactness*, provide better robustness compared to naive softmax loss. This property is applied to each layer of the network using compact convolutional modules (CCN), which significantly reduces the network's vulnerability to adversarial perturbations. In future, we would further analyze the necessary properties for a network to be robust, and build sophisticated architectures that are provably robust to adversarial attacks.

Chapter 6: Conclusions and Future Work

6.1 Summary

In this dissertation, we discussed an end-to-end pipeline for automatic face recognition and propose efficient deep learning algorithms for each of the individual modules of face detection, facial key-points estimation and face verification/identification. Additionally, we presented a new architecture for deep neural networks using feature compactness which is more robust to adversarial attacks.

Chapter 2 presented two algorithms for unconstrained face detection. The DP2MFD algorithm trains DPM for faces on deep feature pyramid. We add a normalization layer to the deep CNN architecture which reduces the bias in the face sizes. The DPSSD algorithm uses the inbuilt feature pyramid present in a DCNN to detect faces at multiple scales. Bottom-up feature aggregation provides the context information to accurately detect tiny faces. Extensive experiments on publicly available unconstrained face detection datasets demonstrate the effectiveness of our proposed approaches.

Chapter 3 presented a multi-task deep learning method called HyperFace for simultaneously detecting faces, localizing landmarks, estimating head pose and identifying gender. Extensive experiments using various publicly available unconstrained datasets demonstrate the effectiveness of our method on all four tasks. Additionally, we presented a multi-task CNN-based method for simultaneous face detection, face alignment, pose estimation, gender and smile classification, age estimation and face verification and recognition. This method performs significantly better than HyperFace, even though both of them use the MTL framework. This work demonstrates that subject-independent tasks benefit from domain-based regularization and network initialization from face recognition task. Also, the improvement in face verification and recognition performance compared to [154] clearly suggests that MTL helps in learning robust feature descriptors.

Chapter 4 proposed Crystal Loss that adds a simple, yet effective, L_2-constraint to the regular softmax loss for training a face verification system. The constraint enforces the features to lie on a hypersphere of a fixed radius characterized by parameter α. We provided bounds on the value of α for achieving a consistent performance. We also presented an overview of modern face recognition systems based on DCNNs. Then, we discussed the major components of our end-to-end face recognition pipeline. Face detection is carried out by the proposed DPSSD face detector while keypoint localization is done using the All-in-One CNN. We also presented the details of our face verification/identification module which uses an ensemble of two networks for feature representation. We discussed training and datasets details for our system and how it relates to existing works on face recognition. We presented the results of our system for four challenging datasets, *viz.*, IJB-A, IJB-B, IJB-C, and IARPA Janus Challenge Set 5 (CS5). Additionally, we performed the most comprehensive examination to date of face identification performance across groups of humans with variable levels of training, experience, talent, and motivation. We compared the accuracy of state-of-the-art face recognition algorithms with humans and show the benefits of a collaborative effort that combines the judgments of humans and machines. The work draws on previous cornerstone findings on human expertise and talent with faces, strategies for fusing human judgments, and computational advances in face recognition. The study provides an evidence-based roadmap for achieving highly accurate face identification. These methods should be extended in future work to test humans and machines on a wider range of face recognition tasks, including recognition across viewpoint and with low-quality images and video as well as recognition of faces from diverse demographic categories.

Chapter 5 showed that learning features by imposing *compactness* constraints can improve a network's robustness against adversarial attacks. The L_2-Softmax Loss, that ensures feature *compactness*, provided better robustness compared to naive softmax loss. This property was applied to each layer of the network using compact convolutional modules, which significantly reduced the network's vulnerability to adversarial perturbations. In future, we plan to further analyze the necessary properties for a network to be robust, and build sophisticated architectures that are provably robust to adversarial attacks.

6.2 Open Issues and Future Direction

Although the proposed face recognition system surpasses human performance on some existing datasets like LFW [70], there are still some open problems that needs to be addressed to make the DCNN-based recognition systems robust and practical. We discuss some of the existing issues with current face recognition pipeline and provide some insights into how we can overcome them.

Reliance on large training data sets: One of the top performing networks in the MegaFace challenge needs 500 million faces of about 10 million subjects. Such large annotated training set may not be always available (e.g. expression recognition, age estimation). So networks that can perform well with reasonable sized training data are needed. It has been shown that having more unconstrained data for training always improves the performance. However, the increment in performance does not linearly scale with the increase in training data. It is always desirable to achieve high accuracy with less training data since it reduces the effort required for data collection. One way to solve this is by using self-supervised learning to pre-train the face verification network. Self supervised learning does not require the input data to be annotated, and hence can use a large number of training samples. Many methods have been proposed for self supervised learning using context, and spatial or temporal ordering [33, 128, 137]. Wiles et al. [188] recently proposed a self-supervised framework for learning facial attributes by simply watching videos of a human face speaking, laughing, and moving over time. A similar approach can be used to pre-train a face recognition system.

Variations in Resolution: Existing face verification/identification algorithms based on deep networks do not perform well on very low resolution faces. Evaluations on recently released IJBS [79] dataset, that contains faces from surveillance videos collected by UAV (see Fig. 6.1), show that the identification accuracy for state-of-the art face recognition systems drops significantly . The faces are blurry and of low resolution which makes it is extremely difficult to extract useful facial features for identification. A major cause for this decrease in performance is the fixed input size used by the existing face identification deep networks. Since the networks are trained on good resolution faces of size 224×224 pixels, they are unable to extract similar meaningful features when a low resolution face of 20×20 pixels is resized to the 224 input size. Moreover, the face alignment also gets affected by resolution which propagates the error down the pipeline. One approach to solve this issue would be to train the network with low resolution and blurry faces. We can generate the training set for these faces using artificial blurring or GANs [52]. One can also think of other techniques such as person re-identification to solve this problem.

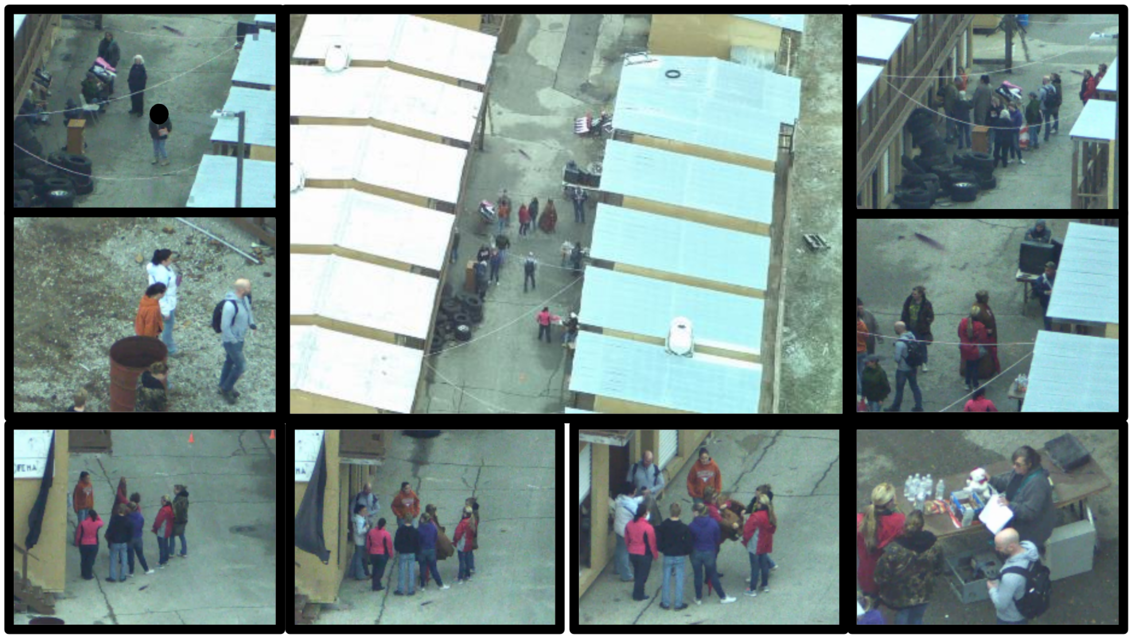


Figure 6.1: Sample frames from the IJBS [79] dataset. The faces are of low resolution which makes them extremely difficult to recognize.

Variations in Appearance: Humans are able to recognize a face even with extreme variations in appearance. For example, if a person disguises himself/herself by wearing a cap, glasses and a fake moustache, he/she can still be comfortably identified by humans. However, the performance of existing deep learning-based face identification systems decreases significantly while identifying disguised faces. The Disguised Faces in the Wild [93] dataset contains genuine, disguise and imposter face pairs. Evaluation of different face recognition algorithms on this dataset reveals that there is still room for improving the algorithms to capture the variations present in disguise and imposter faces. Training the models with large number of disguise and samples will help in improving their performance. Since these images are difficult to collect, we can use GANs [52] to generate different disguised images and use them in training.

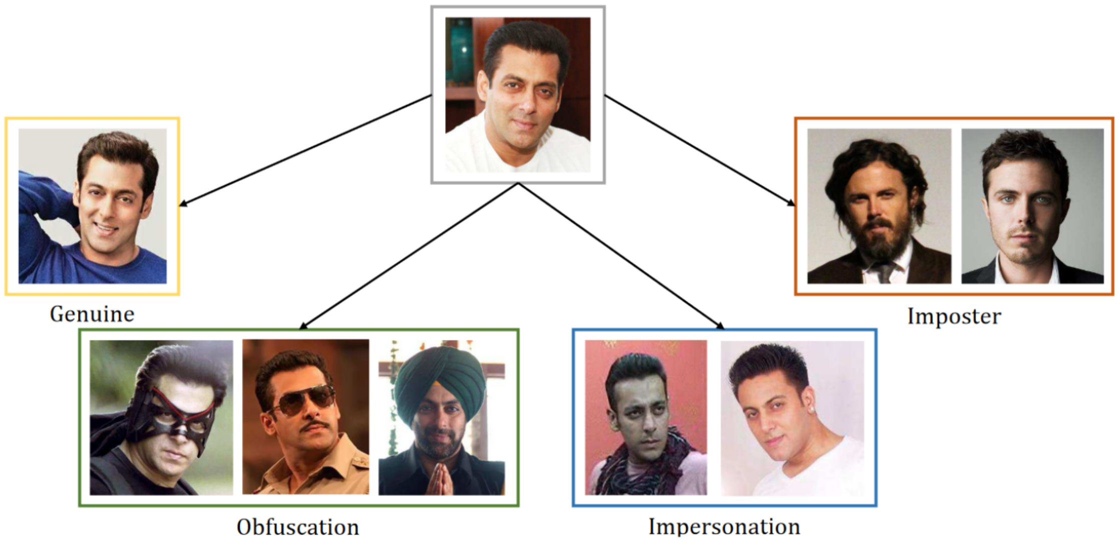


Figure 6.2: Sample genuine, cross-subject impostor, impersonator, and obfuscated face images for a single subject.

Open-set Face Identification: Unlike 1:1 face verification task which requires a single comparison of face feature pairs, 1:N face identification task is much more complex as it requires identifying a given face/template from a set of enrolled subjects present in the gallery. As the gallery size increases, it is more difficult to identify faces. Furthermore, for open-set face identification, most subjects in the probe images are not contained in the gallery which makes this task more challenging. Identity-specific face attributes can be used to provide additional information in filtering out non-relevant subjects from the gallery.

Incorporating Domain Knowledge: Training the face recognition networks on one set of data and evaluating it on a different set can cause performance gap. DCNNs are still not domain invariant, and usually overfit to the domain of training data. For example, training on high resolution faces does not perform as expected on low resolution faces. Similary, training on mostly Caucasian subjects leads to a performance bias against faces of different ethnicity. The current practice to solve this issue is to rely on fine-tuning. For instance, one can train on a generic dataset with large number of samples, and then fine-tune on the data of same domain as that of the test set. However, most of the time there is no information available about the test domain. In such scenarios, domain adaptation techniques for DCNNs can help in overcoming this issue.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Wael AbdAlmageed, Yue Wu, Stephen Rawls, Shai Harel, Tal Hassner, Iacopo Masi, Jongmoo Choi, Jatuporn Lekust, Jungyeon Kim, Prem Natarajan, et al. Face recognition using deep multi-pose representations. In *IEEE Winter Conference on Applications of Computer Vision (WACV), 2016*, pages 1–9.

[3] Ankur Agarwal and Bill Triggs. Multilevel image coding with hyperfeatures. *International Journal of Computer Vision*, pages 15–27, 2008.

[4] Vineeth Nallure Balasubramanian, Jieping Ye, and Sethuraman Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–7. IEEE, 2007.

[5] Ankan Bansal, Carlos Castillo, Rajeev Ranjan, and Rama Chellappa. The Do's and Don'ts for CNN-Based Face Verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2545–2554, 2017.

[6] Ankan Bansal, Anirudh Nanduri, Carlos D Castillo, Rajeev Ranjan, and Rama Chellappa. Umdfaces: An annotated face dataset for training deep networks. In *IEEE International Joint Conference on Biometrics (IJCB), 2017*, pages 464–473. IEEE, 2017.

[7] Ankan Bansal, Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. Deep Features for Recognizing Disguised Faces in the Wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 10–16, 2018.

[8] Xavier P Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark

estimation under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1513–1520, 2013.

[9] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 67–74. IEEE, 2018.

[10] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014.

[11] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[13] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[14] Feng-Ju Chang, Anh Tuan Tran, Tal Hassner, Iacopo Masi, Ram Nevatia, and Gerard Medioni. Faceposenet: Making a case for landmark-free face alignment. In *2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 1599–1608. IEEE, 2017.

[15] Wei-Lun Chao, Jun-Zuo Liu, and Jian-Jiun Ding. Facial age estimation based on label-sensitive learning and age-oriented regression. *Pattern Recognition*, 46(3):628–641, 2013.

[16] D. Chen, G. Hua, F. Wen, and J. Sun. Supervised transformer network for efficient face detection. In *European Conference on Computer Vision*, pages 122–138. Springer, 2016.

[17] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *European Conference on Computer Vision*, volume 8694, pages 109–122. 2014.

[18] J.-C. Chen, R. Ranjan, A. Kumar, C.-H. Chen, V. M. Patel, and R. Chellappa. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In *IEEE International Conference on Computer Vision Workshop on ChaLearn Looking at People*, pages 118–126, 2015.

[19] Jun-Cheng Chen, Amit Kumar, Rajeev Ranjan, Vishal M Patel, Azadeh Alavi, and Rama Chellappa. A cascaded convolutional neural network for age estimation of unconstrained faces. In *IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2016*, pages 1–8. IEEE, 2016.

[20] Jun-Cheng Chen, Wei-An Lin, Jingxiao Zheng, and Rama Chellappa. A real-time multi-task single shot face detector. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 176–180. IEEE, 2018.

[21] Jun-Cheng Chen, Vishal M Patel, and Rama Chellappa. Unconstrained face verification using deep CNN features. In *IEEE Winter Conference on Applications of Computer Vision (WACV), 2016*, pages 1–9. IEEE, 2016.

[22] Jun-Cheng Chen, Rajeev Ranjan, Swami Sankaranarayanan, Amit Kumar, Ching-Hui Chen, Vishal M Patel, Carlos D Castillo, and Rama Chellappa. Unconstrained

still/video-based face verification with deep convolutional neural networks. *International Journal of Computer Vision*, pages 1–20, 2017.

[23] J. Cheney, B. Klein, A. K. Jain, and B. F. Klare. Unconstrained face detection: State of the art baseline and challenges. In *International Conference on Biometrics*, 2015.

[24] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.*, volume 1, pages 539–546. IEEE, 2005.

[25] Aruni Roy Chowdhury, Tsung-Yu Lin, Subhransu Maji, and Erik Learned-Miller. One-to-many face recognition with bilinear CNNs. In *IEEE Winter Conference on Applications of Computer Vision (WACV).*, pages 1–9. IEEE, 2016.

[26] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863, 2017.

[27] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[28] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[29] N. Crosswhite, J. Byrne, O. M. Parkhi, C. Stauffer, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. *IEEE International Conference on Automatic Face and Gesture Recognition*, 2017.

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. 1:886–893 vol. 1, June 2005.

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 248–255. IEEE, 2009.

[32] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018.

[33] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[34] Andrew J Dowsett and A Mike Burton. Unfamiliar face matching: Pairs outperform individuals and provide a route to training. *British journal of psychology*, 106(3):433–445, 2015.

[35] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of JPG compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[36] Max Ehrlich, Timothy J Shields, Timur Almaev, and Mohamed R Amer. Facial attributes classification using multi-task representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 47–55, 2016.

[37] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of*

the IEEE International Conference on Computer Vision, pages 2650–2658, 2015.

[38] Mohamed Y El Dib and Motaz El-Saban. Human age estimation using enhanced bio-inspired features (ebif). In *2010 IEEE International Conference on Image Processing*, pages 1589–1592. IEEE, 2010.

[39] Sergio Escalera, Junior Fabian, Pablo Pardo, Xavier Baró, Jordi Gonzalez, Hugo J Escalante, Dusan Misevic, Ulrich Steiner, and Isabelle Guyon. Chalearn looking at people 2015: Apparent age and cultural event recognition datasets and results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–9, 2015.

[40] Sergio Escalera, M Torres, B Martinez, Xavier Baró, Hugo Jair Escalante, et al. Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition Workshops*, 2016.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[42] M. R. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy" – automatic naming of characters in tv video. In *Proceedings of the British Machine Vision Conference*, pages 92.1–92.10, 2006.

[43] S. S. Farfade, M. J. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. In *International Conference on Multimedia Retrieval*, 2015.

[44] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[45] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008.*, pages 1–8. IEEE, 2008.

[46] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010.

[47] Spyros Gidaris and Nikos Komodakis. Locnet: Improving localization accuracy for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 789–798, 2016.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[49] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446, 2015.

[50] Georgia Gkioxari, Bharath Hariharan, Ross B. Girshick, and Jitendra Malik. R-cnns for pose estimation and action detection. *CoRR*, abs/1406.5212, 2014.

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets.

In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[54] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[55] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[56] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[57] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016.

[58] Hu Han, Charles Otto, and Anil K Jain. Age estimation from face images: Human vs. machine performance. In *International Conference on Biometrics (ICB)*, pages 1–8. IEEE, 2013.

[59] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] Md Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, Liming Chen, et al. von mises-fisher mixture model-based deep learning: Application to face verification. *arXiv preprint arXiv:1706.04264*, 2017.

[61] Md Abul Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, and Liming Chen. Deepvisage: Making face recognition simple yet with powerful generalization skills. In *ICCV Workshops*, pages 1682–1691, 2017.

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer, 2014.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[65] Lijun Hong, Di Wen, Chi Fang, and Xiaoqing Ding. A new biologically inspired active appearance model for face age estimation by using local ordinal ranking. In *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, pages 327–330. ACM, 2013.

[66] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

[67] Nan Hu, Weimin Huang, and Surendra Ranganath. Head pose estimation by nonlinear embedding and mapping. In *IEEE International Conference on Image Pro-*

cessing, 2005., volume 2, pages II–342–5, Sept 2005.

[68] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *CVPR*, volume 1, page 3, 2017.

[69] Ying Hu, Kelsey Jackson, Amy Yates, David White, P Jonathon Phillips, and Alice J OToole. Person recognition: Qualitative differences in how forensic face examiners and untrained people rely on the face versus the body for identification. *Visual Cognition*, 25(4-6):492–506, 2017.

[70] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on Faces in Real-Life Images: Detection, Alignment, and Recognition*, 2008.

[71] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[72] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[73] V. Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 577–584, June 2011.

[74] Vidit Jain and Erik Learned-Miller. FDDB: A benchmark for face detection in unconstrained settings. (UM-CS-2010-009), 2010.

[75] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678, 2014.

[76] Huaizu Jiang and Erik Learned-Miller. Face detection with the Faster R-CNN. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 650–657. IEEE, 2017.

[77] Amin Jourabloo and Xiaoming Liu. Pose-invariant 3D face alignment. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[78] Z. Kalal, J.G. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *Proceedings of the British Machine Vision Conference*, pages 42.1–42.10, 2008.

[79] Nathan Kalka, Brianna Maze, James Duncan, Kevin Connor, Stephen Eliott, Kaleb Hebert, Julia Bryan, and Anil Jain. IJBS: IARPA Janus Surveillance Video Benchmark. *https://pdfs.semanticscholar.org/d1a4/3737ca8be02d65684cf64ab2331f66947207.pdf*, 2018.

[80] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, June 2014.

[81] I. Kemelmacher-Shlizerman, S. M Seitz, D. Miller, and E. Brossard. The Megaface Benchmark: 1 Million Faces for Recognition at Scale. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4873–4882, 2016.

[82] W Kienzle, G BakIr, M Franz, and B Schölkopf. Face detection: Efficient and rank

deficient. In *Advances in Neural Information Processing Systems*, pages 673–680, 2005.

[83] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[84] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, Mark Burge, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus benchmark A. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1931–1939. IEEE, 2015.

[85] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[86] M. Kostinger, P. Wohlhart, P.M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *IEEE International Conference on Computer Vision Workshops*, pages 2144–2151, Nov 2011.

[87] Martin Köstinger, Paul Wohlhart, Peter Roth, and Horst Bischof. Robust face detection by simple means. In *Computer Vision in Applications Workshop CVAW*, 2012.

[88] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[89] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[90] A. Kumar, R. Ranjan, V. Patel, and R. Chellappa. Face alignment by local deep descriptor regression. *arXiv preprint arXiv:1601.07950*, 2016.

[91] N. Kumar, P. N. Belhumeur, and S. K. Nayar. FaceTracer: A Search Engine for Large Collections of Images with Faces. In *European Conference on Computer Vision (ECCV)*, pages 340–353, Oct 2008.

[92] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[93] Vineet Kushwaha, Maneet Singh, Richa Singh, Mayank Vatsa, Nalini Ratha, and Rama Chellappa. Disguised faces in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, volume 8, 2018.

[94] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[95] Yann LeCun, Corinna Cortes, and Christopher JC Burges. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[96] Gil Levi and Tal Hassner. Age and gender classification using convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) workshops*, June 2015.

[97] Chenghua Li, Qi Kang, Guojing Ge, Qiang Song, Hanqing Lu, and Jian Cheng. DeepBE: Learning Deep Binary Encoding for Multi-Label Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 39–46, 2016.

[98] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, June 2015.

[99] Haoxiang Li, Gang Hua, Zhe Lin, J. Brandt, and Jianchao Yang. Probabilistic elastic part model for unsupervised face detector adaptation. In *IEEE International Conference on Computer Vision*, pages 793–800, Dec 2013.

[100] Haoxiang Li, Zhe Lin, J. Brandt, Xiaohui Shen, and Gang Hua. Efficient boosted exemplar-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1843–1850, June 2014.

[101] Jianguo Li, Tao Wang, and Yimin Zhang. Face detection using surf cascade. In *IEEE International Conference on Computer Vision Workshop on Benchmarking Facial Image Analysis Technologies*, pages 2183–2190, Nov 2011.

[102] Jianguo Li and Yimin Zhang. Learning surf cascade for fast and accurate object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3468–3475, 2013.

[103] Y. Li, B. Sun, T. Wu, and Y. Wang. Face detection with end-to-end integration of a convnet and a 3d model. *European Conference on Computer Vision (ECCV)*, 2016.

[104] Lin Liang, Rong Xiao, Fang Wen, and Jian Sun. Face alignment via component-based discriminative search. In *European conference on computer vision*, pages 72–85. Springer, 2008.

[105] S. Liao, A. Jain, and S. Li. A fast and accurate unconstrained face detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[106] W.-A. Lin, J.-C. Chen, and R. Chellappa. A proximity-aware hierarchical clustering of faces. *IEEE Conference on Automatic Face and Gesture Recognition*, 2017.

[107] Jingtuo Liu, Yafeng Deng, Tao Bai, Zhengping Wei, and Chang Huang. Targeting ultimate accuracy: Face recognition via deep embedding. *arXiv preprint arXiv:1506.07310*, 2015.

[108] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C Berg. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[109] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[110] Xin Liu, Shaoxin Li, Meina Kan, Jie Zhang, Shuzhe Wu, Wenxian Liu, Hu Han, Shiguang Shan, and Xilin Chen. Agenet: Deeply learned regressor and classifier for robust apparent age estimation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 16–24, 2015.

[111] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[112] Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *arXiv preprint arXiv:1704.00103*, 2017.

[113] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[114] N. Markus, M. Frljak, I. S. Pandzic, J. Ahlberg, and R. Forchheimer. A method for object detection based on pixel intensity comparisons organized in decision trees, 2014. CoRR.

[115] Iacopo Masi, Anh Tun Trn, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. Do we really need to collect millions of faces for effective face recognition? In *European Conference on Computer Vision*, pages 579–596. Springer, 2016.

[116] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *European Conference on Computer Vision*, volume 8692, pages 720–735. 2014.

[117] Iain Matthews and Simon Baker. Active appearance models revisited. *International journal of computer vision*, 60(2):135–164, 2004.

[118] Brianna Maze, Jocelyn Adams, James A Duncan, Nathan Kalka, Tim Miller, Charles Otto, Anil K Jain, W Tyler Niggel, Janet Anderson, Jordan Cheney, et al. IARPA Janus Benchmark–C: Face dataset and protocol. In *11th IAPR International Conference on Biometrics*, 2018.

[119] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[120] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, pages 69–82, 2004.

[121] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[122] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94. Ieee, 2017.

[123] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[124] E. Murphy-Chutorian and M.M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 31(4):607–626, 2009.

[125] Hajime Nada, Vishwanath A Sindagi, He Zhang, and Vishal M Patel. Pushing the limits of unconstrained face detection: a challenge dataset and baseline results. *arXiv preprint arXiv:1804.10275*, 2018.

[126] Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry Davis. SSH: Single Stage Headless Face Detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4875–4884, 2017.

[127] A. Nech and I. Kemelmacher-Shlizerman. Level playing field for million scale face recognition. *IEEE International Conference on Computer Vision and Pattern*

Recognition (CVPR), 2017.

[128] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[129] E Noyes, PJ Phillips, and AJ OToole. What is a super-recogniser. *Face Processing: Systems, Disorders, and Cultural Differences, eds Bindermann M, Megreya AM (Nova, New York)*, pages 173–201, 2017.

[130] Eshed Ohn-Bar and Mohan M Trivedi. To boost or not to boost? on the limits of boosted trees for object detection. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3350–3355. IEEE, 2016.

[131] Alice J O'Toole, Hervé Abdi, Fang Jiang, and P Jonathon Phillips. Fusing face-verification algorithms and humans. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(5):1149–1155, 2007.

[132] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[133] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[134] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[135] Connor J Parde, Carlos Castillo, Matthew Q Hill, Y Ivette Colon, Swami Sankaranarayanan, Jun-Cheng Chen, and Alice J O'Toole. Deep convolutional neural network features and the original image. *arXiv preprint arXiv:1611.01751*, 2016.

[136] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, volume 1, page 6, 2015.

[137] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[138] P Jonathon Phillips. A cross benchmark assessment of a deep convolutional neural network for face recognition. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 705–710. IEEE, 2017.

[139] P Jonathon Phillips, Fang Jiang, Janet Ayyad, Nils Pénard, et al. Face recognition algorithms surpass humans matching faces over changes in illumination. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1642–1646, 2007.

[140] P Jonathon Phillips and Alice J O'toole. Comparison of human and computer performance across face recognition experiments. *Image and Vision Computing*, 32(1):74–85, 2014.

[141] R. Ranjan, V. M. Patel, and R. Chellappa. A deep pyramid deformable part model for face detection. In *International Conference on Biometrics Theory, Applications*

and Systems, 2015.

[142] Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[143] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. HyperFace: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[144] Rajeev Ranjan, Swami Sankaranarayanan, Ankan Bansal, Navaneeth Bodla, Jun-Cheng Chen, Vishal M Patel, Carlos D Castillo, and Rama Chellappa. Deep learning for understanding faces: Machines may be just as good, or better, than humans. *IEEE Signal Processing Magazine*, 35(1):66–83, 2018.

[145] Rajeev Ranjan, Swami Sankaranarayanan, Carlos D Castillo, and Rama Chellappa. An all-in-one convolutional neural network for face analysis. In *12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), 2017*, pages 17–24. IEEE, 2017.

[146] Rajeev Ranjan, Sabrina Zhou, Jun Cheng Chen, Amit Kumar, Azadeh Alavi, Vishal M Patel, and Rama Chellappa. Unconstrained age estimation with deep convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 109–117, 2015.

[147] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[148] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[149] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Face alignment at 3000 fps via regressing local binary features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1685–1692, 2014.

[150] Karl Ricanek and Tamirat Tesafaye. Morph: A longitudinal image database of normal adult age-progression. In *7th International Conference on Automatic Face and Gesture Recognition, 2006. FGR 2006.*, pages 341–345. IEEE, 2006.

[151] R. Rothe, R. Timofte, and L. V. Gool. Dex: Deep expectation of apparent age from a single image. In *ICCV, ChaLearn Looking at People workshop*, December 2015.

[152] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[153] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 397–403, 2013.

[154] Swami Sankaranarayanan, Azadeh Alavi, Carlos D Castillo, and Rama Chellappa. Triplet probabilistic embedding for face verification and clustering. In *IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2016*, pages 1–8. IEEE, 2016.

[155] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training. *arXiv preprint arXiv:1705.07819*, 2017.

[156] J. M. Saragih, S. Lucey, and J. F. Cohn. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91(2):200–215, 2011.

[157] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[158] Santi Segu, Michal Drozdzal, Petia Radeva, and Jordi Vitri. An integrated approach to contextual face detection. In *International Conference on Pattern Recognition Applications and Methods*, pages 90–97, 2012.

[159] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[160] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.

[161] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[162] Xiaohui Shen, Zhe Lin, J. Brandt, and Ying Wu. Detecting and aligning faces by image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3460–3467, June 2013.

[163] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[164] S. Srinivasan and K.L. Boyer. Head pose estimation using view based eigenspaces. In *Proceedings. 16th International Conference on Pattern Recognition, 2002.*, volume 4, pages 302–305 vol.4, 2002.

[165] V. Subburaman and Sébastien Marcel. Fast bounding box estimation based face detection. In *Proc. Workshop on Face Detection of the European Conference on Computer Vision*, Heraklion, Crete, Greece, 2010.

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

[167] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, pages 1988–1996. 2014.

[168] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900, 2015.

[169] Kah Kay Sung and Tomaso Poggio. Example based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:39–51, 1995.

[170] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[171] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[172] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[173] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[174] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[175] G. Tzimiropoulos and M. Pantic. Gauss-newton deformable part models for face alignment in-the-wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, June 2014.

[176] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[177] Michal Uricár, Radu Timofte, Rasmus Rothe, Jiri Matas, and Luc Van Gool. Structured output svm prediction of apparent age, gender and smile from deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–33, 2016.

[178] Paul A. Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[179] D. Wang, C. Otto, and A. K. Jain. Face Search at Scale: 80 Million Gallery. *arXiv preprint arXiv:1507.07242*, 2015.

[180] Jing Wang, Yu Cheng, and Rogerio Schmidt Feris. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. *arXiv preprint arXiv:1604.06433*, 2016.

[181] David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, page 311, 2016.

[182] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision (ECCV)*, pages 499–515, 2016.

[183] Yandong Wen, Zhifeng Li, and Yu Qiao. Latent factor guided convolutional neural networks for age-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4901, 2016.

[184] David White, A Mike Burton, Richard I Kemp, and Rob Jenkins. Crowd effects in unfamiliar face matching. *Applied cognitive psychology*, 27(6):769–777, 2013.

[185] David White, James D Dunn, Alexandra C Schmid, and Richard I Kemp. Error rates in users of automatic face recognition software. *PLoS One*, 10(10):e0139827,

2015.

[186] David White, P Jonathon Phillips, Carina A Hahn, Matthew Hill, and Alice J O'Toole. Perceptual expertise in forensic facial image comparison. *Proc. R. Soc. B*, 282(1814):20151292, 2015.

[187] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn C Adams, Tim Miller, Nathan D Kalka, Anil K Jain, James A Duncan, Kristen Allen, et al. IARPA Janus Benchmark-B face dataset. In *CVPR Workshops*, pages 592–600, 2017.

[188] Olivia Wiles, A Koepke, and Andrew Zisserman. Self-supervised learning of a facial attribute embedding from video. *arXiv preprint arXiv:1808.06882*, 2018.

[189] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–534. IEEE, 2011.

[190] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light CNN for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018.

[191] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] Weidi Xie, Li Shen, and Andrew Zisserman. Comparator networks. In *European Conference on Computer Vision*, pages 811–826. Springer, 2018.

[193] Weidi Xie and Andrew Zisserman. Multicolumn networks for face recognition. *arXiv preprint arXiv:1807.09192*, 2018.

[194] L. Xiong, K. Jayashree, J. Zhao, J. Feng, S. Pranata, and S. Shen. A good practice towards top performance of face recognition: Transferred deep feature fusion. *arXiv preprint arXiv:1704.00438*, 2017.

[195] Xuehan Xiong and Fernando De la Torre. Global supervised descent method. June 2015.

[196] Hongyu Xu, Jingjing Zheng, Azadeh Alavi, and Rama Chellappa. Template regularized sparse coding for face verification. In *23rd International Conference on Pattern Recognition, ICPR 2016*, pages 1448–1454, 2016.

[197] Xuehan-Xiong and Fernando De la Torre. Supervised descent method and its application to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[198] Junjie Yan, Zhen Lei, Longyin Wen, and S.Z. Li. The fastest deformable part model for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2504, June 2014.

[199] Junjie Yan, Zhen Lei, Dong Yi, and Stan Li. Learn to combine multiple hypotheses for accurate face alignment. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 392–396, 2013.

[200] Junjie Yan, Xuzong Zhang, Zhen Lei, and Stan Z. Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790 – 799, 2014.

[201] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features. In *IEEE International Conference on Computer Vision*, 2015.

[202] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Aggregate channel features for

multi-view face detection. In *IEEE International Joint Conference on Biometrics (IJCB), 2014*, pages 1–8. IEEE, 2014.

[203] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Fine-grained evaluation on face detection in the wild. In *11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2015*, volume 1, pages 1–7. IEEE, 2015.

[204] Jiaolong Yang, Peiran Ren, Dongqing Zhang, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. Neural aggregation network for video face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5216–5225. IEEE, 2017.

[205] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *IEEE International Conference on Computer Vision*, pages 3676–3684, 2015.

[206] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5533, 2016.

[207] Shuo Yang, Yuanjun Xiong, Chen Change Loy, and Xiaoou Tang. Face detection through scale-friendly deep convolutional networks. *arXiv preprint arXiv:1706.02863*, 2017.

[208] Songfan Yang and Deva Ramanan. Multi-scale recognition with DAG-CNNs. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[209] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[210] Xiang Yu, Junzhou Huang, Shaoting Zhang, Wang Yan, and Dimitris Metaxas. Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1944–1951, 2013.

[211] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. *arXiv preprint arXiv:1707.06728*, 2017.

[212] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[213] Cha Zhang and Zhengyou Zhang. A survey of recent advances in face detection. Technical Report MSR-TR-2010-66, Microsoft Research, 2010.

[214] Kaipeng Zhang, Lianzhi Tan, Zhifeng Li, and Yu Qiao. Gender and smile classification using deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[215] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[216] Ning Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1644, 2014.

[217] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S^3 FD: Single Shot Scale-invariant Face Detector. *arXiv preprint*

arXiv:1708.05237, 2017.

[218] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):918–930, 2016.

[219] Zhanpeng Zhang, Ping Luo, ChenChange Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108, 2014.

[220] W. Zhao, R. Chellappa, J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, pages 399–458, Dec. 2003.

[221] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[222] Chenchen Zhu, Yutong Zheng, Khoa Luu, and Marios Savvides. CMS-RCNN: contextual multi-scale region-based cnn for unconstrained face detection. In *Deep Learning for Biometrics*, pages 57–79. Springer, 2017.

[223] S. Zhu, C. Li, C.-C. Loy, and X. Tang. Unconstrained face alignment via cascaded compositional learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3409–3417, 2016.

[224] Shizhan Zhu, Cheng Li, Chen Change Loy, and Xiaoou Tang. Face alignment by coarse-to-fine shape searching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4998–5006, 2015.

[225] Xiangxin Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, June 2012.

[226] Xiangxin Zhu and Deva Ramanan. FaceDPL: Detection, pose estimation, and landmark localization in the wild. preprint 2015.

[227] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Z Li. Face alignment across large poses: A 3D solution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 146–155, 2016.