ABSTRACT

Title of thesis: PARTIAL FACE DETECTION
AND ILLUMINATION ESTIMATION

Sayantan Sarkar
Master of Science, 2018

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

Face Analysis has long been a crucial component of many security applications. In this work, we shall propose and explore some face analysis algorithms which are applicable to two different security problems, namely Active Authentication and Image Tampering Detection. In the first section, we propose two algorithms, "Deep Feature based Face Detection for Mobile Devices" and "DeepSegFace" that are useful in detecting partial faces such as those seem in typical Active Authentication scenarios. In the second section, we propose an algorithm to detect discrepancies in illumination conditions given two face images, and use that as an indication to decide if an image has been tampered by transplanting faces. We also extend the illumination detection algorithm by proposing an adversarial data augmentation scheme. We show the efficacy of the proposed algorithms by evaluating them on multiple datasets.

PARTIAL FACE DETECTION AND ILLUMINATION ESTIMATION

by

Sayantan Sarkar

Thesis 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
Master of Science
2018

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Min Wu
Professor Behtash Babadi

© Copyright by
Sayantan Sarkar
2018

Acknowledgments

I am deeply indebted to my advisor, Dr. Rama Chellappa. His advice and support has been crucial for the completion of this work. He has often gone out of his way to provide me with assistance. I am also grateful to everyone in Dr. Chellappa's group. It has been my pleasure working with this talented set of people.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Face detection and analysis is a well studied problem in recent Computer Vision research. Advances in Deep Learning techniques have enabled leaps in performance in both low and high level vision tasks relating to facial analysis such as

- Low level tasks: Face Detection, Pose and Fiducial Detection and Facial Attribute Detection
- High level tasks: Face Recognition and Face Verification

Face analysis is often a crucial part of many security applications such as Active Authentication (AA) and image tampering detection. In this work, we focus on the problem of detecting partial faces in AA scenarios, finding illumination conditions of a facial image and leveraging that information to detect if an image has been possibly forged.

1.1 Partial Single Face Detection for Active Authentication

The first step in the facial analysis pipeline is Face Detection. Due to the release of many public, large-scale face datasets [2] [3] [4], many state-of-the-art face detectors [5] [6] [7] have been proposed and developed, which approach human performance in detecting faces even in cluttered environments. However, consider the domain of Active Authentication (AA), where one is interested in developing algorithms that continuously verify if the correct user is using a mobile device. Verification of users using their front camera captures is significant part of many proposed AA systems. Detecting faces from front cameras captures comes with a unique set of challenges and advantages, compared to detecting faces in a general setting. Some of the challenges of detecting faces in the AA setting are:

- Partial Visibility: In many cases, the user's face captured by the front camera is only partially visible.

- Running on phone GPU: The face detector should be light-weight enough to run on a phone's GPU.

However, there are some mitigating circumstances that make the task simpler, such as:

- Single Face: Since mobile devices are used by single users at a time, there is only one face to be detected. Also users are present close to the phone, hence we can assume a fairly large face is to be detected.

- Pose: When users interact with mobile devices, usually frontal images are captured. Extreme pose variations are rare.

Thus, in this work, we focus on developing face detectors that take the above mentioned factors into consideration, to perform well in the task of detecting single, but partially visible faces.

1.2 Tampering Detection using Face Illumination Discrepancies

Image tampering detection is an area where facial analysis has recently gained traction. Consider the case where a face has been digitally doctored into an image. The transplanted face image is under a certain lighting condition, depending on the illumination of the image it is originally from. There is a significant chance that the image it is being pasted into has a different illumination condition. Thus, if we check the illumination conditions of faces in a given image and find discrepancies, it is indicative that one of the faces has been transplanted.

In this work, we explore a method of identifying the illumination conditions of a 2D face crop, and then extend that into a verification framework that is trained to directly predict if two faces have the same illumination condition.

Chapter 2: Related Work

In this chapter, we shall review some of the relevant related works in the fields of Active Authentication and Face Detection and the datasets used in the various methods described later.

2.1 Active Authentication Datasets

Active Authentication (AA) is an active field of research. Many methods have been proposed based on different individual modalities and their fusion such as:

- Facial images [8] [9]
- Touch/swipe signature [10]
- Gait recognition [11]
- Device movement and accelerometer patterns [12] [13]
- Multi-modal fusion [14]

In this work, we focus mostly on face detection for AA. While there are many large-scale face detection datasets, they are not suited for face detection in the AA setting. The main characteristics of an ideal AA face detection dataset are:

- Device: The image should have been captured by mobile device or a webcam.
- User awareness: While the user is informed of the data collection process beforehand, it must be ensured that during the process itself, the user is not made conscious that he or she is being recorded and should be left free to perform actions naturally.
- Dataset size: The dataset should be large and varied enough to be able to train complex machine learning models.
- Usability: The dataset should be annotated and available publicly.

A few relevant AA datasets are described below:

2.1.1 MOBIO: Constrained mobile dataset

The MOBIO dataset [15] contains 61 hours of audio-visual data from a NOKIA N93i phone (and a 2008 Mac-book laptop) with 12 distinct sessions of 150 participants spread over several weeks. However, it is a constrained dataset, where subjects were required to position their head inside an elliptical region within the scene while capturing the data. Therefore it does not represent real-life acquisition scenarios.

2.1.2 Abacus and Move: Private datasets

Google's Project Abacus dataset is a 27.62 TB collection of smartphone usage data of 1500 users for six months on $Nexus5$ phones [16]. It contains multiple modalities, such as front-facing camera, touchscreen and keyboard, gyroscope, accelerometer, magnetometer, ambient light sensor, GPS, Bluetooth, WiFi, cell antennae, app usage and on time statistics. Google also collected the 114GB Project Move dataset, which consists of smartphone inertial signals collected from 80 volunteers over two months on $LG3$, $Nexus5$, and $Nexus6$ phones. While these datasets are unconstrained, they are not available publicly for the research community.

2.1.3 AA-01 and UMDAA-02: Publicly available unconstrained mobile datasets

The AA-01 dataset [14] is a challenging dataset for front-camera face detection task which contains the front-facing camera face videos and swipe information for 43 male and 7 female IPhone users under three different ambient lighting conditions. A subset of this dataset, AA-01-FD, contains 8036 frames annotated with a face bounding box. AA-01-FD, contains 1607 frames without faces and 6429 frames with faces [1], [17]. The training split consists of 5202 images, while the rest are used as test images. The images in this are semi-constrained as the subjects perform a set task during the data collection period. However they are not required or encouraged to maintain a certain posture, hence the dataset is sufficiently challenging due to pose variations, occlusions and partial faces.

The UMDAA-02 dataset is an unconstrained dataset containing 141.14 GB of smartphone usage collected from 48 subjects using Nexus 5 phones over a 2 month period. All modalities described in Google's Abacus dataset are collected. A subset UMDAA-02-FD was created with $33,209$ images, which was annotated with facial bounding box information.

2.2 Face Detection

In this section we shall briefly review different face detection methods by categorizing them under 3 subsections.

2.2.1 Traditional methods

Face detection is one of the earliest applications of computer vision dating back several decades [18] [19]. However, most methods before 2004 performed poorly in unconstrained conditions, and therefore were not applicable in real-world settings [20]. Viola and Jones's seminal work on boosted cascaded classification-based face detection [21] was the first algorithm that made face detection feasible in real-world applications. The method, however, works reasonably well only for well illuminated, near-frontal faces without occlusion [22]. Extensions of the boosted architecture for multi-view face detection were proposed [23] [24], but these detectors are difficult to train, and do not perform well because of inaccuracies introduced by viewpoint estimation and quantization [22].

The next step in the evolution of face detectors was the introduction of geometric modeling. Using facial components or parts to construct a deformable part model (DPM) of a face, a robust face detector was proposed in [5]. Similar approaches are found in [25] [26]. In [27], the authors introduced an examplar-based face detection method that does not require multi-scale shifting windows.

A popular face detection paradigm was using some combination of a robust image features like SURF, local binary pattern (LFP) histogram of oriented gradient (HoG) or their variants, along with a classifier, especially support vector machines (SVMs), in a sliding window fashion. Researchers proposed different combinations of features with SVM for robust face detection as surveyed in [20].

Some noteworthy, high performance traditional, non-deep methods are discussed below. In [28] the authors improved the performance of the DPM-based method and also introduced Headhunter, a new face detector that uses Integral Channel Features (ICF) with boosting to achieve state-of-the-art performance in face detection in the wild. A fast face detector that uses the scale invariant and bounded Normalized Pixel Difference (NPD) features is proposed in [29] that uses a single soft-cascade classifier to handle unconstrained face detection. The method is claimed to achieve state-of-the-arts performance on FDDB, GENKI, and CMU-MIT datasets.

2.2.2 Deep learning-based methods

The performance break-through observed after the introduction of Deep Convolutional Neural Networks (DCNN) can be attributed to the availability of large labeled datasets, availability of GPUs, the hierarchical nature of the deep networks and regularization techniques such as dropout and batch normalization [20].

Some of the earliest methods that introduced deep features to the space of face detection relied on transfer learning. Instead of training a full deep network, [30] used features from alexnet [31] and then trained SVM on those deep features to detect faces. Full, end-to-end trainable deep networks for face detection appeared in [32] [33]. Since then there have been a proliferation of deep learning based algorithms for face detection.

Among recent works, HyperFace [7] and All-in-one network [6] are a deep multi-task learning framework that perform face detection, landmark localization, pose Estimation, and gender recognition among other tasks. These methods exploits the synergy among related tasks by fusing the intermediate layers of a deep CNN using a separate CNN and thereby boosting their individual performances.

2.2.3 Methods tailored for active authentication

Continuous authentication of mobile devices requires partially visible face detection and verification to operate reliably [34]. In [1], the authors introduced a face detection method based on facial segments to detect partial faces on images captured for AA with smartphones. The algorithm first produces face proposals by employing a number of weak Adaboost facial segment detectors on each image and then clustering them. After filtering out overlapping proposals and then forming subsets of facial segments from each cluster, statistical features from the face proposals were used to train a support vector machine classifier for face detection. The method worked well on AA-01-FD [14] and UMDAA-02-FD [34] mobile face detection datasets compared to other non-CNN methods [34].

Among deep learning methods that consider face parts, [35] achieves state-of-the-arts performance on the FDDB, PASCAL and AFW datasets by learning to identify parts of the face such as hair, eyes, nose, mouth and beard and then combining those proposals into a face detection.

2.3 Illumination and tampering

Detecting tampering in images usually rely on low level features, such as detecting discrepancies in statistical features of global and local image noise [36]. Another possible indicator is double JPEG compression artifacts that was explored in [37]. Steganalysis feature-based methods [38] [39] have also been used to extract low level information to detect tampering. However, these methods rely on low level features of the image signal, rather than on higher level understanding of the scene.

Some algorithms like [40], rely on higher level image understanding since they seek to detect inconsistencies in illumination and shadows. However, this method is not automatic and requires user interaction.

Deep learning-based methods like [41] have recently been proposed for this problem. In [41], the authors propose a two stream network, where the first stream is used to detect if a face is tampered or not, while the second stream is used to classify each patch of the image as real or fake. The results are combined to produce the final decision.

Chapter 3: Deep Feature based Face Detection for Mobile Devices

Face Detection from mobile front camera captures, which are of importance in domains like Active Authentication (AA), is the primary focus of this section. As mentioned in 1.1, this problem has its own set of challenges and advantages. It is an important problem, because reliable face detection is a necessary condition for further processing like face recognition and authentication.

3.1 Challenges and Advantages

- Partial Visibility: When user's use a mobile phone, they might not be directly looking into it. As shown in Fig. 3.7, there are many cases, when only a partial face is visible.

- Running on phone GPU: While modern mobile devices usually have a GPU, they are not usually Nvidia GPUs. Hence usual CUDA based frameworks like Caffe cannot run on mobile phones. However, OpenCL is an open standard that runs on both Nvidia and non-Nvidia GPUs. For Android phones Render-Script is another alternative for programming GPUs. Thus for the algorithm to run on a mobile platform, it must be reasonably less computationally intensive and be implementable using OpenCL or RenderScript.

Some of the inherent advantages in the task of detecting partial, single faces from front camera captures are as follows:

- Single Face: Usually, a single person uses a mobile at any given time. Therefore, we can focus on detecting a single large face, and can ignore small background faces. Also, one can assume that the user is not very close to the phone. This assumption is checked by plotting a 2D histogram of face height and width, as shown in Fig. 3.1. Analysing the distribution of face sizes, we find that the height of faces vary from around 350 to 700 and the width varies from 300 to 600, which indicates that we can focus on detecting faces in this typical range only.
- Pose: Extreme pose variations are rare since when users tend to interact with mobile devices, usually frontal images are captured.

3.2 Proposed Method

In this section we shall delineate the proposed algorithm, Deep Features based Face Detection on Mobiles (DFFDM) and discuss the motivation for the design choices and its salient features.

3.2.1 Introduction

Transferred learning is an effective way to incorporate deep networks. The first step of the DFFDM algorithm is to extract deep features using the first 5 layers of Alexnet. Different sized sliding windows are considered to account for faces of

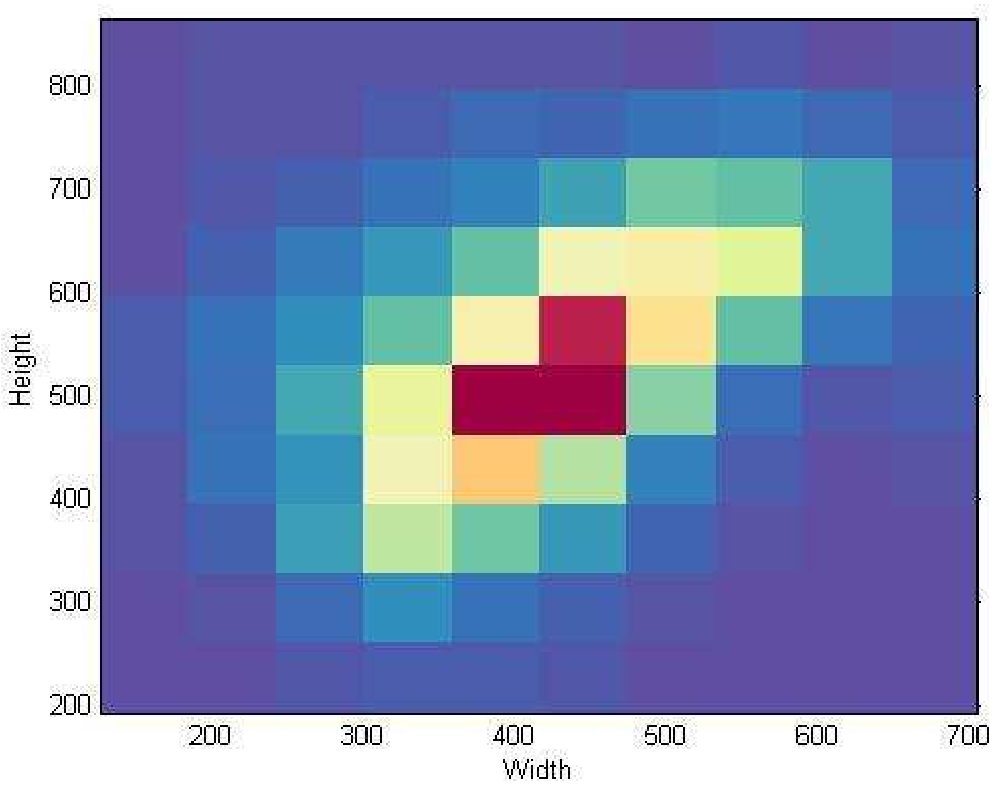

Figure 3.1: Histogram showing distribution of bounding box widths and heights in the training set of the AA-01 dataset

different sizes and an SVM is trained for each window size to detect faces of that particular size. Then detections from all the SVMs are pooled together and some candidates are suppressed based on a overlap criteria. Finally a single bounding box is output by the detector. In the following subsections, the details of the algorithm and model training are provided.

3.2.2 Deep Features Computation

For training 5202 images from the UMDAA-01 database is used. As shown in Fig. 3.1, the 2D histogram of face heights and widths, the height of faces vary from around 350 to 700 and the width varies from 300 to 600. The original videos of the AA-01 dataset are captured at 720p resolution. But since that resolution is too high for our purpose, we resize it to half the resolution, that is 640×360. Therefore typical faces range from 175 to 350 rows and 150 to 300 columns in this reduced resolution.

First we extract deep features from these resized images by forwarding them through AlexNet [42]. We tap the network at the 5th convolutional layer (after the max-pooling). The standard AlexNet reduces an image by a factor of 16 in both dimensions. Thus if the input image is of size $p \times q$, the output is of dimensions $f_r \times f_c \times 256$, where 3.1

$$f_r = \lceil p/16 \rceil, f_c = \lceil q/16 \rceil \tag{3.1}$$

The 3rd dimension is 256 because the 5th layer uses 256 filters. Given the typical face dimensions in the last paragraph, they are reduced by a factor of 16 in the feature space to a heights ranging from 10 to 22 and widths ranging from 9 to 19 approximately.

3.2.3 Training SVM

Once the features have been extracted, we turn to training SVMs to detect the presence of faces. Obviously a single sized sliding window cannot account for these varying sizes, therefore we consider windows of width starting from 8 and increasing to 20 in steps of 2, and height starting from 9 and increasing in steps of 2 to 23. In total we get 56 different window sizes for which we need to train 56 different SVMs. We denote a window by W_{ij}, where i denotes its window height and j denotes its window width.

Let w_k and h_k denote the width and height of the deep feature for the face in the k^{th} training image. The face from the k^{th} training image is used as a positive sample for the SVM W_{ij}, if we have 3.2

$$|i - h_k| \leq t_p \ \& \ |j - w_k| \leq t_p \tag{3.2}$$

for some threshold for selecting positive samples t_p. That is, we select those faces for W_{ij} whose sizes are comparable and close to the window's dimensions.

For negative samples, we extract random patches of size $i \times j$ from those training samples which have no faces. If the k^{th} training sample has a face of size $w_k x h_k$, and for a particular window W_{ij}, if we have 3.3

$$|i - h_k| > t_n \ \& \ |j - w_k| > t_n \tag{3.3}$$

for some threshold for selecting negative samples t_n, then we extract a few random patches from the k^{th} training sample that act as negative samples for W_{ij}.

That is, if the face in an image is of a very different size from the current window W_{ij} under consideration, we extract negative samples from it, so that W_{ij} gives a negative response of faces of different size. Finally, since the UMDAA-01 database does not have many images with no faces, we extract some random negative patches from images of the UPenn Natural Image Database.

Once we have extracted the positive and negative samples for each window size, we discard those window sizes which do not have enough positive examples. Then we flatten the three dimensional deep feature patches into a single dimensional feature vector. Thus for W_{ij}, we get a feature vector of length $i \times j \times 256$. We estimate the mean and standard deviation of features from each window, which are used to normalize the features.

Next we train linear SVMs for each window. Since we get a very long feature vector, it is difficult to train an SVM with all positive and negative samples together. To make the training tractable, we divide the samples into batches and train over many cycles. Specifically let p_{ij} be the number of positive samples for W_{ij}. Then we choose a small number of negative samples say n_{ij} and train the SVM. After that we find the scores of the n_{ij} negative training samples using the weights we get after training and retain only those that are close to the separating hyperplane and discard the rest. We refill the negative samples batch with new negative samples and continue this cycle multiple times. This procedure is repeated for each SVM.

3.2.4 Full Face Detection Pipeline

After the SVMs are trained, we can scan the deep feature extracted from the given image in a sliding window fashion for each SVM. Specifically for an image of size $p \times q$, the deep feature is of f_r rows and f_c columns as given by 3.1 and 256 depth. Therefore for W_{ij}, we can slide the window from position $(1, 1)$, which is the top left, to $(f_r - i, f_c - j)$. Let (r_{ij}, c_{ij}) denote the position where the SVM yields highest score. Then we say that a bounding box, whose top left is at $16 * (r_{ij}, c_{ij})$ and has width $16 \times j$ and height $16 \times i$ is the prediction from W_{ij}. Note that we multiply 16, because the feature space is approximately 16 times smaller than the original image.

Now that we have 1 prediction from each of the 56 SVMs we need to combine them to get a single prediction. A modified version of the non maximal suppression scheme used by Girshick et al. [43] is used for this purpose. First we sort the 56 proposals by their scores and then pick the candidate with highest score. Boxes that overlap significantly with it and have a considerably lower score than it are ignored. This is continued for the next highest scoring candidate in the list, till all boxes are checked. After this we sort the remaining candidates by size. If a larger box significantly overlaps a smaller box, but the larger box has a slightly lower score than the smaller box, we suppress the smaller box. This is useful in the following scenario: A smaller SVM may give a strong response for part of a full face, while the larger SVM responsible for detecting faces of that size may give a slightly lower response. But clearly the larger SVM is making the correct prediction, so we

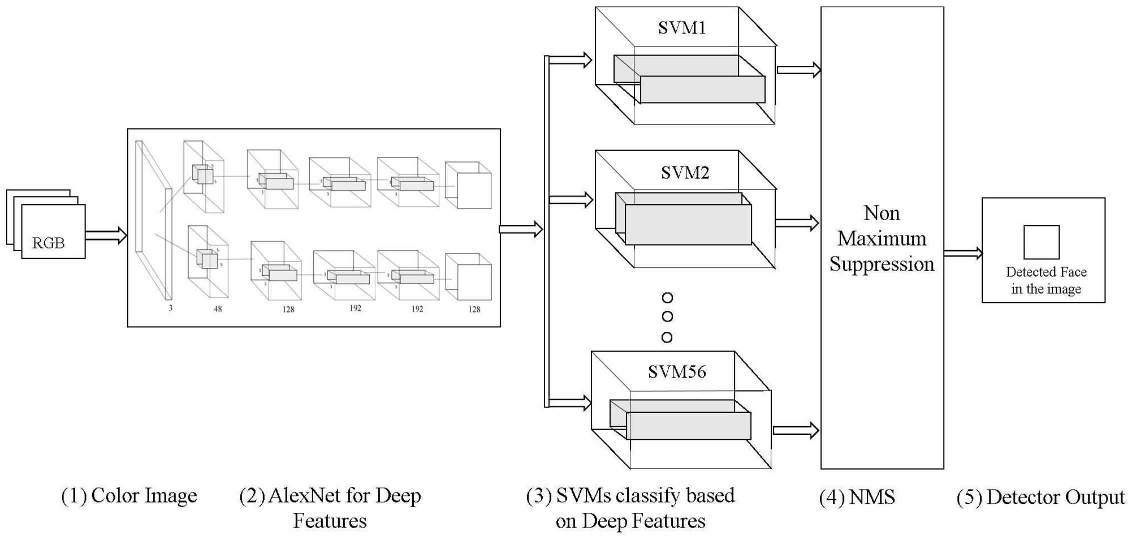
RGB
3
48
128
192
192
128
SVM1
SVM2
SVM56
Non Maximum Suppression
Detected Face in the image
(1) Color Image
(2) AlexNet for Deep Features
(3) SVMs classify based on Deep Features
(4) NMS
(5) Detector Output

Figure 3.2: Overview of the deep feature based face detection algorithm for mobile devices.

need to suppress the overlapping smaller SVM's candidate. After performing these suppressions, we pick the SVM's candidate that has the highest score. We then choose a suitable threshold, and if final candidate's score is larger than that, we declare a face is present at that location, else declare that there is no face present.

3.2.5 Design Choices and Salient Features

In this section, we discuss some of the design choices that were made for DFFDM.

Transferred Learning: The DFFDM algorithm seeks to focus on images typical in the AA case, for which AA-01 is a publicly available dataset. The dataset is very challenging, as can be seen from the samples shown in Fig. 3.7, hence it is clear that a deep network based method must be used to achieve acceptably good performance. While the AA-01 dataset has a large numbered of unannotated frames it only has 8036 images with facial bounding boxes annotated by hand. Given current dataset sizes, it is a small number, and perhaps not enough to train a deep network. One possible solution to this dilemma, is to use transferred learning. As studied in [44], deep features extracted from deep networks trained on large datasets are often transferable to new domains, with a little fine-tuning. With that in mind, we choose to follow this path to address the conflicting requirement of a deep network based method, but on a small training set.

Sliding Window : Sliding window approaches usually work on the principle of extracting appropriate features and then sliding a window and deciding if an object is present in that window or not. This is a very common technique. For example, the classic algorithm on pedestrian detection using HOG features [45] works on this principle. In the proposed algorithm, DFFDM can be thought of as using DCNs to extract the features for the sliding window approach.

Scale Invariance and Robustness to Occlusion : However to make the sliding window approach work for detecting objects of varying scales, we need to extract features across scaled versions of the input image. The approach followed by Ranjan et al. in [30] is based on extracting deep features at multiple resolutions of the image and then training a single SVM to detect faces. Clearly extracting deep features is a very costly operation because of the sheer number of convolutions involved. Passing the image at multiple resolutions through the network increases the workload even more. Therefore the proposed algorithm passes the image through the DCN only once, but trains SVMs of different sizes to achieve scale invariance. Also the different

SVM sizes helps detect partial faces. For example a tall and thin windowed SVMs are usually trained with left half or right half faces, while short and fat windowed SVMs are trained for top half of faces. SVMs whose aspect ratio match a normal face's aspect ratio are trained on full faces. Thus different sized windows help in scale invariance as well as in detecting partial faces.

Number of Parameters : The Alexnet deep feature extractor has 2,334,080 parameters and the 56 SVMs have 3,211,264 parameters, making the total 5,545,344 parameters.

3.3 Implementation

Popular deep learning platforms include Caffe, Theano and Torch. Although these platforms have a CPU only version, they are significantly slower than the GPU enabled versions. These platforms have a CUDA based backend that offloads the heavy, but parallelizable, computations involved in a convolutional deep network to an Nvidia GPU. Thus, although there are multiple platforms in the deep learning ecosystem, the computational backend is dominated by CUDA based code and Nvidia GPUs. Unfortunately, CUDA is proprietary and works only for Nvidias CUDA enabled GPUs. Current mobile devices have GPUs that are predominantly provided by Adreno, Mali and PowerVR. Therefore existing deep learning frameworks are difficult to port on to GPUs made by other vendors.

OpenCL [46] is an open standard, developed by Khronos Group, to support multiple vendors and facilitate cross platform heterogeneous and parallel computing.

All major vendors like Qualcomm, Samsung, Apple, Nvidia, Intel and ARM conform to the OpenCL standard. Thus OpenCL is a portable option or implementing convolutional networks in GPUs other than those made by Nvidia. Recently though, Google has developed RenderScript to facilitate heterogeneous computing on the Android platform.

Mobile devices are obviously not an ideal platform to perform training on massive datasets. But once the model has been trained, we can hope to run the forward pass on mobile platforms. Thus to harness GPUs of mobile devices to perform the convolution heavy forward pass, we have implemented OpenCL and RenderScript based libraries. The OpenCL library is general and should work on any GPU, while the RenderScript library is specifically tailored for Android. An Android specific example is the use of Schraudolp's fast exponentiation [47] to approximately but quickly compute the normalization layer in AlexNet. Full exponentiation takes a significant amount of time and can become bottlenecks in weaker mobile GPUs.

The primary ingredients for a basic convolutional deep network are convolution and activation layers, max pooling layers and normalization layers, each of which can be parallelized on GPUs. By appropriately stacking up these layers in the correct combination and initializing the network with pre-trained weights we can build a CNN easily. For our purpose we have implemented the AlexNet network as described earlier. For an image of size 360×640, a single forward pass, running on a machine with 4th generation Intel Core i7 and Nvidia GeForce GTX 850M GPU, takes about 1 second for the OpenCL implementation. For an image of the same size, on the Renderscript implementation running on different phones, we summarize

Phone	Runtime	GPU	CPU
Moto G	36.7 s	Adreno 305	Qualcomm Snapdragon 400
HTC One (M7)	31.2 s	Adreno 320	Qualcomm Snapdragon 600
Samsung Galaxy S4	28.0 s	Adreno 320	Qualcomm Snapdragon 600
Nexus 5	11.9 s	Adreno 330	Qualcomm Snapdragon 800
LG G3	10.3 s	Adreno 330	Qualcomm Snapdragon 801
Nexus 6	5.7 s	Adreno 420	Qualcomm Snapdragon 805

Table 3.1: Run times of DFFDM on different mobile platforms

the run time results in Table 3.1. Only about 10% or less of this run time is due to max-pooling layer, normalization layer and SVMs. The rest of the time is due to the heavy computations of the convolutional layers. Continuously running the algorithm drains the battery at 0.42% per minute, while leaving the phone undisturbed drains the battery at around 0.16% per minute.

3.4 Evaluation and Results

For evaluation, we consider common metrics like Precision-Recall plots, F1 scores and Accuracy. We compare the performance of our algorithm on the AA-01 [48] and MOBIO [49] [50] datasets with Deep Pyramid Deformable Part Model (DP2MFD) [30], which is among the state-of-the-art algorithms for some challenging datasets like AFW [51] and FDDB [2], deformable part model for face detection (DPM) [52] and Viola Jones detector (VJ) [53].

We compute detections based on 50% intersection over union criteria. Let d be the detected bounding box, g be the ground truth box and s be the associated score of the detected box d. Then for declaring a detection to be valid, we need Eq. (3.4) to be satisfied for some threshold t

$$\frac{area(d \cap g)}{area(d \cup g)} > 0.5 \ \& \ s \geq t. \tag{3.4}$$

3.4.1 AA-01 Dataset

Results on AA-01 dataset are summarized in Table 3.2.

Metric	DFFDM	DP2MFD	DPM	VJ
Max F1	92.8%	89.0%	84.1%	67.7%
Max Accuracy	88.0%	82.3%	76.4%	58.0%
Recall at 95% precision	85.7%	81.7%	72.6%	-

Table 3.2: Comparision of different metrics for various detectors on UMD-AA database

To check the robustness of the detector, we vary the intersection-over-union threshold as defined in Eq. (3.4) from 0.1 to 0.9 and plot the resulting F1 score in Figure 3.5 and accuracy in Figure 3.6. We see that the DFFDM algorithm gives better performance at higher overlap thresholds too.

A few example positive and negative detections are shown in Fig. 3.7. The detections are marked in red, while the ground truth is in yellow. The first row shows a few examples of positive detections with partial faces and the second row shows positive detections with pose variations. The third row shows some false detections, or detections with score lesser than 1. The detector is quite robust to illumination change and is able to detect partial or extremely posed faces.

3.4.2 MOBIO Dataset

Results on MOBIO dataset are summarized in Table 3.3. The MOBIO dataset has full frontal faces only, therefore we get very high performance. DP2MFD beats

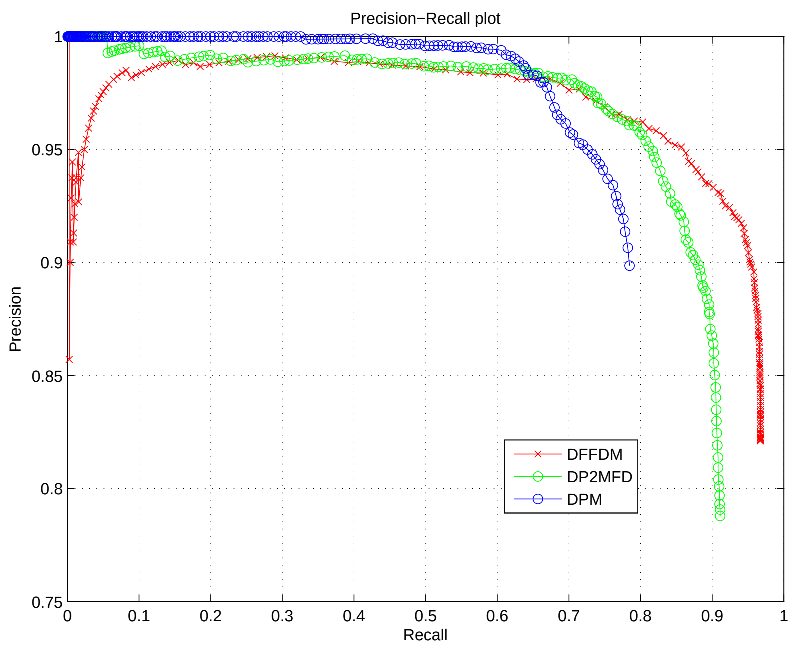

Figure 3.3: Precision Recall plot corresponding to the AA-01 dataset.

our algorithm for this dataset, which can be attributed to the fact that DP2MFD is one of the best algorithms, trained on a large, varied dataset, and for full frontal faces it has near perfect performance over multiple scales. For DFFDM, SVMs of different sizes were trained, based on the typical size of faces captured by the front camera. But sometimes for very large or small faces, the training dataset of UMD-AA may not have enough samples, therefore for extremely scaled faces, DFFMD may fail. This can be remedied by training on a larger database, and also by training SVMs on more scales. A few example positive and negative detections are shown in Figure 3.7. The first row shows positive detections while the second row shows failures. As the examples show, there are some false detections for really large faces, of which we did not have many examples in the AA-01 training dataset on which DFFDM was trained.

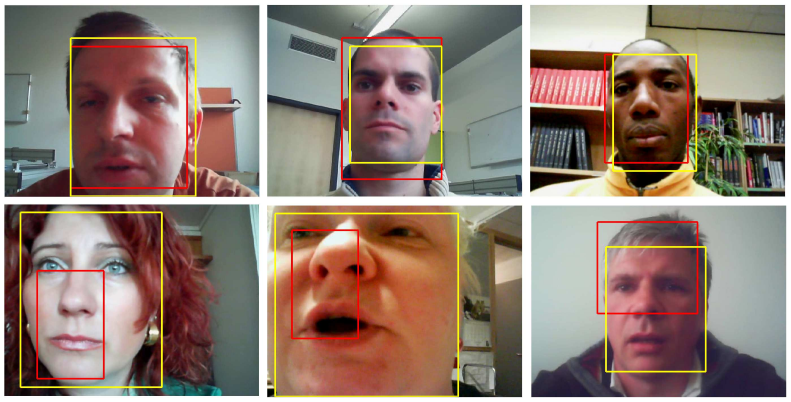

Figure 3.4: Examples of positive (1st row) and negative (2nd row) detections on MOBIO. The detector's output is in red, while ground truth is in yellow.

Metric	DFFDM	DP2MFD	DPM	VJ
Max F1	97.9%	99.7%	97.8%	92.6%
Max Accuracy	96.0%	99.3%	95.8%	86.3%

Table 3.3: Comparision of different metrics for various detectors on MOBIO database

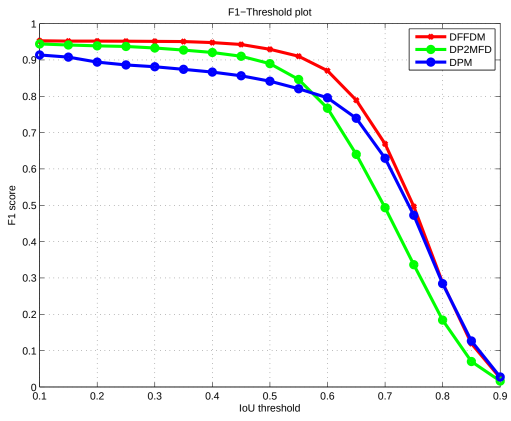

Figure 3.5: Plot showing variation of F1 score with respect to overlap threshold corresponding to the AA-01 dataset.

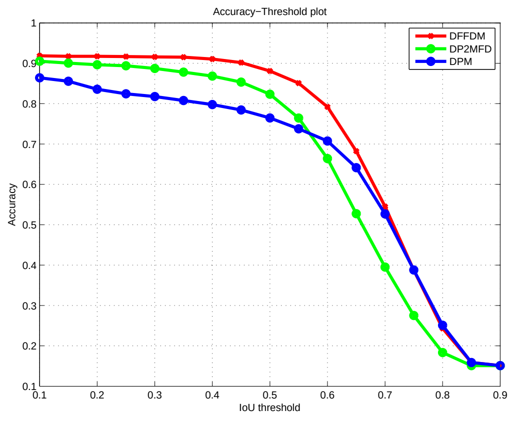

Figure 3.6: Plot showing variation of accuracy with respect to overlap threshold corresponding to the AA-01 dataset.

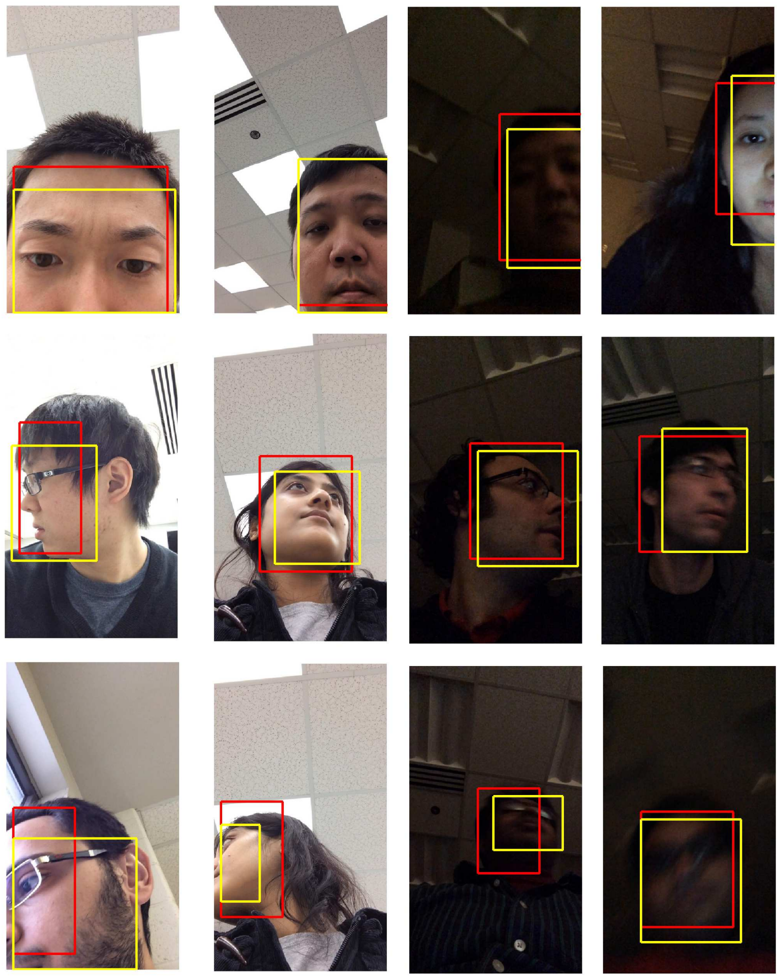

Figure 3.7: Examples of positive detections (with pose variations and occlusion) and examples of negative detections (due to insufficient overlap or small score) in the AA-01 dataset. The detector's output is in red, while ground truth is in yellow.

Chapter 4: DeepSegFace: Pooling Facial Segments to Detect Partial Faces

In this section, we shall introduce DeepSegFace, a deep convolutional neural network (DCNN) that is designed to detect partial faces, by pooling together features from facial segments.

4.1 Proposed Algorithm

In this section we shall propose the DeepSegFace algorithm and look at its relation to other associated methods.

4.1.1 Introduction

DeepSegFace consists of 2 stages, a proposal generation stage, followed by a classification stage. A 'proposal' in this context is a collection of bounding boxes of facial segments (like left half, upper half etc), which might constitute a whole face. In the proposal generation phase, weak, fast classifiers detects facial segments, and comes to a consensus if a face is present. Then these detected segments are sent to a DCNN classifier, which outputs a single probability value about the presence or absence of a face.

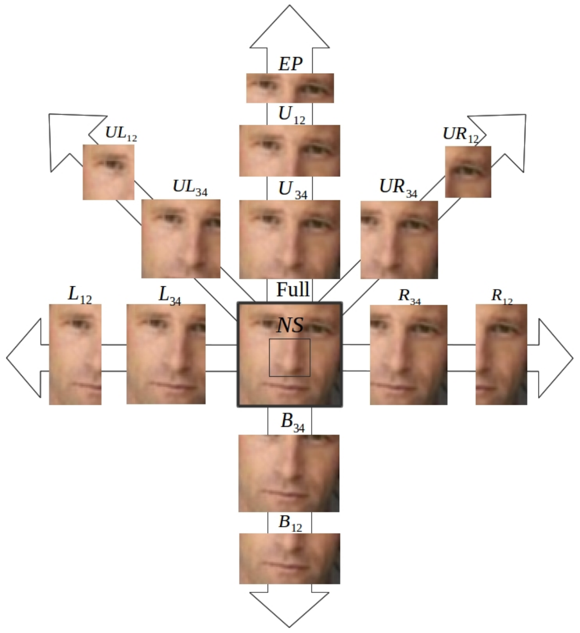

Figure 4.1: Facial segments as proposed in [1]

4.1.2 Proposal generation

In our proposal generation scheme, 9 parts are used, namely: nose ($Nose$), eye-pair (Eye), upper-left three-fourth (UL_{34}), upper-right three-fourth (UR_{34}), upper-half (U_{12}), left three-fourth (L_{34}), upper-left-half (UL_{12}), right-half (R_{12}) and left-half (L_{12}. They are shown in Fig. 4.1. These 9 parts are chosen based on the analysis in selecting the best combination C_{best} for detecting faces in [1].

The set of facial segments is denoted by $S = \{a_k \mid k = 1, 2, \dots M\}$, where $M = 9$ is the number of segments under consideration and a_k is a particular facial segment. $M = 9$ weak Adaboost facial segment detectors are trained to detect each of the segments in S. Following the method described in [1], first we estimate the face centers given a detected segment by simple geometrical considerations. For example, for the U_{12} segment, the face center is estimated to lie at the center of the last row of the detected segment.

Close estimates of face centers are considered to belong to the same face proposal. Once the face centers are estimated, they are clustered into clusters CL_j, $j = \{1, 2, \ldots c_I\}$ as discussed in [1]. Here, c_I is the number of clusters formed for image I. A bounding box for the whole face B_{CL_j} is calculated based on the constituent segments. The proposals are winnowed by deleting clusters that give rise to the exact same full face bounding box proposal.

Consider a cluster composed of segments S. Denoting the power set by $\mathcal{P}$, possible non-empty sets of segments are $S_P \in \mathcal{P}(S) - \{\varnothing\}$ and $\mathcal{P}$. A proposal P can be the full face estimate computed from any of these sets S_P. However, we only consider those sets which have a atleast c segments, so that it is ensured we get a robust estimate. We set $c = 2$, a low value, so that it does not miss out any face (high recall), at the cost of generating lots of false positives (low precision). This lets one generate a large number of proposals, so that any face is not missed in this stage. However since the number of subsets can be very large, since it is exponential in the number of segments, we choose at most $\zeta = 10$ face proposals from each cluster by selecting random subsets of face segments constituting that cluster.

4.1.3 Proposal Classification

DeepSegFace integrates deep CNNs and segments-based face detection from proposals such as [1]. DeepSegFace, accepts as inputs, subsets of the $M = 9$ face parts as discussed earlier, for each image. DeepSegFace is then trained to calculate

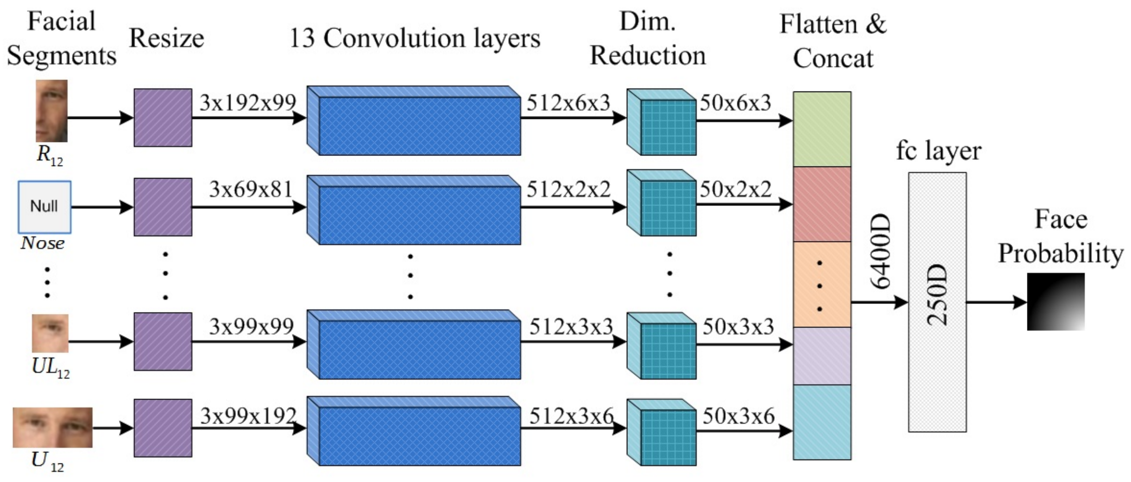

Figure 4.2: Block diagram showing DeepSegFace architecture.

the probability values of the proposal being a face. Finally, a re-ranking step adjusts the probability values from the network. The proposal with the maximum re-ranked score is deemed as the detection for that image.

The architecture of DeepSegFace is arranged according to the classic paradigm in pattern recognition: feature extraction, dimensionality reduction followed by a classifier. A block diagram of the architecture is shown in Fig. 4.2. Different components of the figure are discussed here.

Convolutional Feature Extraction: There are nine convolutional network columns, structurally similar to VGG16 [54] and initialized with its pretrained weights, for each of the nine segments. Thus each network has thirteen convolution layers arranged in five blocks. Each segment in the proposal is resized to standard dimensions for that segment, then the VGG mean value is subtracted from each channel. If a segment is absent in the proposal, zero-input is fed into the network input corresponding to that segment, as shown for the *Nose* segment in the figure.

Dimensionality reduction: The last convolutional feature map has 512 channels, hence naively concatenating them results in a very large $65,536$-D feature vector. Hence, a randomly initialized convolutional layer with filter size 1×1 and 50 feature maps is appended to provide a learnable dimension reduction, which reduces the feature size to 6400 only, as shown in Table 4.1.

Classifier: The classifier receives a 6400 dimensional feature vector from the dimensionality reduction block as shown in Table 4.1. That is passed through a fully connected layer of 250 nodes, followed by a softmax layer of two nodes (both randomly initialized). The two outputs of the softmax layer sum to one and correspond to the probability of the presence or absence of a face.

Re-ranking: The DeepSegFace network outputs the face detection probabilities for each proposal in an image, which can be used to rank the proposals and then declare the highest probability proposal as the face in that image. We only consider the highest score proposal the face, because in the AA scenario, we assume that there is only one face present in the image. However there is some prior knowledge that some segments are more effective at detecting the presence of faces than others. After computing the prior probabilities of detecting a face given that a segment was present in the proposal, we multiply the score from the DeepSegFace network with the mean of the priors.

Table 4.1: Structure of DeepSegFace's Convolutional layers (feature extraction and dimensionality reduction)

Segment	Input	Feature	Dim. Reduce	Flatten
$Nose$	$3 \times 69 \times 81$	$512 \times 2 \times 2$	$50 \times 2 \times 2$	200
Eye	$3 \times 54 \times 162$	$512 \times 1 \times 5$	$50 \times 1 \times 5$	250
UL_{34}	$3 \times 147 \times 147$	$512 \times 4 \times 4$	$50 \times 4 \times 4$	800
UR_{34}	$3 \times 147 \times 147$	$512 \times 4 \times 4$	$50 \times 4 \times 4$	800
U_{12}	$3 \times 99 \times 192$	$512 \times 3 \times 6$	$50 \times 3 \times 6$	900
L_{34}	$3 \times 192 \times 147$	$512 \times 6 \times 4$	$50 \times 6 \times 4$	1200
UL_{12}	$3 \times 99 \times 99$	$512 \times 3 \times 3$	$50 \times 3 \times 3$	450
R_{12}	$3 \times 192 \times 99$	$512 \times 6 \times 3$	$50 \times 6 \times 3$	900
L_{12}	$3 \times 192 \times 99$	$512 \times 6 \times 3$	$50 \times 6 \times 3$	900

4.1.4 Segment Drop-out for Regularization and Data Augmentation

As mentioned in the proposal generation scheme, subsets of face segments in a cluster are used to generate new proposals. For example, if a cluster of face segments contains n segments and each proposal must contain atleast c segments, then it is possible to generate $\sum_{k=t}^{n} \binom{n}{k}$ proposals. However, all the nine parts are redundant for detecting a face, because of significant overlaps and often many segments are not detected by the weak segment detectors.

To make the network robust to missing segment detections, generalize better and also to effectively perform data augmentation, we use segment drop-out when training, i.e. some of the input signals are randomly missing and set to zero. Around sixteen proposals are generated per image. Many of these proposals are actually training the network to detect the same face using different combination of segments.

4.1.5 Comparision with related methods

Table 4.2 shows a comparision between DeepSegFace and two related methods, namely FSFD and SegFace. FSFD introduced the proposal generation scheme described above and SegFace extends it in [55].

Table 4.2: Comparison of the proposed methods

Component	FSFD	SegFace	DeepSegFace
Proposal Generation	Clustering detections from cascade classifiers for facial segments	Clustering detections from cascade classifiers for facial segments	Clustering detections from cascade classifiers for facial segments
Low level features	Prior probabilities	HoG and Priors	Deep CNN features
Intermediate stage	none	SVM for segment i outputs a score on HoG features of segment i	Dimension reduction and concatenation to single 6400D vector
Final classifier	SVM trained on priors	SVM trained on scores from part SVMs and priors	Fully connected layer, followed by a softmax layer
Using priors	Used as features in the final SVM	Used as features in the final SVM	Used for re-ranking of face probabilities in post processing
Trade offs	Very fast but less accurate	Fast but less accurate	Slower but more accurate

4.2 Experimental Results

4.2.1 Experimental Setup

The proposal generator used in FSFD, which was used in trained on LFW, is used as the proposal generator for DeepSegFace. The DeepSegFace DCNN is trained on the training set of UMDAA-02-FD dataset.

For testing DeepSegFace, experimental results on the AA-01-FD and UMDAA-02-FD datasets are compared with a) Normalized Pixel Difference (NPD)-based detector [29], b) Hyperface detector [7], c) Deep Pyramid Deformable Part Model detector [56], d) DPM baseline detector [28], and e) Facial Segment-based Face Detector (FSFD) [1]. Both SegFace and DeepSegFace are trained on 3964 images from AA-01-FD and trained models are validated using 1238 images. The data augmentation process produces $57,756$ proposals from the training set, that is around 14.5 proposals per image. The remaining 2835 images of AA-01-FD are used for testing. For UMDAA-02-FD, $32,642$ images are used for testing. In all experiments with SegFace and DeepSegFace, $c = 2$ and $\zeta = 10$ is considered.

Some samples detections can be seen in Fig. 4.7. The quantitative results are evaluated by comparing the ROC curve and precision-recall curves of these detectors since all of them return a confidence score for detection. The goal is to achieve high True Acceptance Rate (TAR) at a very low False Acceptance Rate (FAR) and also a high recall at a very high precision. Hence, numerically, the value of TAR at 1% FPR and recall achieved by a detector at 99% precision are the two metrics that

Table 4.3: Comparison at 50% overlap on AA-01-FD and UMDAA-02-FD datasets

Methods	AA-01		UMDAA-02	
	TAR at 1% FAR	Recall at 99% Prec.	TAR at 1% FAR	Recall at 99% Prec.
NPD [29]	29.51	11.0	33.49	26.79
DPMBaseline [28]	85.08	83.25	78.48	72.79
DeepPyramid [56]	66.17	42.35	71.19	66.07
HyperFace [7]	90.52	90.32	73.01	71.14
FSFD C_{best} [1]	59.06	55.65	55.74	26.88
SegFace	67.12	63.09	66.44	61.47
DeepSegFace	87.16	86.49	82.26	76.28

are used to compare different methods.

In table 4.3, the performance of SegFace and DeepSegFace are compared with state-of-the-arts methods for both datasets. From the measures on the AA-01-FD dataset, it can be seen that SegFace, in spite of being a traditional feature based algorithm, outperforms several algorithms like FSFD and even DCNN based algorithms such as NPD and DeepPyramid. On the other hand, DeepSegFace outperforms all the other methods except HyperFace in terms of the two evaluation measures on the AA-01-FD dataset. Hyperface is a state-of-the-art algorithm that is trained on over $20,000$ images, compared to only 5202 images used to train DeepSegFace. Also

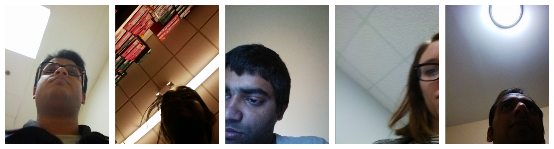

Figure 4.3: Images without even one good proposal returned by the proposal generation mechanism. This bottleneck can be removed by using better proposal generation schemes.

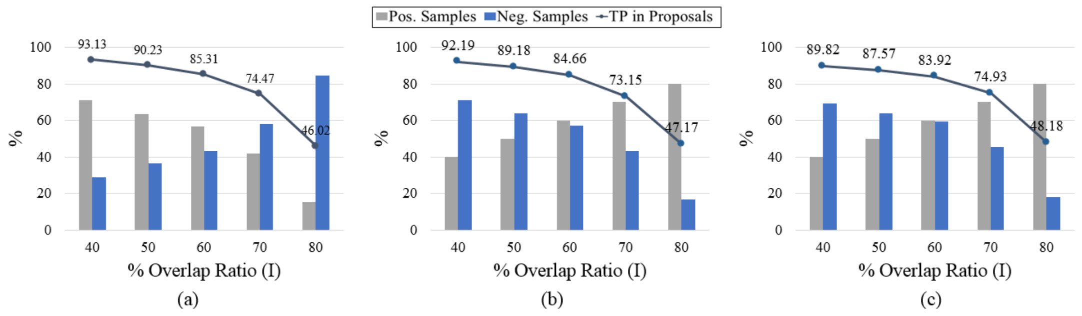

Figure 4.4: (a) for 57756 Train Proposals from AA-01-FD Dataset, (b) 39168 Test Proposals from AA-01-FD Dataset, and (c) 410138 Test Proposals from UMDAA-02-FD dataset. In all cases $c = 2$ and $\zeta = 10$.

Hyperface uses R-CNN to generate face proposals, compared to the fast weak classifiers used by DeepSegFace. Furthur analysis reveals that one of the bottlenecks of DeepSegFace's performance is the proposal generation phase, thus its performance can increase if it uses a more powerful proposal generation scheme, such as R-CNN [57].

In Fig. 4.3, some images are shown for which the proposal generator did not return any proposals or returned proposals without sufficient overlap, even though there are somewhat good, visible faces or facial segments in them. The percentage of true faces that are represented by at least one proposal in the list of proposals for the training and test sets are counted. The result of this analysis is shown in Fig. 4.4. The bar graphs denote the percentage of positive samples and negative samples present in the proposal list generated for a certain overlap ratio. For example, out of the 55, 756 proposals generated for training, there are approximately 62% positive samples and 35% negative samples at an overlap ratio of 50%. Considering the overlap ratio fixed to 50% for this experiment, it can be seen from the line plot in Fig. 4.4(b), corresponding to the AA-01-FD test set, that the proposal generator actually represent 89.18% of the true faces successfully and fails to generate a single good proposal for the rest of the images. Hence, the performance of the proposed detectors are upper-bounded by this number on this dataset, a constraint that can be mitigated by using advanced proposal schemes like R-CNN which generates around 2000 proposals per image for Hyperface, compared to just around sixteen proposals that are generated by the fast proposal generator employed by DeepSegFace.

However, when considering the UMDAA-02-FD test set, which is completely unconstrained and has almost ten times more images than AA-01-FD test set, this upper bound might not be so bad. From Fig. 4.4(c) it can be seen that the upper bound for UMDAA-02-FD is 87.57% true positive value. Now, in Fig. 4.5, the ROC for this dataset is shown. It can be seen that the DeepSegFace method outperforms all the other methods, including HyperFace, with a large margin even with the upper bound (the curve flattens around 87.5%). This is because all the traditional methods suffer so much more when detecting mobile faces in truly unconstrained settings that a true acceptance rate of even 87% is hard to achieve. It is to be noted that the data collection process for AA-01-FD was task-based [14] and hence,

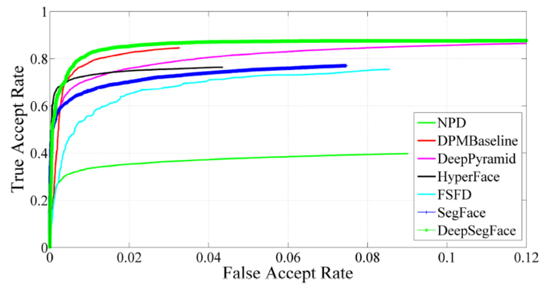

Figure 4.5: ROC curve for comparison of different face detection methods on the UMDAA-02 dataset

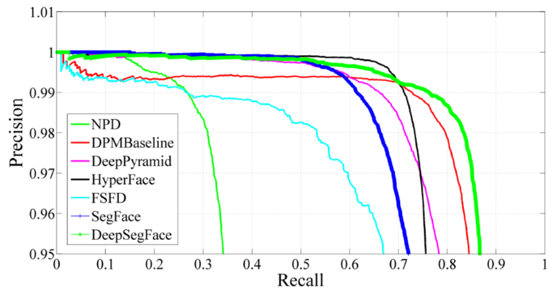

Figure 4.6: Precision-Recall curve for comparison of different face detection methods on the UMDAA-02 dataset.

supervised, while UMDAA-02-FD was collected in a completely natural setting [34]. The precision-recall curve for UMDAA-02-FD dataset is shown in Fig. 4.6. It can be seen that the DeepSegFace method has much better recall at 99% precision than any other method. In both figures, the performance of SegFace is found satisfactory given its dependency on traditional features. In fact, the curves are not too far off from the DeepPyramid method, which is DCNN-based.

The proposals for both test sets are analyzed to reveal that on an average only three segments per proposal are present for both datasets. Thus, while there are nine convolutional networks in the architecture, only three of them need to fire on an average for generating scores from the proposals. When forwarding proposals in batch sizes of 256, DeepSegFace takes around 0.02 seconds per proposal on a GTX Titan-X GPU. SegFace takes around 0.49 seconds when running on a Intel Xeon CPU E-2623 v4 (2.604 GHz) machine with 32GB Memory without multi-threading, hence it is possible to optimize it to run on mobile devices in reasonable time without specialized hardware.

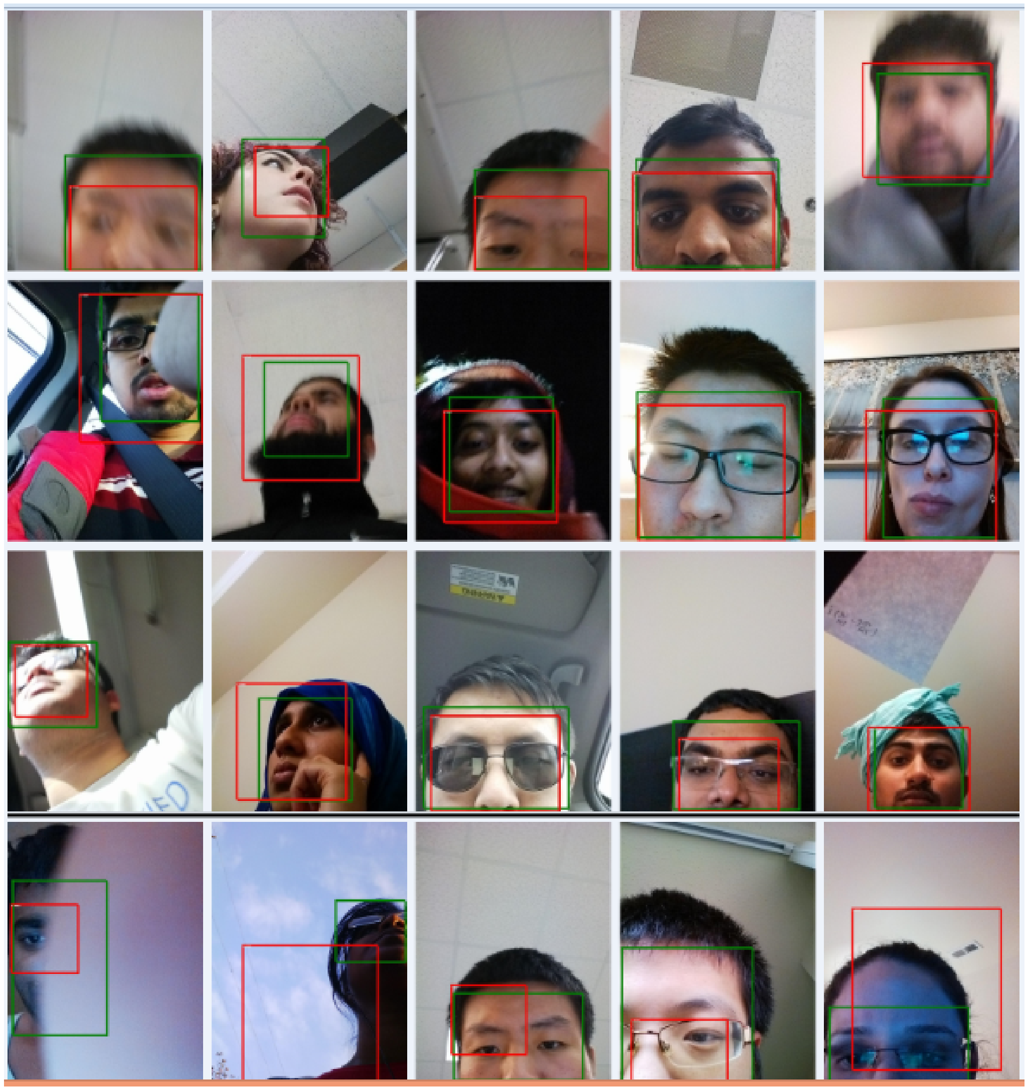

Figure 4.7: Sample face detection outcome of the proposed DeepSegFace method on the UMDAA-02-FD dataset. The first three rows show correct detections at different illumination, pose and partial visibility scenarios, while the last row shows some incorrect detections. Red boxes denote DeepSegFace detection, while green boxes denote ground truth

Chapter 5: Face Illumination Detection and its Applications in Tampering Detection

In this section we shall look at techniques of determining the illumination conditions of a face and use that information to determine if an image has been tampered. Specifically, we hope to identify cases when a face is taken from one image and transplanted into another.

5.1 Introduction

In this section we shall establish some preliminary concepts before diving into details in the next section.

In this work, we assume a single point source light. As such, it can be characterized by 2 numbers, namely, elevation and azimuth angles. It is enough to consider an elevation angle in the range $[0, \pi]$ and an azimuth in the range $[-\pi/2, \pi/2]$, when considering faces. Another possible way of characterizing illumination is by using Spherical Harmonic Lighting [58]. As input to the system, we assume that we already have a face bounding box from a face detector. Thus a 2D face crop is input to the system, which outputs the elevation and azimuth angles of the illumination.

When discussing about tampered images, we focus on a particular case of tampering: One where a face or a person has been cropped from a *source* image and been pasted into a new *target* or *tampered* or *spoofed* image.

To detect tampering, we basically wish to perform some sort of illumination verification where we ask the question if two faces have the same or different illuminations. This is analogous to the problem of face verification, where we ask if two face images belong to the same person. We can adopt two strategies:

- *Compute and Compare*: In this approach, we train a network to compute the illumination and then compare them. The comparision can be done using a simple metric like euclidean distance or by learning a more discriminative metric. This is analogous to face verification by first performing face recognition.
- *Direct Verification*: Here, we train a network that focuses on learning if two illumination conditions are same, rather than trying to predict the actual illumination conditions. This is analogous to face verification by Siamese networks.

5.2 Proposed Method

As mentioned in the previous section 5.1, there are 2 methods of performing verification. However, one can also combine both, that is attempt to predict the actual illumination as well as learn to predict if two illuminations are the same. The proposed solution, IllumNet is described below.

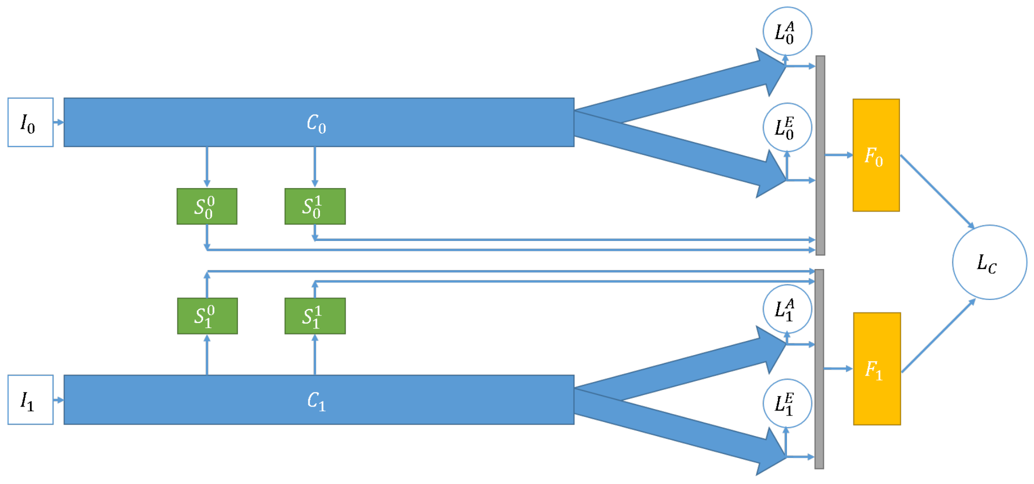

Figure 5.1: Network Architecture of IllumNet, showing the main identical classifiers C_0 and C_1 arranged in Siamese style.

5.2.1 Network Architecture

5.2.1.1 Inputs and Outputs

Firstly, the Hyperface [7] face detector is run on the images, to get face bounding boxes. These crops are then normalized to lie in the range $[-0.5, 0.5]$. The faces crops are resized to 224×224.

We consider the elevation angle to be in the range $[0, \pi]$ and an azimuth in the range $[-\pi/2, \pi/2]$. These ranges are enough to cover light sources lying on a hemisphere in front of the face. Also, instead of trying to predict the exact angles, we focus on predicting an approximate location by considering that there are 13 linearly spaced divisions dividing the π radians considered in the range. Discretizing the space allows us to treat the problem as a classification problem, rather than a regression.

5.2.1.2 Illumination Classifier

Our proposed method consists of a pair of identical networks C_0 and C_1. Both C_i networks are classification networks, which have a common trunk that is made of resnet blocks. At the end of the common trunk, the network branches into 2 subnetworks that each produce 13 probabilities, one number each for the 13 classes of elevation and azimuth angles. The two classifier networks are in blue in Fig. 5.1

5.2.1.3 Siamese Network for Illumination Verification

While directly attempting to predict and compare the illumination to determine if two illuminations are the same might work, it might be a simpler task to answer the question if they are the same. Therefore to bake in verification, along with classification, a Siamese network like arrangement is proposed. As shown in green in Fig. 5.1, convolutional side channel networks S_0^k and S_1^k tap into intermediate layers of the main classifiers C_0 and C_1. Outputs of the side channel networks S and the outputs of the main classifiers C are concatenated to create an intermediate verification feature vector. These feature vectors are passed through some fully connected layers F_0 and F_1 to yield the final verification feature v_0 and v_1. The fully connected layers F act as a learnt metric, to produce well separated verification feature vectors.

5.2.2 Losses

The main classifiers are trained with categorical crossentropy losses. Thus losses $L_0^A, L_0^E, L_1^A, L_1^E$, shown in Fig. 5.1 are all crossentropy losses. L_0^A and L_1^A are losses for the azimuth prediction, while L_0^E and L_1^E penalize the azimuth prediction. To train the network as a verifier, a contrastive loss L_C is used. The contrastive loss is defined as follows:

$$L_C = 0.5(\|v_0 - v_1\|(1 - Y) + max(0, 1 - \|v_0 - v_1\|)Y) \tag{5.1}$$

where Y is an indicator variable that is 0 when the two illuminations are similar, and 1 otherwise.

5.2.3 Training and Testing Data

Datasets with illumination conditions are rare indeed. One such dataset is the CMU Multi-PIE Face Dataset [59]. It contains subjects faces captured under multiple illuminations and viewpoints. While it has sufficient variations in illumination azimuth, it does not have a lot of variation in elevation. From this dataset, we choose $34,211$ training images, $1,925$ validation images and $5,362$ testing images.

Generating synthetic face images is one way to get around this paucity of labelled data. To that end, we use the Basel Face 3D Morphable Face Model [60] (BFM). The BFM geometry consists of $53,490$ 3D vertices connected by $160,470$ mesh triangles. This basic model can be varied by using different linear combinations of 199 principal components that control shape and texture. Random coefficients for the principal components were generated to render faces under random lighting conditions. This was used to create a synthetic dataset of $17,000$ training images, $1,000$ validation images and $4,223$ test images.

Finally, only for testing illumination verification performance we consider two more datasets: AFW and Portraits. AFW is a well known face detection dataset. We consider all the images in AFW that have 5 or less faces. We leave out other images with more than 5 faces, because those faces wold be too small to correctly deduce illumination conditions from. These chosen images yield 230 pairs of faces that belong to the same images, and of the many possible pairs of faces that belong to different images, we randomly choose 230, to keep the dataset balanced. The idea is to create a dataset of pairs of faces from the same or different images.

Portraits dataset was developed at the University of Maryland to study image tampering detection. It consists of pairs of faces that come from tampered and non-tampered images, along with labels indicating if they have been tampered.

5.2.4 Experimental Setup

First, we train the network on the training split of Multi-PIE and then test it on the test split. The ROC of the test results are shown in Fig. 5.2. The solid lines show the verification performance of IllumNet when using the verification features v_i, while the dotted lines show the verification results when directly comparing the illumination predictions from the main classifiers C_0 and C_1. This graph indicates that training in a Siamese fashion helps with verification.

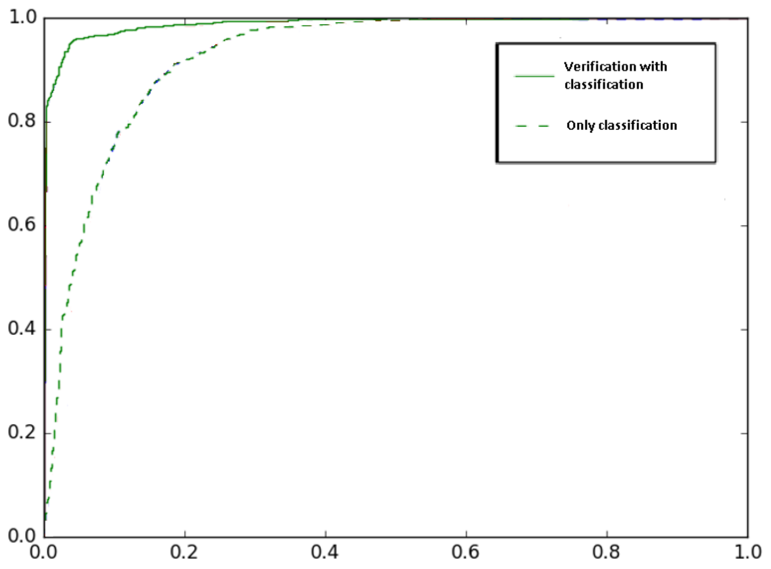

Figure 5.2: ROC of test split of Multi-PIE from a IllumNet trained on Multi-PIE train set. We observe that the adding the siamese style verification network given better performance, compared to just using classification results for verification.

Next, consider the pairs constructed from the AFW dataset and the Portraits dataset as described in section 5.2.3. The ROC plots for these 2 datasets are shown in Fig. 5.3. The histograms of the euclidean distances of the verification features of both datasets are shown in Fig. 5.4. The histograms clearly show that similar illumination conditions create verification feature vectors that produce smaller euclidean distance, compared to dissimilar pairs.

5.3 Adversarial Training

As mentioned in the last section, there is a scarcity of data with annotated ground truth illumination. While synthetic data can be generated, networks trained

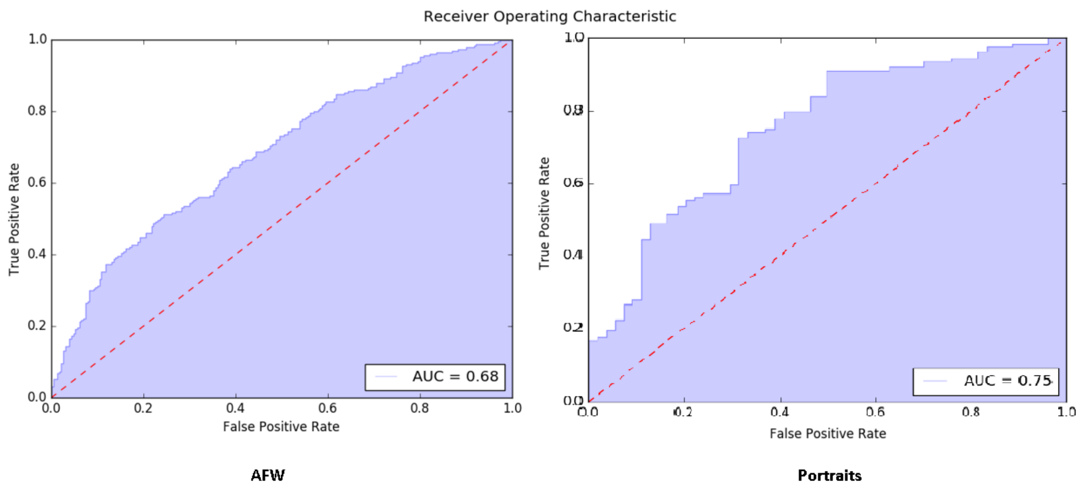

Figure 5.3: Verification ROC of AFW and Portraits datasets

on that cannot be expected to perform as well on real data. Hence we must turn to intelligent data augmentation techniques. We propose the intelligent data augmentation for training an illumination classifier.

5.3.1 Proposed Method

Consider the setup shown in Fig. 5.7. It shows 2 networks A and I lined up serially. For this consider I to be the main classifier C of IllumNet. The other network A is an adversarial network, that tries to generate occlusions that cause network I to produce incorrect classifications. However, it is desirable that network A introduces very small occlusions to fool network I, otherwise without that constraint network A might occlude the whole image.

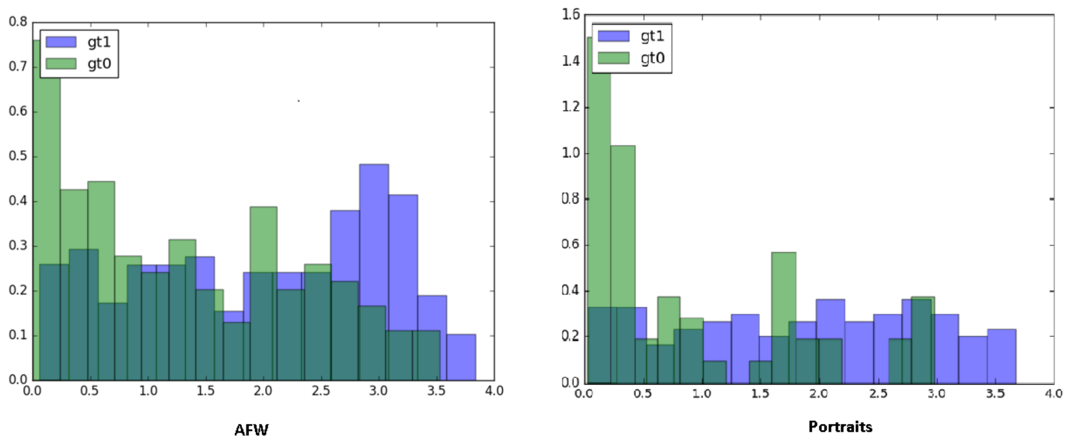

Figure 5.4: Score histograms of similar and dissimilar pairs of AFW and Portraits datasets, showing a clear separation of scores between tampered and non-tampered pairs

5.3.2 Adversarial Occlusion Generation

First we shall discuss Binary Stochastic Neurons, then the architecture of the adversarial network, its losses and other training details.

5.3.2.1 Adversarial Occlusion Generation

Before delving into the adversarial network, let us discuss the use of the Binary Stochastic Neuron (BSN) briefly here. The adversarial network A is being used to generate a binary mask of 1 and 0. To do so it produces a probability. But to generate the actual mask, we need to threshold the probability. However, it is well known that a thresholding function is non-differentiable and thus makes back propagation difficult. To get around this difficulty, we can use Binary Stochastic

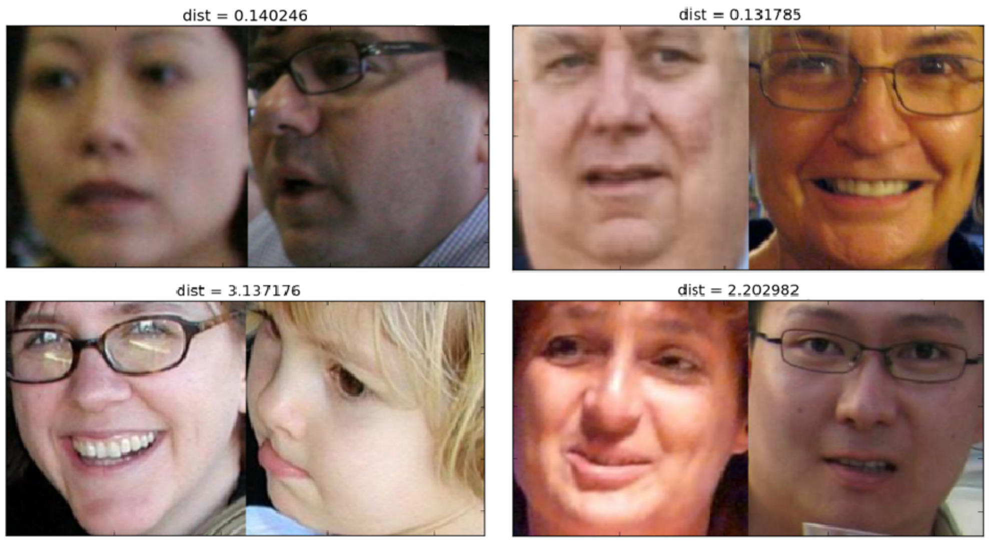

Figure 5.5: Samples from the AFW dataset. The first column contains face pairs from the same images, while the second column contains face pairs from different images. The first row contains face pairs which were predicted to be in the same image, while the second row contains pairs predicted to be in different images

Neurons (BSN). BSNs allow us to simulate a hard threshold activation. The forward pass of BSN is a simple thresholding given by eq. 5.2.

$$BSN(x) = \mathbb{1}_{z<sigm(x)}, z \sim U[0,1] \tag{5.2}$$

There are two varieties of BSN: Straight-through and REINFORCE. The forward equation for both variants is simple thresholding and is given by eq. 5.2. However, they differ in the estimators they use for the gradient during back propagation.

For the straight-through variant, we simply use 1 instead of 0 as the derivative.

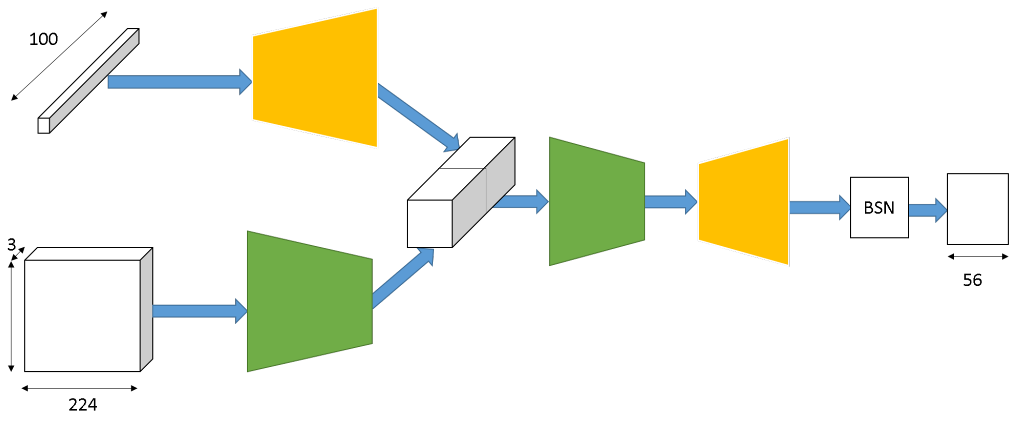

Figure 5.6: Architecture of the adversarial network. Its inputs are a 100-D noise vector and the face crop image, while its output is a binary mask. The green sections are convolutional blocks while the yellow sections are deconvolutional blocks.

Another possibility is to use the derivative of the sigmoid function, along with slope-annealing [61]. The REINFORCE estimator was proposed in [62] and estimates the gradient of the loss L as shown in eq. 5.3.

$$\mathbb{E}\left[\frac{\partial L}{\partial x}\right] = \mathbb{E}[(BSN(x) - sigm(x))(L - c)], c = \frac{\mathbb{E}[(BSN(x) - sigm(x))^2 L]}{\mathbb{E}[(BSN(x) - sigm(x))^2]} \tag{5.3}$$

5.3.2.2 Adversarial Network Architecture

We want to train an adversarial network A accepts an input face image and produces a binary mask of the same size. Pixels that have corresponding mask value 1 are allowed to pass through, while pixels with mask value 0 are suppressed.

To do so, we create an adversarial network, which accepts the image and a 100 dimensional noise vector as input. The image is passed through convolutional layers while the noise is passed through deconvolutional layers. They are then concatenated and passed through more convolutional and deconvolutional layers. Finally, the last deconvolutional layer outputs a $56 \times 56 \times 1$ tensor, which is passed through a BSN layer. The architecture is summarized in fig. 5.6.

Once the 56×56 thresholded map is obtained, it is upsampled by a factor of 4 to yield a 224×224 occlusion map. This is multiplied pointwise with the input image to produce the occluded image.

5.3.3 Training and Losses

The training proceeds as follows: We train the illumination network I with one batch of data, while keeping the adversarial network A fixed, then we train network A while keeping network I fixed. The loss for network I is simply categorical cross entropy loss L_E, since we model the illumination detection as a classification problem rather than a regression.

To ensure we get certain desirable properties in the generated occlusion masks, we employ the following losses listed below:

- *Misclassifiation Loss* L_M: First and foremost, the occlusion should be successful in inducing an incorrect classification in network I. Let N_C be the number of classes, the network predicts class p, while the ground truth is class c. If $|c-p|$ is large, then L_M should be small. This is achieved by defining defining L_M according to eq. 5.4.

- *Distortion Loss* L_D: The occluded area should be as small. We define this

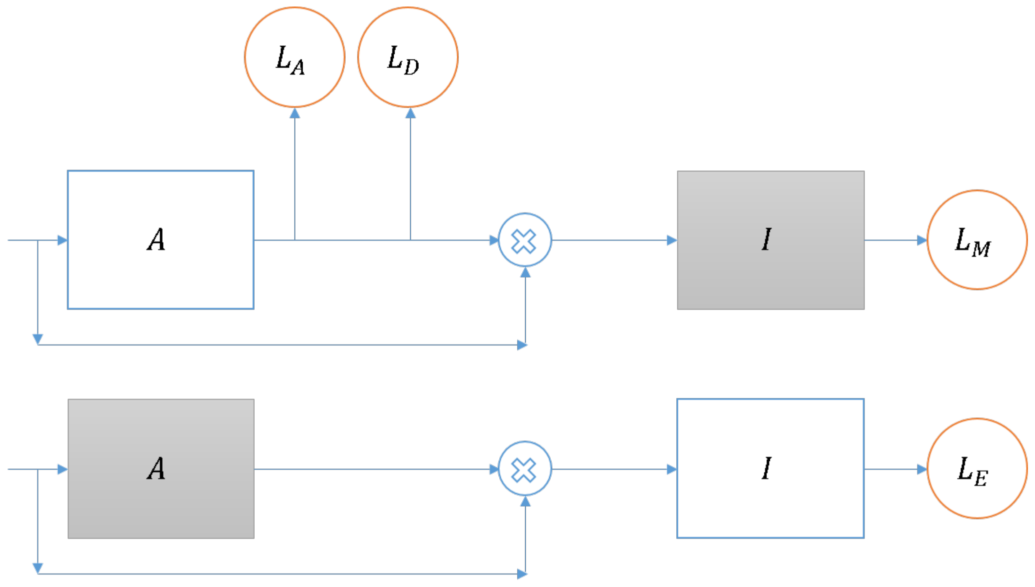

Figure 5.7: Network architecture, training scheme and losses of the proposed adversarial training scheme. In each training iteration, first network A trains with affinity loss L_A, distortion loss L_D and misclassification loss L_M, then network I trains with categorical crossentropy loss L_E.

loss as $L_D = R(P(mask) - 0.2)$. R represents the ReLU function, while P represents the percentage of masked pixels. This loss allows 20% of the pixels to be modified without any loss, and after that the loss scales linearly.

- *Affinity Loss* L_A: The occluded area should be clustered together. If it is spread out randomly, the mask will look like random noise and not an occlusion. A tightly clustered occluded area is useful in simulating real life occlusions. To compute this loss, we find the absolute difference of each element of the mask with its 4 connected neighbours, and then add their averages.

$$L_M = 1 - \frac{|c - p|}{N_C - 1} \tag{5.4}$$

The total loss for the adversarial network A is a weighted sum of the losses discussed above and is given by equation 5.5

$$L_A = L_A + 2L_M + L_D \tag{5.5}$$

The training process of altering between training network A and network I and their corresponding losses are summarized in fig. 5.7.

5.3.4 Experimental Results

Fig. 5.8 shows the effect of adversarial occlusion for some images from the Multi-PIE and synthetic datasets. We observe that the occlusions are well-clustered, occupy only a small region of the image and seem to focus on the cheek region. Cheeks are smooth, reflective regions on the face, which are useful in inferring the source of illumination. By choosing to occlude the cheeks, the adversarial occluder A, makes the job of the illumination network more difficult. The adversarial mask also gives us an indication of the parts of the face that the illumination network I considers important for inferring illumination.

For training, we use both CGI and Multi-PIE datasets as described in the last section. We evaluate by reporting the test set accuracy. When trained without adversary, we get 71.45% accuracy, while on training with adversarial network, we get 73.93% accuracy. Thus this data augmentation scheme, seems to help in

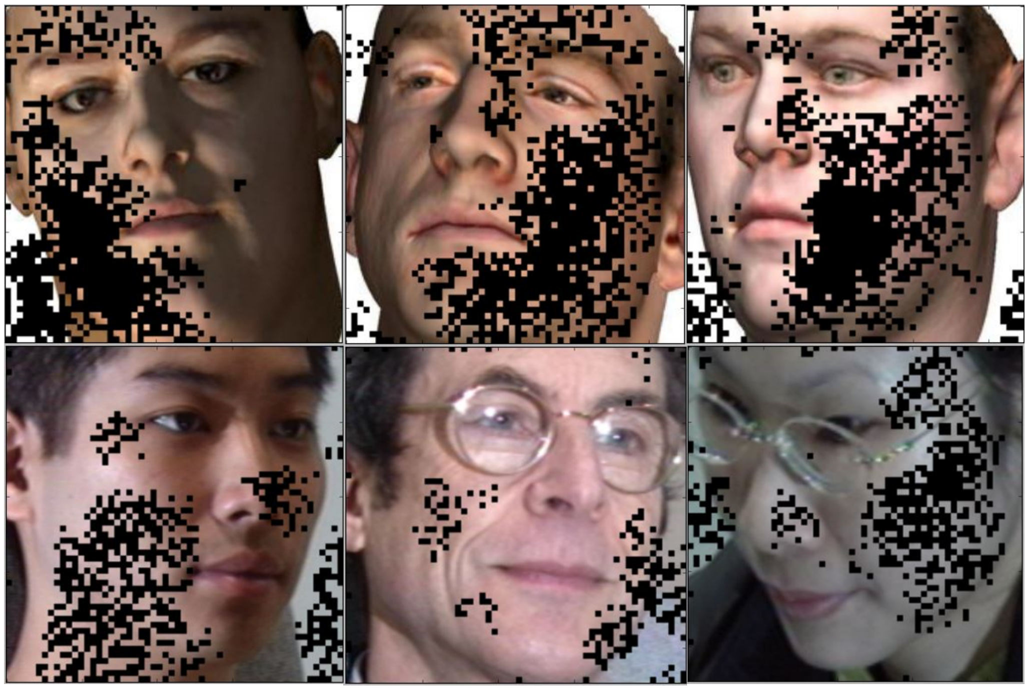

Figure 5.8: Some sample images from Multi-PIE and synthetic datasets that have been adversarially occluded.

generalizing better.

Chapter 6: Conclusion

In this work, we looked at two problems relating to security applications using face analysis.

The first problem we address is detecting faces in the Active Authentication (AA) setting, where we tackle partial face detection. We first propose a transfer learning-based solution to the problem DFFDM, and then a deep learning-based solution, DeepSegFace. Extensive experiments on AA datasets such as AA-01 and UMDAA-02 show that the methods are very effecting in the AA setting.

The second problem addressed is using illumination information to decide if a face image has been transplanted into an image. Under the assumption that faces in the same image should have similar illumination, we propose an algorithm that detects illumination discrepancies as an indicator of tampering. We extend the proposed method by implementing an adversarial data augmentation scheme.

Future directions include incorporating network compression methods into the face detection algorithms that will allow them to run faster on mobile platforms. Another possible extension would be to develop an algorithm that detects tampering by finding inconsistencies in illumination of general objects, and not just faces.

Bibliography

[1] U. Mahbub, V. M. Patel, D. Chandra, B. Barbello, and R. Chellappa. Partial face detection for continuous authentication. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2991–2995, Sept 2016.

[2] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[3] 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 *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2144–2151, Nov 2011.

[4] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2879–2886, Washington, DC, USA, 2012. IEEE Computer Society.

[6] Rajeev Ranjan, Swami Sankaranarayanan, Carlos D. Castillo, and Rama Chellappa. An all-in-one convolutional neural network for face analysis. *CoRR*, abs/1611.00851, 2016.

[7] 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. *CoRR*, abs/1603.01249, 2016.

[8] M. E. Fathy, V. M. Patel, and R. Chellappa. Face-based active authentication on mobile devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[9] P. Samangouei, V. M. Patel, and R. Chellappa. Attribute-based continuous user authentication on mobile devices. In *International Conference on Biometrics Theory, Applications and Systems (BTAS), Arlington, VA*, Sept 2015.

[10] Abdul Serwadda, Vir V. Phoha, and Zibo Wang. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *BTAS*, pages 1–8. IEEE, 2013.

[11] M.O. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Inform. Hiding and Multimedia Signal Process. (IIH-MSP), 2010 6th Int. Conf.*, pages 306–311, Oct. 2010.

[12] A. Primo, V.V. Phoha, R. Kumar, and A. Serwadda. Context-aware active authentication using smartphone accelerometer measurements. In *Comput. Vision and Pattern Recognition Workshops, IEEE Conf.*, pages 98–105, June 2014.

[13] D. Crouse, H. Han, D. Chandra, B. Barbello, and A. K. Jain. Continuous authentication of mobile user: Fusion of face image and inertial measurement unit data. In *International Conference on Biometrics (ICB)*, May 2015.

[14] Heng Zhang, V.M. Patel, M. Fathy, and R. Chellappa. Touch gesture-based active user authentication using dictionaries. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 207–214, Jan 2015.

[15] Chris McCool, Sébastien Marcel, Abdenour Hadid, Matti Pietikainen, Pavel Matejka, Jan Cernocky, Norman Poh, J. Kittler, Anthony Larcher, Christophe Levy, Driss Matrouf, Jean-François Bonastre, Phil Tresadern, and Timothy Cootes. Bi-modal person recognition on a mobile phone: using mobile phone data. In *IEEE ICME Workshop on Hot Topics in Mobile Multimedia*, July 2012.

[16] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham W. Taylor. Learning human identity from motion patterns. *CoRR*, abs/1511.03908, 2015.

[17] Sayantan Sarkar, Vishal M. Patel, and Rama Chellappa. Deep feature-based face detection on mobile devices. *ArXiv e-prints*, abs/1602.04868, 2016.

[18] H. Chan and W.W. Bledsoe. A man-machine facial recognition system: Some preliminary results. 1965.

[19] T. Sakai, T. Nagao, and T. Kanade. Computer analysis and classification of photographs of human faces. *Seminar*, pages 55–62, October 1973.

[20] Stefanos Zafeiriou, Cha Zhang, and Zhengyou Zhang. A survey on face detection in the wild. *Comput. Vis. Image Underst.*, 138(C):1–24, September 2015.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

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

[23] Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. High-performance rotation invariant multiview face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(4):671–686, April 2007.

[24] StanZ. Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 67–81. Springer Berlin Heidelberg, 2002.

[25] S.M. Bileschi and B. Heisele. Advances in component based face detection. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*, pages 149–156, Oct 2003.

[26] M.J. Marin-Jimenez, A. Zisserman, M. Eichner, and V. Ferrari. Detecting people looking at each other in videos. *International Journal of Computer Vision*, 106(3):282–296, 2014.

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

[28] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014.

[29] Shengcai Liao, Anil K. Jain, and Stan Z. Li. A fast and accurate unconstrained face detector. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):211–223, February 2016.

[30] 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.

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

[32] Sachin Sudhakar Farfade, Mohammad J. Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks. In *Proceedings of*

the 5th ACM on International Conference on Multimedia Retrieval, ICMR '15, pages 643–650, New York, NY, USA, 2015. ACM.

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

[34] U. Mahbub, S. Sarkar, V. M. Patel, and R. Chellappa. Active user authentication for smartphones: A challenge data set and benchmark results. In *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 7th Int. Conf.*, Sep. 2016.

[35] Chen Change Loy Shuo Yang, Ping Luo and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[36] Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. *Int. J. Comput. Vision*, 110(2):202–221, November 2014.

[37] T. Bianchi, A. De Rosa, and A. Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2444–2447, May 2011.

[38] J. Fridrich and J. Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, June 2012.

[39] D. Cozzolino, D. Gragnaniello, and L. Verdoliva. Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5302–5306, Oct 2014.

[40] Tiago Carvalho, Hany Farid, and Eric Kee. Exposing photo manipulation from user-guided 3-d lighting analysis, 02 2015.

[41] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1831–1839, July 2017.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[44] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[45] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[46] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010.

[47] Nicol N. Schraudolph. A fast, compact approximation of the exponential function. 1999.

[48] M. E. Fathy, V. M. Patel, and R. Chellappa. Face-based active authentication on mobile devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[49] Chris McCool and Sébastien Marcel. Mobio database for the icpr 2010 face and speech competition. Idiap-Com Idiap-Com-02-2009, Idiap, 11 2009.

[50] Chris McCool, Sébastien Marcel, Abdenour Hadid, Matti Pietikainen, Pavel Matejka, Jan Cernocky, Norman Poh, J. Kittler, Anthony Larcher, Christophe Levy, Driss Matrouf, Jean-François Bonastre, Phil Tresadern, and Timothy Cootes. Bi-modal person recognition on a mobile phone: using mobile phone data. In *IEEE ICME Workshop on Hot Topics in Mobile Multimedia*, July 2012.

[51] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild.

[52] 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.

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

[54] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[55] U. Mahbub, S. Sarkar, and R. Chellappa. Pooling facial segments to face: The shallow and deep ends. In *12th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2017)*, May 2017.

[56] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. A deep pyramid deformable part model for face detection. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th Int. Conf.*, pages 1–8, 2015.

[57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 580–587, Washington, DC, USA, 2014. IEEE Computer Society.

[58] Robin Green. Spherical harmonic lighting: The gritty details. 03 2003.

[59] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image Vision Comput.*, 28(5):807–813, May 2010.

[60] IEEE. *A 3D Face Model for Pose and Illumination Invariant Face Recognition*, Genova, Italy, 2009.

[61] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations 2017 (Conference Track)*, 2017.

[62] Yoshua Bengio. Estimating or propagating gradients through stochastic neurons for conditional computation. arxiv preprint arxiv:1308.3432.